

Министерство образования и науки Российской Федерации  
Федеральное агентство по образованию  
Южно-Уральский государственный университет  
Кафедра информатики

681.3.06 (07)  
К647

**Е.А. Конова, Г.А. Поллак, А.М. Ткачев**

# **ПРАКТИЧЕСКИЙ КУРС ПРОГРАММИРОВАНИЯ НА ЯЗЫКАХ С и С++**

Учебное пособие

Под редакцией Е.А. Коновой

Челябинск  
Издательство ЮУрГУ  
2004

УДК [681.3.06: 519.6] (075.8)

Конова Е.А., Поллак Г.А., Ткачев А.М. Практический курс программирования на языках С и С++: Учебное пособие/ Под ред. Е.А. Коновой – Челябинск: Изд-во ЮУрГУ, 2004. – 170с.

Главная цель пособия – помочь студентам, изучающим языки С и С++, в приобретении навыков практического программирования путем решения специально подобранных задач. В пособии собраны различные задачи по программированию. Всего определены 16 тем, и в каждой около 30-ти заданий, примерно одинаковых по уровню сложности. Уровень сложности постепенно возрастает от одной темы к другой. Такая методика позволяет выработать технику программирования для начинающих и развить имеющиеся навыки для студентов, продолжающих изучение программирования. Особый интерес представляют содержательные задачи, в которых процессу кодирования должен предшествовать этап постановки задачи.

В каждой теме приводятся минимум теоретических сведений и большое количество примеров. В каждом примере назван типовой алгоритм и приведен код программы с подробными комментариями.

Учебное пособие интегрирует опыт многолетней работы авторов. Оно подготовлено на основе курсов программирования, прочитанных авторами в разное время разному контингенту учащихся.

Пособие предназначено, в первую очередь, для студентов специальности 351400, изучающих программирование на С или С++. Может быть полезно для самостоятельного освоения программирования и, безусловно, будет интересно преподавателям, ведущим подобные курсы.

Ил. 1, список лит. – 23 назв.

Одобрено учебно-методической комиссией факультета экономики и управления.

Рецензенты: Тумашов В.И., Прохорова И.А.

© Издательство ЮУрГУ, 2004

## Введение

Главная цель этого пособия – помочь студентам, изучающим языки С и С++, в приобретении навыков практического программирования, потому что нельзя научиться программировать без решения множества практических задач. Процесс работы с данным пособием предполагает знакомство с теорией проектирования алгоритмов, изучение синтаксических правил и механизмов языков программирования С и С++, помогает закреплению знаний теоретического материала, позволяет освоить многие технические и алгоритмические приемы программирования. В итоге ожидается получение базовых знаний, приобретение навыков практического программирования и овладение приемами отладки и тестирования программ.

Практикум предусматривает прохождение всех ступеней программирования при решении каждой задачи: выбор структур данных и алгоритма, кодирование, отладка программы, решение контрольных тестов, анализ результатов.

В пособии приведены различные задачи для программирования. Определены 16 тем, в каждой более 30-ти заданий, которые одинаковы по уровню сложности. Уровень сложности возрастает от одной темы к другой. Так, первая тема, это знакомство с базовыми типами данных, вторая, это использование условного оператора, и так далее по возрастанию уровня. Такая методика позволяет выработать технику программирования для начинающих. Для тех, кто знаком с основными приемами программирования, можно рекомендовать эти задачи для закрепления знаний и развития навыков. Большинство задач разработано авторами, но многие собраны из различных источников, в том числе известных задачников по программированию.

В каждой теме приводятся минимум теоретических сведений, но большое количество примеров. В каждом примере назван типовой алгоритм, и приведен код программы с подробными комментариями. Все коды примеров протестированы в среде Borland C++ версии 3.0. Чтение текстов примеров и разбор типовых решений поможет начинающим в освоении практических приемов программирования и выработке собственного стиля.

Особый интерес представляют содержательные задачи, в которых постановка задачи выполнена на вербальном уровне, то есть, не формализована. Здесь студент должен самостоятельно осмыслить задачу, выбрать модель, предложить структуру данных и выбрать или разработать алгоритм ее решения. Опыт показывает, что часто именно этот этап в практическом программировании является наиболее трудным.

Учебное пособие консолидирует многолетний опыт работы авторов в преподавании различных курсов программирования на различных языках различного контингенту учащихся.

Пособие предназначено для студентов специальности 351400 «прикладная информатика», изучающих программирование на С или С++. Будет полезно для самостоятельного изучения программирования и, безусловно, будет интересно преподавателям, ведущим различные дисциплины программирования.

## Тема 1. Базовые типы данных. Вычисления по формулам.

### Простые программы на языке C++

Простыми можно считать такие задачи, у которых алгоритм решения линейный, то есть все действия выполняются в порядке записи описывающих их операторов. Простые задачи сводятся к линейному процессу со следующими основными пунктами:

- ввести исходные данные;
- выполнить вычисления по формулам;
- вывести результат.

Последовательность решения должна складываться из следующих этапов.

1. Анализ условия задачи.
2. Определение необходимых входных данных: сколько их, какого типа каждое данное, какое имя следует присвоить переменной, обозначающей данное. Тип переменной должен быть определен в объявлении, а имя должно быть значимое, то есть подчеркивать назначение переменной, ее логический смысл.
3. Определение выходных данных программы: сколько их, какие их типы, как назвать переменные для обозначения результатов.
4. Определение списка рабочих переменных, если необходимо.
5. Определение необходимых формул вычисления, их последовательности и правил записи.
6. Кодирование алгоритма.

При кодировании необходимо соблюдать синтаксические правила и следить за стилем программирования. Эти требования можно сформулировать в пунктах.

1. Объявить все переменные программы.
2. Правильно оформить ввод данных. Перед каждым оператором ввода необходимо записывать инструкцию, которая подскажет пользователю, что он должен сделать в данный момент времени.
3. Правильно записать формулы вычислений.
4. Грамотно оформить вывод результата. Вывод следует дополнить необходимыми заголовками и пояснениями, числовые данные форматировать.

**Пример №1.** Пусть требуется вычислить длину окружности, площадь круга и объем шара одного и того же радиуса.

Опишем последовательность решения.

1. Анализ алгоритма показывает, что он линейный.
2. Входная переменная одна, это значение радиуса, в общем случае величина вещественная. Обозначим его именем  $r$ .
3. Выходных данных три. Обозначим их именами  $l$ ,  $s$ ,  $v$ . Все имеют вещественный тип.
4. Формулы вычислений общеизвестны.

// Код программы примера №1.

```
#include <stdio.h>           // Библиотека стандартного ввода-вывода.  
#include <math.h>            // Библиотека математических функций.  
#define PI 3.1415926        // Значение константы  $\pi$ .
```

```

void main (void)
{
    float      r;                // Объявление всех переменных.
    float      l, s, v;
    printf ("\nНаходим значение длины окружности, площади круга и объема шара \
            указанного радиуса: \n");    // Вывод заголовка.
    printf ("\nВведите значение радиуса:\n"); // Приглашение для ввода данных.
    scanf ("%f", &r);                // Ввод исходного данного.
    // Вычисления по формулам:
        l = 2.*PI*r;
        s = PI*r*r;
        v = 4.*PI*r*r*r / 3.;
    // Вывод результатов
    printf ("r=%6.2f, l=%6.2f, s=%6.2f, v=%6.2f\n ", r, l, s, v);
} // End of main

```

Многие ошибки начинающих программистов вызваны незнанием информации о типах данных. Объявление переменной в программе указывает ее имя и тип. Тип переменной определяет способ хранения переменной, а именно, сколько памяти выделено для переменной и как ее значение размещено в этом объеме. Тип также определяет диапазон значений, которые может принять переменная, и операции, которые можно к ней применить. Способ сохранения переменных вещественного типа (double, float) радикально отличается от способа хранения данных целых типов (int, char). Тип констант, если они есть в тексте программы, определяется их записью, так константа 2 будет целочисленной, а константы 2.0 или 2. будут вещественными. При вычислении значений выражений, в которых смешаны данные разных типов, выполняется преобразование (приведение) типов. Например, если в программе объявлены целочисленные переменные:

```

int      a = 1, b = 2;
float    f;

```

выполнено деление, и значение присвоено переменной вещественного типа:

```

f = a / b;

```

то результат будет равен 0.0, потому что сначала выполняется целочисленное деление, а потом присваивание. Чтобы результат был равен 0.5, нужна запись с использованием явного преобразования типов:

```

f = (float) a / float (b);

```

**Пример №2.** Особенности, связанные с преобразованием типов, рассмотрим на следующем примере. Дано значение угла, измеренное в градусах, минутах и секундах. Требуется перевести значение угла в радианную меру. Формула

$$\text{вычисления } \lambda_{\text{рад}} = \frac{\lambda_{\text{град}} \cdot 360}{2 \cdot \pi}.$$

// Код программы примера №2. Программа имеет скрытую ошибку.

```

#include <stdio.h>
#include <math.h>

```

```

void main (void)
{
int      Grad, Min, Sec;          // Исходные данные всегда целые.
float     GRAD;                  // Новая переменная для
                                // вычисления полного значения угла.

float     Rad;                   // Итоговое значение.
printf ("Введите значение угла в градусах, минутах и секундах:\n");
scanf ("%d%d%d", &Grad, &Min, &Sec);
// Полное значение угла:
    GRAD = Grad + Min / 60 + Sec / 3600;
// Перевод в радианы:
    Rad = Grad * M_PI / 360;      // Константа M_PI объявлена в <math.h>.
printf ("Значение угла в радианах: %12.6f\n", Rad);
} // End of main

```

Для проверки стоит выполнить два тестовых примера, вот их результаты:

Введите значение угла в градусах, минутах и секундах:	
2 0 0	
Значение угла в радианах:	<b>0.017453</b>
Введите значение угла в градусах, минутах и секундах:	
1 59 59	
Значение угла в радианах:	<b>0.008727</b>

Жирным начертанием выделены результаты решения задачи. Как видим, почти одинаковые входные данные дают разные результаты. Источником ошибки величиной в один градус является строка, вычисляющая полное значение угла. При делении целых результат равен нулю, поэтому значения минут и тем более секунд отброшены при вычислении GRAD.

// Код программы примера №2.

// Для устранения ошибки используется явное преобразование типов.

```

#include <stdio.h>
#include <math.h>
void main(void)
{
int      Grad, Min, Sec;
float     Rad;
float     GRAD;
printf ("Введите значение угла в градусах, минутах и секундах\n");
scanf ("%d%d%d", &Grad, &Min, &Sec);
// Полное значение угла:
    GRAD = (float) Grad + (float) Min / 60. + (float) Sec / 3600.;
// Перевод в радианы:
    Rad = GRAD * M_PI / 360;
printf ("Значение угла в радианах = %12.6f\n", Rad);
} // End of main

```

**Пример №3.** Рассмотрим особенности вычислений с использованием данных целого типа. Сформулируем содержательную задачу. Пусть для покупки подарков выделена некоторая сумма в рублях. Требуется узнать, сколько можно купить подарков, если известна цена одного подарка, и определить, сколько денег останется. Наоборот, если нужно купить  $N$  подарков, то найти наибольшую стоимость одного подарка, и сколько останется денег.

// Код программы примера №3. Решение в целых числах.

```
#include <stdio.h>
void main (void)
{
    int      Sum;           // Выделена сумма денег.
    int      Cost;          // Стоит один подарок.
    int      Cou;           // Количество, которое можно купить на эту сумму.
    int      N;             // Количество, которое хочется купить.
    int      Rest;          // Останется денег.
    printf ("Введите сумму, которой Вы располагаете, и стоимость одного подарка\n");
    scanf ("%d%d", &Sum, &Cost);
    // Найдем количество и остаток:
        Cou = Sum / Cost;    // Целочисленное деление.
        Rest = Sum % Cost;   // Остаток от деления.
    printf ("Можно купить %d подарков, останется %d рублей.\n", Cou, Rest);
    // Найдем стоимость и остаток.
    printf ("Введите число подарков, которое хочется купить\n");
    scanf ("%d", &N);
        Cost = Sum / N;
        Rest = Sum % N;
    printf ("Наибольшая стоимость подарка %d, останется %d рублей.\n", Cost, Rest);
} // End of main
```

Замечание. Операции целочисленного деления и вычисления остатка от деления можно применять только к данным целых типов. Если такие операции нужно выполнить для данных вещественных типов, используются функции `ceil`, `floor`, `fmod` библиотеки `<math.h>`. Функция `ceil` округляет вещественное число вверх, функция `floor` округляет вниз, а функция `fmod` находит остаток деления по модулю. Все функции имеют аргументы типа `double` и возвращают значение типа `double`. Приведение типа при переходе от `float` к `double` не обязательно, так как не приведет к потере данных.

Рассмотрим тот же пример с использованием вещественного типа данных. Все денежные суммы должны вычисляться в рублях и копейках, следовательно, переменные `Sum`, `Cost` и `Rest` имеют вещественный тип. При вычислениях необходимо округление вниз для получения целочисленного значения, и для округления с точностью до копеек. Деление должно быть по модулю.

// Код программы примера №3. Решение в вещественных числах.

```
#include <stdio.h>
#include <math.h>
```

```

void main(void)
{
float      Sum;           // Выделена сумма денег.
float      Cost;          // Стоит один подарок.
int        Cou;           // Количество, которое можно купить на эту сумму.
int        N;             // Количество, которое хочется купить.
float      Rest;          // Останется денег.
printf("Введите сумму, которой Вы располагаете, и стоимость одного подарка\n");
scanf("%f%f", &Sum, &Cost);
// Найдем количество и остаток:
    Cou = (int) floor (Sum / Cost); // Целое число получено округлением вниз.
    Rest = fmod (Sum, Cost);        // Остаток от деления.
printf("Можно купить %d подарков, останется %6.2f рублей.\n", Cou, Rest);
// Найдем стоимость и остаток.
printf("Введите число подарков, которое хочется купить\n");
scanf("%d", &N);
    Cost = Sum / (float) N;
    // Точность вычислений должна быть 0,01.
    Cost = floor (Cost * 100) / 100.;
    Rest = fmod (Sum, Cost);
// Rest = Sum – Cost * N; // Аналог предыдущей строки без функции fmod.
printf("Наибольшая стоимость подарка %6.2f, останется %6.2f руб.\n", Cost, Rest);
} // End of main

```

Важное замечание. Если этот пример получить как точную копию первого, то хорошо будет работать только первая формула вычислений. Вторая даст ошибку, вызванную наличием погрешности вычислений, которая должна быть менее одной копейки. Для устранения погрешности добавлена строка  $\text{Cost} = \text{floor}(\text{Cost} * 100) / 100.;$

О смысле этой строки предлагается подумать читателю.

### Варианты заданий

Задание 1. Даны длины катетов прямоугольного треугольника. Найти его острые углы, вывести в градусной мере с точностью до минут. Указание: функция  $\text{arctg}$  объявлена в библиотеке `math.h`.

Задание 2. Даны длины катетов прямоугольного треугольника. Найти его периметр и площадь. При выводе округлить до двух знаков.

Задание 3. Дан положительный угол в радианах. Найти, сколько градусов и минут содержит данный угол.

Задание 4. Дано вещественное число. Найти и напечатать его целую, дробную части и округленное значение. Использовать функции округления.

Задание 5. Камень бросили вверх со скоростью  $V$ . Определить расстояние от земли в некоторые моменты времени  $t_1$  и  $t_2$ . Вывести с точностью два знака.

Задание 6. Даны координаты точек  $A(x_1, y_1)$  и  $B(x_2, y_2)$  гиперболы  $y = k/x + b$ . Найти и напечатать значения  $k$ ,  $b$ .

Задание 7. От полуночи минутная стрелка описала угол  $K$  градусов



(целочисленное значение). Определить, какое время (часы, минуты) показывают часы. Указание: использовать операции целочисленного деления.

Задание 8. Даны значения переменных  $A$  и  $B$ . Найти частное от деления  $A$  на  $B$  и частное от деления  $B$  на  $A$ . Найти остаток от деления  $A$  на  $B$  и остаток от деления  $B$  на  $A$ . Найти вещественные значения дробей  $\frac{A}{B}$  и  $\frac{B}{A}$ .

Задание 9. Даны координаты левой верхней и правой нижней вершин квадрата. Найти длину стороны квадрата, найти длину диагонали, найти координаты точки пересечения диагоналей квадрата.

Задание 10. Две прямые на плоскости проходят через начало координат и точки  $A(x_1, y_1)$  и  $B(x_2, y_2)$ . Найти в градусной мере с точностью до градусов угол наклона биссектрисы угла, образованного прямыми.

Задание 11. Дано натуральное четырехзначное число. Найти сумму его цифр. Указание: использовать операции целочисленного деления.

Задание 12. Определить, как наименьшим числом купюр можно выдать сумму  $K$  рублей ( $K < 9999$ ), если есть купюры достоинством 1000 руб., 500 руб., 100 руб., 50 руб. и 10 руб. Оставшуюся мелочь выдать рублями.

Задание 13. Камень бросили в колодец, и через  $t$  секунд послышался всплеск. Определить расстояние от сруба до воды.

Задание 14. Две прямые на плоскости заданы своими уравнениями. Известно, что они пересекаются. Найти координаты точки пересечения заданных прямых.

Задание 15. Дано натуральное число  $N$ , в записи которого ровно пять знаков. Определить цифры, составляющие число. Учесть, что наибольшее целочисленное значение (int) равно 32767.

Задание 16. На плоскости задан треугольник координатами своих вершин. Найти длины сторон треугольника.

Задание 17. Найти скорость камня, брошенного под углом  $\lambda$  к горизонту и пролетевшего  $b$  метров за  $t$  секунд.

Задание 18. На плоскости заданы два луча, выходящие из начала координат под углами  $\lambda$  и  $\beta$  к оси  $Ox$ . Найти в градусной мере с точностью до градусов углы наклона биссектрисы угла, образованного лучами, а также угол между лучами.

Задание 19. Квадрат задан длиной стороны. Найти радиус окружности, площадь которой равна площади этого квадрата.

Задание 20. На плоскости заданы две окружности своими радиусами. Найти процентное соотношение площади первой окружности к площади второй.

Задание 21. Зарплата сотрудника складывается из ставки минимальной зарплаты, умноженной на разрядный коэффициент (вещественное значение в диапазоне от 1 до 3), уральского коэффициента (15%) и премии (в %). Из зарплаты вычитается подоходный налог (12%). Вводя необходимые исходные данные, вычислить и вывести зарплату сотрудника в рублях.

Задание 22. Известно, что прямая, заданная уравнением  $y = Kx + b$  пересекает ось координат в точках  $A$  и  $B$ . Найти координаты точек пересечения, вывести с точностью два знака после запятой.

Задание 23. Участок земли треугольной формы имеет длины сторон  $L_1$ ,  $L_2$ ,  $L_3$ .

Найти площадь этого участка, вывести с точностью два знака после запятой.

Задание 24. Прямоугольной формы строительный блок имеет ширину  $w$ , высоту  $h$  и глубину  $l$ . Найти объем и площадь поверхности строительного блока.

Задание 25. Прямоугольной формы строительный блок имеет ширину  $w$ , высоту  $h$  и глубину  $l$ . Найти, сколько блоков войдет в квадратный контейнер с длиной стороны  $L$ .

Задание 26. В первый день тренировок спортсмен пробежал 10 км. Каждый день он пробегает на 10% больше, чем в предыдущий день. Найти и вывести на экран путь, пройденный спортсменом во второй, третий и четвертый дни тренировок.

Задание 27. В пачке  $N$  листов бумаги размера А4 плотностью  $P$  гр. на кв. см. Найти вес пачки бумаги с точностью два знака после запятой.

Задание 28. На плоскости заданы две точки своими координатами. Найти расстояние от начала координат до каждой точки и расстояние между точками.

Задание 29. Дан круг радиуса  $R$ . Найти площадь круга. Найти площадь сектора круга с указанным значением центрального угла (в градусах).

Задание 30. Известны стоимость конфет, печенья и яблок. Найти общую стоимость покупки, если куплено  $X$  кг. конфет,  $Y$  кг. печенья и  $Z$  кг. яблок.

Задание 31. Известны стоимость системного блока, монитора, клавиатуры и мыши. Найти, во сколько обойдется покупка  $N$  компьютеров.

Задание 32. Известно, что для кормления одного лабораторного животного нужно  $K_1$  гр. корма для мыши,  $K_2$  гр. корма для крысы,  $K_3$  гр. корма для морской свинки. Найти, сколько съедят в день  $X$  мышей,  $Y$  крыс,  $Z$  морских свинок. Найти, сколько корма нужно в месяц.

Задание 33. В подарке 2 шоколадки по цене  $K_1$  руб., 2 яблока по  $K_2$  руб. и коробка печенья стоимостью  $K_3$  руб. Найти, сколько будут стоить  $N$  подарков.

Задание 34. В очереди к врачу  $N$  человек. Врач беседует с пациентом примерно 15 минут. Если сейчас 12 часов, а до бассейна бежать 30 минут, стоит ли занимать очередь, если надо успеть на сеанс в  $Ч$  часов (сеансы начинаются каждый час).

Задание 35. Одна жемчужина диаметром  $d$  стоит Cost рублей. Сколько будет стоить ожерелье на шею, обхват которой  $D$  см.

Задание 36. Летит стадо гусей, а навстречу ворона. Ворона говорит: «Как много вас, гусей, наверное,  $N$ ?», на что вожак отвечает «Если взять нас столько, и еще пол столько, и четверть столько, то будет  $N$ ». Сколько же гусей в стае? Иметь в виду, что гуси и вороны считают не очень хорошо, и  $N$  не может быть любым.

## Тема 2. Простые типы данных. Управление алгоритмом с использованием условного оператора if или переключателя switch

Условный оператор if изменяет последовательность выполнения операторов программы в зависимости от условий, сложившихся при ее выполнении. Классическая схема оператора if в C++ предполагает разветвление на две ветки, одна из которых может быть пустой. В более сложных случаях, когда проверяется множество условий, используются сложные (вложенные) условные операторы или сложные логические выражения.

Оператор переключатель switch используется для тех же целей, но позволяет организовать множественное ветвление. Если оператор if может передать управление только на две ветви, то switch на произвольное число ветвей.

**Пример №1.** Требуется найти корни квадратного уравнения вида  $ax^2 + bx + c = 0$ . Выполним этапы решения.

1. Анализ условия задачи. Квадратное уравнение может иметь одно или два решения, или не иметь их вообще. Результат решения зависит от определителя уравнения, вычисляемого по формуле  $d = b^2 - 4 \cdot a \cdot c$ .

2. Входными данными являются коэффициенты уравнения  $a, b, c$ . Чтобы уравнение осталось квадратным, первый коэффициент должен быть отличен от нуля. Имена переменных, обозначающих коэффициенты, могут совпадать с их математической записью, то есть  $a, b, c$ . Тип коэффициентов вещественный.

3. Выходными данными программы будут значения корней уравнения, если их удастся найти по формулам  $x_{1,2} = \frac{b^2 \pm \sqrt{d}}{2 \cdot a}$ . Если же корней нет, об этом следует напечатать сообщение.

4. Для вычисления и хранения значения определителя требуется ввести новую переменную, которая не является результатом задачи, но необходима для ее решения. Такие переменные называются рабочими переменными программы. Обозначим ее буквой  $d$ , как и в постановке задачи.

5. Определение необходимых формул вычисления очевидно.

6. Кодирование алгоритма. При выполнении алгоритма можно выполнить предварительное словесное описание:

- 1) ввести значения коэффициентов (названы  $a, b, c$ );
- 2) вычислить дискриминант (назван  $d$ );
- 3) если дискриминант отрицательный, решения нет, вывести сообщение;
- 4) если дискриминант  $>$  или  $= 0$ , решение есть, вычислить значения корней и вывести на печать.

В первой ветви условного оператора нужно выполнить не одно, а три действия, поэтому используется блок  $\{...\}$ , который показывает компилятору, что эти действия следует воспринимать как единое целое.

// Код программы примера №1.

```
#include <stdio.h>
```

```
#include <math.h>
```

```

void main (void)
{
float      a, b, c, d;
float      x1, x2;
// Ввод входных данных
printf ("\nВведите коэффициенты квадратного уравнения.\n");
scanf ("%f%f%f", &a, &b, &c);
    d = b*b - 4*a*c;                //Вычисление дискриминанта d.
    if (d >= 0.0)
    {
        x1 = ( - b + sqrt(d)) / (2.*a);    //Вычисление корней.
        x2 = ( - b - sqrt(d)) / (2.*a);
        printf ("Корни равны x1=%6.2f x2=%6.2f\n", x1, x2);
    }
    else
        printf ("Корней нет\n");          //Печать сообщения.
} // End of main

```

В этом примере не предусмотрен случай равных корней, например, при  $a=2$ ,  $b=4$ ,  $c=2$ . Для того чтобы отследить этот случай, потребуется внести изменение в текст программы.

**Пример №2.** Когда для принятия решения требуется проверка более одного условия, появляется множественное ветвление. В данном примере следует предусмотреть случай равных корней, при этом в условном операторе появляется второе условие:  $d = 0$ , а значит, и второй оператор проверки условия.

Схема алгоритма условно может быть изображена так:

```

if (d > 0)
{
    // Блок операторов, чтобы вычислить два корня.
}
else
    if (d == 0)
    {
        //Блок операторов, чтобы вычислить один корень.
    }
    else
    {
        //Вывод сообщения о том, что решения нет.
    }
}

```

Структура ветвления рассматривает только взаимоисключающие варианты. Выполнение оператора `if` при использовании многих условий, если они не выделены блоком, разворачивается по принципу матрешки, то есть каждый `else` принадлежит ближайшему сверху `if`. Отступы в тексте программы позволяют лучше увидеть логику выбора решения.

// Код программы примера №2.

```

#include <stdio.h>
#include <math.h>
void main (void)
{
float      a, b, c, d;
float      x1, x2;
printf ("\nВведите коэффициенты квадратного уравнения\n");
scanf ("%f%f%f", &a, &b, &c);
d = b*b - 4*a*c;
if (d > 0.0)
{
    x1 = ( - b + sqrt (d)) / (2.*a);
    x2 = ( - b - sqrt (d)) / (2.*a);
    printf ("Корни равны x1=%6.2f x2=%6.2f\n", x1, x2);
}
else
// Дискриминант d вещественное значение, сравнивать его с нулем можно только
// с некоторой степенью точности, например, так: |d| < 0.00001.
    if ( fabs (d) <= 0.00001)           // Приблизительно равен нулю.
    {
        x1 = ( - b + sqrt (d)) / (2.*a);
        printf ("Корни одинаковы x1=x2=%6.2f\n", x1);
    }
    else
        printf ("Корней нет\n");
} // End of main

```

**Пример №3.** Диалог с программой в общепринятом виде. Для полного тестирования программы, алгоритм которой проверяет множество условий, нужно выполнить столько примеров, сколько вариантов ветвления возможно. В программе примера №3 нужно выполнить три тестовых примера. Кроме того, пользователь с помощью программы может решать несколько уравнений. Поэтому необходимо уметь организовать многократное обращение к программе.

Для организации повторного выполнения программы или ее фрагмента используется циклический алгоритм. Этот цикл будет управляться внешним событием. Событие порождает пользователь, нажимая клавишу на клавиатуре. Общепринято использовать клавишу Esc для завершения процесса, а для продолжения клавишу Enter или любую другую.

Кроме того, при вводе данных необходимо предусмотреть, чтобы введенное значение коэффициента  $a$  было бы отлично от 0. Проверка выполняется в цикле сразу же после ввода данных. Цикл будет повторен всегда, когда введенное значение не соответствует правилам, и только при правильных данных программа продолжит работу.

// Код программы примера №3. Показывает пример диалога в общепринятом виде.  
 // Для управления используется оператор do...while с выходом по условию

```

// «нажата клавиша Esc».
#include <stdio.h>
#include <math.h>
#include <conio.h>          // Библиотека консольного ввода-вывода.
#define ESC 27             // Код клавиши Esc в символьном представлении.
void main (void)
{
float    a, b, c, d;
float    x1, x2;
char     key;              // Переменная key для обработки события.
// Первое выполнение алгоритма обязательно, поэтому необходим цикл do...while
do
{
    // Проверка корректности вводимых данных.
    // Если введенное значение a близко к нулю, цикл ввода повторяется.
    do
    {
        printf ("\nВведите коэффициенты квадратного уравнения \n");
        scanf ("%f%f%f", &a, &b, &c);
    } while (fabs (a) < 0.001)
    // Ввод выполнен правильно.
    d = b*b - 4*a*c;
    if (fabs (d) <= 0.00001)
    {
        x1 = (- b + sqrt (d)) / (2.*a);
        printf ("Корни одинаковы x1 = x2 = %6.2f\n", x1);
    }
    else
        if ( d > 0)
        {
            x1 = (- b + sqrt (d)) / (2.*a);
            x2 = (- b - sqrt (d)) / (2.*a);
            printf ("Корни равны x1=%6.2f x2=%6.2f\n", x1, x2);
        }
        else
            printf ("Корней нет\n");
    // Обработка события «нажатие клавиши».
    printf ("Клавиша ESC – завершение работы, Any Key – продолжение...");
    // Ожидание события «нажатие клавиши».
    key = getch ();        // Функция getch () читает символьный код клавиши.
} while ( key != ESC )    // Код клавиши Esc прописан в директиве define.
} // End of main

```

**Пример №4.** Использование вложенных условных операторов в программах с

проверкой многих условий. Примером сложного выбора является оценка знаний ученика по результатам тестирования. Пусть известен результат ЕГЭ (от 0 до 100 баллов). Оценка по пятибалльной шкале должна быть выставлена по правилам:

$$\text{Оценка} = \begin{cases} 2, & \text{если } \text{ЕГЭ} \leq 40 \\ 3, & \text{если } 40 < \text{ЕГЭ} \leq 60 \\ 4, & \text{если } 60 < \text{ЕГЭ} \leq 80 \\ 5, & \text{если } 80 < \text{ЕГЭ} \leq 100. \end{cases}$$

В записи этих правил есть избыточность с точки зрения механизмов выполнения условного оператора. Ветвление, это взаимоисключающие варианты, значит, в первой ветви истинность условия  $\text{ЕГЭ} \leq 40$  означает получение результата. Если же это условие ложно, то к проверке второго условия  $40 < \text{ЕГЭ} \leq 60$  приходят только те значения ЕГЭ, которые не удовлетворяют первому условию, то есть только  $\text{ЕГЭ} > 40$ , а значит, их не нужно включать в запись условия. Так же и в записи всех последующих условий. Логика вложенных операторов условия присоединяет каждый else к ближайшему сверху if.

// Код программы примера №4.

```
#include <stdio.h>
#include <conio.h>
void main (void)
{
    int      Test;          // ЕГЭ.
    int      Ball;          // Оценка.
    do
    {
        printf ("Введите результат ЕГЭ испытуемого\n");
        scanf ("%d", &Test);
        if (Test <= 40)
            Ball = 2;
        else
            if (Test <= 60)
                Ball = 3;
            else
                if (Test <= 80)
                    Ball = 4;
                else
                    Ball = 5;
        printf("Оценка в аттестате: %d\n", Ball);
        printf("Если больше нет испытуемых, нажмите ESC\n");
    } while (getch() != 27);
} // End of main
```

Обратите внимание, что цикл по-прежнему управляется событием, но в этом примере вспомогательная переменная key не используется.

**Пример №5.** Использование сложных логических выражений в программах с проверкой многих условий.

Использование множественных вложений операторов условия затрудняет «читабельность» программы. Зачастую его можно избежать, объединяя простые условия в логические выражения везде, где для поиска решения требуется проверка более одного условия. Пусть требуется проверить, принадлежит ли точка с произвольными координатами  $(x, y)$  некоторой области, например, заштрихованной области, изображенной на рис. 1. Известны длина стороны квадрата (обозначим  $L$ ) и радиус окружности (обозначим  $R$ ), где не обязательно  $L > R$ , то есть область может быть пустой.

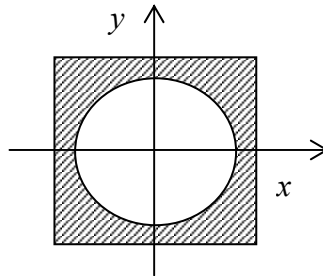


Рис. 1. Иллюстрация к примеру №5.

Запишем условие математически с помощью системы неравенств:

$$\left\{ \begin{array}{ll} x^2 + y^2 \geq R^2 & \text{условие "точка находится вне окружности"} \\ |x| < \frac{L}{2} & \text{условие "точка находится внутри квадрата"} \\ |y| < \frac{L}{2} & \end{array} \right.$$

Для соблюдения условия нужно, чтобы все три неравенства выполнялись одновременно, поэтому при объединении их в одно требуется использовать логическую операцию «И» (конъюнкция), следующим образом:

$$x*x + y*y \geq R*R \ \&\& \ \text{fabs}(x) < 0.5*L \ \&\& \ \text{fabs}(y) < 0.5*L$$

Тогда условный оператор запишется:

```
if (x*x + y*y >= R*R && fabs(x) < 0.5*L && fabs(y) < 0.5*L)
    printf("Точка принадлежит области \n");
```

```
else
```

```
    printf("Точка не принадлежит области \n");
```

Если не использовать сложное выражение, то приходится использовать вложенные операторы условия, следующим образом:

```
if (x*x + y*y >= R*R)
    if (fabs(x) < 0.5*L)
        if (fabs(y) < 0.5*L)
            printf("Точка принадлежит области \n");
        else
            printf("Точка не принадлежит области \n");
```

Квалифицированный программист отдаст предпочтение первому способу.



**Пример №6.** Использование оператора переключателя switch для организации множественного ветвления. Оператор switch использует для управления значение выражения целочисленного (символьного) типа, и позволяет выбрать один из нескольких возможных вариантов ветвления алгоритма. Покажем, как можно использовать этот оператор для управления алгоритмом с консоли, например, при организации меню пользователя. Значение кода нажатой пользователем клавиши управляет выбором ветви алгоритма. Код клавиши возвращает функция getch(). Если нажата одна из функциональных (управляющих) клавиш, то функция возвращает 0 (0xE0). Ее повторный вызов получает расширенный код клавиши.

// Код программы примера №6.

```
void main (void)
{int      key;
do
{
    printf ("Выберите действие\n");
    key = getch();
    if ( key == 0 )           // Нажата управляющая клавиша.
    {
        key = getch(); // Ввод не повторяется, символ получен из буфера ввода.
        switch key      // Значение key управляет ветвлением.
        {
            case 77 : {printf ("Стрелка вправо\n"; break;)}
            case 75 : {printf ("Стрелка влево\n";  break;)}
            case 72 : {printf ("Стрелка вверх\n";  break;)}
            case 80 : {printf ("Стрелка вниз\n";   break;)}
            case 27 : {printf ("Esc\n";            break;)}
            default:  {printf ("Не стрелка\n"; }
        }
    }
} while (key != 27);      // Выход из цикла по нажатию Esc.
} // End of main
```

Оператор break в каждой ветви передает управление на оператор, следующий за switch. Если break опущен, то после ветви, на которую пал выбор, выполняются все операторы, стоящие далее в тексте switch.

### Варианты заданий

Задание 1. Прямая  $L$ , заданная на плоскости координатами двух точек  $(a,b)$  и  $(c,d)$ , разбивает координатную плоскость на две полуплоскости. Известны также координаты двух точек  $(x_1, y_1)$  и  $(x_2, y_2)$ , не лежащих на данной прямой.

Определить, находятся ли точки в разных полуплоскостях или в одной, и напечатать сообщение. Иметь возможность повторного обращения в диалоге.

Задание 2. Составить программу для определения корней системы двух линейных алгебраических уравнений по правилу Крамера. Решение вывести в виде системы уравнений. Иметь возможность повторного обращения в диалоге.

Задание 3. Дано натуральное число  $N$ , в записи которого ровно пять знаков.

Определить, является ли это число палиндромом или нет, напечатать сообщение текстом. Значение числа вводить в диалоге, иметь возможность повторного обращения.

Задание 4. Дано натуральное число  $N$ , определяющее возраст человека в годах.

Дать для этого числа наименование «год», «года» или «лет». Например, «Вам 21 год» или «Вам 43 года». Иметь возможность повторного обращения в диалоге.

Задание 5. Окружность на плоскости с центром в начале координат имеет радиус  $R$ . Известны также координаты концов некоторого отрезка  $(x_1, y_1)$  и  $(x_2, y_2)$ .

Определить, пересекает ли отрезок окружность и сколько раз. Значения координат вводить в диалоге, иметь возможность повторного обращения.

Задание 6. На плоскости заданы две окружности координатами центров и радиусами.

Определить, пересекаются ли они или касаются друг друга. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 7. Даны длины трех отрезков.

Определить, можно ли построить треугольник с такими длинами сторон. Если да, то определить, какой это треугольник: прямоугольный, остроугольный или тупоугольный. Значения длин вводить в диалоге, иметь возможность повторного обращения.

Задание 8. Заданы декартовы координаты точки на плоскости.

Перевести в полярные координаты с учетом номера четверти, где находится точка. Угол перевести в градусную меру с точностью до минут. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 9. По введенным координатам точки  $(x, y)$  определить номер четверти координатной плоскости, где находится точка.

Значения координат вводить в диалоге, иметь возможность повторного обращения.

Задание 10. На плоскости задан треугольник длинами своих сторон.

Найти наименьший из углов треугольника с точностью до одного градуса. Значения сторон вводить в диалоге, иметь возможность повторного обращения.

Задание 11. На плоскости заданы три точки своими координатами.

Определить расстояния от точек до начала координат и напечатать, какая из точек расположена ближе к началу координат. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 12. Дано натуральное число  $N$ , в записи которого ровно пять знаков.

Определить, имеет ли это число одинаковые цифры или нет. Значение числа вводить в диалоге, иметь возможность повторного обращения.

Задание 13. Окружность на плоскости с центром в начале координат имеет радиус 1. Известны координаты левого верхнего угла квадрата со стороной 1.

Определить, принадлежит ли окружности хотя бы одна вершина квадрата. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 14. Окружность с центром в начале координат имеет радиус 1. Определить, пересекает ли прямая  $y = kx + b$  окружность или хотя бы касается один раз. Значения  $k$  и  $b$  вводить в диалоге, иметь возможность повторного

обращения.

Задание 15. На плоскости заданы два квадрата координатами левого верхнего угла и длинами сторон.

Определить, пересекаются ли они. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 16. На плоскости задано кольцо с центром в начале координат и радиусами  $r_1$  и  $r_2$ , где  $r_1 < r_2$ . Дана точка своими координатами  $(x, y)$ .

Определить, находится ли точка внутри кольца. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 17. Прямоугольной формы кирпич имеет стороны  $A, B, C$ . Определить, пройдет ли кирпич в прямоугольное отверстие размером 5 на 8.

Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 18. Даны четыре числа.

Определить, являются ли они элементами арифметической прогрессии. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 19. На плоскости задан треугольник длинами своих сторон.

Вычислить его медианы

$$ma = 0.5 * \sqrt{2 * b^2 + 2 * c^2 - a^2}$$

$$mb = 0.5 * \sqrt{2 * a^2 + 2 * c^2 - b^2}$$

$$mc = 0.5 * \sqrt{2 * b^2 + 2 * a^2 - c^2},$$

и найти наибольшую медиану. Значения длин сторон вводить в диалоге, иметь возможность повторного обращения.

Задание 20. Прямоугольной формы контейнер имеет размеры  $8*8*12$ , где 12 – высота контейнера. Дано  $K$  прямоугольных блоков размером  $n*n*2n$ .

Определить, войдут ли эти блоки в контейнер. Если не войдут, то определить, сколько останется. Размер блоков вводить в диалоге, иметь возможность повторного обращения.

Задание 21. На плоскости заданы три точки своими координатами.

Определить длины сторон охватывающего их прямоугольника наименьшего размера. Пусть его стороны параллельны координатным осям.

Значения координат вводить в диалоге, иметь возможность повторного обращения.

Задание 22. Даны два интервала числовой оси  $[a_1, b_1]$  и  $[a_2, b_2]$ .

Найти соотношение интервалов: пересекаются или нет, первый принадлежит второму или наоборот. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 23. На плоскости задан квадрат координатами левого верхнего угла и длиной стороны. Задан также отрезок координатами концов.

Определить, находится ли отрезок полностью внутри квадрата или один его конец внутри, или отрезок полностью снаружи. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 24. На плоскости задан треугольник координатами своих вершин.

Найти наибольшую из сторон треугольника. Значения координат вводить в диалоге, иметь возможность повторного обращения.

Задание 25. Дано цилиндрическое ведро радиусом  $r$  и высотой  $h$ . Требуется с его помощью переместить  $K$  литров жидкости из одной емкости в другую.

Определить, можно ли это сделать за одно действие. Если нет, то определить, сколько раз нужно воспользоваться ведром. Размеры вводить в диалоге, иметь возможность повторного обращения.

Задание 26. На плоскости заданы три точки своими координатами.

Определить, сколько из них и какие находятся внутри окружности, для которой известны радиус и координаты центра. Все значения вводить в диалоге, иметь возможность повторного обращения.

Задание 27. Дан возраст двоих людей в формате «Число, месяц, год». Найти который из них старше, сравнивая по отдельности год, затем месяц, затем день.

Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 28. Даны прямоугольная коробка размером  $w*w*h_1$  и цилиндрическое ведро радиуса основания  $r$  и высотой  $h_2$ .

Найти, какая емкость больше по объему. Значения вводить в диалоге, иметь возможность повторного обращения.

Задание 29. На плоскости заданы две прямые каноническими уравнениями.

Найти, пересекаются эти прямые, или нет. Значения коэффициентов вводить в диалоге, иметь возможность повторного обращения.

Задание 30. Дано числовое значение денежной суммы не более 100 руб.

Вывести значение числа прописью, например, «2 рубля», «12 рублей», «51 рубль». Анализировать остаток от деления на 10. Так, для остатка, равного 1 наименование «рубль», для остатка, равного 2, 3, 4, наименование «рубля», для остатка, равного 5, 6, 7, 8, 9, 0 наименование «рублей». Числа второго десятка – исключение. Значение денежной суммы вводить в диалоге, иметь возможность повторного обращения.

Задание 31. У каждого из трех братьев есть некоторая сумма денег. Они решили поделиться поровну.

Определить, кто из них и кому должен передать какую сумму. Значение сумм вводить в диалоге, иметь возможность повторного обращения.

Задание 32. В продаже есть два вида системных блоков, по цене  $K_1$  и  $K_2$  руб., и три вида мониторов по цене  $M_1$ ,  $M_2$  и  $M_3$  руб. Известны также стоимость клавиатуры и мыши.

Найти, сколько будет стоить самый дешевый компьютер, и сколько самый дорогой. Значения стоимостей вводить в диалоге, иметь возможность повторного обращения.

Задание 33. При кормлении одного лабораторного животного нужно в день  $K_1$  калорий для мыши,  $K_2$  калорий для крысы. Есть корма калорийностью  $P_1$  по цене  $X_1$  руб.,  $P_2$  по цене  $X_2$  руб. и  $P_3$  по цене  $X_3$  руб. за кг.

Подобрать самый дешевый дневной рацион для  $X$  мышей и  $Y$  крыс. Значения данных вводить в диалоге, иметь возможность повторного обращения.

Задание 34. Для одного подарка выделено  $K$  руб. В подарок можно положить

яблоки по цене  $K_1$  руб., апельсины по цене  $K_2$  руб. и печенье по цене  $K_3$  руб. за штуку. В подарок каждый из предметов должен войти хотя бы один раз. Если получается, то по два одинаковых или по три.

Найти состав подарка. Найти сумму, которая останется от покупки. Значения данных вводить в диалоге, иметь возможность повторного обращения.

### Тема 3. Инструменты C++ для реализации циклических алгоритмов

Если какой-либо фрагмент алгоритма должен быть выполнен многократно, то это циклический алгоритм (цикл).

Циклические алгоритмы можно условно разделить на две группы.

1. Арифметический цикл, у которого заранее известно число повторений.
2. Итерационный цикл, у которого заранее неизвестно число повторений.

Управление циклом выполняет некоторая переменная величина, которая называется «параметр цикла» или «управляющая переменная». Это переменная программы, которая, как правило, изменяется в теле цикла, определяет число повторений цикла и позволяет вовремя завершить его работу.

Можно выделить четыре составные части цикла.

1. Подготовка цикла: присваивание начальных значений переменным, в том числе параметру цикла.
2. Тело цикла: фрагмент, который должен быть повторен многократно.
3. Изменение параметра цикла: как правило, выполняется в теле цикла.
4. Проверка условия завершения цикла: в проверке условия, явно или нет, присутствует параметр цикла.

Не всегда эти составляющие присутствуют явным образом.

В C++ существуют три вида операторов цикла, которые одинаково можно использовать для организации любого циклического алгоритма.

#### 1. while...do:

while (Условие)

```
{    //Количество повторений любое.  
    Тело цикла  
}
```

#### 2. do...while:

do

```
{    //Количество повторений любое.  
    Тело цикла  
}
```

while (Условие)

#### 3. for:

for (объявление параметра цикла)

```
{    //Количество повторений фиксировано.  
    Тело цикла  
}
```

**Пример №1.** Алгоритм построения таблиц значений различных функций. Это, чаще всего, арифметический цикл. Обычно параметром цикла является аргумент

функции. Для функции задана формула вычисления значения  $y(x) = F(x)$ . Известны диапазон изменения аргумента  $x \in [x_0, x_n]$ , и шаг изменения  $\Delta x$ .

Общая схема этого алгоритма на основе цикла while ... do выглядит так:

// Печать заголовка таблицы.

$x = x_0$ ; // Подготовка цикла.

while ( $x \leq x_n$ ) // Проверка условия завершения.

{

$y = F(x)$ ; // Сколь угодно сложный алгоритм вычисления значения.

// Вывод строки таблицы.

$x += \Delta x$ ; // Приращение управляющей переменной.

}; // Выход из цикла.

Общая схема этого алгоритма на основе цикла do ... while выглядит так:

// Печать заголовка таблицы.

$x = x_0$ ; // Подготовка цикла.

do

{

$y = F(x)$ ; // Сколь угодно сложный алгоритм вычисления значения.

// Вывод строки таблицы.

$x += \Delta x$ ; // Приращение управляющей переменной.

}

while ( $x \leq x_n$ ); // Проверка условия завершения.

Общая схема этого алгоритма на основе цикла for выглядит так:

// Печать заголовка

for ( $x = x_0$ ;  $x \leq x_n$ ;  $x += \Delta x$ ) // Все составляющие цикла в заголовке.

{

$y = F(x)$ ; // Сколь угодно сложный алгоритм вычисления значения.

// Вывод строки таблицы.

};

**Пример №2.** Функция  $F(x)$  может быть достаточно сложной, тогда в теле цикла нужно позаботиться о правильной записи блока, вычисляющего функцию. Пусть для  $x \in [-\pi/2; +\pi/2]$  требуется вычислить таблицу значений функции, имеющей разрыв в точках  $|x| = \pi/4$ , по формуле

$$y(x) = \begin{cases} \sin x & \text{для } |x| < \frac{\pi}{4} \\ \cos x & \text{для } |x| \geq \frac{\pi}{4} \end{cases}$$

// Код программы примера №2.

#include <stdio.h>

#include <math.h>

// В библиотеке math.h определена константа M\_PI – значение числа  $\pi$ .

void main (void)

{

float  $x, y$ ; // Аргумент и значение функции.

```
// Вывод шапки таблицы в текстовом режиме экрана.
printf ("\n Таблица значений функции\n");
printf ("-----\n");
printf ("      x      y      \n");
printf ("-----\n");
// x – параметр цикла, в заголовке цикла for задано полное управление.
for ( x = - M_PI/2.; x <= M_PI/2.; x += 0.2 )    // Цикл вычисления таблицы.
{ // Блок вычисления значения функции использует оператор if.
  if (fabs (x) <= M_PI/4.)    // Логическая запись формулы вычисления.
    y = sin(x);
  else
    y = cos(x);
  printf ("%11.2f %11.2f\n" ,x, y); // Вывод строки таблицы.
}
} // End of main
```

**Пример №3.** Итерационный цикл. Если в формулировке задачи явно или неявно присутствуют условия «пока не будет выполнено» или «до тех пор, пока не выполнено», то число повторений заранее неизвестно, и это итерационный процесс. Содержание тела цикла может быть произвольным. Для организации таких алгоритмов используются, как правило, циклы while ... или do ... while.

Пример использования цикла, управляемого событием, уже был ранее показан в примерах №3 и №4 предыдущего раздела. Еще раз покажем, как использовать цикл do ... while для проверки корректности ввода. Если программа дружелюбна пользователю, то при вводе данных она сначала подскажет, что и как следует вводить, затем выполнит проверку введенных значений, и в случае ошибки будет возвращать пользователя к этапу ввода данных.

Требуется найти площадь треугольника, построенного по значениям длин трех заданных отрезков. Известно, что треугольник существует только тогда, когда длина каждой его стороны меньше, чем сумма длин двух других сторон. Если введенные данные не удовлетворяют этому условию, ввод повторяется.

// Код программы примера №3.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main (void)
{
  float      a, b, c;          // Длины сторон треугольника.
  float      S;               // Площадь треугольника.
  // Цикл проверки корректности введенных данных.
  do          // Не перейдет к вычислениям, пока не получит корректных данных.
  {
    printf ("Введите длины сторон треугольника\n");
    scanf ("%f%f%f", &a, &b, &c); // Тут может появиться ошибка данных.
  }
}
```

```

while (! (a < b + c && b < a + c && c < a + b));
// Данные введены корректно.
float      pp;
    pp = (a + b + c) / 2.;
    S = sqrt (pp*(pp - a)*(pp - b)*(pp - c));
    printf ("Площадь треугольника равна: %6.2f\n",S);
} // End of main

```

**Пример №4.** Программист, организовав управление итерационным циклом, должен позаботиться и об инициализации управляющей переменной, и об ее приращении, и об условии завершения цикла.

Сформулируем условие задачи. Мяч брошен вертикально вверх со скоростью  $V$ . Требуется построить таблицу зависимости высоты  $Y$  от времени  $t$ , начиная с момента броска до момента падения мяча на землю, если

$$Y(t) = V \cdot t - \frac{g \cdot t^2}{2},$$

где  $g$  – константа тяготения, равная 9,8.

При движении мяч сначала взлетает вверх, затем падает. Высота подъема и время полета мяча зависят от начальной скорости броска. Очевидно, что циклом вычисления должна управлять переменная  $t$ . Для нее известно начальное значение ( $t = 0$ ). Шаг изменения можно оценить из физического смысла задачи и выбрать произвольно, например,  $t = 0.1$  сек. Момент завершения вычислений неизвестен в числовом выражении, но известно, что значение высоты сначала будет возрастать, затем убывать, и при падении мяча на землю станет равно нулю:  $Y \leq 0$ . Величина  $Y$  – прямая функция  $t$ , значит, в этом условии переменная  $t$  присутствует, но неявно. Поскольку в постановке задачи условие завершения звучит «до момента падения», кажется естественным выбрать цикл `do`.

// Код программы примера №4.

```

#include <stdio.h>
#include <conio.h>
#define      G      9.8
void main (void)
{
    float      V, y, t;           // Имена переменных имеют физический смысл.
    printf ("Введите значение скорости броска\n");
    scanf ("%f", &V);
    printf ("Таблица зависимости высоты от времени Y(t)\n");
    printf ("-----\n");
    printf ("    t          y(t) \n");
    printf ("-----\n");
    // Подготовка цикла. Момент времени t = 0.1
    t = 0.1;
    do                               // Выбран цикл do.
    {

```



```

    y = V * t - 0.5 * G * t * t;
    printf ("\t %6.2f \t %6.2f \n", t, y);
    t += 0.1;                // Переход на новую итерацию.
} while (y >= 0);           // В условии t присутствует неявно.
getch ();                  // Для того, чтобы держать таблицу на экране.
} // End of main

```

Замечание. Этот вариант имеет недостаток. Последнее значение таблицы будет отрицательным, потому что точного совпадения координаты  $y$  с нулем не будет никогда, а цикл `do` проверяет условие завершения *после* печати строки таблицы. Чтобы избежать этой некорректности, следует выбрать цикл `while`, в котором отчетливо прописать условие завершения цикла: `while (V * t - 0.5 * G * t * t >= 0)`.

В рассмотренных примерах циклические алгоритмы являются простыми. Сложным (вложенным) циклом называется такой, у которого содержанием циклически выполняемого фрагмента является также циклический алгоритм. Типы циклических алгоритмов внешнего и внутреннего циклов при этом могут быть любыми. Внутренний цикл, от подготовки до завершения, должен находиться внутри внешнего, точнее, быть его телом.

**Пример №5.** Сложный арифметический цикл. Функция двух переменных  $F(x, y)$  или функция с параметром  $F(A, x)$  порождает необходимость использования сложного арифметического цикла тогда, когда оба аргумента изменяются. Здесь решается задача полного перебора, то есть нахождения всех возможных значений функции при всех возможных значениях ее аргументов.

При решении таких задач необходимо выделить обе управляющие переменные, описать закон их изменения, и выбрать, какая из них по логике задачи главная или условно главная. Главная переменная будет параметром внешнего цикла, другая переменная будет параметром внутреннего цикла. Для каждого зафиксированного значения главной переменной другая переменная должна пробегать все значения своего диапазона. В сложном арифметическом цикле можно строить таблицу значений функции вида  $F(x, y)$  для всех возможных значений  $x$  и  $y$ . Чтобы значение параметра внешнего цикла не повторялось во многих строках таблицы, его следует выводить однократно во внешнем цикле как заголовок таблицы.

Пусть требуется вычислить объемы нескольких цилиндрических емкостей

$$V = H \cdot \pi \cdot \left(\frac{d}{2}\right)^2,$$

где  $H$  – высота, а  $d$  – диаметр основания цилиндра.

Высота может изменяться в диапазоне  $H \in [-1; 5]$ ,  $\Delta H = 1$ , а диаметр в диапазоне  $d \in [0.5; 3.5]$ ,  $\Delta d = 0.5$ . Переменные равноправны, поэтому внешним циклом может быть любой. Пусть это высота, тогда для одного фиксированного значения высоты нужно выводить таблицу значений функции  $F(d)$ .

// Код программы примера №5.

```

#include <stdio.h>
#include <conio.h>

```

```

#include <math.h>
void main(void)
{
    float    d, h;           // Диаметр основания и высота емкости.
    float    V;              // Объем емкости.
    // Управление внешним циклом, параметр h.
    for (h = 1.; h <= 5.; h += 1.)
    {
        clrscr ();          // Очистка экрана перед выводом очередной таблицы.
        printf ("Таблица зависимости объема от диаметра основания\n");
        printf ("-----\n");
        printf (" Высота = %6.2f\n", h);
        printf ("      d      V      \n");
        printf ("-----\n");
        // Управление внутренним циклом, параметр d.
        for (d = 0.5; d <= 3.5; d += 0.5)
        {
            V = M_PI * pow (0.5*d, 2.);
            printf ("%6.2f\t %6.2f\n", d, V);
        }
        getch();            // Удержание экрана после вывода очередной таблицы.
    }
} // End of main

```

Очистка экрана перед выводом очередной таблицы функцией `clrscr ()` не необходима. В этом варианте на очередном экране мы будем видеть одну таблицу для одного значения высоты. Если убрать эту строку, то таблицы будут выводиться подряд одна за другой. Удержание экрана после вывода очередной таблицы обязательно, так как программа выводит данных больше, чем вмещается на один экран.

**Пример №6.** Сложный (вложенный) цикл, управляемый событием. Изменим текст примера №3 добавлением к нему цикла многократного повторения, как в примере №3 предыдущего раздела. Новый внешний цикл просто охватывает весь текст старой программы, и позволяет в диалоге управлять ее многократным повторением.

// Код программы примера №6.

```

#include <stdio.h>
#include <conio.h>
#include <math.h>
void main (void)
{
    float    a, b, c;         // Длины сторон треугольника.
    float    S, pp;           // S – площадь, pp – полупериметр.
    // Цикл, управляющий процедурой многократных вычислений.
    do

```

```

{
// Цикл проверки корректности введенных данных.
do
{
    printf ("Введите длины сторон треугольника\n");
    scanf ("%f%f%f", &a, &b, &c);    // Тут может появиться ошибка.
}
while (!(a < b + c && b < a + c && c < a + b));
pp = (a + b + c) / 2.;
S = sqrt (pp*(pp - a)*(pp - b)*(pp - c));
printf("Площадь треугольника равна: %6.2f\n",S);
// Управление внешним циклом.
printf("Если больше нет треугольников, нажмите ESC\n");
} while (getch () != 27);
} // End of main
Как видим,

```

**Пример №7.** В сложном цикле типы алгоритмов могут быть произвольными. Оба цикла могут быть одного типа, или внешний цикл может быть итерационным, а внутренний арифметическим, или наоборот. Изменим пример №4, использующий итерационный цикл, добавлением необходимости исследования решения задачи для различных значений стартовой скорости  $V$ . Пусть  $V$  может изменяться в диапазоне  $V \in [-1;7]$  с шагом 1. Как видно из постановки задачи, ее решение должно быть повторено многократно по правилам арифметического цикла. Переменная  $V$  из простой превращается в управляющую. Текст примера нужно изменить добавлением к нему управления по переменной  $V$  во внешнем цикле, при этом запись внутреннего цикла не изменяется ни в одном символе.

// Код программы примера №7.

```

#include <stdio.h>
#include <conio.h>
#define      G      9.8
void main (void)
{
float      V, y, t;           // Имена переменных имеют физический смысл.
// Управление внешним циклом, параметр V.
for (V = 1; V <=7 ; V += 1.)
{
    clrscr ();
    printf ("Таблица зависимости высоты от времени при V=%6.2f\n", V);
    printf ("-----\n");
    printf ("    t          y(t) \n");
    printf ("-----\n");
    // Управление внутренним циклом, параметр t.
    // Подготовка цикла. Момент времени t = 0.1

```

```

t = 0.1;
while (V * t - 0.5 * G * t * t >= 0)
{
    y = V * t - 0.5 * G * t * t;
    printf ("t %6.2f\t %6.2f\n", t, y);
    t += 0.1;          // Переход на новую итерацию.
}
getch ();             // Для того, чтобы держать таблицу на экране.
}
} // End of main

```

### Варианты заданий

Задание 1. Составить программу для вычисления таблицы значений функции

$$y(x) = \begin{cases} a \cdot x^2 & \text{для } -2 < x < 3 \\ -x + \frac{1}{a} & \text{для остальных } x \end{cases}$$

Значение  $a$  ввести. На интервале  $x \in [-4; 6]$  иметь в цикле не менее 15-ти точек. Иметь возможность повторного обращения в диалоге.

Задание 2. Составить программу для вычисления таблицы значений функции, вычисляющей силу тяготения между двумя материальными точками с массами  $m_1$  и  $m_2$  по формуле

$$F = g \cdot \frac{m_1 \cdot m_2}{r^2}$$

где  $g = 6,67 \cdot 10^{-8}$ ,  $m_2 = 4 \cdot 10^8$ .

$r \in [0,1; 0,9]$ , шаг 0,2;  $m_1 \in [3 \cdot 10^8; 4 \cdot 10^8]$ , шаг  $0,5 \cdot 10^8$ .

Результаты свести в таблицу. Для каждого  $m_1$  строить отдельную табличку  $F(r)$ , где  $m_1$  напечатано в заголовке.

Задание 3. Составить программу, вычисляющую таблицу значений функции

$$y(x) = \begin{cases} \sin x + a & \text{при } x < 0 \\ \cos \pi \cdot x & \text{при } 0 \leq x \leq 2 \\ a^2 + x^2 & \text{при } x > 2, \end{cases}$$

если  $x \in [-4; 4]$ , шаг 0,3. Параметр  $a$  принимает значения 1.2, 1.3, 1.4, 1.5, 1.6. Для каждого значения параметра строить отдельную табличку  $y(x)$ .

Задание 4. Составить программу, вычисляющую таблицу значений функции

$$y(x) = \begin{cases} \frac{2 \cdot x^3}{x^2 + 1} & \text{при } |x| < 3 \\ 1,5 \cdot \left| \operatorname{tg} \frac{\pi}{x} \right| & \text{для остальных } x, \end{cases}$$

если  $x \in [-6; +6]$  с шагом 0.5.

Задание 5. Составить программу, вычисляющую таблицу значений функции

$$F(x, y) = \begin{cases} x \cdot e^{x \cdot y}, & \text{если точка } (x, y) \text{ лежит в круге радиуса } r \\ y + e^{x+y}, & \text{если точка } (x, y) \text{ лежит в кольце внутреннего радиуса } r \\ & \text{и внешнего } R \\ x \cdot y \cdot e^{x-y}, & \text{если точка } (x, y) \text{ лежит вне круга радиуса } R. \end{cases}$$

Значения радиусов  $r$  и  $R$  вводить.

Для  $y \in [0; 2.4]$  с шагом 0.8 и  $x \in [0; 2.4]$  с шагом 0.6 построить таблицу на решетке из полного перебора значений  $x, y$ , где  $y$  выводить в заголовке.

Задание 6. Составить программу для вычисления таблицы значений функции

$$T(x, \varphi) = (9,036 + x^2)^{-\sin \varphi},$$

если  $x \in [-1; 5]$ , шаг 1. Угол  $\varphi$  принимает последовательно значения  $\pi/2, \pi/3, \pi/6, \pi/9$ .

Результаты свести в таблицу; для каждого  $\varphi$  строить отдельную табличку  $T(x)$ , где  $\varphi$  напечатано в заголовке.

Задание 7. Напечатать в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр. Иметь возможность повторного обращения в диалоге.

Задание 8. Составить программу, которая находит наибольший общий делитель двух натуральных чисел  $a$  и  $b$  по следующему алгоритму: до тех пор, пока  $a$  и  $b$  не сравняются, вычитать  $a = a - b$  при  $a > b$ , или  $b = b - a$  при  $b > a$ . Исходные числа вводить, иметь возможность повторного обращения в диалоге.

Задание 9. Натуральное число является простым, если оно делится на 1 и на самого себя. Натуральное число является совершенным, если оно равно сумме своих делителей, включая 1, например:

$$6 = 1 + 2 + 3,$$

$$28 = 1 + 2 + 4 + 7 + 14.$$

Составить программу, которая определит, является ли некоторое число  $N$  простым, а если нет, то является ли оно совершенным. Иметь возможность повторного обращения в диалоге.

Задание 10. Для любого действительного числа  $x$  вычислить значение  $f(x)$ , где  $f$  – периодическая функция с периодом  $T=2$ , совпадающая на отрезке  $[0, 1]$  с функцией  $y(x) = x^2 - 2,25x$ , а на отрезке  $[1, 2]$  с функцией  $y(x) = x - 1.25$ .

Проверить в цикле на интервале  $x \in [-4; 4]$  для четного числа точек.

Задание 11. Натуральное число является простым, если оно делится только на 1 и на самого себя. Составить программу, которая найдет все простые числа от  $N_1$  до  $N_2$  включительно. Иметь возможность повторного обращения в диалоге.

Задание 12. Составить программу для проверки знания таблицы умножения для столбца  $N$ , где  $N = 2, 3, \dots, 10$  (выбирать в диалоге).

Подсчитывать правильные ответы, выставить оценки: «отлично», «хорошо», «удовлетворительно» или «неудовлетворительно» за 10, 9, 8, 7 и менее

соответственно. Иметь возможность повторного обращения в диалоге.

Задание 13. Составить программу для проверки навыков сложения и вычитания. Программа в диалоге генерирует необходимую процедуру "+" или "-", и два операнда, затем спрашивает сумму или разность. При правильном ответе печатать поощрительный текст и предлагать повторить ввод, иначе печатать сообщение об ошибке и предлагать повторить ввод результата до получения правильного ответа.

Задание 14. Для любого действительного  $x$  вычислить значение  $f(x)$ , где  $f$  – периодическая функция с периодом  $T = 2.5$ , совпадающая на отрезке  $[0, 1]$  с функцией  $y(x) = x^2$ , а на отрезке  $[1, 1.5]$  с функцией  $y(x) = -x$ .

Проверить на интервале  $x \in [-3; 5]$ , взять не менее 10-ти точек.

Задание 15. Для любого натурального числа  $N$  найти все такие натуральные числа  $x$  и  $y$ , для которых выполняется  $N = x^2 + y^2$ . Иметь возможность повторного обращения в диалоге.

Задание 16. Найти все целочисленные координаты точек, попадающих в круг радиуса  $R$  с центром в точке  $(A, B)$ . Иметь возможность повторного обращения в диалоге.

Задание 17. Составить программу, вычисляющую температуру воздуха на разных высотах

$$T(h) = \begin{cases} 88,16 - 0,0065 \cdot h, & \text{если } 0 < h < 11000 \\ 216,16 + 0,0076 \cdot h, & \text{если } 11000 \leq h < 25000 \\ 216,16 + 0,0027 \cdot (h - 25000), & \text{если } h \geq 25000. \end{cases}$$

Построить таблицу значений для  $h \in [0; 45000]$ , шаг 1000.

Задание 18. Составить программу для вычисления таблицы значений функции

$U(y) = 16 + y + y^2$ , где

$$y(x) = \begin{cases} 5 \cdot x^2 - 0,5 & \text{при } x \leq 0, \\ 3 \cdot x^2 + 4 \cdot x & \text{при } 0 < x < 3, \\ x^2 - \frac{1}{x} & \text{при } x \geq 3, \end{cases}$$

если  $x \in [-2; 4]$ , шаг 0.25.

Задание 19. Найти все натуральные числа от  $N_1$  до  $N_2$ , запись которых есть палиндром. Иметь возможность повторного обращения в диалоге.

Задание 20. Найти все натуральные числа от  $N_1$  до  $N_2$ , запись которых совпадает с последними цифрами записи их квадрата, например:  $6^2 = 36$ ,  $25^2 = 625$  и т.д. Иметь возможность повторного обращения в диалоге.

Задание 21. Напечатать в возрастающем порядке все трехзначные числа, в десятичной записи которых есть одинаковые цифры. Иметь возможность повторного обращения в диалоге.

Задание 22. Составить программу, вычисляющую таблицу значений функции

$$Y(x) = \begin{cases} x + a & \text{при } x < -1 \\ a & \text{при } -1 \leq x \leq 1 \\ -x + a & \text{при } x > 1 \end{cases}$$

если  $x \in [-4; +4]$ , шаг 0.5;  $a \in [1..5]$ , шаг 1. Считать  $a$  параметром, и строить отдельные таблицы  $Y(x)$  для каждого  $a$ , печатая его в заголовке таблицы.

Задание 23. Найти все целочисленные степени произвольного числа  $K$  от 1 до  $N$  включительно. Иметь возможность повторного обращения в диалоге.

Задание 24. На клетчатой бумаге нарисовали окружность целого радиуса  $R$  с центром на пересечении линий. Найти количество клеток, целиком лежащих в этой окружности. Например, если  $R = 5$ , то  $K = 60$ .

Задание 25. Найти все делители некоторых натуральных чисел в диапазоне от  $N_1$  до  $N_2$ . Иметь возможность повторного обращения в диалоге.

Задание 26. Найти разложение произвольного натурального числа на простые множители. Иметь возможность повторного обращения в диалоге.

Задание 27. Найти все числа в диапазоне от  $N_1$  до  $N_2$ , которые одновременно кратны  $m$  и не кратны  $n$ . Иметь возможность повторного обращения в диалоге.

Задание 28. Составить программу для вычисления таблицы значений функции

$$F(x, y) = \begin{cases} 1 - e^{-(x+y)} & \text{при } x > 0, y > 0, \\ x + y & \text{при } x > 0, y < 0, \\ \sin^2(x + y) & \text{в остальных случаях.} \end{cases}$$

$y \in [0; 2.5]$  с шагом 0.5 и  $x \in [0; 2.5]$  с шагом 0.5. Построить таблицу на решетке из полного перебора значений  $x, y$ , где  $y$  выводить в заголовке.

Задание 29. Поле шахматной доски определяется парой значений (вертикаль, горизонталь), где первая буква, а вторая цифра, например,  $e2, f5$ . По двум заданным значениям полей определить, угрожает ли ферзь, стоящий на первом поле, второму полю. Иметь возможность повторного обращения в диалоге

Задание 30. Числа Фибоначчи образуются по закону:

$f_1 = 1, f_2 = 1, \dots, f_{k+1} = f_{k-1} + f_k$ . Найти все числа Фибоначчи, не превышающие  $N$ . Иметь возможность повторного обращения в диалоге

Задание 31. Написать программу, работающую как простой калькулятор, выполняющий операции «+», «-», «\*», «/» для двух операндов. Реализовать в диалоге ввод операндов и выбор операции.

Задание 32. Составить программу для вычисления таблицы значений функции

$$F(x, y) = \begin{cases} x^2 + y^2 & \text{при } x < 0, y < 0, \\ \sqrt{x + y} & \text{при } x > 0, y > 0, \\ \sqrt{x^2 + y^2} & \text{в остальных случаях.} \end{cases}$$

$y \in [0; 2.5]$  с шагом 0.5 и  $x \in [0; 2.5]$  с шагом 0.5. Построить таблицу на решетке из полного перебора значений  $x, y$ , где  $y$  выводить в заголовке.

Задание 33. Напечатать в возрастающем порядке все числа в диапазоне от  $1$  до  $N$ , в десятичной записи которых нет одинаковых цифр. Иметь возможность повторного обращения в диалоге.

Задание 34. Найти все натуральные числа в диапазоне от  $N_1$  до  $N_2$ , равные сумме кубов своих цифр. Иметь возможность повторного обращения в диалоге.

Задание 35. Составить программу, вычисляющую таблицу значений функции

$$f(x) = \begin{cases} \sin \frac{\pi}{x} & \text{при } |x| \geq 1 \\ \sin((x-1) \cdot \frac{\pi}{2}) & \text{при } -1 \leq x \leq 1, \end{cases}$$

если  $x \in [-2, +2]$ , шаг 0.2.

Задание 36. Составить программу для определения наибольшего общего делителя (НОД) двух простых чисел  $m$  и  $n$  по алгоритму Евклида:  $\text{НОД} = m$ , если  $m = n$ , иначе, если  $m > n$ , то  $m = m - n$ , иначе, если  $m < n$ , то  $n = n - m$ . Значения  $n, m$  вводить в диалоге, иметь возможность повторного обращения.

## Тема 4. Циклические алгоритмы вычисления сумм, произведений, количеств, пределов, последовательностей.

### Сложные циклы

В алгоритмах вычисления сумм, произведений, количеств, пределов, последовательностей особенностью является содержание тела цикла. При вычислении суммы к значению суммы многократно прибавляются новые значения слагаемых. При вычислении произведения значение многократно помножается на очередной сомножитель. При вычислении количеств значение счетчика увеличивается на 1. При вычислении предела или последовательности значение многократно вычисляется на базе предыдущего значения. Итоговое значение, кроме вычисления последовательностей, чаще единственное, так как все остальные вычисленные значения являются промежуточными.

Управление циклами этого вида выполняется также с использованием управляющих переменных, которыми фактически служит номер вычисляемого значения (слагаемого, множителя, элемента последовательности). Если число повторений известно, цикл должен быть арифметическим. В задачах вычисления с указанной точностью цикл должен быть итерационным, так как заранее не может быть известно число повторений, которое понадобится, чтобы достичь заданной точности.

**Пример №1.** Вычисление суммы известного числа слагаемых. Пусть требуется вычислить сумму  $N$  чисел натурального ряда

$S = 1 + 2 + 3 + 4 + \dots + N$ , где  $N$  – любое наперед заданное число.

Это арифметический цикл, у которого параметром является номер слагаемого, который также определяет и значение очередного слагаемого, включаемого в сумму. Обозначим его буквой  $n$ , тогда общая формула тела цикла запишется так:

$$S = S + n.$$

Смысл цикличности в том, что к значению суммы многократно прибавляются



новые значения слагаемых, обновляя ее. Число повторений цикла равно числу действий сложения, которое нужно выполнить, чтобы достичь результата.

Номер слагаемого (и его значение)  $n$  меняется в диапазоне от 1 до  $N$  с шагом, равным 1.

// Код программы примера №1.

```
void main (void)
{
    int      n;           // Управляющая переменная.
    int      S;           // Сумма ряда.
    int      N;           // Число слагаемых, включенное в сумму.
    printf ("Вычислю сумму чисел натурального ряда. \nВведите число слагаемых>1");
    scanf ("%d", &N);
    S = 0;                // Инициализация переменной S нулем обязательна.
    n = 1;                // К нулю готовимся прибавить первое слагаемое.
    do
    {
        S += n;           // Тело цикла.
        n ++;            // Приращение параметра цикла.
    } while (n <= N);
    // Печать результата вне цикла.
    printf ("При %d слагаемых сумма = %d", N, S);
} // End of main
```

Поскольку данный цикл арифметический, использование оператора цикла do... while не необходимо, но подчеркивает, что любой тип цикла в C++ можно реализовать с помощью любого оператора цикла.

**Пример №2.** Организация итерационного цикла на примере алгоритма суммирования. Пусть требуется найти сумму прогрессии

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} + \dots \text{ с точностью } \varepsilon \text{ (например, } \varepsilon = 0.001 \text{)}.$$

Количество слагаемых, которое нужно включить в сумму для достижения заданной точности, неизвестно, но известно условие, определяющее точность вычислений. Предел значения очередного слагаемого стремится к нулю:

$$\lim_{n \rightarrow \infty} \frac{1}{n} = 0,$$

поэтому можно считать, что именно это значение определяет требуемую точность вычислений, и можно закончить вычисления, когда очередное слагаемое настолько мало, что им можно пренебречь. Все переменные должны иметь вещественный тип, так как участвуют в вычислении вещественного значения.

// Код программы примера №2.

```
void main (void)
{
    float      S;
    float      eps;       // Значение точности вычислений.
```

```

float    n;                // Номер слагаемого, определяет также его значение,
                           // изменяется от 1 с шагом 1
printf("Вычислю сумму ряда. \nВведите точность вычислений <1");
scanf("%f", &eps);
    n = 1;
    S = 0;                // Входит в подготовку цикла.
    do
    {
        S += 1 / n;
        n += 1;
    } while ( 1. / n >eps);    // eps достаточно мало.
    printf("Слагаемых %5.0f, Сумма %f8.5", n, S);
} // End of main

```

В рассмотренных примерах циклические алгоритмы суммирования являются простыми. При организации сложных циклов внутренний цикл полностью, от подготовки до завершения должен находиться внутри внешнего.

**Пример №3.** Пусть требуется вычислить сумму ряда по формуле

$$S = x + \frac{x}{2} + \frac{x}{3} + \frac{x}{4} + \dots + \frac{x}{n} + \dots$$

Очевидно, что значение суммы будет зависеть от значения  $x$ . Добавить сложность к этому алгоритму может одно из следующих условий.

1. Вычислить суммы для различного числа слагаемых ( $n1$ ,  $n2$ ,  $n3$  и так далее) в арифметическом цикле или цикле, управляемом событием.
2. Вычислить суммы для различных значений  $x$ , например, для  $x \in [x_0, x_n]$  с шагом  $\Delta x$ , или вводимых с клавиатуры.
3. Вычислить суммы для различных степеней точности, например, для  $\varepsilon \in [\varepsilon_0, \varepsilon_n]$  в итерационном цикле.

Рассмотрим второй случай. Сумма заново вычисляется для каждого значения  $x$ , поэтому цикл вычисления суммы должен быть внутренним, а цикл изменения переменной  $x$  внешним. Управляющей переменной внутреннего цикла является номер слагаемого  $n \in [1, N]$ , ее шаг равен 1. Число повторений  $N$  должно быть известно, его можно ввести. Внешним циклом управляет переменная  $x$ , которая изменяется по закону арифметического цикла, пусть для определенности  $x \in [0, 1]$ , шаг 0,1.

// Код программы примера №3.

```

void main (void)
{
    float    x;                // Параметр внешнего цикла.
    float    n;                // Параметр внутреннего цикла.
    float    S;
    int      N;                // Число слагаемых, включенных в сумму.
    printf("Вычислю сумму чисел натурального ряда.\nВведите число слагаемых>1");
    scanf("%d", &N);

```

```
// Управление внешним циклом.
for (x = 0; x <= 1; x += 0.1)
{
    // Тело внешнего цикла.
    // Управление внутренним циклом.
    S = 0;
    n = 1;
    while (n <= N)
    {
        // Тело внутреннего цикла.
        S += x / n;
        n += 1;
    }
    printf ("При x=%6.2f    сумма = %6.2f\n", x, S);
}
} // End of main
```

**Пример №4.** Использование рекуррентных соотношений при вычислении последовательностей. Кстати, слагаемые сумм и сомножители произведений фактически тоже образуют вычисляемые последовательности. Часто имеет смысл использовать соотношения, существующие между соседними элементами последовательности, чтобы каждое новое значение вычислять рекуррентно, то есть исходя из значения, вычисленного на предыдущем шаге, например,

$$a_n = f(a_{n-1}).$$

Используем этот прием для вычисления значения функции  $S(x)$  в произвольной точке  $x$  по формуле разложения в ряд для 5-ти, 10-ти, 15-ти, 20-ти слагаемых, если формула задана:

$$S(x) = x - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^{2n} \cdot \frac{x^{2n}}{(2n)!}.$$

Переменная  $x$  произвольна, и не задана в условии, значит, ее следует ввести.

Помимо собственно вычисления суммы требуется сравнить вычисленные значения для разного числа слагаемых, для чего нужен сложный цикл, где внешним циклом будет арифметический цикл с управлением по количеству слагаемых, включенных в сумму:  $N = 5, 10, 15, 20$ , т. е.  $N \in [1, 20]$ , шаг 5.

В теле внешнего цикла для каждого из значений количества слагаемых  $N$  номера слагаемых изменяются от 1 до  $N$ , и если дать управляющей переменной со смыслом «количество слагаемых» имя  $k$ , то  $k \in [1, N]$ .

Внутренний цикл имеет некоторые особенности.

1. При  $k = 1$  начальное значение суммы  $S$  равно 0, а очередное слагаемое  $a$  не определено.

2. Во внутреннем цикле очередное слагаемое вычисляется по формуле  $a = \frac{x^n}{n!}$ .

Для возведения числа в степень в C++ есть функция `pow`, а для вычисления

факториала функции нет. Факториал, во-первых, растет очень быстро, и для  $n = 10$  и более имеет очень большие значения. Во-вторых, для его вычисления нужен циклический алгоритм. Если для вычисления слагаемого использовать рекуррентное соотношение, то мы избавимся от этих двух недостатков.

Вычисление степени, это произведение  $x^n = x \cdot x \cdot x \cdot \dots \cdot x$ . Вычисление факториала, это произведение чисел натурального ряда,  $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$ . Если обозначить именем  $a$  значение очередного слагаемого, тогда последующее значение может быть вычислено по отношению к предыдущему по формуле:

$$a = a \cdot \frac{x}{k}.$$

Действительно,

при  $k = 1$ ,  $a = x$ ,

$$\text{при } k = 2, a = a \cdot \frac{x}{2} = \frac{x^2}{2},$$

$$\text{при } k = 3, a = a \cdot \frac{x}{3} = \frac{x^2 \cdot x}{2 \cdot 3} = \frac{x^3}{6},$$

и так далее. Значит, в теле внутреннего цикла для вычисления очередного слагаемого может быть использована именно эта формула.

3. Ряд является знакопеременным, то есть знак слагаемого отрицательный для четных слагаемых, и положительный для нечетных. Однако нет необходимости в формуле вычислений возводить  $(-1)$  в степень. Обычно для решения подобных задач вводится переменная, хранящая знак, например, переменная  $z$  принимает значения  $+1$  или  $-1$ . Смена знака в цикле достигается присваиванием  $z = -z$  при переходе к очередной итерации.

// Код программы примера №4.

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void main (void)
```

```
{
```

```
int      N;           // Управление внешним циклом, целая переменная.
```

```
float    x;
```

```
float    k;           // Управление внутренним циклом, вещественная  
// переменная, потому что участвует в вычислении.
```

```
float    a;           // Очередное слагаемое.
```

```
float    S;           // S – сумма.
```

```
int      z;           // Знак слагаемого.
```

```
printf ("Введите x = \n");
```

```
scanf ("%f", &x);
```

```
// Управление внешним циклом по числу слагаемых.
```

```
for (N = 5; N < 20; N += 5)
```

```
{
```

```
    k = 1;           // Начинаем с номера 1.
```

```

S = 0;           // Сумма = 0.
a = 1;           // Для рекуррентного соотношения.
z = 1;           // Знак слагаемого «плюс» для k = 1.
// Управление внутренним циклом.
while (k <= N)
{
    a = a * x / k; // Очередное слагаемое рекуррентно.
    S = S + z * a; // Сложение с учетом знака.
    // Переход к следующему слагаемому.
    k += 1;
    z = - z;
};
printf ("При %4.0f слагаемых сумма равна= %8.4f\n", k - 1, S);
};
} // End of main

```

Небольшой итог. При вычислении сумм в подготовку цикла входит обнуление суммы. В теле цикла сумма вычисляется многократным прибавлением к значению суммы очередного слагаемого. Итоговое значение одно, оно получено при завершении цикла.

При вычислении произведений в подготовку цикла входит присваивание произведению значения 1. В теле цикла произведение вычисляется многократным умножением значения произведения на очередной сомножитель. Итоговое значение также одно, получено при завершении цикла

**Пример №5.** Вычисление последовательностей. Классическая задача, это нахождение арифметической или геометрической последовательностей. Для нахождения значения очередного элемента используется один или несколько предыдущих элементов  $a_n = f(a_{n-1})$ , почти как при использовании рекуррентных соотношений.

Пусть требуется вычислить элементы арифметической прогрессии, если заданы первый элемент, знаменатель и число элементов. Пусть требуется вычислить элементы геометрической прогрессии, если заданы первый элемент и знаменатель. Для возрастающей прогрессии вычислять значения, пока очередной элемент не превысит 1000, для убывающей прогрессии вычислять значения, пока очередной элемент не станет меньше, чем 0,001.

В этом примере еще раз обратим внимание на использование операторов цикла. Для организации арифметического цикла логичнее использовать for, для организации итерационного цикла while или do... while.

// Код программы примера №5.

```

#include <stdio.h>
#include <conio.h>
void main (void)
{
    float    Ar, d_Ar;           // Первый элемент и знаменатель.

```

```

float    Geom, d_Geom;           // Первый элемент и знаменатель.
int      Cou_Ar;                 // Число элементов прогрессии.
//----- Задача 1. -----
printf ("Введите первый элемент и знаменатель арифметической прогрессии\n");
scanf ("%f%f", &Ar, &d_Ar);
printf ("Введите число элементов\n");
scanf ("%d", &Cou_Ar);
// Управление первым циклом по числу слагаемых.
for (int k = 2; k <= Cou_Ar; k++)    // Номер начинается с 2.
{
    Ar = Ar + d_Ar;
    printf ("%d-й элемент равен: %6.2f\n", k, Ar);
}
//----- Задача 2. -----
printf ("Введите первый элемент и знаменатель геометрической прогрессии\n");
scanf ("%f%f", &Geom, &d_Geom);
if (d_Geom > 1)
{
    printf ("Возрастающая последовательность\n");
    k = 2;
    while (Geom < 1000)           // while, так как Geom известно.
    {
        Geom = Geom * d_Geom;
        printf ("%d-й элемент равен: %6.2f\n", k, Geom);
        k ++;
    }
}
else
    if (d_Geom < 1)
    {
        printf ("Убывающая последовательность\n");
        k = 2;
        while (Geom > 0.001)
        {
            Geom = Geom * d_Geom;
            printf ("%d-й элемент равен: %8.4f\n", k, Geom);
            k ++;
        }
    }
    else    // Здесь знаменатель равен 1.
        printf ("Все элементы одинаковы\n");
} // End of main

```

**Пример №6.** Использование операторов break и continue для циклов do, while, for. Оператор прерывания break внутри любого цикла прекращает выполнение

цикла с передачей управления следующему за циклом оператору. Осуществляя выход из цикла при наступлении каких-либо обстоятельств, позволяет корректно завершить цикл. В случае вложения прерывает только непосредственно охватывающий цикл.

Найдем сумму чисел натурального ряда, не превышающую некоторого заданного значения. Узнаем также, сколько элементов будет включено в сумму.

// Код программы примера №6.1.

```
#include <stdio.h>
void main (void)
{
    int      Max, Sum = 0, k;           // Max – наибольшее значение суммы.
    printf ("\nВведите наибольшее значение\n");
    scanf ("%d", &Max);
    for (k = 1; ;k ++ )                 // Бесконечный цикл.
    {
        Sum += k;
        if (Sum > Max)                 // Условие завершения цикла.
            break;                     // Прерывание. При выходе известно k.
    }
    printf ("количество элементов %d", k);
} // End of main
```

Оператор continue внутри любого цикла выполняет переход к следующей итерации тела цикла. Противоположен break. В любой точке цикла прервет текущую итерацию и перейдет к проверке условия завершения цикла.

Найдем сумму слагаемых вида  $S = \frac{1}{x}$  для  $x \in [-1; +1]$  с шагом 0,1. Особой точкой в этом ряду является точка  $x = 0$ , функция в этой точке не определена. При  $x = 0$  вычисление выполнять нельзя, значит, это действие следует пропустить.

// Код программы примера №6.2.

```
#include <stdio.h>
#include <math.h>
void main (void)
{
    float      Sum=0;
    float      x;
    for (x = - 1; x <= 1.1; x += 0.1)
    {
        if (fabs (x) < 0.0001)
            continue;                 //Если в знаменателе 0.
        Sum += 1 / x;
    }
    printf ("Сумма %6.2f", Sum);
} // End of main
```

**Пример №7.** Бесконечный цикл. В тех случаях, когда управление циклом осуществляется не по событиям или условиям извне, а возникает непосредственно в теле цикла, можно использовать бесконечные циклы. Бесконечным он может быть назван только с точки зрения синтаксиса, потому что условие выхода из цикла не прописано непосредственно в управлении циклом. Содержимое тела цикла может быть произвольным, но какие-то действия внутри должны привести к тому, что процесс когда-то закончится. Прерывание и выход из цикла выполняет обычно оператор break.

```
while (1)
{
    ...
    // Проверка условия и выход.
}
for (; ; )
{
    ...
    // Проверка условия и выход.
}
```

### Варианты заданий

Задание 1. Найти число  $\pi$ , используя произведение

$$\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \dots$$

Точность для печати не менее 5-ти знаков. Во внешнем цикле выполнить вычисления для 50-ти, 100, 200, 400 сомножителей.

Задание 2. Найти число  $\pi$ , используя формулу суммы ряда

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots + (-1)^{k+1} \cdot \frac{1}{2 \cdot k + 1}.$$

Во внешнем цикле выполнить вычисления для 50-ти, 100, 200 слагаемых. Точность для печати не менее 5-ти знаков.

Задание 3. Вычислить значение полинома  $P(x)$  в произвольной точке  $x$ , если

$$P(x) = 1 + x + x^2 + x^3 + \dots + x^n.$$

Выполнить вычисления для 100 слагаемых. Во внешнем цикле составить таблицу, выводящую на экран значение полинома для  $x \in [-0.5; 1,1]$ .

Задание 4. Вычислить значение функции  $Y(x)$  в произвольной точке  $x$  по формуле разложения в ряд

$$Y(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}.$$

Величину  $x$  вводить. Во внешнем цикле выполнить вычисления для 5-ти, 10, 15, 20, 25 слагаемых. Точность для печати не менее 5-ти знаков.

Задание 5. Вычислить значение суммы ряда  $Y(x)$  в произвольной точке  $x$ , если

$$Y(x) = \sin x + \sin^2 x + \sin^3 x + \dots + \sin^n x.$$

Величину  $x$  вводить в градусной мере. Во внешнем цикле выполнить



вычисления для 10-ти, 20, 30 слагаемых. Точность для печати не менее 5-ти знаков.

Задание 6. Вычислить значение функции  $Y(x)$  в произвольной точке  $x$  по формуле разложения в ряд

$$Y(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^n \cdot \frac{x^{2n}}{2n!}.$$

Величину  $x$  вводить. Во внешнем цикле выполнить вычисления для 5-ти, 10, 15, 20 слагаемых. Точность для печати не менее 5-ти знаков.

Задание 7. Вычислить значение функции  $Y(x)$  в произвольной точке  $x$  по формуле разложения в ряд

$$Y(x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \dots + (-1)^{n-1} \cdot \frac{x^n}{n}.$$

Величину  $x$  вводить. Во внешнем цикле выполнить вычисления для 5-ти, 10, 15, 20 слагаемых. Точность для печати не менее 5-ти знаков.

Задание 8. Составить программу для вычисления таблицы значений функции

$$Y(x) = x \cdot (x - 0,5) \cdot (x - 1) \cdot (x - 1,5) \cdot \dots \cdot (x - 5),$$

для  $x \in [0; 0,9]$ , шаг 0,1. Параметром внутреннего цикла является величина  $a$  в сомножителе вида  $(x - a)$ .

Задание 9. Вычислить значение суммы ряда

$$S = \sqrt{\underbrace{\left(2 + \sqrt{\left(2 + \sqrt{\left(2 + \dots + \sqrt{2}\right) \dots}\right)}\right)}_{n \text{ корней}}}$$

Во внешнем цикле выполнить вычисления для  $n = 5, 10, 15, 20$ . Один из тестовых примеров проверить вручную. Точность для печати не менее 5-ти знаков.

Задание 10. Для некоторого натурального числа  $N$  найти все целые  $x, y$ , такие, что выполняется равенство:  $x^2 + y^2 = N$ .

Во внешнем цикле значение  $N$  вводить в диалоге.

Задание 11. Вычислить значение суммы ряда  $Y(x)$  в произвольной точке  $x$ , если

$$Y(x) = x \cdot \sin\left(\frac{\pi}{4}\right) + x^2 \cdot \sin\left(2 \cdot \frac{\pi}{4}\right) + \dots + x^n \cdot \sin\left(n \cdot \frac{\pi}{4}\right).$$

Величину  $x$  вводить в градусной мере. Во внешнем цикле выполнить вычисления для 5-ти, 10, 15, 20 слагаемых. Точность для печати не менее 5-ти знаков.

Задание 12. Найти произведение  $n$  сомножителей

$$S = \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdot \frac{7}{8} \cdot \dots$$

Во внешнем цикле выполнить вычисления для 50-ти, 100, 200 сомножителей. Точность для печати не менее 5-ти знаков.

Задание 13. Вычислить значение функции  $Y(x)$  в произвольной точке  $x$  по формуле разложения в ряд

$$Y(x) = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \dots + (-1)^{n-1} \cdot \frac{x^n}{n!}.$$

Во внешнем цикле выполнить вычисления для 5-ти, 10, 15, 20, 25 слагаемых. Один из тестовых примеров проверить вручную.

Задание 14. Составить программу для вычисления таблицы значений функции

$$Y(x) = 1 + 2x + 3x^2 + 4x^3 + \dots + 20x^{19},$$

для  $x \in [0; 1,2]$ , шаг 0,05.

Задание 15. Составить программу для вычисления таблицы значений функции

$$Y(x) = (x-1) + \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} + \frac{(x-1)^4}{4} + \dots$$

Отследить значение функции для числа взятых слагаемых, равного 5, 10, 15, 20. Во внешнем цикле задать значение  $x \in [0; 1,2]$ , шаг 0,2.

Задание 16. Вычислить значение функции  $Y(x)$  в произвольной точке  $x$  по формуле разложения в ряд

$$Y(x) = \cos x + \frac{\cos 2x}{2} + \frac{\cos 3x}{3} + \dots + \frac{\cos nx}{n}.$$

Значение  $x$  ввести в градусной мере. Во внешнем цикле выполнить вычисления для 5-ти, 10, 15, 20 слагаемых. Точность для печати не менее 5-ти знаков.

Задание 17. Вычислить значение функции  $Y(x)$  в произвольной точке  $x$

$$Y(x) = \frac{(x-2) \cdot (x-4) \cdot (x-8) \cdot \dots \cdot (x-64)}{(x-1) \cdot (x-3) \cdot (x-7) \cdot \dots \cdot (x-63)}.$$

Выполнить вычисления для  $x \in [-1, 1]$ , шаг = 0,1.

Задание 18. Вычислить значение суммы

$$S = 1 \cdot 2 + 2 \cdot 3 \cdot 4 + 3 \cdot 4 \cdot 5 \cdot 6 + \dots + n \cdot (n+1) \cdot \dots \cdot 2n.$$

Во внешнем цикле вычислить сумму для значений  $n = 10, 20, 30, 40, 50$ .

Задание 19. Составить программу для вычисления суммы:

$$S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{9999} - \frac{1}{10000},$$

а) последовательно слева направо;

б) вычислить отдельно суммы положительных и отрицательных слагаемых, а затем вторую сумму вычесть из первой.

Если результаты различны, объяснить.

Задание 20. Вычислить значение суммы

$$S = \sqrt{\underbrace{\left( 3 + \sqrt{\left( 6 + \dots \sqrt{\left( 3(n-1) + \sqrt{3n} \right)} \right)} \right)}_{n \text{ корней}}}$$

Во внешнем цикле выполнить вычисления для  $n = 5, 10, 15, 20$ . Один из тестовых примеров проверить вручную. Точность для печати пять знаков.

Задание 21. Вычислить значение суммы

$$S = \frac{1}{a} + \frac{1}{a^2} + \frac{1}{a^3} + \dots + \frac{1}{a^n}.$$

Величину  $a$  вводить в диалоге во внешнем цикле. Для каждого  $a$  выполнить вычисления при различном числе слагаемых  $n = 5, 10, 15, 20$ .

Задание 22. Вычислить значение цепной дроби

$$\frac{1}{1 + \frac{1}{3 + \frac{1}{5 + \frac{1}{7 + \frac{1}{\dots}}}}} \\ n + \frac{1}{n+2}$$

Выполнить вычисления в диалоге, чтобы можно было ввести произвольное значение  $n$ . Один из тестовых примеров проверить вручную.

Задание 23. Последовательность вычисляется по закону

$$1, \quad 1 + \frac{1}{2}, \quad 1 + \frac{1}{2} + \frac{1}{3}, \quad 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}, \dots$$

Известно, что существует предел последовательности. Найти его с точностью  $\varepsilon$  знаков после запятой. Величину точности во внешнем цикле последовательно задать равной 0,1, 0,01, 0,001, 0,0001, 0,00001. Выводить в виде таблицы значение предела последовательности, и значение числа слагаемых, включенных в сумму.

Задание 24. Для некоторого наперед заданного натурального числа  $N$  найти такую наименьшую степень двойки  $2^k$ , которая не превосходит  $N$ . Вывести значение  $k$  и все степени двойки. Во внешнем цикле значение  $N$  вводить в диалоге.

Задание 25. Составить программу для вычисления суммы

$$S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{9999} - \frac{1}{10000},$$

а) последовательно справа налево;

б) вычислить отдельно суммы положительных и отрицательных слагаемых, а затем вторую сумму вычесть из первой.

Если результаты различны, объяснить.

Задание 26. Составить программу для вычисления произведения  $n$  сомножителей

$$P = \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdot \dots \cdot \frac{n-1}{n},$$

а) последовательно справа налево;

б) последовательно слева направо.

Если результаты различны, объяснить. Значение  $n$  вводить в диалоге.

Задание 27. Вычислить значение полинома в точке  $x$

$$Y(x) = x^{10} + 2x^9 + 3x^8 + \dots + 10x + 11.$$

Составить таблицу значений полинома, где во внешнем цикле переменная  $x$

принимает значения  $x \in [-2, 2]$  при шаге 0,2.

Задание 28. Вычислить значение полинома в точке  $x$

$$Y(x) = 11x^{10} + 10x^9 + 9x^8 + \dots + 2x + 1.$$

Составить таблицу значений полинома, где во внешнем цикле переменная  $x$  принимает значения  $x \in [-2, 2]$  при шаге 0,2.

Задание 29. В промежутке между значениями функций  $y(x) = \ln x$  и  $y(x) = e^x$  при некотором заданном  $x > 0$  найти все целочисленные значения  $y$ , попадающие в интервал, а также найти их количество.

Значение  $x$  вводить в диалоге.

Задание 30. Найти все натуральные трехзначные числа, сумма цифр которых равна некоторому натуральному числу  $N$ . Подсчитать их количество. Значение  $N$  вводить в диалоге.

Задание 31. Вычислить значение суммы ряда  $Y(x)$  в произвольной точке

$$Y(x) = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}.$$

Величину  $x$  вводить в диалоге. Во внешнем цикле выполнить вычисления для 5-ти, 10, 15, 20 слагаемых. Точность для печати не менее 5-ти знаков.

Задание 32. В диапазоне от 1 до некоторого натурального  $N$  найти и вывести на печать все числа, которые делятся на 3, на 2, но не делятся на 5. Найти их количество. Величину  $N$  вводить в диалоге.

Задание 33. Для заданных  $x$ ,  $a$  и  $n$  найти сумму

$$y(x, a) = \underbrace{((\dots ((x + a)^2 + a)^2 + \dots + a)^2 + a)}_{n \text{ скобок}}$$

Значения  $x$  и  $a$  вводить в диалоге. Во внешнем цикле  $n$  принимает значения от 5 до 25 с шагом 5.

Задание 34. Числа Фибоначчи определяются формулами:

$$f_0 = f_1 = 1, f_2 = 2, \text{ и каждое последующее число равно сумме двух предыдущих.}$$

Составить программу, которая найдет  $n$  чисел Фибоначчи, а также сумму всех чисел Фибоначчи, не превосходящую некоторого  $N$ , введенного в диалоге.

## **Тема 5. Использование циклических алгоритмов в решении содержательных задач**

Наиболее важным в программировании является подготовительный этап, который называют постановкой задачи, тесно связанный с этапом формализации задачи. От правильного их выполнения во многом зависит время и качество программирования.

Постановка задачи обычно заключается в ее словесном описании. Как известно, на вербальном уровне точное определение модели невозможно, поэтому следующий этап, это формализация, то есть выбор математической или иной модели, адекватно отражающей суть задачи. Точных рекомендаций при выборе модели дать невозможно.

Важным также является выбор структур данных и определение их типов.

До сих пор мы рассматривали уже готовые, формализованные задачи, где математическое описание было дано или лежало на поверхности. Теперь приведем пример неформализованной задачи

**Пример.** Червячок ползет по дереву вверх, стартуя со скоростью  $V$  м./час. Каждый последующий час движения его скорость падает на 10% от предыдущего значения. Если высота дерева  $H$ , то за какое время червячок достигнет вершины, если только это возможно.

Для решения задачи нужно выяснить, прежде всего, что выполняется многократно: бежит время, медленно растет пройденный путь, падает скорость. В задаче все, как в жизни. Очевидно, что управлять этим циклом должна переменная, обозначающая время, так как все три переменные связаны в формуле вычисления пути при заданной скорости  $S = V \cdot t$ . Очевидно, что точное условие завершения процесса сформулировать невозможно, поэтому нужно использовать итерационный цикл с условием завершения  $S < H$ .

В подготовку цикла входит присваивание начальных значений пути  $S = 0$ , скорости  $V$  и времени  $t = 1$ . Причем, если  $S$  и  $V$  имеют значения до момента первого вычисления, то  $t = 1$  уже готовится к первой итерации. Для убедительности будем выводить на экран значения пути, пройденного к исходу каждого часа, и значение скорости в каждый час времени.

Момент, завершающий цикл, интересен многими особенностями, и сильно зависит от входных данных.

1. Первый вариант завершения цикла, это когда червячок сможет добраться до вершины. Условием завершения является  $S < H$ , при этом в теле цикла оператор `break` отсекает лишнюю итерацию.

2. Второй вариант, это когда червячок не сможет добраться до конца дерева, так как скорость падает очень быстро, а высота большая. Должна быть выполнена проверка условия  $V > 0$  с некоторой степенью точности, так как возможно, что скорость упадет практически до нуля, а до конца дерева останется еще очень долгий путь. Кстати, проверка  $S < H$  также выполняется приближенно.

```
#include <stdio.h>
void main (void)
{
    float    V;           // Скорость предыдущего часа и скорость нового часа.
    float    S;           // Путь.
    float    H;           // Высота дерева.
    float    t;           // Время пути.
    printf ("Введите высоту дерева и начальную скорость движения\n");
    scanf ("%f%f", &H,&V);
    printf (" Таблица зависимости пути от времени. \n");
    printf ("-----\n");
    printf ("Время \tПуть \tСкорость \n");
    printf ("----- \n");
```

```

// В подготовку цикла входит присваивание значений пути, скорости и времени.
S = 0.;
t = 1.;
while (1)      // Бесконечный цикл.
{
    S = S + V * 1.;           // Путь, пройденный за t - тый час:
                              //  $S = S + V * t$ , где  $t=1.0$ 

    if ( S > H )
    {
        printf ( "Я уже почти добрался до вершины.\n ");
        break;               // Чтобы исключить лишнюю итерацию.
    }

    if (V <= 0.01)
    {
        printf ( "Я никогда не доберусь до вершины.\n ");
        break;               // Цикл закончен.
    }

    printf ("%6.0f %8.3f %8.3f\n", t, S, V);
    V *= 0.9;                 // Аналог записи  $V = V * 0.9$ ;
    t += 1.;
}
} // End of main

```

### Варианты заданий

Задание 1. Директор школы набирает группу для обучения школьников по факультативной программе. Обучение платное, общая стоимость курса  $K$  рублей. Сколько же должен платить каждый ученик? Очевидно, эта сумма зависит от значения  $K$  и от количества учеников.

Вычислить и вывести на экран таблицу сумм, которую должен внести один ученик, если группа будет состоять из трех, четырех, и т. д. до 20-ти учеников.

Задание 2. Валяльная фабрика ежегодно увеличивает объем продаж на 2%, снижая себестоимость продукции на 1%. В текущем году объем продаж составил 500 тыс. руб., а себестоимость пары валенок была равна 55 руб.

Вычислить и вывести на экран таблицу прогнозируемого увеличения объема продаж и снижения себестоимости на ближайшие  $K$  лет.

Задание 3. Ученица швеи начинает работу, сострачивая в день 2 пары рукавиц. Совершенствуя свое мастерство, она каждый день выполняет в два раза больше работы, чем в предыдущий день. Больше, чем 100 пар в день, состроить нельзя.

Найти, на который день ученица достигнет вершин мастерства. Сколько всего пар рукавиц ей при этом придется сшить? Вывести на экран таблицу роста мастерства швеи по дням.

Задание 4. На день рождения ребенка бабушка открыла счет в банке и положила на него 5 долларов. Каждый год она добавляет 5 долларов. Годовой процент по банковскому счету равен 12%.

Какая сумма накопится к совершеннолетию ребенка (к 16-ти годам),

включая последний взнос. Вывести на экран таблицу ежегодного состояния счета.

Задание 5. Процент по банковскому вкладу равен 6%. Если положить в банк сумму  $N$  руб., то эта сумма будет ежегодно увеличиваться.

Как будет изменяться сумма в течение ближайших 10-ти лет? Если годовая инфляция составляет 3,5%, то сколько же на самом деле будут стоить эти деньги? Вывести на экран таблицу ежегодного состояния счета.

Задание 6. Пара кроликов дает приплод раз в четыре месяца, в среднем по 5 крольчат.

Вычислить и вывести в виде таблицы, каков будет ежегодный приплод от пары, двух, трех и так далее до 20-ти пар.

Задание 7. Оплата труда приходящей няни осуществляется по часам. За срок до 6-ти часов она получает по 25 руб. в час. Начиная с 6-ти часов работы, каждый последующий час стоит в два раза дороже, то есть 50, потом 100, и т.д. Родители, отправляясь на вечеринку, хотят знать сумму, которую они заплатят няне, но не знают, насколько задержатся.

Вычислить и вывести на экран таблицу оплат услуг няни, начиная с одного часа до 24-х часов.

Задание 8. Поженившись, молодые супруги решили откладывать деньги на покупку автомобиля. Муж может вложить ежемесячно  $M$  руб., жена  $V$  руб. Если положить деньги в банк, то по срочному вкладу годовой процент равен 12%. Автомобиль мечты стоит  $N$  тыс. руб.

Через какой срок молодые поедут на юг в собственном авто? Для убедительности выведите таблицу ежемесячных накоплений с учетом процента по банковскому вкладу.

Задача 9. Маркетинговое исследование, проведенное фирмой «Рога и копыта», выявило, что каждый третий год работы предприятия является очень прибыльным, а каждый пятый убыточным. Известно, что этот год (текущий) является убыточным, а предыдущий был очень прибыльным.

Выяснить, какие из ближайших 15-ти лет будут прибыльными, а какие убыточными. Вывести на экран только года и комментарии.

Задание 10. По окончании сессии всегда есть некоторое количество «хвостистов». Деканат решил провести курсы для отстающих в объеме 40 часов, и установил стоимость оплаты часа, равной 100 руб. Из суммы, оплаченной студентами, преподавателю причитается 40%.

Найти, сколько денег получит преподаватель, если будет заниматься с одним, двумя, тремя, и т.д. до  $M$  студентов. Может ли он озолотиться, если золотой горой считать сумму 20 тыс. руб. Скольких бездельников ему придется выучить?

Задание 11. Незнайка учит английский язык. В первый день он выучил два слова, а каждый последующий день собрался выучивать на два слова больше, чем в предыдущий.

Найдите и выведите в виде таблицы, на который день Незнайка выучит 100 слов, 200, 300, 400 и т. д. до 1000? В английском языке около 50 тыс. слов, а срок жизни Незнаек примерно 30 лет. Успеет ли до своей кончины Незнайка выучить английский язык? Если нет, то сколько Незнаечих жизней понадобится, чтобы

выучить английский язык?

Задание 12. Спортсмен начал тренировки, пробежав  $N$  км. Ежедневно он увеличивает длину пробегаемого пути на 20%.

Найти, к исходу какого дня спортсмен пробежит марафонскую дистанцию. Вывести таблицу длин ежедневно пройденного пути.

Задание 13. Спортсмен начал тренировки, пробежав  $N$  км. Ежедневно он увеличивал длину пробегаемого пути на 10%. Пусть его средняя скорость  $V$  км./ч.

Найти, сколько времени ежедневно занимает тренировка, если время не должно превысить 4 часов. Найти, какое расстояние будет пробегать спортсмен. Вывести таблицу ежедневно пройденного расстояния.

Задание 14. Старушка решила купить телевизор, когда внук подарил ей 1000 руб. Она положила их в сбербанк под 8% годовых. Ежемесячно на этот же счет старушка вносит 200 руб. Самый дешевый телевизор стоит  $K$  руб.

Через сколько месяцев старушка посмотрит кино на СТС? Вычислить и вывести на экран состояние счета помесечно.

Задание 15. Известно, что заяц бежит в  $K$  раз быстрее, чем ползет черепаха. Они стартуют из пункта  $A$  в пункт  $B$ , расстояние между которыми  $N$  км.

Вычислить, сколько раз заяц сбегает туда и обратно, пока черепаха доберется до пункта назначения.

Задание 16. В комнате  $N$  человек,  $M$  кошек и  $K$  мух. Вместе у них 100 ног и лап. Вычислить, сколько в комнате людей, сколько кошек, и сколько мух?

Задание 17. Мама и дочка идут навстречу друг другу со скоростью  $V_1$  км./час. Расстояние между ними  $S$  км. В момент начала движения их собачка, которая была у мамы, видит дочку и начинает бегать от одной к другой со скоростью  $V_2$  км./час., большей, чем  $V_1$ .

Найти, какое расстояние пробежала собака, прежде чем мама с дочкой встретились. Вычислить число пробегов собаки от мамы к дочке.

Задание 18. Улитка ползет вверх по дереву с начальной скоростью  $V$  м./сек. При этом она устает, и ее скорость движения падает по прямолинейному закону на  $0,1 \cdot V$  за каждый час.

Вычислить и вывести в таблицу, на какую высоту улитка поднимается в течение каждого часа. Узнать, через сколько часов она смертельно устанет.

Задание 19. В стаде  $K$  коров и  $k$  коз. Корова дает примерно 25 л. молока в сутки, коза примерно 2,5 л.

Найти, сколько молока приносит ежедневно стадо, в котором число коров и коз может быть от двух до десяти.

Задание 20. Заяц бежит в  $K$  раз быстрее, чем ползет черепаха, скорость которой равна  $V$ . Они стартуют из пункта  $A$  в пункт  $B$ , расстояние между которыми  $N$  км.

Составьте таблицу перемещения объектов, если стартовый момент времени равен 0, а интервал времени 0,5 часа. В таблице учесть, что когда заяц закончил движение, черепаха еще ползет.

Задание 21. Одноклеточная амеба каждые три часа делится на 2 клетки.

Вычислить и вывести в виде таблицы, сколько клеток будет через каждые три



часа в течение двух суток.

Задание 22. Составить таблицу стоимости порций товара весом от 100 гр. до 1 кг. с шагом 100 гр., и от 1 кг до 10-ти кг. с шагом 500 гр. Цена вводится с клавиатуры.

Задание 23. Фабрика по производству тапочек ежегодно увеличивает объем продаж на 5 процентов. Себестоимость продукции при этом уменьшается на 1 процент. В текущем году объем продаж составил  $N$  тыс. руб., а себестоимость пары тапочек  $S$  руб.

Вычислить и вывести на экран таблицу увеличения объема продаж и снижения себестоимости на ближайшие  $K$  лет.

Задание 24. Ученик мастера начинает с изготовления в день одной табуретки. Совершенствуясь, он каждый день изготавливает на одну табуретку больше, чем в предыдущий день. Больше, чем 20 табуреток в день изготовить нельзя.

Найти, на который день ученик достигнет вершин мастерства, и сколько табуреток он смастерит за все время обучения? Для убедительности вывести на экран таблицу ежедневной производительности.

Задание 25. Татьяна Ларина, читая очередной французский роман, подсчитала сумму номеров прочитанных страниц. Обозначим эту сумму через  $Q$ .

Определить номер последней прочитанной страницы. Учесть, что юная девица может быть не в ладах с арифметикой.

Задание 26. Для продавщицы Несчитайкиной разработайте программу, которая по стоимости 1 кг некоторого продукта выдает таблицу стоимости 50, 100, 150,..., 1000, 2000, 3000,..., 10000 г этого продукта.

Задание 27. Спортсмен бежит по кругу длиной 400 метров, а тренер измеряет среднюю скорость движения на каждом круге. Скорость на первом круге была  $V$  км./час, но на каждом круге она падает на 10%.

Если спортсмен пробежит  $N$  кругов, то какая скорость будет на последнем круге? Вычислить и вывести на экран таблицу скоростей на каждом круге пути.

Задание 28. Спортсмен бежит по кругу длиной 400 метров, а тренер измеряет среднюю скорость движения на каждом круге. Начальная скорость была  $V_1$  км./час, но на каждом круге она падает на 10%.

Узнать, на каком круге нужно закончить движение, если скорость не должна упасть ниже  $V_2$  км./час? Вычислить и вывести на экран таблицу скоростей на каждом пройденном круге пути.

Задание 29. Богатый дядя подарил племяннику на пятилетний юбилей один доллар. Мальчик отнес их в банк и положил на счет под 8 % годовых. Каждый год на день рождения дядя дарит мальчику столько долларов, сколько лет мальчику.

Вычислить, сколько набегит к совершеннолетию мальчика (18 лет). Вывести состояние счета ежегодно с учетом последнего вклада.

Задание 30. Богатый дядя подарил племяннику на рождение 1 доллар. Каждый день рождения сумма подарка удваивается.

Вычислить, сколько набегит к совершеннолетию мальчика (18 лет). Вывести состояние счета ежегодно.

Задание 31. Карамель стоит  $K$  руб., мармелад  $M$  руб., шоколад  $S$  руб. за кг.

Вычислить и вывести на экран таблицу стоимости каждого вида сладостей весом 100 гр., 200 гр., и т.д. до 1 кг. включительно.

Задание 32. Составить таблицу умножения в виде Пифагоровой таблицы.

Задание 33. Напечатать таблицу перевода температуры из градусов по шкале Цельсия ( $C$ ) в градусы по шкале Фаренгейта ( $F$ ) для значений от  $C_0$  до  $C_n$  с шагом 1 градус. (Перевод осуществляется по формуле  $F=1,8*C+32$ ).

Задание 34. Два пловца тренируются в бассейне, длина дорожки которого 50 м. Первый пловец начинает тренировку со скоростью  $V_1$  км./час., второй со скоростью  $V_2$  км./час. Оба начинают движение одновременно с одного края бассейна. Скорость первого падает на 5% за час, скорость второго на 3%.

Встретятся ли пловцы в точке начала движения, если время тренировки  $T$ ? Если нет, то какое время им придется плавать, чтобы встретиться в первый раз?

Задание 35. На прилавке в один ряд лежат  $N$  арбузов. Известно, что каждый арбуз на 100 граммов легче, чем среднее арифметическое весов его соседей.

Найти вес второго арбуза с точностью до грамма, если первый арбуз весит  $A$  килограммов, а последний  $B$  килограммов.

## Тема 6. Практическое использование механизма функций C++

Принципы модульного программирования требуют использования функций для решения любой задачи, алгоритм которой может быть описан абстрактно. Чтобы правильно использовать функции, необходимо четко представлять:

1. Что такое функция.
2. Как написать собственную функцию.
3. Как обратиться к функции.
4. Как правильно передать данные в функцию и как вернуть их.

### Что такое функция

Функция – это абстрактный алгоритм решения некоторой самостоятельной задачи. Фактически, функция является одним из конструируемых типов данных:

- имеет имя;
- имеет тип;
- может иметь параметры (аргументы функции), которые обеспечивают связь функции с внешним окружением;
- имеет тело, в котором разработан алгоритм решаемой задачи.

### Описание функции

В C++ все функции описываются на одном уровне, вложений не допускается. Синтаксис описания функции:

```
тип_возвращаемого_значения имя_функции (тип_параметров имена_параметров)
{
    описания локальных переменных;
    описание алгоритма;
    return    возвращаемое_значение; //Отсутствует, если функция void.
}
```

Первая строка описания называется заголовком функции. Используется для объявления функции в теле программы, содержит все ее *внешние* характеристики.

1. Тип функции. Это тип возвращаемого значения или void для функций, не возвращающих значения. Если тип функции опущен, то по умолчанию тип int. Оператор return в теле функции содержит выражение, тип которого совпадает с типом функции. Для функций void этот оператор отсутствует или пустой (return;).

2. Имя функции. Это имя main для главной функции программы, или любое, не совпадающее с ключевыми словами и именами других объектов программы.

3. Формальные параметры функции. Это перечисленные через запятую имена аргументов функции вместе с их типами, или void, если параметров нет.

Тело функции содержит:

- 1) описания локальных переменных, действующих только в теле функции;
- 2) описание алгоритма функции;
- 3) возврат в точку вызова посредством оператора return.

Формальные параметры функции, это ее внешние данные. Это название подчеркивает, что данные описания формальны, то есть не участвуют в реальных действиях, а только описывают взаимосвязь данных в теле функции. Количество формальных параметров функции и их типы могут быть любыми.

### Обращение к функции

Под обращением к функции понимается реальное выполнение алгоритма функции, каждый раз с различными входными данными. Выполняется из любой другой функции программы с использованием механизма вызова функции через операцию обращения. Операция обращения к функции зависит от типа функции, и может иметь две формы.

1. Если функция возвращает значение, то им является одно значение базового типа или указатель. Оно может быть использовано в выражениях или печати в виде обращения к функции, которое имеет название «оператор-выражение»:

переменная = имя\_функции (фактические параметры);

printf ("форматная\_строка ", имя\_функции (фактические\_параметры));

Например, при обращении к библиотечной функции sin:

y = sin(x); // Значение функции вычислено и присвоено y.

printf ("%6.2f", sin(x)); // Значение функции напечатано.

sin(x); // Значение функции вычислено, но

// что происходит с вычисленным значением?

2. Если функция не возвращает значения (функция типа void), то обращение к ней выглядит как обычный оператор программы, и имеет название «оператор-функция»:

имя\_функции (фактические параметры);

Например, при обращении к библиотечной функции printf:

printf ("%d,%d", a, b);

При обращении к функции ей передаются фактические параметры. Это выражения, имеющие реальные значения на момент обращения к функции, с которыми функция отрабатывает очередной вызов.

### Формальные и фактические параметры функции

Тип параметров, их количество и порядок следования называются совместно «сигнатура параметров». В описании функции и в обращении сигнатуры параметров должны строго совпадать, то есть, формальные и фактические параметры должны соответствовать друг другу по количеству, типу и порядку следования. При несовпадении типов формальных параметров в описании и фактических в обращении, C++ выполняет автоматическое приведение типов фактических параметров к типу формальных параметров.

Формальные параметры функции, это имена переменных с указанием их типов. Фактические параметры, это выражения, то есть константы, переменные или выражения.

В C++ существуют два способа передачи параметров в функцию.

1. По значению. При этом создается локальная копия фактического параметра в теле функции, следовательно, фактические значения параметров при обращении не могут быть изменены. Так можно защитить входные данные от их нежелательного изменения функцией.

2. По ссылке. Изменение механизма передачи данного в функцию формально выглядит добавлением знака & к имени формального параметра:

имя\_функции (тип\_параметра & имя\_параметра)

При этом функция и вызывающая программа работают с адресом объекта в памяти, следовательно, функция может изменить значение параметра. При этом фактическим параметром, соответствующим формальному параметру – ссылке, может быть только имя переменной (левостороннее выражение).

**Пример №1.** Функция с одним параметром, возвращающая значение. Функция имеет тип, имя, один параметр. Приведем пример описания функции, которая вычисляет значение некоторой функции в указанной точке, например,  $y = x^2$ .

```
float sqr (float x)           // Имя функции sqr .
{
    return x*x;              // Одно возвращаемое значение.
}
```

При обращении к функции используется оператор-выражение. Фактическим параметром может быть константа, переменная или выражение.

```
#include <stdio.h>
void main (void)
{
    float    a = 2, b;
        b = sqr (a);           // Фактический параметр – переменная.
        printf ("\n%f", b);
        b = sqr (3.5);         // Фактический параметр – константа.
        printf ("\n%f", b);
        b = (a+1.1);           // Фактический параметр – выражение.
        printf ("\n%f", b);
// Запись обращения сокращается, если оно используется в функции вывода.
        printf ("\n%f", sqr (b);
        printf ("\n%f", sqr (3.5);
        printf ("\n%f", sqr (a+b*5.);
} // End of main
```

**Пример №2.** Функция со многими параметрами, возвращающая значение. Функция имеет тип, имя, несколько параметров. Приведем пример описания функции, которая находит среднее арифметическое трех чисел. Имя функции Avg. Тип функции и ее параметров float.

```
float Avg (float a, float b, float c)
{
    float    S;                // Локальная переменная.
        S = (a + b + c) / 3.;
        return S;              // Тип совпадает с типом функции.
}
```

Можно привести пример простой записи той же функции.

```
float Avg (float a, float b, float c)
{
    return (a + b + c) / 3.;    // Тип совпадает с типом функции.
}
```

При обращении к функции используется оператор-выражение. Фактическим параметром может быть константа, переменная или выражение.

```
#include <stdio.h>
void main (void)
{
// Фактические параметры – переменные. Результат присвоен переменной у.
float x1 = 2.5, x2 = 7. , x3 = 3.5;
    float y = Avg (x1, x2, x3);
// Фактические параметры – константы. Результат присвоен переменной у.
    y = Avg (2., 4., 7.);
printf ("x1=%f, x2=%f, x3=%f y= %f\n" ,2., 4. ,7., y);
// Фактические параметры – выражения. Результат выводится на печать.
    printf ("x1=%f, x2=%f, x3=%f y= %f\n" , x1, x2, x3, y);
} // End of main
```

**Пример №3.** Функция, возвращающая значение. Алгоритм функции может быть достаточно сложный. Приведем пример описания функции, вычисляющей сумму  $n$  чисел натурального ряда.

```
int Sum (int n)
{
int    S = 0;                // Переменная для накопления значения суммы.
int    i;                   // Управляющая переменная для цикла суммирования.
    for (i = 1; i<= n; i++)
        S += i;
return    S;
}
```

В этом примере локальная переменная  $S$  необходима для накопления суммы, а переменная  $i$  не необходима. Зная, что параметр, передаваемый по значению, не будет изменен функцией, его можно использовать как рабочую переменную для управления циклом. По завершении работы функции фактический параметр имеет то же значение, что и до обращения.

```
int Sum (int n)
{
    for (int S = 0; n > 0; n --)    // При n == 0 цикл будет завершен.
        S += n;
return    S;
} // Функция не изменит значения n.
```

Обращение к функции особенностей не имеет.

```
#include <stdio.h>
void main (void)
{
int Cou;
    printf ("\n Введите число слагаемых\n ");
    scanf ("%d", &Cou);
    printf ("Сумма=%d", Sum(Cou));
}
```

```
} // End of main
```

Имя любого локального данного не должно совпадать с именем функции. Вот пример распространенной ошибки, когда имя функции совпадает с именем ее локального данного. Результат в этом случае непредсказуем.

```
int Sum (int n)
{
    int      Sum = 0;
    int      i;
    for (i = 1; i<= n; i++)
        Sum += i;
    return   Sum;
}
```

**Пример №4.** Функция, не возвращающая значения, и не имеющая параметров. Приведем пример описания функции, которая выводит на экран 50 символов «звездочка». Имя функции print. Список формальных параметров пуст. Функция, не возвращающая значения, не имеет в своем теле оператора return.

```
#include <stdio.h>
void print (void)           // Функция типа void, не возвращает значения.
{
    int      i;             // Внутренняя рабочая переменная.
    for (i = 1; i <= 50; i++)
        printf ("%c", '*'); // Вывод символа «звездочка».
    printf ("\n");
}
```

При обращении к функции используется оператор-функция. Фактические параметры не передаются, однако скобки после имени функции опускать нельзя. Обращение к функции выполняется дважды.

```
void main (void)
{
    print ();               // Обращение к print ().
    printf (" \tПример вызова функции.\n");
    print ();
}
```

Вывод на экран будет иметь вид:

***** Пример вызова функции. *****
--

**Пример №5.** Функция с параметрами, не возвращающая значения. Приведем пример описания функции, которая выводит на экран произвольное количество символов «звездочка». Поскольку количество произвольно, его значение может быть определено только на момент обращения к функции, значит, его следует сделать параметром функции. Функция, не возвращающая значения, не имеет в своем теле оператора return.

```

void print (int count)                // Печать строки «*» длиной count.
{
    int      i;
    for (i = 1; i <= count; i++)      // count – количество символов в строке.
        printf ("%c", '*');
    printf ("\n");
}

```

Если задуматься о том, почему же именно звездочки нужно печатать, можно вынести в список параметров еще и вид выводимого символа, тогда функция будет иметь два параметра, по смыслу количество выводимых символов и их вид. Второй параметр функции имеет тип char.

```

void print (int count, char symbol)    // Печать строки символов symbol
                                      // длины count .
{
    int      i;
    for (i = 1; i <= count; i++)      // count – количество символов в строке.
        printf ("%c", symbol);
    printf ("\n");
}

```

При обращении к функции оператор-функция может использоваться в цикле. Фактический параметр один. На экран будет выведено 12 строк, в первой строке один символ, в каждой последующей на один больше.

```

void main (void)
{
    int cou;
    clrscr();
    for (cou = 1; cou <= 12; cou++)
        print (cou);                // Здесь cou – количество при обращении.
    getch();
} // End of main

```

При обращении к функции с двумя параметрами, первый параметр означает число выводимых символов, второй вид символа. Оператор-функция выводит при первом обращении 80 символов тире ('-'), при втором 25 символов плюс ('+'). При первом обращении фактические параметры константы, при втором переменные.

```

void main (void)
{
    char      ch;
    ch = '+';
    int      cou = 25;
    clrscr ();
    print (80, '-');                // 80 символов тире ('-').
    print (cou, ch);                // 25 символов плюс ('+').
    getch();
} // End of main

```



**Пример №6.** Логические функции. Логические функции используются во многих случаях, когда нужно выполнить проверку условия, как правило, сложного, или в алгоритмах поиска. Особенностью их является то, что функция возвращает одно значение логического типа, которого в C++ нет. Логический тип заменяется целочисленным, интерпретация которого соответствует принятой идеологии: ложно то, что равно 0, истинно то, что отлично от 0 (не обязательно это 1). Это правило используется при вычислении значений логических выражений, а также при проверке условий в операторе if и в операторах цикла.

Примером логической функции с простым алгоритмом может служить функция определения четности числа. Функция должна вернуть значение логического выражения «остаток от деления числа на два равен 0».

```
int Chet (int num)           // Функция возвращает логическое значение.
{
    return    num%2 == 0;    // Остаток от деления на два.
}
```

Примером логической функции с довольно сложным алгоритмом может служить алгоритм определения, является число простым или нет. Как известно, натуральное число является простым, если оно не имеет делителей, кроме 1 и самого себя. Но для того, чтобы найти, есть делители или нет, нужно последовательно делить число на все числа натурального ряда, которые могут быть его делителями, начиная с двух до значения, равного половине самого числа. Как только делитель найден, дальнейшие проверки не имеют смысла, так как уже известно, что число не является простым. Если же выполнены все возможные проверки, и ни один делитель не найден, то число простое.

```
int Easy (int num)           // Функция возвращает логическое значение.
{
    int      mod;              // Делитель числа, управляющая переменная.
    // Управление по возможным значениям делителей числа.
    for (mod = 2; mod <= num/2; mod++)
        if (num%mod == 0)
            return 0;          // Прерывание поиска, делитель найден.
    return 1;                  // Поиск завершен по управлению.
}
```

При обращении к функции имеет смысл выполнить проверку, четно число или нет, так как любое четное число не является простым.

```
#include <stdio.h>
#include <conio.h>
void main (void)
{
    int      Number;
    do
    {
        printf ("Введите число\n");
        scanf ("%d", &Number);
```

```

    if (! Chet (Number) && Easy (Number) )    // Если число нечетное и простое.
        printf ("Число простое\n");
    else
        printf ("Число не простое\n");
    printf ("Продолжение – любая клавиша, выход – Esc\n");
} while (getch () != 27)
} // End of main

```

Логика оператора условия звучит как перевод с русского на C++. Однако, если эта запись вызывает трудности в понимании, используйте явные проверки соответствия возвращаемого значения логической константе: 1, это «истина», 0, это «ложь».

```

    if (Chet (Number) != 0)                    // Число нечетно.
    {
        if (Easy (Number) == 1)              // Число простое.
            printf ("Число простое\n");
    }
    else
        printf ("Число не простое\n");
    ...

```

**Пример №7.** Несколько слов о роли оператора return и о сложности алгоритма функции. Оператор return относится к операторам управления программой. Его назначение не только определить и вернуть вычисленное функцией значение, но и прервать работу алгоритма функции, выполнив выход из нее правильным образом. Синтаксис return имеет две формы:

```

return Выражение;
return;

```

Первая форма используется для функций, возвращающих значение, вторая для функций типа void. В любом случае оператор return прервет выполнение алгоритма функции и передаст управление в точку вызова.

Рассмотрим пример функции, которая возводит вещественное число в целочисленную степень. Для данной задачи фактически возможны три алгоритма, три ветви, выбор одной из которых зависит от входных данных. Для положительной степени вычисляется простое произведение  $n$  сомножителей, для отрицательной значение равно дроби  $1/(\text{произведение } n \text{ сомножителей})$ , а для нулевой степени значение всегда равно 1. Решение начинается с анализа входных данных, и в зависимости от степени сразу расходуется на три независимые ветки, каждая из которых заканчивается своим выходом со своими данными.

// В алгоритме этой функции три варианта выхода.

```

float pow_1 (float x, int n)                // Основание float, степень int.
{
    float S;                                // Локальная переменная.
    if ( n > 0 )                             // Положительная степень.
    {
        for( S = 1.0; n > 0; n --)

```

```

        S *= x;
    return S;                                     // Решение найдено при n>0.
}
else
    if ( n < 0 )                                 // Отрицательная степень.
    {
        for(S = 1.0; n < 0; n ++)
            S *= x;
        return 1. / S;                         // Решение найдено при n<0.
    }
    else
        return 1.0;                            // Решение найдено при n = 0
printf ("Кому нужны эти функции?");           // Оператор вне всех ветвей.
                                              // Он никогда не будет выполнен.
}

```

Этот алгоритм интересен еще и тем, что механизм передачи параметров по значению защищает внешние данные от изменения их функцией. Это позволяет не вводить рабочую переменную для организации цикла в теле функции, а использовать входное данное  $n$  в качестве рабочей переменной. При входе в функцию ей передано входное значение  $n$ . В процессе выполнения алгоритма функция изменяет  $n$ , но не внешнее данное, а его локальную копию. По завершении работы функции внешняя переменная  $n$  сохранит свое значение, каким оно было до обращения.

```

#include <stdio.h>
void main (void)
{
    int      n;
    float    x;
    printf ("Введите вещественное основание и целую степень\n");
    scanf ("%f%d", &x, &n);
    printf ("\n\nДанные и решение:%6.2f %4d %6.2f\n", x, n, pow_1 (x,n));
} // End of main

```

**Пример №8.** Функция, возвращающая значение через список параметров. Использование механизма возвращения данных через параметры необходимо всегда, когда функция должна вернуть более одного значения. Возвращаемые значения должны передаваться по ссылке.

Синтаксически в заголовке функции к имени параметра добавляется `&`, например `int & Cou`. Технически радикально изменяется механизм передачи данных, в этом случае в тело функции передается адрес объекта с именем `Cou`, который выделен для него в вызывающей программе. Иллюстрацией отличия двух механизмов передачи данных будет пример функции, которая должна поменять значения двух переменных.

// Функция Swap1 с параметрами по значению.

```

void Swap1 (int x, int y)
{
    int    tmp;
    tmp = x;
    x = y;
    y = tmp;    // Переменные переменились своими значениями.
}
// Функция Swap2 с параметрами по ссылке.
void Swap2 (int &x, int &y)
{
    int    tmp;
    tmp = x;
    x = y;
    y = tmp;    // Переменные переменились своими значениями.
}

```

Как видим, функции одинаковы во всем, кроме механизма передачи параметров. Обратимся к этим функциям одинаковым образом, получим разный результат.

```

void main (void)
{
    int    a=5, b=10;
    printf ("Было: a=%d b=%d\n", a, b);
    Swap1 (a, b);
    printf ("Передача по значению: a=%d b=%d\n", a, b);
    Swap2 (a, b);
    printf ("Передача по ссылке: a=%d b=%d\n", a, b);
} // End of main

```

Вывод на экран покажет, что функция Swap1 работает с локальными копиями данных, а Swap2 работает с адресами данных.

Было: a=2 b=10 Передача по значению: a=2 b=10 Передача по ссылке: a=10 b=2
--

**Пример №9.** Еще один пример функции, возвращающей значение через параметры, покажет, что такая функция не всегда типа void, а возвращаемое функцией значение расширяет логику задачи, добавляя функции новый смысл.

Пусть функция находит площадь и периметр треугольника, заданного длинами сторон. Возвращаемых значений два, площадь и периметр, следовательно, их нужно возвращать через параметры. Функция будет возвращать значение типа int, смысл которого заключается в проверке условия существования треугольника. Если треугольник существует, функция вернет 1, если не существует, вернет 0.

```
#include <stdio.h>
```

```

#include <math.h>
int Triangle (float a, float b, float c, float & p, float &s)
{    // p и s – внешние данные, могут быть входными и выходными.
float pp;                                // Полупериметр, локальная переменная.
    if (a + b <= c || a + c <= b || b + c <= a)
        return 0;                        // Треугольник не существует.
    else
    {
        p = a + b + c;
        pp = p / 2.;
        s = sqrt (pp*(pp – a)*(pp – b)*(pp – c));
        return 1; // Треугольник существует.
    }
}

```

Обращение к функции имеет особенности. Во-первых, функция приобрела статус логической, значит, при обращении к ней необходимо проанализировать возвращенный результат. Во-вторых, фактическими параметрами, подставляемыми на место формальных параметров – ссылок, при обращении могут быть только адресуемые выражения, то есть только имена переменных.

```

void main (void)
{
float    A, B, C;                        // Длины сторон фактические.
float    Perim, Square;                  // Периметр и площадь фактические.
printf("Введите длины сторон треугольника\n");
scanf ("%f%f%f", &A, &B, &C);
if (Triangle (A, B, C, Perim, Square)==1) // Проверка условия.
    printf("Периметр равен %6.2f, площадь равна %6.2f\n", Perim, Square);
else
    printf("Треугольник не существует\n");
} // End of main

```

Ошибкой будет попытка обращения:

```
Triangle (A, B, C, Perim+1, 6.)
```

**Пример №10.** Функция, возвращающая указатель. Такие функции используются, когда тип возвращаемого данного отличен от базового типа, или когда функция возвращает новое динамическое значение.

Пусть параметрами функции являются две целочисленные переменные. Если первый параметр больше второго, нужно создать новый объект и присвоить ему значение, равное 1. В противном случае функция не создает объект, и должна вернуть пустой адрес (NULL). Возвращаемое значение, это адрес вновь созданного объекта или пустой указатель NULL. Тип функции – указатель на целое. Возвращаемое значение должно быть присвоено указателю, объявленному в вызывающей программе.

```

#include <stdio.h>
#include <conio.h>

```

```

#include <math.h>
int * New_obj (int a, int b) // Тип функции указатель.
{
    int * Obj; // Новый объект будет создан или нет в функции.
    if (a > b)
    {
        Obj = new int; // Создается новый объект.
        *Obj = 1; // Ему присваивается значение.
        return Obj; // Возвращается его адрес.
    }
    else
        return NULL;
}

В вызывающей программе объявлен указатель int * ip, которому
присваивается возвращаемое значение.
void main (void)
{
    int p1, p2; // Переменные.
    int * ip; // Указатель на новый объект.
    printf ("Введите две переменные\n");
    scanf ("%d%d", &p1, &p2);
    // Обращение к функции обычное.
    ip = New_obj (p1, p2);
    if (ip != NULL)
        printf ("Новый объект имеет адрес %p, значение %d\n", ip, *ip);
    else
        printf ("Новый объект не создан, использовать ip нельзя,
его значение %s\n", ip);
} // End of main

```

В первом выводе на экран для вывода адреса ip использован спецификатор формата %p, во втором для вывода константы NULL использован спецификатор формата %s. Спецификатор %s позволит увидеть текстовое представление константы. Если использовать спецификатор %p, адрес будет показан как 0000.

### Варианты заданий

#### Задание 1.

1. Описать функцию  $F(x)$ , вычисляющую значение периодической функции в произвольной точке. Период функции  $T = 2$ . На интервале  $[-1, 0]$  она совпадает с функцией  $y = x + 1$ , на интервале  $(0, 1]$  совпадает с функцией  $y = -x + 1$ . Обратиться к функции на интервале  $x \in [-2, +4]$  с не менее чем десятью точками.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести все результаты, полученные при обращении к функции  $F(x)$ . Вывести в это окно таблицу значений функции  $F(x)$ .

## Задание 2.

1. Описать логическую функцию  $Yes(x, y)$ , которая определит, принадлежит ли точка с координатами  $(x, y)$  единичной окружности, центр которой совпадает с началом координат. Обратиться с координатами точек, лежащими на параболе  $y = x^2$  для  $x \in [-2; +2]$ , шаг = 0.5.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты всех обращений к первой функции. Вывести в это окно таблицу значений логической функции.

## Задание 3.

1. Описать логическую функцию  $Is\_letter(c)$ , которая определит, является ли некий произвольный символ  $c$  (параметр функции) одной из строчных или заглавных букв русского алфавита. Обратиться к функции в диалоге, передавая ей посимвольно текст, вводимый с клавиатуры.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждый новый символ вводился бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

## Задание 4.

1. Описать функцию  $S(x, \varepsilon)$ , вычисляющую значение суммы ряда в точке  $x$  с указанной точностью  $\varepsilon$ , если формула суммы:

$$S = x - \frac{x}{2} + \frac{x}{3} - \frac{x}{4} + \dots$$

Обратиться с координатами точек  $x \in [-0.5; +0.5]$ , шаг = 0.1.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести все результаты, полученные при обращении к первой функции. Вывести в это окно таблицу значений функции  $S(x, \varepsilon)$ .

## Задание 5.

1. Описать логическую функцию  $Is\_Tri(a, b, c)$ , которая по значениям длин трех отрезков  $a, b, c$  определит, можно ли построить треугольник с такими сторонами. Обратиться в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая тройка значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

## Задание 6.

1. Описать функцию  $Pi(\varepsilon)$ , вычисляющую значение числа  $\pi$  по формуле:

$$\frac{\pi}{2} = \frac{2 \times 4}{3 \times 3} \cdot \frac{4 \times 6}{5 \times 5} \cdot \frac{6 \times 8}{7 \times 7} \cdot \dots$$

с произвольной точностью  $\varepsilon$ . Значение точности передать в функцию как

аргумент. Обратиться к функции в цикле, вычисляя значение с точностью 0.01, 0.001, 0.0001.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести все три результата, полученные первой функцией. Вывести в это окно таблицу значений функции  $P_i(\varepsilon)$ .

Задание 7.

1. Описать функцию  $P(x)$ , вычисляющую значение полинома в произвольной точке  $x$  по формуле:

$$P(x) = 1 + x + x^2 + x^3 + x^4 + \dots \text{ для 50 слагаемых.}$$

Обратиться к функции со значениями  $x \in [-0.5; +0.5]$  с шагом 0.1.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести все результаты, полученные при обращении к первой функции. Вывести в это окно таблицу значений функции  $P(x)$ .

Задание 8.

1. Описать функцию  $Min(x, y, z)$ , которая вернет значение наименьшего из трех своих аргументов. Обратиться в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая тройка значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

Задание 9.

1. Описать функцию  $Square(a, b, c)$ , которая найдет площадь треугольника по значениям длин сторон  $a, b, c$ . Если треугольник не существует, функция вернет значение 0. Обратиться к функции в цикле со значениями длин сторон (1,2,3), (2,3,4), (3,4,5), (4,5,6), (5,6,7) и т.д. до (10,11,12).

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести все результаты, полученные при обращении к первой функции. Вывести в это окно таблицу значений функции  $Square(a, b, c)$ .

Задание 10.

1. Описать функцию  $Transform(n)$ , которая преобразует натуральное число  $n$ , «приписывая» к нему по единичке в начале и в конце. Обратиться к функции со значениями 23, 234, 2345.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести все три результата, полученные при обращении к первой функции. Вывести в это окно таблицу значений функции  $Transform(n)$ .

Задание 11.

1. Описать функцию  $R(x1, y1, x2, y2)$ , которая вычисляет расстояние между двумя точками на координатной плоскости. Обратиться к функции в диалоге,



чтобы определить расстояния между началом координат и вершинами некоторого квадрата, заданного координатой верхнего левого угла и длиной стороны.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты, полученные при всех обращениях к первой функции. Вывести в это окно все четыре результата.

Задание 12.

1. Описать функцию  $Count(r)$ , которая определит, сколько точек с целочисленными координатами попадают в круг радиуса  $R$  с центром в начале координат. Обратиться в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждое новое значение радиуса вводилось бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

Задание 13.

1. Описать логическую функцию  $P(x)$ , которая определит, является ли ее аргумент  $x$  простым числом. Обратиться к функции со значениями чисел натурального ряда от 7 до 99.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть 10 строк по 50 позиций в строке. Вывести в это окно результаты, полученные при обращении к первой функции. После каждого 10-го вывода восстанавливать окно.

Задание 14.

1. Описать функцию  $R(a, b, c)$ , определяющую радиус вписанной окружности для треугольника со сторонами  $a, b, c$ . Обратиться к функции с равносторонними треугольниками со сторонами 2,3,4,5,6. Предусмотреть условие существования треугольника. Если он не существует, функция должна вернуть значение 0.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести все результаты, полученные при обращениях к первой функции. Вывести в это окно таблицу значений функции  $R(a, b, c)$ .

Задание 15.

1. Описать функцию  $S(a, b, h)$ , которая найдет площадь равнобокой трапеции с заданными основаниями и высотой. Обратиться к функции в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая тройка значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

Задание 16.

1. Описать функцию  $Square(r1, r2)$ , которая найдет площадь кольца с заданными радиусами. Если первый параметр меньше второго, возвращать

значение 0. Обратиться к функции в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая пара значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

Задание 17.

1. Описать функцию  $S\_C(r, angle)$ , которая найдет значение площади сектора круга с заданными значениями радиуса и угла (в градусах). Если угол отрицательный или больше чем  $360^\circ$ , возвращать значение 0. Обратиться к функции в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая пара значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

Задание 18.

1. Описать функцию  $Sum\_AP(a1, d, n)$  которая найдет сумму арифметической прогрессии. Предусмотреть проверку исходных данных, и в случае ошибки возвращать значение 0. Обратиться к функции в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая тройка значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

Задание 19.

1. Описать логическую функцию  $Is\_in(x1, y1, x2, y2, x3, y3)$ , которая по заданным координатам вершин треугольника определит, находится ли начало координат внутри этого треугольника. Возвращать значение  $-1$ , если треугольник не существует,  $0$ , если снаружи, и  $1$ , если внутри. Обратиться к функции в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая порция входных значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

Задание 20.

1. Описать функцию  $R(long\ int\ N)$ , которая по заданному значению произвольного натурального числа определит его разрядность. Обратиться к функции со значениями 3, 13, 130, 1300, 13000.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты, полученные при всех обращениях к первой функции. Вывести в это окно таблицу значений функции  $R(N)$ .

#### Задание 21.

1. Описать логическую функцию  $Yes(N)$ , которая по заданному значению натурального числа определит, является ли оно палиндромом. Предусмотреть анализ входных данных. Если они заданы некорректно, функция вернет значение  $-1$ . Обратиться к функции в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждое новое число вводилось бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

#### Задание 22.

1. Описать функцию  $Kvadr(x, y)$ , которая по заданным координатам точки определит номер четверти координатной плоскости, где находится точка. Если точка на осях координат, вернуть значение 0. Обратиться к функции, передавая ей поочередно значения  $(1,1)$ ,  $(-1,1)$ ,  $(-1,-1)$ ,  $(1,-1)$ ,  $(0,0)$ .

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести все результаты, полученные при обращениях к первой функции. Вывести в это окно таблицу значений функции  $Kvadr(x, y)$ .

#### Задание 23.

1. На плоскости даны две окружности. Описать логическую функцию  $Yes$ , которая определит, имеют ли окружности точки пересечения. Возможны варианты: окружности не пересекаются, окружности пересекаются, окружности касаются, окружности концентрические. Возвращать значение номера варианта или текстовую строку, содержащую значение соответствующего текста. Обратиться к функции в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая порция входных значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

#### Задание 24.

1. На плоскости даны окружность радиуса  $R$  и отрезок координатами концов. Описать логическую функцию, которая определит, находится ли отрезок полностью внутри окружности. Обратиться к функции в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая порция значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

#### Задание 25.

1. Описать функцию  $Sum(x, n)$ , которая найдет сумму знакопеременного ряда вида:

$$S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + (-1)^{k+1} \cdot \frac{1}{k}$$

Обратиться к функции в цикле со значениями  $k = 50, 100, 150, 200$ .

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой, и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты, полученные при всех обращениях к первой функции. Вывести в это окно таблицу значений функции  $Sum(x, n)$ .

Задание 26.

1. Описать функцию  $Pi(eps)$ , которая находит значение числа  $\pi$  по формуле произведения:

$$\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \times \frac{4}{3} \cdot \frac{4}{5} \times \frac{6}{5} \cdot \frac{6}{7} \times \dots \text{ с указанной точностью } eps.$$

Обратиться к функции со значениями точности 0,01, 0,001, 0,0001.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты, полученные при трех обращениях к первой функции. Вывести в это окно таблицу значений функции  $Pi(eps)$ .

Задание 27.

1. Описать функцию, которая вычисляет произведение первых  $N$  натуральных чисел (от 1 до  $N$ ). Обратиться к функции со значениями  $N = 5, 10, 15, 20, 25, 30$ .

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты, полученные при всех обращениях к первой функции. Вывести в это окно таблицу значений первой функции.

Задание 28.

1. Описать функцию, которая находит значение многочлена вида

$P(x) = 1 + x + x^2 + x^3 + x^4 + \dots + x^n$  для заданных значений  $x$  и  $n$ . Обратиться к функции при  $n = 10$  со значениями  $x \in [-0.9; +0.9]$ , шаг 0.1.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты, полученные при всех обращениях к первой функции. Вывести в это окно таблицу значений функции  $P(x)$ .

Задание 29.

1. Описать функцию, которая возводит произвольное число с плавающей точкой в целочисленную положительную или отрицательную степень путем многократного умножения (в цикле). Обратиться к функции, передавая ей поочередно значения  $x \in [-2; +2]$ . Для каждого  $x$  вычислять положительную степень, отрицательную и нулевую.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты, полученные при всех обращениях к первой функции. Вывести в это окно таблицу значений первой функции.

Задание 30.

1. Два интервала числовой оси заданы координатами своих концов. Описать

логическую функцию, которая определит, имеют ли эти интервалы общие точки. Корректность данных проверить. Если интервалы заданы некорректно, функция вернет значение  $-1$ . Обратиться к функции в диалоге.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждая новая четверка значений вводилась бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

Задание 31.

1. Дано натуральное число. Описать функцию, которая поменяет порядок цифр числа на обратный. Обратиться со значениями чисел 5, 15, 150, 1500, 15000.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты, полученные при всех обращениях к первой функции. Вывести в это окно таблицу обращений к первой функции.

Задание 32.

1. Описать функцию, которая найдет объем цилиндра, если заданы радиус основания и высота. Обратиться к функции, чтобы найти объемы цилиндров высотой  $H = 12, 13, 14, 15$  см. и для каждой высоты с радиусами основания  $R = 5, 10, 15, 20$  см.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть 7 строк на 50 позиций. Использовать эту функцию для вывода результатов, полученных первой функцией. В каждом окне выводить таблицу  $V(R)$  при фиксированном значении  $H$ .

Задание 33.

1. Описать функцию, которая найдет время, через которое встретятся два тела, равномерно движущиеся навстречу друг другу, если заданы их скорости и расстояние между телами. Обратиться к функции со значениями  $r = 100$  м. и  $v_1 = 2, 3, 4, 5$  м./сек., и для каждого  $v_1$  значение  $v_2 = 2, 3, 4, 5$  м./сек.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть 7 строк на 50 позиций. Использовать эту функцию для вывода результатов, полученных первой функцией. В каждом окне выводить таблицу  $T(v_2)$  при фиксированном значении  $v_1$ .

Задание 34.

1. Описать функцию, которая найдет предел последовательности, вычисляемой по закону:

$$1, 1 + \frac{1}{2}, 1 + \frac{1}{2} + \frac{1}{3}, 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}, \dots$$

с точностью  $\varepsilon$  знаков после запятой, где  $\varepsilon$  – параметр функции. Обратиться к функции, передавая ей последовательно значения точности 0,1, 0,01, 0,001, 0,0001.

2. Описать консольную функцию, которая выводит на чистый экран окно с

рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести все результаты, полученные при обращении к первой функции. Вывести в это окно таблицу значений пределов при указанных значениях точности.

Задание 35.

1. Описать функцию, которая вычислит значение полинома в точке  $x$ ,

$$Y(x) = 11x^{10} + 10x^9 + 9x^8 + \dots + 2x + 1.$$

Обратиться к функции, чтобы составить таблицу значений полинома, где переменная  $x$  принимает значения  $x = [-2, +2]$  при шаге 0,2.

2. Описать консольную функцию, которая выводит на чистый экран окно с рамкой и позиционирует курсор внутри него. Размер окна должен быть достаточен, чтобы вывести результаты всех обращений к первой функции. Вывести в это окно таблицу значений первой функции.

Задание 36.

Две прямые на плоскости заданы коэффициентами своих уравнений:

$$a_1x + b_1y = c_1$$

$$a_2x + b_2y = c_2$$

1. Описать функцию, которая определит, параллельны ли эти прямые или пересекаются. Если пересекаются, найти координаты точки пересечения. Иметь возможность повторного обращения.

2. Описать консольную функцию, которая выводит на экран окно с рамкой, позиционирует курсор примерно посередине этого окна и удаляет изображение курсора. Использовать ее для организации диалога, чтобы каждые новые значения вводились бы в чистом окне, примерно посередине, а в следующей строке выводился результат.

## **Тема 7. Массивы и указатели. Алгоритмы работы с одномерными массивами. Использование функций при работе с массивами**

Прежде чем приступить к программированию задач этого раздела, необходимо четко знать ответы на следующие вопросы.

1. Что такое массив?
2. Как описать массив?
3. Как размещаются в памяти элементы массива?
4. Как обратиться к элементу массива?
5. Как присвоить значения элементам массива?

Основные алгоритмы для работы с массивами можно условно назвать в следующем перечне.

1. Использование массивов переменной длины.
2. Присваивание значений элементам массива: инициализация, ввод, генерация случайным образом, вычисление по определенному закону.
3. Поиск в массиве, то есть способы нахождения элементов массива, удовлетворяющие какому-то критерию.
4. Определение количества элементов, удовлетворяющих какому-то критерию.
5. Суммирование элементов массива, всех или только соответствующих какому-то условию.
6. Поиск максимального и минимального значения.
7. Преобразование элементов массива по какому-либо принципу.
8. Получение программным путем нового массива, если известен алгоритм его формирования.
9. И так далее.

Следуя правилам модульного подхода в программировании, мы здесь и далее будем использовать функции для решения задач обработки данных, тем более что основных алгоритмов немного. Если выделить абстрактный алгоритм, описать и отладить функцию, то ею можно пользоваться, как говорится, всю оставшуюся жизнь. Функции программиста komponуются в библиотеки. Для библиотек исходных текстов используются заголовочные файлы, содержащие описания функций. Их тексты инклидируются в программу пользователя и компилируются вместе с ней. Для объектных библиотек исходные тексты функций предварительно компилируются и сохраняются в виде объектного кода (файлы с расширением .obj). Тогда в отдельный заголовочный файл, который также инклидируется в текст программы пользователя, выносятся только прототипы функций, содержащихся в объектной библиотеке.

В соотношении массивов и функций можно выделить два механизма:

- 1) массив как параметр функции;
- 2) массив как возвращаемое функцией значение.

В любом случае массив, это указатель на нулевой элемент массива.

Когда массив используется как параметр функции, то в функцию фактически передается адрес массива. Механизм передачи по адресу, значит, функция

возвращает измененный массив.

Пусть в вызывающей программе объявлен массив:

```
int    Array[10];        // Имя Array, это адрес нулевого элемента массива.
```

В записи заголовка функции, которой передается любой целочисленный массив, формальный параметр должен быть указателем. Синтаксически это можно записать двумя способами.

Первый способ:

```
void Function1 (int Array[], int len)
{
    for (int i = 0; i < len i++)
        // Обращение к Array[i];
}
```

Пустые скобки у формального имени массива в списке параметров функции подчеркивают, что передается адрес массива. Длина массива, если ее не передавать (опустить `int len`), будет равна длине массива в вызывающей программе (как он объявлен, в данном примере 10).

Второй способ:

```
void Function2 (int * Array, int len)
{
    for (int i = 0; i < len i++)
        // Обращение к Array[i];
}
```

Признак «\*» у формального имени массива в списке параметров означает, что функция получает указатель на массив, то есть при обращении фактически передается адрес массива в памяти.

При обращении к функции фактическим параметром должно быть имя любого целочисленного массива, который объявлен в вызывающей программе и получил в ней значения.

```
Function1 (Array, 10);        // Передается весь массив.
```

```
Function2 (Array, 5);         // Передается весь массив, но его длина ограничена 5.
```

Еще раз подчеркнем, что механизм передачи массива только по адресу, и после такого обращения, если функция изменит элементы массива, это станет известно вызывающей программе. Передача массива в функцию через формальный параметр, это одновременно и его возвращение после вызова функции.

Массив как возвращаемое функцией значение используется, если массив возвращается через имя функции. Тип функции должен быть указателем, он будет содержать адрес массива в памяти. В теле функции должен быть оператор `return`, возвращающий адрес массива. В вызывающей программе возвращаемое значение должно быть присвоено указателю. Указатель должен быть объявлен в главной программе, и должен адресовать область памяти, объем которой достаточен, чтобы вместить в себя весь массив.

Синтаксически это можно записать так:

```
int * Function3 (int * Array, int len)
```



```

{
    for (int i = 0; i < len; i++)
        // Обращение к Array[i];
    return Array;
}

```

В этом примере мало смысла, так как массив в списке параметров.

Имеет смысл использовать второй способ для динамических массивов, создаваемых в теле функции. Например, функция создает динамический массив указанной длины. Динамическая память для массива выделена в теле функции.

```

int *New_mas (int len)           // Тип функции указатель.
{
    int *mas = new int [len];     // Выделено len * sizeof(int) байт памяти,
                                   // адрес которой присвоен переменной mas.

    for (int i = 0; i < len; i++)
        *(mas+i) = i+1;          // Присвоены значения по возрастанию.
    return mas;                   // Адрес необходимо вернуть.
}

```

Вызывающая программа должна быть готова к получению нового данного. Для этого в ней объявлен указатель \*Array. До обращения к функции он пустой, и примет значение только после обращения. Возвращаемое функцией значение присваивается указателю Array = New\_mas (len\_A).

```

#include <stdio.h>
void main (void)
{
    int *Array;                  // Пустой указатель.
    int len_A;
    printf ("Введите длину массива\n");
    scanf ("%d", &len_A);
    Array = New_mas (len_A);     // Теперь получил адрес.
    for (int i = 0; i < len_A; i++)
        printf ("%3d", Array[i]);
    printf ("\n");
} // End of main

```

Приведем полный текст файла, содержащего описание множества функций, реализующих различные алгоритмы работы с массивами. Это отдельный файл, который хранит абстрактные описания алгоритмов. Такие файлы могут быть доступны многим программам. Файл следует назвать заголовочным, и дать расширение имени файла "имя.hpp" или "имя.h". Далее этот файл с помощью директивы #include легко можно подключить к любой программе, которой требуется обработка массивов. Например, если имя заголовочного файла "Task.h" ("Task.hpp"), то он будет доступен любой программе, в тексте которой есть директива #include "Task.h".

Физически файл может находиться в том же директории, в котором находится использующая его программа (по умолчанию). Он может размещаться в

системном директории, путь к которому прописан в настройках интегрированной среды разработчика. Третий вариант, это собственная директория INCLUDE, на которую дополнительно нужно настроить интегрированную среду.

Каждый пример, это описание решения какой-нибудь задачи для массивов в общем виде, то есть безотносительно к способу создания и инициализации массива. В каждой задаче значимыми параметрами массива являются тип элементов и их количество (длина массива), которые являются параметрами функций. В этом примере используются статические массивы, в том числе массивы переменной длины. Для сканирования элементов массивов используется либо адресация с помощью индексов (прямая), либо адресация с использованием указателей (косвенная). В любом случае массивы можно считать массивами условно переменной длины, так как функция получает длину массива в виде переменной величины.

Итак, текст заголовочного файла.

```
#include <stdio.h>
```

```
#include <math.h>
```

// **Пример №1.** Функция вывода на экран целочисленного массива в общем виде.

// Формальные параметры функции – имя массива (указатель), и длина массива.

```
void Print_mas (int mas[], int len)    // len – длина массива.
{
    int      i;                        // Рабочая переменная.
    printf ("Массив:\n");              // Вывод заголовка.
    for (i = 0; i < len; i++)
        printf ("%5d", mas[i]);        // Вывод элемента массива в строку.
    printf ("\n");                     // По завершении вывода переход на
                                        // новую строку.
}
```

// **Пример №2.** Функция ввода целочисленного массива в общем виде.

// Формальные параметры функции – имя массива (указатель), и длина массива.

// Длина массива определена в теле функции. Чтобы функция могла вернуть это значение, формальный параметр len должен передаваться по ссылке.

void Input\_mas (int \*mas, int &len) // Длина изменяется, &len – ссылка.

```
{
    int      *ip;                      // Для адресации используется указатель
                                        // на элемент массива.

    printf ("Введи кол-во элементов массива \n");
    scanf ("%d", &len);
    printf ("Введи элементы массива \n");
    for (ip = mas; ip < mas + len; ip++)
        scanf ("%5d", ip)              // & не нужен, .
}
```

// **Пример №3.** Функция преобразования целочисленного массива увеличивает

// значение каждого элемента в два раза.

```

// Поскольку массив всегда передается в функцию как указатель, то измененные в
// теле функции значения элементов массива доступны и вызывающей программе.
// Формальные параметры функции – имя массива (указатель), и длина массива.
void Transform_mas(int *mas, int len)    // Длина массива len не изменяется.
{
    int i;
    for (i = 0; i < len; i++)
        mas[i] = mas[i] * 2;
}

// Пример №4. Поиск в массиве. Смысл задачи в том, что для массива задан
// критерий отбора, то есть условие, которому должны соответствовать элементы
// массива. Задачу поиска можно решать многими способами, из них простейший,
// это прямой поиск. Прямым он называется потому, что условие отбора просто
// примеряется по очереди ко всем элементам массива, и для тех, кто
// соответствует условию, выполняется какое-то действие, например, печать.
// Для примера найдем все числа, кратные пяти.
// 1. Поиск всех вхождений значения.
void Find_all (int mas[], int len)
{ // Поиск всех вхождений чисел, кратных пяти.
    for (int i = 0; i < len; i++)
        if (mas[i] % 5 == 0)
            // Здесь с этим элементом можно что-то сделать, например, вывести.
            printf ("Кратен пяти элемент с номером %d", i)
}
// Замечание. В случае, когда в массиве нет ни одного искомого значения,
// функцией будет выполнено len сравнений, но ни одной печати, соответственно,
// никакого сообщения пользователь не получит.
// 2. Поиск первого вхождения значения. Особенностью является то, что при
// первом же удачном сравнении алгоритм должен закончить свою работу.
// Но если нужного элемента не окажется вообще, выход из функции произойдет
// при завершении цикла, и результатом должно быть значение, отличное от
// любого номера элемента массива, например, отрицательное, –1.
int Find_first (int mas[], int len)
{
    // Поиск первого вхождения числа, кратного пяти.
    for (int i = 0; i < len; i++)
        if (mas[i] % 5 == 0)
            return i;    // Прерывание при найденном значении.
                        // Функция вернет номер вхождения.
    return –1;          // Неудачный поиск, функция вернет –1.
}
// При обращении к функции следует учесть, что значение –1, возвращенное
// функцией, означает неудачный поиск.
// Num = Find_first (A, 5);

```

```

// if (Num >= 0)
//     printf ("Номер %d\n", Num);
//     else
//     printf ("Элемент не найден");

// Пример №5. Нахождение суммы элементов массива.
// Функция возвращает одно значение, поэтому ее тип int. В этой задаче приведем
// пример использования различных способов адресации элементов массива,
// прямую и косвенную. Выбор способы адресации не зависит от способа
// передачи в функцию параметра – массива.
// 1. Прямая адресация. Обращение к элементам массива через операцию [].
int Sum1 (int *mas, int len)
{
    int i;
    int Sum = 0;
    for (i = 0; i < len; i++)
        Sum += mas[i];
    return Sum;
}

// 2. Косвенная адресация. Обращение к элементам массива через операцию *,
// примененную к указателю (на очередной элемент массива).
int Sum2 (int *mas, int len)
{
    int * ip;
    int Sum = 0;
    for (ip = mas; ip < mas+len; ip++)
        Sum += *ip;
    return Sum;
}

// 3. Косвенная адресация. Обращение к элементам массива через операцию *
// и смещение указателя относительно начала массива.
int Sum3 (int * mas, int len)
{
    int Sum;
    for (int i = 0, Sum = 0; i < len; i++)
        Sum += *(mas+i);
    return Sum;
}

// Пример №6. Вывод на экран номеров и адресов элементов массива
// иллюстрирует отличие способов адресации. Для вывода адреса используется
// спецификатор формата %p.
void Out (int *mas, int len)
{
    int i; *ip;

```

```

    for (ip = mas, i=0; ip < mas + len; ip++, i++)
        printf("Номер %d Адрес %p\n", i++, ip);
}

// Пример №7. Функция поиска наибольшего (наименьшего) элемента массива.
// Алгоритм решения следующий: введем переменную, обозначающую
// максимальное значение, и сначала предположим, что наибольшим (или
// наименьшим) является первый элемент массива. Далее в цикле все
// последующие значения сравним с максимальным (минимальным) и, если
// найдется такое значение, что больше максимума (меньше минимума), то
// значение максимального (минимального) заменяется значением найденного
// элемента массива.
// 1. Использование прямой адресации.
int Max1 (int mas[], int len)
{
    int      i, max;                // Наибольшее значение max.
    max = mas[0];                  // Пусть первое наибольшее.
    for (i = 0; i < len; i++)
        if (mas[i] > max) max = mas[i]; // Если найден элемент (mas[i]),
                                        // больший, чем тот, которого
                                        // считали максимальным (max),
                                        // то запоминаем его значение.
    return max;                    // Возвращаем значение наибольшего.
}

// 2. Использование косвенной адресации.
int Max2 (int mas[], int len)
{
    int      *ip;                  // Указатель на элемент массива.
    int      *imax;                // Указатель на наибольшее значение
    imax = mas;                    // Запоминаем адрес максимального.
    for (ip = mas+1; ip < mas + len; ip++)
        if (*ip > *imax) imax = ip; // Если найден элемент (*ip),
                                        // больший, чем тот, которого
                                        // считали максимальным (*imax),
                                        // то запоминаем адрес.
    return imax - mas;              // Возвращаем номер наибольшего, вычисленный как
                                    // разность адресов найденного и начального.
}

// Пример №8. Функция удаления элемента из массива. Для определенности
// удалим все отрицательные элементы массива. Удаление элемента заключается в
// сдвиге части массива влево на одну позицию, начиная с удаляемого значения.
// Длина массива уменьшается на 1 при каждом сдвиге, поэтому длина массива
// возвращается по адресу. В этом алгоритме объединены поиск (пример №3)
// и удаление, причем цикл удаления внутренний, так как для каждого найденного

```

// значения выполняется сдвиг оставшейся части массива влево на одну позицию.  
 // Управление циклом выполняет оператор for, из которого изъято приращение  
 // параметра цикла. Это необходимо, так как приращение номера элемента должно  
 // происходить лишь тогда, когда удаления не было. Это позволит избежать  
 // ошибки, когда удаляемые элементы будут идти подряд.

```
void Del (int mas[], int &len)
{
  // Поиск отрицательных элементов выполняется в цикле с рабочей переменной i.
  // Если найден отрицательный элемент, его номер будет i.
  int i; // Рабочая переменная для поиска.
  int j; // Рабочая переменная для удаления.
  for (i = 0; i < len; ) // Приращение изъято.
  {
    if (mas[i] < 0) // Элемент должен быть удален.
    {
      for (j = i; j < len - 1; j++)
        mas[j] = mas[j+1];
      len --; // Длина стала меньше по завершении цикла.
    }
    else
      i ++; // Переход к следующему поиску, только
  } // если не было удаления .
}
```

**Пример №9.** Элементы массива можно менять местами.  
 // Например, переставим их по кругу, это называется циклический сдвиг.  
 // Последний элемент не упадет за край массива, а станет первым.

```
void Cyrclе_Mas (int mas [], int len)
{
  // Переменная tmp запомнит значение последнего элемента.
  int tmp = mas[len - 1];
  // Сдвиг массива вправо, начиная с первого элемента, до len-1.
  for (int i = len-1; i > 0; i--)
    mas[i] = mas[i-1];
  mas[0] = tmp; // Запомненный элемент записывается вперед.
}
```

**Пример №10.** Функция формирования нового массива.  
 // Получим массив, в котором содержатся только положительные элементы  
 // исходного массива. Все необходимые данные должны быть объявлены вне тела  
 // функции, и должны быть известны вызывающей программе. Длина нового  
 // массива и его значения получены функцией и возвращаются в вызывающую  
 // программу.

```
int New_mas (int mas[],int len, int *mas_new, int &len_new)
{
```

```

len_new = 0;          // В новом массиве нет ни одного значения. Переменная
                      // len_new индексирует новый массив, а по завершению
                      // работы будет знать число записанных элементов.
// Прямой поиск положительных элементов:
for (int i = 0; i < len; i++)
    if ( mas[i] > 0 )    // Найден положительный элемент:
                        // запись его в новый массив.
                        // Индексация в массивах различна.
        mas_new [len_new++] = mas[i];
// По завершении цикла поиска определена длина нового массива, это переменная
// len_new. Она может оказаться равной 0, поэтому имеет смысл возвращать
// значение длины нового массива. Именно поэтому функция имеет тип int, что
// дает ей статус функции, возвращающей логическое значение.
return len_new;
}
// При обращении к этой функции можно выполнить проверку: если функция
// вернула 0, новый массив имеет нулевую длину, то есть, не сформирован.
// if ( New_mas (Array_old, 10, Array_new, len) != 0)
//     print_mas(Array_new,len);
// В противном случае любые действия с новым массивом бессмысленны.
// Пример №11. Функция может работать с данными базовых типов, а при
// обращении ей передаются элементы массива. Например, есть функция, которая
// находит расстояние между двумя точками на плоскости.
float R(float x1, float y1, float x2, float y2)
{
    return sqrt (pow (x1-x2, 2.)+pow (y1-y2 ,2.));
}

```

Если существует множество абстрактных функций для работы с массивом, то роль вызывающей (главной) программы заключается в том, чтобы:

- 1) подготовить данные, которые должны быть обработаны;
- 2) управлять вызовами функций их обработки.

Любой функции все равно, как объявлен и получил значения массив, с которым он работает, важно только, чтобы массив существовал и получил значения до момента, когда он будет передан в функцию (до момента обращения). Способов подготовки фактических данных для функций много. Можно объявить массив как статический или как динамический. Можно присвоить значения элементам массива любым способом: инициализировать, ввести, получить случайно, вычислить и прочие. Функции для решения многих из этих задач описаны выше.

В тексте примера будут только объявления переменных, их инициализация, и обращения к функциям. Здесь мы покажем, как главная программа управляет вызовами функций.

```

#include <stdio.h>
#include "Task.hpp"

```

// Включен текст заголовочного файла.

```

#define N 10 // Это объявление наибольшей длины массива.
void main(void)
{
int a[] = {1, -2, 3, -4, 5, -6}; // Объявление и инициализация массива a.
// Длина массива определена длиной списка инициализации.
int na = sizeof(a) / sizeof(int);
// Чтобы увидеть размещение в памяти элементов массива, выведем его на печать:
Print_mas (a, na);
// Изменим значения элементов массива:
Transform_mas (a, na);
// Покажем, что они изменились:
Print_mas (a, na);
// Массив условно переменной длины.
int b[N]; // Выделена память для N элементов массива b.
int nb = 0; // Реальная длина массива b < N.
printf ("Длина массива должна быть < %d", N);
// Введем его значения.
Input_mas (b, nb); // Длина будет определена при вводе.
// Найдем сумму элементов массива тремя функциями вычисления суммы,
// чтобы показать, что способы адресации в массиве могут быть различны,
// а результат один и тот же.
printf ("Прямая адресация: Sum1 = %d\n", Sum1 (b, nb) );
printf ("Косвенная адресация: Sum2 = %d\n", Sum2 (b, nb) );
printf ("Косвенная адресация: Sum3 = %d\n", Sum3 (b, nb) );
// Покажем сходство и различие механизмов адресации:
Out (b, nb);
// Найдем наибольший элемент массива b:
printf ("Номер = %d, Значение = %d\n", Max1 (b, nb), b[Max1 (b,nb)]);
// Удалим из массива a отрицательные элементы.
Del (a, na);
// Покажем, что осталось.
printf ("После удаления:\n");
Print_mas (a, na); // Длина массива уменьшилась.
// Выполним циклический сдвиг в массиве b:
Circle_Mas (b, nb);
printf ("После сдвига:\n");
Print_mas (b, nb);
// Циклический сдвиг можно выполнить много раз, если обратиться к функции в
// цикле, например, трижды:
for (int C = 1; C<=3 ; C++) // C – обычный счетчик.
Circle_Mas (b, nb);
printf ("После сдвига:\n");
Print_mas (b, nb);
// Объявим новый массив и проинициализируем его.

```



```

int      d [5] = {-1, 1, -2, 2, -3};
// Объявим новый массив, длина которого не более 5-ти.
int      c[5];
int      nc;                                // Реальная длина нового массива.
// Получим новый массив, в котором только положительные элементы массива d.
// Данные нового массива и его длина будут получены функцией и возвращены
// в вызывающую программу. Функция дважды возвращает длину, через имя и
// через список параметров,
    nc = New_mas (d, 5, c, nc);
    if (nc != 0)                            // Длина нового массива больше 0,
                                            // значит, массив получен.
    {
        printf ("Новый массив:\n");
        print_mas (c, nc);
    }
    else
        printf ("Массива нет\n"); // Массив не сформирован.
// Обращение к функции с элементами массива.
// Два массива определяют значения координат точек на плоскости.
float     x[3] = {1., 2., 3.};
float     y[3] = {3., 2., 1.};
printf ("С элементами массива обращаемся к функции,\n
        чтобы найти расстояние от точек до начала координат\n");
    for (int i = 0; i < 3; i++)
    { // При обращении к функции ей передается только одна точка.
        printf ("%6.2f %6.2f, R = %6.2f\n", x[i], y[i], R(0, 0, x[i], y[i]) );
    }
} // End of main

```

### Варианты заданий

Предлагаемые задания имеют основную тему «обработка массивов». Не менее важной является тема «использование механизма функций», рассмотренная ранее. Функции обработки массивов имеют особенности, связанные с передачей данных в функцию и возвращением их оттуда. Не следует писать просто программу обработки массива, а затем «переписывать ее на функцию», это порочный путь. Следует воспитывать в себе алгоритмическое мышление, научаясь видеть абстрактный алгоритм в конкретной задаче, научаясь выделять главное и отсекал детали, а также совершенствуя свой навык многократным использованием одних и тех же механизмов.

В каждом задании предлагается три задачи. В любой из них предполагается, что функция обрабатывает абстрактный массив, для которого известны лишь тип элементов и их количество (длина массива).

В первой задаче нужно объявить массив и проинициализировать его в главной программе. Инициализация, это самый простой путь получения массива, она позволяет легко протестировать задачу изменением данных прямо в тексте

программы. Алгоритмами обработки массива являются простые алгоритмы сканирования его элементов, или проход по массиву от начала до конца. Входным данным для функции обработки массива является имя массива и его длина. Результатом обработки является одно значение, которое и возвращает функция.

Во второй задаче нужно объявить массив в главной программе. Главная программа будет управлять процессом обработки массива путем определения последовательности вызовов функций. Для увеличения функциональности массива предлагается написать и использовать функции ввода и вывода на экран элементов массива, а также функцию или функции его обработки, особенностью которых во многих задачах будет изменение длины массива, выполняемое функцией. Поскольку длина массива всегда передается в функцию как параметр, следует всего лишь не забыть передать ее по адресу, а не по значению, для чего в заголовке функции приписать знак & к имени параметра, например,

```
void (int mas, int &len)
{
    ...
    len = ...;
}
```

В третьей задаче функциональность массива увеличивается добавлением функции случайной генерации элементов массива. Задачей функции обработки является получение нового массива. При решении этой задачи все реальные массивы объявляет главная программа, и она же управляет процессом получения, обработки и вывода данных. Функция должна получить в качестве параметров все входные данные и указатели на переменные, которые будут результатом обработки.

Задание 1.

1. Проинициализировать массив. Описать функцию, которая найдет количество отрицательных элементов массива. Описать функцию, которая найдет значение наибольшего из отрицательных элементов.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит из массива наибольший из отрицательных элементов.

3. Описать функцию случайной генерации элементов массива. Описать функцию для нахождения среднего арифметического элементов массива. Описать функцию, которая получит в новом массиве все значения, меньшие, чем среднее арифметическое своих соседей. Использовать механизм указателей.

Задание 2.

1. Проинициализировать массив. Описать функцию, которая найдет количество элементов массива, имеющих четные порядковые номера, но являющихся нечетными числами. Описать функцию, которая найдет наибольшее четное значение.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит повторяющиеся элементы массива.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая получит в новом массиве только те элементы исходного, которые имеют четные порядковые номера, но являются нечетными числами.

Использовать механизм указателей.

Задание 3.

1. Проинициализировать два массива произвольной длины, в каждом из которых нет повторяющихся значений. Описать логическую функцию, которая проверит, нет ли в массиве повторяющихся элементов. Описать функцию, которая найдет количество элементов, которые одинаковы в двух массивах.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая определит, является ли массив упорядоченным по возрастанию. Описать логическую функцию, которая для массива и некоторого заданного  $N$  определит, сохранится ли упорядоченность, если  $N$  «приписать» в конец массива. Если это возможно, функция добавит число в конец массива, и увеличит его длину.

3. Описать функцию случайной генерации элементов массива  $Y$ , в котором могут быть как положительные, так и отрицательные значения. Описать функцию получения нового массива  $Z$  по следующему закону:  $Z_i = Y_i$ , если  $Y_i$  по модулю больше 1, и  $Z_i = 1$  в противном случае. Найти и вернуть число единиц в новом массиве. Использовать механизм указателей.

Задание 4.

1. Проинициализировать массив. Описать функцию, которая найдет количество элементов массива, принадлежащих интервалам  $[-2, -5]$  или  $[2, 5]$ . Описать функцию, которая найдет сумму элементов, не входящих в указанные интервалы.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая проверит, является ли массив упорядоченным по возрастанию. Описать функцию, которая выполнит вставку в упорядоченный массив некоторого произвольного значения таким образом, чтобы упорядоченность не была нарушена.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая сохранит в новом массиве только те элементы исходного массива, которые принадлежат интервалам  $[-2, -5]$  или  $[2, 5]$ . Использовать механизм указателей.

Задание 5.

1. Проинициализировать массив. Описать две функции для нахождения номеров наибольшего и наименьшего элементов массива.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит из массива наибольший и наименьший элементы.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая выполнит разделение массива на два – массив положительных значений и массив отрицательных значений. Использовать механизм указателей.

Задание 6.

1. Проинициализировать два массива, которые задают  $n$  точек координатами  $(X, Y)$  в двумерном пространстве. Описать функцию, которая находит расстояние между двумя произвольными точками. Описать функцию, которая найдет расстояние между всеми точками и выведет их на экран в виде таблицы. Описать функцию, которая найдет наибольшее расстояние.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая выполнить сжатие массива (удаление всех чисел, меньших нуля).

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая в новом массиве получит только положительные значения элементов исходного массива. Использовать механизм указателей.

#### Задание 7.

1. Проинициализировать два массива, которые задают  $n$  точек координатами  $(X, Y)$  в декартовой системе координат. Описать функцию перевода декартовых координат точки в полярные (углы вычислять с точностью до градусов). Для всех точек перевести декартовы координаты в полярные, сохранить в массивах. Описать функцию поиска наибольшего значения. Найти номер точки, имеющей наибольший радиус-вектор, и номер точки, имеющей наибольший угол. Вывести значение декартовых и полярных координат этих точек.

2. Описать функции ввода и вывода элементов массива. Описать функции поиска и удаления из массива произвольных цепочек чисел  $(M1, M2, M3)$ .

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая находит в массиве все элементы, значения которых принадлежат некоторому указанному диапазону  $[M1; M2]$ , и формирует из них новый массив. Использовать механизм указателей.

#### Задание 8.

1. Проинициализировать массив. Описать функцию, которая найдет среднее арифметическое элементов массива. Описать функцию, которая найдет номер элемента массива, ближайшего к среднему арифметическому элементов массива.

2. Даны координаты и массы  $N$  материальных точек, расположенных на прямой. Описать функции ввода и вывода  $N$  точек. Описать функцию, которая найдет координату центра тяжести системы. Описать функцию, которая найдет номер точки, наиболее близко расположенной к центру тяжести. Описать функцию, которая удалит точку, наиболее близко расположенную к центру тяжести.

3. Описать функцию случайной генерации элементов массива. Описать функцию, один из параметров которой символьная переменная «знак», принимающая значения '>' или '<'. Функция переписет в новый массив только те элементы исходного, которые больше среднего арифметического, если знак '>', и те, которые меньше, если знак '<'. Использовать механизм указателей.

#### Задание 9.

1. Проинициализировать массив. Описать две функции, которые определяют, являются ли элементы массива упорядоченными по возрастанию, и являются ли элементы массива упорядоченными по убыванию. Описать функцию, которая одна выполняет полную проверку. Если порядок возрастания, функция вернет 1, если убывания, -1, если нет порядка, 0.

2. Даны координаты  $N$  материальных точек расположенных на прямой, все одной массы  $M$ . Описать функции ввода и вывода точек. Описать функцию, которая найдет координату центра тяжести системы. Описать функцию, которая

найдет номер точки, наиболее удаленной от центра тяжести. Описать функцию, которая удалит эту точку.

3. Описать функцию случайной генерации элементов массива. Описать функцию, формирующую новый массив, с параметром «знак». Если знак '+', то функция формирует новый массив, в котором записаны только положительные элементы исходного массива, если '-', то функция формирует новый массив, в котором записаны только отрицательные элементы исходного массива. Использовать механизм указателей.

#### Задание 10.

1. Проинициализировать массив. Описать функции, которые найдут число положительных и число отрицательных элементов массива. Описать функцию преобразования массива, которая возводит в квадрат все отрицательные элементы массива, а из положительных элементов извлекает квадратный корень.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит из массива все элементы, значения которых по модулю больше некоторого заданного  $N$ .

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая выполнит разделение массива на два – массив четных чисел и массив нечетных чисел. Использовать механизм указателей.

#### Задание 11.

1. Проинициализировать массив, в котором могут быть положительные и отрицательные числа. Описать функцию, которая найдет количество элементов массива до первого отрицательного, и функцию, которая найдет сумму элементов после первого отрицательного. Описать функцию, которая решит обе эти задачи.

2. Даны координаты  $n$  точек на координатной плоскости массивами координат. Описать функции ввода и вывода  $n$  точек. Описать функцию, которая оставит в массивах только те точки, которые принадлежат полосе, заданной системой неравенств:

$$\begin{cases} y \geq -1 \\ y \leq +1 \end{cases}, \text{ а остальные удалит.}$$

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая в новом массиве получит все значения исходного массива, которые находятся после первого отрицательного элемента. Использовать механизм указателей.

#### Задание 12.

1. Проинициализировать массив, в котором могут быть как положительные, так и отрицательные числа. Описать функцию, которая проверит, являются ли элементы массива упорядоченными по возрастанию. Описать функцию сортировки массива по возрастанию методом пузырька. Алгоритм заключается в следующем: массив просматривается по перекрещивающимся парам чисел  $(a_i, a_{i+1})$ . Если  $a_i > a_{i+1}$ , они меняются местами. Перестановки подсчитываются. Алгоритм завершает работу, если при просмотре массива нет ни одной перестановки. Сортировка выполняется в том же массиве.

2. Даны координаты  $n$  точек на координатной плоскости массивами координат  $(X, Y)$ . Описать функции ввода и вывода массивов, определяющих точки. Описать функцию, которая найдет координаты центра системы и добавит найденные значения в конец массивов координат.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая формирует новый массив, содержащий только положительные элементы исходного массива. Описать функцию, которая формирует новый массив, содержащий только отрицательные элементы исходного массива. Использовать механизм указателей.

#### Задание 13.

1. Дан массив произвольной длины, определяющий коэффициенты многочлена степени  $N$ . Проинициализировать исходный массив. Описать функцию, которая определит, не имеет ли нулевое значение коэффициент при наивысшей степени. Описать функцию, которая найдет значение наибольшего элемента массива. Определить, при какой степени коэффициент наибольший. Описать функцию, которая найдет значение многочлена в произвольной точке  $x$ .

2. Описать функции ввода и вывода массива. Описать функцию выравнивания элементов массива, которое заключается в удалении из массива всех элементов, по модулю больших некоторого  $M$ .

3. Описать функцию случайной генерации элементов массива. Задать массив произвольной длины, определяющий коэффициенты многочлена степени  $N$ . Описать функцию, которая вычислит и сохранит в новом массиве коэффициенты многочлена, являющегося его производной первой степени. Используя ее, найти производную  $n$ -й степени. Использовать механизм указателей.

#### Задание 14.

1. Проинициализировать массив. Описать функции поиска наименьшего, наибольшего элементов массива и функцию выравнивания массива (замена нулем минимального и максимального его элементов).

2. Описать функции ввода и вывода массива. Описать функцию, которая проверит, является ли массив упорядоченным по убыванию. Описать функцию, которая в упорядоченный по убыванию массив включит некоторый элемент  $b$  с сохранением упорядоченности.

3. Описать функцию случайной генерации элементов массива. Локальным минимумом называется любой элемент, который меньше своих соседей. Описать функцию, которая найдет все локальные минимумы, с записью их в новый массив. Использовать механизм указателей.

#### Задание 15.

1. Проинициализировать массив. Описать функцию, которая получит сумму нечетных чисел данного массива. Описать функцию, которая получит сумму отрицательных чисел данного массива. Описать функцию, которая получит сумму тех чисел данного массива, которые нечетны и отрицательны.

2. Описать функции ввода и вывода массива. Описать функцию, которая определит, является ли некоторое число простым. Описать функцию, которая удалит из массива все простые числа.

3. Описать функцию случайной генерации элементов массива. Получить два массива одинаковой длины. Описать функцию, которая сложит или вычитет массивы поэлементно с записью в новый массив. Операция (знак '+' или '-') передается в функцию как параметр. Использовать механизм указателей.

Задание 16.

1. Проинициализировать массив. Описать функцию, которая определит, являются ли палиндромом элементы массива. Описать функцию, которая расположит его элементы в обратном порядке, не используя вспомогательный массив.

2. Описать функции ввода и вывода массива. Описать функцию нахождения среднего арифметического элементов массива. Описать функцию, которая добавит найденное среднее значение в конец массива.

3. Описать функцию случайной генерации элементов массива. Описать функцию копирования массива. Использовать механизм указателей.

Задание 17.

1. Проинициализировать массив. Описать функцию, которая найдет наибольшее из нечетных по значению чисел. Описать функцию, которая найдет наименьшее из четных чисел. Описать функцию, которая одновременно решит обе эти задачи.

2. Описать функции ввода и вывода массива. Описать функцию, которая определит, является ли массив упорядоченным, например, по возрастанию. Описать функцию, которая выполнит вставку в упорядоченный массив некоторого произвольного значения таким образом, чтобы упорядоченность не была нарушена.

3. Описать функцию случайной генерации элементов массива. Получить массив  $B$ . Описать функцию, которая сольет некоторый упорядоченный массив  $A$  со случайным массивом  $B$ , многократно выполняя вставки элементов из массива  $B$  в массив  $A$ . Использовать механизм указателей.

Задание 18.

1. Проинициализировать массив. Описать функцию поиска наибольшего значения. Описать функцию сортировки массива по возрастанию на основе алгоритма обмена. Путем просмотра отыскивается максимальное значение и меняется местами с последним элементом, затем этой же операции подвергается оставшаяся часть массива кроме последнего элемента, затем кроме двух последних и т.д. всего  $(n-1)$  раз, где  $n$  – число элементов массива. Исходные данные не сохранять, т.е. выполнить сортировку в том же массиве.

2. Описать функции ввода и вывода массива. Описать функцию преобразования массива, которая удаляет из него все отрицательные и равные нулю значения.

3. Описать функцию случайной генерации элементов массива. Пусть два массива произвольной длины хранят коэффициенты двух многочленов степеней  $N$  и  $M$ . Описать функцию сложения (вычитания) многочленов, получающую значения коэффициентов нового многочлена в новом массиве. Характер операции (знак '+' или '-') задать как параметр функции. Использовать механизм

указателей.

#### Задание 19.

1. Проинициализировать массив. Описать функцию, которая определит, образуют ли элементы массива арифметическую прогрессию. Если да, вернуть знаменатель прогрессии, иначе 0. Описать функцию, которая определит, образуют ли элементы массива геометрическую прогрессию. Если да, вернуть знаменатель прогрессии, иначе 1.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит из массива первый и последний элементы.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая получит в новом массиве упорядоченный по возрастанию исходный массив. Использовать механизм указателей.

#### Задание 20.

1. Проинициализировать массив. Описать функцию, которая найдет количество четных элементов массива. Описать функцию, которая найдет количество нечетных элементов массива. Описать функцию, которая одновременно решит обе задачи.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит из массива те значения, которые повторяются.

3. Описать функцию случайной генерации элементов массива. Описать функцию нахождения среднего арифметического элементов массива. Описать функцию, которая получит в новом массиве те значения исходного, которые отличаются от среднего арифметического элементов массива не более чем на 1. Использовать механизм указателей.

#### Задание 21.

1. Проинициализировать массив. Описать функцию, которая определит, является ли массив знакопеременным. Описать функцию, которая найдет число смен знака в массиве.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая определит, является ли массив упорядоченным по возрастанию. Описать функцию, которая объединит два упорядоченных массива в третий, тоже упорядоченный, используя механизм вставок очередного элемента второго массива в подходящее место первого.

3. Описать функцию случайной генерации элементов массива. Использовать ее, чтобы задать в виде двух массивов произвольной длины координаты  $n$  точек на плоскости. Описать функцию вычисления расстояния между двумя точками. Использовать ее, чтобы получить в новом массиве номера всех точек, принадлежащих кругу радиуса  $R$  с центром в начале координат. Использовать механизм указателей.

#### Задание 22.

1. Проинициализировать в виде двух массивов произвольной длины координаты  $n$  точек на плоскости  $(X, Y)$ . Описать функцию вычисления расстояния между двумя точками. Описать функцию, которая найдет суммарное расстояние в порядке обхода от первой точки до последней.



2. Описать функции ввода и вывода элементов массива. Описать функцию, которая проверит, находится ли точка с координатами  $(x, y)$  в диапазоне значений  $|x| < R$  и  $|y| < R$ . Описать функцию, которая выполнит для  $n$  точек на плоскости удаление из массивов координат тех точек, что не принадлежат области.

3. Описать функцию случайной генерации элементов массива. Задать случайно координаты  $n$  точек на плоскости. Описать функцию, которая для  $n$  точек сформирует массив номеров тех точек, радиус-вектор которых меньше некоторого заданного  $R$ . Использовать механизм указателей.

Задание 23.

1. Проинициализировать массив. Описать функцию для нахождения первого положительного элемента массива, и функцию для нахождения последнего отрицательного элемента массива.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит из массива первый положительный и первый отрицательный элементы.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая выполнит слияние двух массивов одинаковой длины в третьем (элементы первого и второго массивов в нем чередуются). Использовать механизм указателей.

Задание 24.

1. Проинициализировать массив. Описать логическую функцию, которая найдет, есть ли в массиве хотя бы одно нулевое значение. Описать функцию, которая найдет общее число элементов до первого нулевого и после последнего нулевого. Если ни одного нулевого элемента нет, функция вернет  $-1$ . Если нулевыми являются первый элемент и последний, функция возвращает  $0$ , иначе значение, отличное от  $0$ .

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит из массива все значения до первого нулевого и после последнего нулевого.

3. Описать функцию случайной генерации элементов массива. Описать логическую функцию, которая в новом массиве получит все значения исходного, которые находятся от первого нулевого значения до последнего нулевого значения. Использовать механизм указателей.

Задание 25.

1. Проинициализировать два массива, которые задают  $n$  точек координатами  $(X, Y)$  в декартовой системе координат. Описать логическую функцию, которая проверит, принадлежит ли точка с координатами  $(x, y)$  кругу радиуса  $R$  с центром в начале координат. Описать функцию, которая для  $n$  точек определит, какая из них принадлежит указанной области и выведет в таблицу.

2. Описать функции ввода и вывода элементов массивов как координат точек. Описать функцию, которая удалит из массивов все точки, не принадлежащие указанной области.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая находит в массиве все элементы, значения которых не

принадлежат некоторому указанному диапазону  $[A;B]$ , и формирует из них новый массив. Использовать механизм указателей.

#### Задание 26.

1. Проинициализировать массив. Описать функцию, которая найдет наименьший элемент массива. Описать функцию, которая найдет наибольший элемент массива. Описать функцию, которая переменит местами наименьшее и наибольшее значения соответственно с первым и последним элементами массива.

2. Описать функции ввода и вывода массивов. Описать функцию, которая определит, является ли число четным. Описать функцию, которая удалит из массива все четные числа.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая определит, является ли число простым. Описать функцию, которая перепишет в новый массив только те элементы исходного, которые являются простыми числами. Использовать механизм указателей.

#### Задание 27.

1. Проинициализировать массив. Описать функцию, которая определит, являются ли элементы массива арифметической прогрессией. Если прогрессия возрастающая, функция вернет 1, если убывающая,  $-1$ , если не прогрессия, 0.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удаляет из массива все нулевые значения. Описать функцию, которая меняет местами элементы массива, передвигая в начало все положительные элементы, а в конец все отрицательные, выполняя перестановки в том же массиве.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая по заданным значениям первого элемента и знаменателя, генерирует арифметическую прогрессию длиной  $n$ . Использовать механизм указателей.

#### Задание 28.

1. Проинициализировать массив. Описать функции поиска наибольшего и наименьшего элементов массива (возвращать номера). Описать функцию, которая найдет, сколько элементов массива находятся в промежутке между номером наибольшего и номером наименьшего элементов массива.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит из массива все элементы до первого экстремума и после последнего.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая получит в новом массиве все элементы исходного, которые находятся в промежутке между двумя экстремумами. Использовать механизм указателей.

#### Задание 29.

1. Проинициализировать массив, в котором могут быть как положительные, так и отрицательные числа. Описать логическую функцию, которая найдет, есть ли в массиве хотя бы один отрицательный элемент. Описать функцию, которая найдет количество элементов массива до первого отрицательного. Описать функцию, которая найдет сумму элементов массива до первого отрицательного.

2. Описать функции ввода и вывода элементов массива. Описать функцию, которая удалит из массива все элементы до первого отрицательного.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая в новом массиве получит все значения исходного массива, которые находятся после первого отрицательного элемента. Использовать механизм указателей.

Задание 30.

1. Проинициализировать массив. Описать функцию, которая выполнит сглаживание элементов массива (сглаживание, это замена каждого числа значением среднего арифметического трех стоящих рядом элементов). Описать функцию, которая найдет среднее арифметическое элементов массива.

2. Описать функции ввода и вывода массива. Описать функцию, которая заменит нулями наименьшее и наибольшее значения элементов массива. Дописать в конец массива значение среднего арифметического.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая формирует массив отклонений для каждого элемента исходного массива от среднего арифметического. Использовать механизм указателей.

Задание 31.

1. Дан массив произвольной длины, определяющий коэффициенты многочлена степени  $N$ . Проинициализировать исходный массив. Описать функцию, которая найдет значение многочлена в произвольной точке  $x$ . Описать функцию, которая найдет значение первой производной многочлена в произвольной точке  $x$ .

2. Описать функции ввода и вывода массива. Описать функцию, которая найдет коэффициенты первой производной многочлена с сохранением в том же массиве.

3. Описать функцию случайной генерации элементов массива. Пусть два массива произвольной длины хранят коэффициенты двух многочленов степеней  $N$  и  $M$ . Описать функцию сложения многочленов, которая получит значения коэффициентов нового многочлена в новом массиве. Описать функцию вычитания многочленов, которая получит значения коэффициентов нового многочлена в новом массиве. Использовать механизм указателей.

Задание 32.

1. Проинициализировать два массива. Описать функцию, которая определит, есть ли в массиве одинаковые элементы. Описать функцию, которая определит, образуют ли элементы массива монотонную возрастающую последовательность. Описать функцию, которая определит, образуют ли элементы массива монотонную убывающую последовательность.

2. Описать функции ввода и вывода массива. Описать функцию, которая удалит из массива все повторяющиеся значения.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая объединит два упорядоченных массива в один, тоже упорядоченный. Использовать механизм указателей.

Задание 33.

1. Проинициализировать массив. Описать функцию, которая выполнит циклический сдвиг влево на одну позицию. Описать функцию, которая выполнит циклический сдвиг вправо на одну позицию. Описать функцию, которая может выполнить циклический сдвиг влево или вправо, в зависимости от значения дополнительного параметра (знак '>' или '<').

2. Описать функции ввода и вывода массива. Описать функцию, которая удалит из массива все повторяющиеся значения.

3. Описать функцию случайной генерации элементов массива. Получить два массива произвольной длины. Описать функцию, которая сольет два массива в третьем, исключая одинаковые элементы. Использовать механизм указателей.

Задание 34.

1. Массивы хранят данные об измерениях размеров деталей цилиндрической формы (диаметр и высота) с точностью три знака. Проинициализировать массивы. Описать функцию, которая найдет объем цилиндра. Описать функцию, которая найдет поверхность цилиндра. Описать функцию, которая найдет объемы и поверхности всех цилиндров с сохранением результатов.

2. Описать функцию, которая найдет в массиве все вхождения некоторого числа  $N$ , и удалит их из массива.

3. Описать функцию случайной генерации элементов массива. Для массива натуральных чисел размером  $n$  описать функцию, которая запишет в новый массив все элементы исходного массива, которые кратны трем и не кратны пяти. Использовать механизм указателей.

Задание 35.

1. Проинициализировать массив действительных чисел длиной  $N$ . Эти числа образуют  $N/2$  интервалов числовой оси:  $(A_1, A_2), \dots, (A_{N-1}, A_N)$ . Описать функцию логического типа, определяющую, имеют ли эти интервалы общие точки. Описать функцию, которая найдет общий интервал, если он есть.

2. Описать функции ввода и вывода массива. Описать функцию, которая удалит из массива элемент с указанным номером.

3. Описать функцию случайной генерации элементов массива. Описать функцию, которая формирует новый массив включением в него только тех чисел исходного, которые по модулю не превышают некоторого заданного значения  $N$ . Использовать механизм указателей.

## **Тема 8. Работа с одномерными массивами. Использование массивов в решении содержательных задач**

В задачах этого раздела основная трудность заключается в том, что задания не формализованы. Следовательно, предваряет кодирование процесс формализации, в который должны входить:

- 1) выбор математической модели;
- 2) выбор способа представления данных;
- 3) функциональная декомпозиция;
- 4) кодирование и отладка алгоритмов обработки данных.

В качестве рекомендации напомним тему раздела: использование массивов,

значит, способ представления данных, это массив (массивы). Алгоритмов обработки массивов в природе не так и много, большинство из них рассмотрены в примерах предыдущего раздела, следовательно, нужно лишь выбрать подходящий алгоритм, и, возможно, модифицировать его для конкретной задачи. Следует активно пользоваться подходящими функциями, написанными ранее, такими как ввод и вывод элементов массива.

### **Варианты заданий**

Задание 1. Сорок разбойников сдали экзамен по охране окружающей среды. Использовать функции обработки массивов, чтобы найти, сколько разбойников будут охранять среду отлично, сколько хорошо, а сколько посредственно.

Задание 2. Для каждой детали цилиндрической формы проводят три измерения: диаметр нижнего основания, диаметр центра и диаметр верхнего основания с точностью три знака. Данные сохранены в массиве. Бракованными считаются детали, у которых конусность или бочковатость более 3%. Пометить бракованные детали. Определить процент бракованных деталей в партии.

Задание 3. Дети встают по кругу и начинают считалку, в которой выбывает  $n$ -й ребенок, после чего круг смыкается. Считалка повторяется, пока не останется один ребенок. Использовать функции обработки массивов, чтобы узнать, кто останется.

Задание 4. Дневная и ночная температура воздуха измеряются ежедневно и записываются в таблицу. Когда среднесуточная температура в течение трех дней подряд ниже 8 градусов Цельсия, начинается отопительный сезон. Использовать функции обработки массивов, чтобы определить, какого числа сезон был начат в этом году, если известно, что это произошло в текущем месяце.

Задание 5. В таблице хранятся данные о расходе электроэнергии в школе ежемесячно в течение года. Использовать функции обработки массивов, чтобы узнать средний расход электроэнергии, минимальный и максимальный расходы, а также узнать, на сколько процентов отличаются минимальный и максимальный расходы от среднемесячного.

Задание 6. На метеостанции в компьютер введены сведения о среднесуточной температуре за март. Использовать функции обработки массивов, чтобы найти:

- среднюю температуру месяца;
- день, когда температура ближе всего подходила к среднемесячной.

Задание 7. Банк провел мониторинг с целью улучшения обслуживания клиентов. Собранные данные хранят номер менеджера, время регистрации клиента, время начала обслуживания и время завершения. При обработке информации требуется найти наибольшее, наименьшее и среднее время ожидания, а также наибольшее, наименьшее и среднее время обслуживания. Использовать функции обработки массивов.

Задание 8. В течение суток через каждый час произведены замеры напряжения в сети. Использовать функции обработки массивов, чтобы определить максимальный скачок напряжения и наибольшее его падение, а также узнать, в какое время суток это произошло.

Задание 9. Кот Матроскин завел 10 коров. Каждый вечер он записывает,

сколько молока дала каждая корова за день. Потом ему нужно узнать, сколько всего молока получено за день, а также какая из коров сегодня отличилась, а какая дала молока меньше всех. Использовать функции обработки массивов.

Задание 10. Каждому ученику 1 класса полагается стакан молока, если его вес меньше 30 кг. Количество учеников и вес каждого известны. Выяснить, сколько литров молока необходимо для класса (1 стакан равен 0,2 л). Использовать функции обработки массивов.

Задание 11. Безумное чаепитие. За круглым столом сидят толстяки. Вес каждого известен. Каждый час они пересаживаются по кругу вправо на один стул. Известно, что один из стульев (он помечен) не выдержит максимального веса толстяка. Используя функцию обработки массивов, определить, в котором часу все повеселятся.

Задание 12. Для того чтобы выявить наиболее популярного политического деятеля из десяти участников, проведен экспертный опрос. Три эксперта каждому деятелю проставляют балл от 1 до 10. Использовать функции обработки массивов, чтобы найти самого популярного деятеля по сумме баллов.

Задание 13. Кот Матроскин и Шарик загадывали числа в произвольном порядке и записывали их на печке, пока не кончилось место. Использовать функции обработки массивов, чтобы определить, каких чисел, четных или нечетных, загадано больше.

Задание 14. Информация о валеологическом обследовании призывников известна. Для каждого хранится вес, рост, объем груди на вдохе. Известны предельные параметры, которым должен удовлетворять призывник. Использовать функции обработки массивов, чтобы найти, какие призывники годны к несению службы.

Задание 15. Филя, Каркуша и Степашка проводят кастинг ведущих передачи «Спокойной ночи, малыши». Каждый кандидат оценивается по трем параметрам: артистичность, фотогеничность, эрудированность. Каждый эксперт выставляет оценку от 0 до 10-ти баллов. Критерий выбора кандидата эксперты не смогли сформулировать, но они хотят выбрать лучшего. Также они хотят узнать лучшего в каждой отдельной номинации. Использовать функции обработки массивов.

Задание 16. Овцы пасутся примерно вместе. Отбившейся от стада считается овца, которая удалилась на максимальное расстояние от условного «центра стада». Если известны координаты всех овец (как точек на плоскости), то проверить, есть ли овца, отбившаяся от стада. Она достанется волку, который с ноутбуком сидит под ближайшим к стаду кустом. Использовать функции обработки массивов.

Задание 17. Коротышки собирали огурцы. Число огурцов, собранных каждым коротышкой, записано. В оплату каждому коротышке выдается два огурца, а тому, кто собрал больше всех, три огурца. Найти, кому три огурца. Найти, сколько собрали. Найти, сколько осталось для засолки. Использовать функции обработки массивов.

Задание 18. Роща ценных деревьев расположена в живописном уголке. Известны координаты каждого дерева. Требуется огородить рощу от зайцев и

олений забором прямоугольной формы, причем потратить на это как можно меньше денег. Найти периметр охватывающего прямоугольника, используя функцию обработки массивов.

Задание 19. Улитка упорно ползет по склону вверх. В солнечный день она способна проползти  $S_1$  м., в пасмурный  $S_2$  м. Сведения о погоде за месяц известны. Использовать функции обработки массивов, чтобы узнать:

- сколько проползала улитка за каждую неделю;
- сколько проползла за месяц;
- какова средняя скорость перемещения.

Задание 20. На метеостанции в компьютер введены сведения о среднесуточной температуре за март. Использовать функции обработки массивов, чтобы найти:

- количество дней, когда температура ниже 0 градусов;
- количество дней, когда температура выше 0 градусов.

Задание 21. Спортсмен бежит по кругу. На каждом круге тренер определяет время прохождения дистанции и записывает его. Определить, сколько метров пробежал спортсмен. Определить, какова средняя скорость бега. Определить, на каком круге скорость была наибольшая, на каком круге наименьшая. Использовать функции обработки массивов.

Задание 22. Роща ценных деревьев расположена в живописном уголке. Известны координаты каждого дерева. Требуется огородить рощу от нашествия мартышек забором круглой формы высотой 5 метров. Найти наименьший радиус окружности, охватывающей все деревья, найти площадь сетки, которая будет потрачена на забор. Использовать функции обработки массивов.

Задание 23. Популяция кроликов изменяется по следующему закону: в «хороший» год она удваивается, в «обычный» увеличивается в 1,25 раза, в «плохой» уменьшается на 0,25%. Характеристики  $N$  лет хранятся, начиная с 1990 года. В этом (нулевом) году популяция составляла  $M$  особей. Использовать функции обработки массивов, чтобы узнать численность популяции к концу  $N$ -го года, вывести на экран данные о ежегодном изменении популяции.

Задание 24. Пес Шарик каждый день фотографирует дачников и обитателей ближайшего леса. Себестоимость одной фотографии  $K$  руб. Шарик записывает в журнал, кого он сфотографировал. Дачникам фотографии стоят денег, причем Шарик берет  $2 \cdot K$  руб. за фотографию. Обитателям леса фотографии раздаются бесплатно, по 1 штуке на морду. Каждый день Шарик хочет знать, каков размер его прибылей (а может, убытков). Использовать функции обработки массивов

Задание 25. Царевна Несмеяна каждому из претендентов на ее руку и сердце задает  $M$  вопросов. За очень понравившийся ответ она присуждает 2 балла, за не очень понравившийся – 6 баллов, за очень не понравившийся – 8. Определить самого понравившегося претендента. Использовать функции обработки массивов

Задание 26. В течение суток через каждый час произведены замеры температуры и влажности воздуха. Использовать функции обработки массивов, чтобы определить максимальное значение температуры и влажности, а также узнать, в какое время суток это произошло.

Задание 27. Безумное чаепитие. За круглым столом сидят толстяки. Вес

каждого известен. Каждый час они пересаживаются по кругу вправо на один стул. Использовать функции обработки массивов, чтобы определить раскладку весов по стульям через  $K$  часов.

Задание 28. Сведения о росте учеников одного класса хранятся в алфавитном порядке фамилий учеников. Использовать функции обработки массивов, чтобы найти:

- самого высокого ученика;
- самого маленького ученика.

Задание 29. Сведения о росте и весе учеников одного класса хранятся в алфавитном порядке фамилий учеников. Параллельно хранится пол ребенка (мальчик, девочка). Известен средний вес ребенка в этом возрасте (для мальчиков и для девочек). Использовать функции обработки массивов, чтобы найти:

- детей, которые выше средней упитанности;
- детей, которые ниже средней упитанности.

Задание 30. Компания, производящая электроэнергию, взимает плату со своих клиентов по следующему тарифу:

- 1,20 руб. за 1 кВт/ч за первые 500 кВт/ч;
- если потребление свыше 500 кВт/ч, но не более 1000 кВт/ч, надбавка 10%;
- если потребление свыше 1000 кВт/ч, надбавка 20%.

Услугами компании пользуются  $n$  клиентов, сведения о них известны и хранятся в таблице. Использовать функции обработки массивов, чтобы посчитать плату для каждого клиента, а также узнать, какова максимальная и минимальная плата.

Задание 31. Есть список школьников одного класса. Для каждого из них известны вес и рост. Использовать функции обработки массивов, чтобы найти самого высокого и самого толстенького школьника.

Задание 32. Участники конкурса балльных танцев получили оценки по трем видам выступлений. Использовать функции обработки массивов, чтобы найти пару, которая победила в соревновании.

Задание 33. Банк провел мониторинг с целью улучшения обслуживания клиентов. Собранная информация хранит номер менеджера, время начала обслуживания и время завершения. При обработке информации требуется найти наибольшее, наименьшее и среднее время обслуживания. Выяснить также, кому из менеджеров принадлежат рекорды скорости обработки клиента. Использовать функции обработки массивов.

Задание 34. Участники соревнований по плаванию в трех видах многоборья показали время в каждом виде соревнований. Использовать функции обработки массивов, чтобы расположить участников в порядке занятых мест.

Задание 35. Информация об успеваемости студентов группы по трем дисциплинам известна. Использовать функции обработки массивов, чтобы найти:

- студентов, которые получили только пятерки;
- студентов, имеющих тройку хотя бы по одному предмету.

Задание 36. Два медвежонка делят  $N$  головок сыра с весами  $P_i$ , где  $i=1, \dots, N$ . Можно ли разделить сыр поровну по весу так, чтобы каждый получил не меньше



$K$  ( $K \leq N/2$ ) головок сыра? Использовать функции обработки массивов

## Тема 9. Работа с двумерными массивами.

### Использование функций для работы с двумерными массивами.

#### Использование файлов для хранения данных

Синтаксис C++ разрешает использовать только одномерные массивы. Массивы большей размерности трактуются как массивы массивов. Использование многомерных массивов имеет некоторые особенности, которые можно рассмотреть на примере двумерных массивов (матриц).

Объявление двумерного массива:

```
int a[8][10]; // Массив из 8-ми одномерных массивов,  
             // в каждом из которых 10 элементов.
```

Элементы массива размещаются в выделенном пространстве построчно по возрастанию индексов. Индексы нумеруются с нуля.

Инициализация двумерных массивов синтаксически выглядит как инициализация нескольких одномерных массивов, например:

```
int a[3][4] = {  
               {1, 2, 3, 4}, // Значения элементов нулевой строки.  
               {2, 4, 6, 8}, // Значения элементов первой строки.  
               {9, 8, 7, 6}  // Значения элементов второй строки.  
};
```

При размещении в памяти будет выделено место под запись  $3 \cdot 4 = 12$  элементов, и они будут размещены линейно в следующем порядке:

1	2	3	4	2	4	6	8	9	8	7	6
Нулевая строка				Первая строка				Вторая строка			

Как и одномерные массивы, матрицы могут быть условно переменной длины. При описании матрицы число строк и столбцов задают константные выражения, это целые значения, которые определяют механизм выделения памяти. Реально из выделенного пространства можно использовать меньшее число данных. Следует объявить переменные, обозначающие реальное число строк и столбцов (меньшее или равное указанному в описании), которые будут управлять процессом сканирования элементов матрицы.

Обращение к элементу массива  $a_{ij}$  можно выполнить по индексу (прямая адресация) или указателю (косвенная адресация):

```
a[i][j] // Адресуется элемент массива, стоящий на  
        // пересечении i-той строки и j-того столбца.  
*(a+i * M + j) // Адресуется элемент массива, отстоящий от его  
               // начала на i * M + j значений.
```

При косвенной адресации важно, что константа  $M$  (количество элементов в строке) определяет разбиение адресного пространства, выделенного для матрицы, на строки определенной длины.

Алгоритмы работы с двумерными массивами основаны на использовании сложного цикла, у которого во внешнем цикле управляющей переменной служит номер строки матрицы, а во внутреннем – номер столбца. Тогда просмотр

элементов происходит построчно. Если во внешнем цикле управляющей переменной служит номер столбца матрицы, а во внутреннем – номер строки, то просмотр элементов происходит по столбцам.

Пусть размер матрицы по описанию определяют define константы N, M. Пусть реальный размер обозначен переменными n, m.

```
#define      N      5
#define      M      7

...
int  a[N][M];
int      n, m;                // Реальное число строк, столбцов.

...
printf ("Введите размер матрицы (строк<%d, столбцов<%d)\n", N, M);
scanf ("%d%d", &n, &m);    // Теперь это наибольшие значения параметров цикла.
// Для обращения к элементам матрицы по строкам цикл записывается так:
    for (i = 0; i < n; i++)
        for(j = 0; j < m; j++)
            {
                // В теле цикла обращение к переменной a[i][j].
            }
// Если поменять эти циклы местами, просмотр будет происходить по столбцам:
    for( j = 0; j < m; j++)
        for (i = 0; i < n; i++)
            {
                // В теле цикла обращение к переменной a[i][j].
            }
```

Зная, что матрица хранится как одномерный массив, можно ее описать как одномерный массив, а элементы адресовать как элементы матрицы, например:

```
int      a[25];                // Матрица 5 на 5.

...
int      n, m;
    n = 5;
    m = 5;
```

Независимо от способа описания двумерного массива, обращение к его элементам с использованием указателя, выполнит те же действия, что и обращение по индексам, а синтаксически должно быть записано так:

```
    for (i = 0; i < n; i++)
        for(j = 0; j < m; j++)
            {
                // В теле цикла обращение к переменной * (a+i * m + j)
            }
```

Как видим, запись сложного цикла несколько не изменилась, а изменился только синтаксис обращения к переменной.

Использование функций при обработке матриц предполагает два различных подхода.

В первом случае матрица рассматривается как самостоятельная структура данных, к которой необходимо применить какой-либо алгоритм обработки. Функции передается вся матрица целиком, для этого в параметрах функции следует указать имя матрицы, при этом функция получает матрицу как адрес. C++ должен знать, каков способ разбиения этой структуры на строки, поэтому число строк данных у имени массива можно опустить, а число данных в строке опускать нельзя. Оно должно быть обязательно указано как константное выражение в квадратных скобках при передаче матрицы. Для массивов переменного размера длина, как правило, передается отдельным параметром. Для матриц следует передать число строк, столбцов. Так, прототип функции, получающей матрицу в качестве входного данных, может выглядеть так:

```
int function (int a [][][M], int n, int m);      // Здесь M – константное выражение.
```

**Пример №1.** В качестве простого примера рассмотрим функции для ввода и вывода матрицы на экран. Матрица условно переменного размера, поэтому число строк и столбцов матрицы по описанию определено define константами  $N = 5$  и  $M = 5$ , а реальный размер матрицы определяют переменные  $n$  и  $m$ .

```
#include <stdio.h>
```

```
#define      N      5
```

```
#define      M      5
```

```
void input_matr (int a [][][M], int &n, int &m);    // Длина строки – константа.
```

```
void print_matr (int a [][][M], int n, int m);      // Число строк можно не передавать.
```

```
// Главная программа описывает входные данные.
```

```
void main(void)
```

```
{
```

```
int      n, m;                                // Реальные размеры матрицы.
```

```
int      matr [N][M];                        // Описан двумерный массив.
```

```
    input_matr (matr, n, m);                  // Передан в функцию ввода.
```

```
    print_matr (matr, n, m);                  // Передан в функцию вывода.
```

```
} // End of main
```

```
// Описание функции ввода. Параметры функции – имя и размеры массива.
```

```
void input_matr (int a [][][M], int &n, int &m)    // n, m возвращаются по ссылке.
```

```
{
```

```
int      i, j;
```

```
    printf ("Введите размер матрицы не более %d на %d\n", N, M);
```

```
    scanf ("%d%d", &n, &m);
```

```
    printf ("Введите матрицу.\n");
```

```
    for (i = 0; i < n; i ++)
```

```
        for (j = 0; j < m; j ++)
```

```
            scanf ("%d", &matr[i][j]);
```

```
}
```

```
// Описание функции вывода. Параметры функции – имя и размеры массива.
```

```
void print_matr (int a [][][M], int n,      int m)
```

```

{
int      i, j;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
            printf ("%5d", mas[i][j]);
        printf ("\n");          // Разбиение вывода на строки.
    }
}

```

Во втором случае, если матрица, это массив из одномерных массивов, то для обработки отдельных строк матрицы можно использовать функции обработки одномерных массивов. Зная, что элементы матрицы, это одномерные массивы, каждый из них можно по очереди передавать в функцию, которая умеет работать с одномерным массивом.

**Пример №2.** При решении задач на обработку одномерных массивов нами были отлажены функции обработки одномерных массивов. Их смело можно применить для решения задач обработки многомерных массивов. Пусть есть функции вывода одномерного массива, и функция его преобразования, которая находит наименьшее значение и переставляет его на первое место. Для ввода матрицы используем функцию ввода из примера №1.

```

#include <stdio.h>
#define N 5
#define M 5
// Есть функция вывода одномерного массива.
void print_mas(int mas[], int len)          // Вывод массива в общем виде
{
    for (int i = 0; i < len; i++)
        printf ("%5d", mas[i]);
    printf ("\n");
}
// На ее основе можно легко описать функцию вывода матрицы как совокупности
// одномерных массивов.
void print_matr (int mas[][M], int n, int m)
{
    printf ("Матрица:\n");
    for (int i = 0; i < n; i++)
        print_mas(mas[i],m); // Обращение к функции вывода массива.
}
// Есть функция преобразования одномерного массива в соответствии с условием.
void Change (int mas[], int len)
{
    // Найдем наименьший элемент.
    int *ip;
    int *min = mas;

```

```

    for (ip = mas; ip < mas+len; ip++)
        if (*ip < *min)
            min = ip;
// Перестановка при завершении поиска.
    *ip = *min;
    *min = *mas;
    *mas = *ip;
}
// main должна объявить матрицу и выполнить управление вызовами функций.
void main(void)
{
    int      n, m;
    int      matr [N][M];           // Матрица объявлена размером 5 на 5.
    int      i;
    input_matr (matr, n, m); // Передан в функцию ввода.
// Для преобразования матрицы ее строки передаются в функцию по очереди как
// одномерные массивы. Цикл управления находится в основной программе.
// и управляет передачей строк в функцию.
    for (i = 0; i < n; i++)
        Change (matr[i], m);       // matr[i], это i – тая строка матрицы.
// Функция вывода матрицы вызывается после завершения обработки.
    print_matr (matr, n, m);
} // End of main

```

**Пример №3.** Просмотр матрицы по столбцам вызовет существенные изменения в основной программе, потому что столбец нельзя передать как одномерный массив простой записью `matr [j]` в силу того, что двумерный массив, это массив массивов (строк для матрицы). Однако функции обработки одномерных массивов можно применить для решения задач обработки столбцов, если предварительно столбец копировать в некую промежуточную структуру, а после преобразования возвращать измененное значение столбцу матрицы. Так, основной цикл главной программы примера №2 будет выглядеть так:

```

int      Tmp[M];                    // Временный массив, его длина равна длине столбца.
    for (j = 0; j < m; j++)        // Цикл по номерам столбцов.
    {
        // Копирование перед каждым обращением.
        for (i = 0; i < n; i++)    // Цикл по номеру элемента в строке.
            Tmp [i] = matr [i][j]; // Tmp получил копию j-того столбца.
        Change (Tmp,m);           // Изменение в Tmp.
        // Копирование после каждого обращения.
        for (i = 0; i < n; i++)
            matr [i][j] = Tmp [i]; // Возвращаем значение j-тому столбцу.
    }
// Вывод обычный.
print_matr(matr,n,m);

```

...

**Пример №4.** Чтобы убедиться в преимуществах косвенной адресации покажем, что можно работать с одномерным массивом как с матрицей. Найдем сумму элементов матрицы.

```
#include <stdio.h>
// Функция ввода матрицы получает одномерный массив. Он разбит на n строк
// по m элементов в каждой. Косвенная адресация позволяет рассматривать
// двумерный массив как линейный.
// Функция ввода матрицы.
void input_matr (int mas [], int &n, int &m)
{
    int i, j;
    printf ("Введи размер матрицы\n");
    scanf ("%d%d", &n, &m);
    printf ("Введи матрицу\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            scanf ("%d", mas + i*m + j);
}
// Функция вывода матрицы.
void print_matr (int mas[], int n, int m)
{
    int i, j;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
            printf ("%5d", *(mas+i*m+j));
        printf ("\n");
    }
}
// Функция нахождения суммы элементов матрицы.
int Sum (int mas[], int n, int m)
{
    int i, j;
    int S = 0;
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            S += *(mas+i*m+j);
    return S;
}
// В главной программе объявлен одномерный массив. Он передается во все
// функции.
void main(void)
{
```

```

int      matr[25];
int      n, m;
    input_matr (matr, n, m);
    printf("Сумма элементов равна %d\n", Sum (matr,n,m) );
    print_matr (matr, n, m);
} // End of main

```

### Варианты заданий

Задание 1. Дана матрица размером  $n \times m$ . Найти суммы элементов в каждой строке матрицы, для чего использовать функцию, находящую сумму элементов одномерного массива. Дополнить матрицу найденными значениями, поместив их в конце каждой строки.

Задание 2. Дана матрица размером  $n \times m$ . Упорядочить матрицу по возрастанию элементов первого столбца. Использовать метод пузырька. Алгоритм заключается в следующем: матрица просматривается по перекрещивающимся парам чисел ( $a[i][0]$ ,  $a[i+1][0]$ ). Если нулевой элемент  $i$ -той строки больше, чем нулевой элемент  $i+1$ -ой строки, то строки меняются местами. Поскольку строки матрицы, это одномерные массивы, имеет смысл написать функцию перемены указателей, адресующих строки  $a[i]$  и  $a[i+1]$ . Перестановки подсчитываются. Алгоритм завершает работу, если при просмотре матрицы нет ни одной перестановки.

Задание 3. Дана матрица размером  $n \times m$ . Найти и заменить нулем максимальное и минимальное значения в каждой строке матрицы. Использовать функцию, находящую минимум, и функцию, находящую максимум из элементов одномерного массива. Передавать им по очереди строки матрицы.

Задание 4. Дана матрица размером  $n \times m$ . Определить, является ли она симметричной относительно главной диагонали. Использовать логическую функцию.

Задание 5. Дана матрица размером  $n \times 3$ . Она содержит результаты измерений прямых круговых конусов: радиус основания, высота, длина образующей. Найти и сохранить в этой же матрице значения объема и площади поверхности каждого конуса. Использовать функции. Найти конус наибольшей площади и наибольшего объема, для чего использовать функцию поиска максимума в одномерном массиве.

Задание 6. Дана матрица размером  $n \times m$ . Выполнить сглаживание в строках матрицы, которое заключается в замене каждого элемента значением среднего арифметического трех стоящих рядом значений. Использовать функцию, выполняющую сглаживание в одномерном массиве. Передавать ей по очереди строки матрицы.

Задание 7. Дана матрица размером  $n \times m$ . Преобразовать матрицу, удалив из нее  $i$ -тую строку и (или)  $j$ -тый столбец (по желанию пользователя). Использовать функцию преобразования матрицы.

Задание 8. На плоскости заданы  $n$  точек своими координатами. Построить матрицу расстояний между всеми точками. Найти наибольшее расстояние. Найти, между какими точками расстояние наибольшее. Использовать

функции для вычисления расстояния между двумя точками и для поиска наибольшего значения.

Задание 9. Дана матрица размером  $n \times m$ . Определить, являются ли упорядоченными по возрастанию данные в каждой строке матрицы. Напечатать исходную матрицу по строкам, в конце каждой строки напечатать сообщение о том, упорядочена строка или нет. Использовать функцию, выполняющую проверку в одномерном массиве. Передавать ей по очереди строки матрицы. Использовать функцию печати строки сообщения.

Задание 10. Заданы несколько матриц произвольного размера. Описать функцию сложения (вычитания) матриц. Передавать как один из параметров функции знак действия ('+' или '-'). Чтобы контролировать входные данные, функция должна быть логической.

Задание 11. Дана матрица размером  $n \times m$ . Преобразовать матрицу следующим образом: найти в каждой строке минимальный и максимальный элементы и поменять их местами с первым и последним элементами строки. Использовать функции, выполняющие поиск минимума и максимума в одномерном массиве. Использовать функцию для выполнения перестановки. Передавать им по очереди строки матрицы.

Задание 12. Получить трехдиагональную матрицу размером  $n \times n$ , у которой на главной диагонали и двух, лежащих выше и ниже ее расположены единицы, а остальные элементы равны нулю.

Задание 13. Дана матрица размером  $n \times m$ . Заменить нулями элементы  $i$ -той строки и (или)  $j$ -того столбца (по желанию пользователя). Использовать функцию преобразования матрицы.

Задание 14. Дана матрица размером  $n \times m$ . Определить, являются ли упорядоченными по возрастанию данные в каждом столбце. Напечатать исходную матрицу по строкам, внизу каждого столбца напечатать сообщение о том, упорядочен столбец, или нет. Использовать функцию, которая решает задачу для элементов одномерного массива. Передавать ей по очереди столбцы матрицы.

Задание 15. Исходные данные, это два массива коэффициентов  $A$  и  $B$ . Вычислить матрицу решений всех линейных уравнений вида  $A_i \cdot x + B_j = 0$ . Использовать функцию.

Задание 16. Дана матрица размером  $n \times m$ . Выполнить выравнивание в матрице, заменяя те значения, которые по абсолютной величине больше некоторого  $M$  (введенного в диалоге), значением  $M$  с учетом знака. Использовать функцию преобразования одномерного массива или матрицы.

Задание 17. Дана матрица размером  $n \times m$ . Найти и заменить нулем максимальное и минимальное значения в каждом столбце матрицы. Использовать функции, выполняющие поиск минимума и максимума в одномерном массиве. Передавать им по очереди столбцы матрицы.

Задание 18. Дана матрица размером  $n \times m$ . Найти норму матрицы: максимальное значение из сумм элементов в каждой строке. Использовать функцию суммирования с сохранением всех найденных сумм. Использовать функцию поиска максимума в одномерном массиве.



Задание 19. Дана матрица размером  $n \times m$ . Найти суммы элементов в каждом столбце матрицы и дополнить матрицу найденными значениями, поместив их в конце каждого столбца. Использовать функцию, находящую сумму элементов одномерного массива. Передавать ей по очереди столбцы матрицы.

Задание 20. На плоскости заданы  $n$  точек своими координатами. Построить матрицу расстояний между всеми точками. Найти равноудаленные точки, если такие есть, и сохранить в отдельном массиве их номера. Использовать функции построения и поиска.

Задание 21. Дана матрица размером  $n \times m$ . Упорядочить каждую строку матрицы по возрастанию элементов. Использовать метод пузырька для одномерного массива, алгоритм которого заключается в следующем: массив просматривается по перекрещивающимся парам чисел ( $a[i]$ ,  $a[i+1]$ ). Если  $a[i] > a[i+1]$ , они меняются местами. Перестановки подсчитываются, алгоритм завершает работу, если при просмотре массива нет ни одной перестановки. Передавать в эту функцию поочередно строки матрицы.

Задание 22. Дана матрица размером  $n \times m$ . Найти седловую точку матрицы и напечатать ее индексы. Седловой точкой называется элемент, имеющий наименьшее значение в строке и наибольшее в столбце. Использовать функцию обработки матрицы.

Задание 23. Дана треугольная матрица размером  $n \times n$ . Описать функцию ввода матрицы, пустую половину при вводе заполнить нулями. Использовать функцию обработки матрицы, чтобы найти значения и индексы наибольшего и наименьшего элементов матрицы. Описать функцию вывода матрицы на экран, выделить цветом найденные значения.

Задание 24. Дана матрица размером  $n \times m$ . Найти сумму элементов каждой строки матрицы. Найти наибольшее и наименьшее значение суммы, а также определить, в какой строке матрицы это значение найдено. Использовать функции для суммирования элементов одномерного массива, для поиска минимума и максимума.

Задание 25. Дана матрица размером  $n \times m$ . Определить и напечатать, в каких строках матрицы ее элементы образуют монотонную последовательность, и какого характера (возрастающую или убывающую). Использовать функцию, выполняющую проверку в одномерном массиве. Передавать ей по очереди строки матрицы.

Задание 26. Дана матрица размером  $n \times m$ . Получить новую матрицу, каждый столбец которой получен сортировкой по убыванию элементов столбца исходной матрицы. Использовать функцию сортировки одномерного массива, где алгоритм сортировки произвольный. Передавать в эту функцию поочередно столбцы исходной матрицы.

Задание 27. Дана матрица размером  $n \times m$ . Найти среднее арифметическое значений ее столбцов, и дописать в последнюю строку матрицы. Найти строку, в которой значения элементов наиболее близки к среднему арифметическому. Использовать функцию поиска среднего в одномерном массиве, передавая ей поочередно столбцы матрицы. Использовать функцию поиска «самой средней

строки».

Задание 28. Дана матрица размером  $n \times m$ . Описать функцию, чтобы найти сумму ее элементов. Получить на основе исходной новую целочисленную матрицу такого же размера, у которой на месте нулевых элементов записаны нули, а на месте значений, отличных от нуля, записаны единицы. Этот алгоритм реализовать функцией, возвращающей число ненулевых элементов.

Задание 29. Дана матрица размером  $n \times m$ . Расстояние между  $k$ -той и  $l$ -той строками матрицы вычисляется по формуле:

$$R_{kl} = \sum_{j=0}^{m-1} a_{kj} \times a_{lj}$$

Определить, между какими строками матрицы расстояние минимально. Использовать функцию для вычисления расстояний и для поиска максимума.

Задание 30. Дана матрица размером  $n \times m$ . В каждой строке матрицы найти первое вхождение значения 10, и все последующие значения заменить нулем. Если числа 10 нет в строке, оставить ее без изменения. Использовать функцию поиска и замены для одномерного массива. Передавать ей по очереди строки матрицы.

Задание 31. Дана матрица размером  $n \times 3$ . Найти сумму элементов, стоящих в строках матрицы, для чего использовать функцию суммирования в одномерном массиве. Найти три строки, в которых содержатся наибольшие значения сумм. Использовать функцию обработки матрицы.

Задание 32. Дана матрица размером  $n \times 3$ . Она содержит результаты измерений сторон  $n$  треугольников. Описать функцию, вычисляющую площадь треугольника. Вычислить площади всех треугольников, сохранить в матрице. Найти номер наибольшего по площади треугольника, используя функцию поиска наибольшего значения в одномерном массиве.

Задание 33. Дана матрица размером  $n \times m$ . Локальным минимумом называется любой элемент  $a_{ij}$ , который меньше своих соседей:  $a_{i-1,j} > a_{ij} < a_{i+1,j}$  и  $a_{i,j-1} > a_{ij} < a_{i,j+1}$ . Найти все локальные минимумы. Сохранить их в матрице, где записаны значение элемента, и его индексы. Использовать функцию обработки матрицы. Отыскать наименьший из локальных минимумов с использованием функции поиска минимального значения в одномерном массиве.

Задание 34. Дана матрица размером  $n \times m$ . Найти суммы элементов тех ее строк, в которых расположены наибольший и наименьший элементы матрицы.

Задание 35. Дана матрица размером  $n \times m$ . Поменять местами строку, содержащую наибольшее значение, со строкой, содержащей наименьшее значение. Использовать функции поиска максимума, минимума в одномерном массиве. Использовать функцию перестановки строк матрицы.

Задание 36. Дана матрица размером  $n \times m$ . Описать и использовать функции для выполнения следующих действий:

- в матрицу M1 поместить все строки исходной, элементы которых образуют монотонную последовательность;
- в матрицу M2 поместить все строки исходной, элементы которых являются

только отрицательными числами.

Задание 37. Дана матрица размером  $n \times m$ . Описать функцию, которая найдет сумму элементов одномерного массива, предшествующих первому по порядку отрицательному элементу, а если такого нет, то сумму всех. Использовать ее для нахождения сумм в строках матрицы.

Задание 38. Дана матрица размером  $n \times m$ . Определить, является ли она симметричной относительно главной диагонали. Использовать логическую функцию.

Задание 39. Дана матрица размером  $n \times m$ . Переставить строки матрицы по убыванию значений наибольших элементов строк. Использовать функции поиска наибольшего значения в одномерном массиве, и функцию перестановки строк матрицы.

Задание 40. Дана матрица размером  $n \times m$  и вектор размером  $m$ . Строки матрицы упорядочены по возрастанию первых элементов. Получить новую матрицу, путем вставки в исходную матрицу строки с элементами данного вектора, не нарушая упорядоченности строк матрицы по первому элементу. Использовать функцию преобразования матрицы.

Задание 40. Дана матрица размером  $n \times m$ . Назовем допустимыми преобразованиями матрицы перестановку двух строк или двух столбцов. С помощью допустимых преобразований добиться того, чтобы максимальный элемент матрицы располагался в левом верхнем углу. Использовать функцию преобразования матрицы.

Задание 41. Дана матрица размером  $n \times m$ . Сформировать вектор  $x$ , элементы которого определяются как максимальные значения в столбцах матрицы, и вектор  $y$ , элементы которого определяются как минимальные значения в каждой строке матрицы. Использовать функции для получения векторов.

Задание 42. Даны две целочисленных таблицы  $A[1,10]$  и  $B[1,15]$ . Разработать алгоритм и написать программу, которая проверяет, являются ли эти таблицы похожими. Две таблицы называются похожими, если совпадают множества чисел, встречающихся в этих таблицах.

## **Тема 10. Использование файлов, одно и двумерных массивов в решении содержательных задач**

### **Варианты заданий**

Задание 1. При записи данных о соревнованиях по шахматам формируется матрица турнира особого вида. Результат матча может быть 1 (выигранная партия), 0 (проигранная) или 0,5 (ничья). При вводе данных нужно получить симметричную матрицу турнира, где на главной диагонали нули, для обоих участников 0,5 в случае ничьей, а если участник выиграл, то его партнер проиграл, и ему записывается 0. Написать функцию для ввода данных турнира в диалоге с формированием матрицы турнира, которую сохранить в текстовом файле. Написать функцию обработки турнира, чтобы определить победителя. Написать функцию обработки турнира, чтобы распределить участников по

убыванию набранных очков.

Задание 2. Фабрика производит валенки, тапочки и боты. Данные об объемах сбыта продукции каждого вида за прошлый год помесечно хранятся в файле. При подведении итогов года дирекция требует найти суммы накопительным итогом для каждого вида продукции, и сохранить эту информацию в файле, а также выяснить, в каком месяце имеет место наибольший сбыт каждого вида продукции. Использовать функции для накопления итогов и для поиска максимума.

Задание 3. Валяльная фабрика производит валенки. Данные об объемах сбыта продукции и о ценах продаж за прошлый год помесечно хранятся в файле в виде таблицы следующего вида:

Месяц	Объем продаж (пар)	Цена продажи (руб.)	Себестоимость (руб.)
Январь	4500	100	20
Февраль	3900	100	25
...			

При подведении итогов года необходимо выяснить динамику сбыта, то есть найти и упорядочить по убыванию информацию о прибылях, приносимых производством. Использовать функцию для вычисления ежемесячной прибыли. Для сортировки матрицы по столбцу «прибыль» использовать функцию сортировки матрицы методом пузырька.

Задание 4. Кот Матроскин владеет стадом коров, а также организовал производство молока, сметаны, творога, масла, сыра. Ежедневно он записывает в текстовый файл дневной сбыт каждого вида продукции, причем цены товара у Матроскина могут изменяться, и цены ему приходится тоже ежедневно записывать. В конце месяца необходимо подвести итог по результатам торговли, чтобы выяснить, какая продукция пользуется наибольшим и наименьшим спросом, а также какая продукция приносит наибольшую и наименьшую прибыль.

Задание 5. В соревнованиях по фигурному катанию каждый спортсмен принимает участие в трех видах соревнований. Каждый вид судят десять судей. Для более точной оценки самый низкий и самый высокий баллы судей отбрасывают. Составить протокол судейства для  $n$  участников в виде матрицы, сохраненной в файле. Найти победителя соревнований. Использовать функции для обработки результатов соревнований.

Задание 6. Коротышки собирают урожай огурцов, помидор, гороха, моркови и прочих плодов земли. Работают  $n$  коротышек, а Знайка подводит итоги их трудовой деятельности. Он завел текстовый файл, и написал программу, которая ежедневно в диалоге позволяет ввести данные об итогах трудового дня. Эти данные суммируются с итогами предыдущих дней работы, и файл обновляется. По итогам работы в конце уборочной кампании производится оплата. Стоимость сбора одного вида овощей Знайка хранит в отдельном файле. Представьте себя на месте Знайки.

Задание 7. Разборчивая невеста занесла в файл данные о своих потенциальных женихах: внешность и богатство она оценила в баллах.

	Возраст	Внешность	Богатство
Иван	25	15	10
Петр	30	15	10
Панкратий	40	12	50
Борис	45	8	50
Магомед	45	10	100

Составить оценочную таблицу в виде матрицы. Возраст невесты  $N$  лет. Требования к претенденту сформулированы просто: средняя красота, среднее богатство, средний возраст, но не моложе ее самой. Помогите сделать выбор. Использовать функцию выбора, возвращающую номер потенциального жениха.

Задание 8. Сорок разбойников провели турнир, в котором каждый разбойник бился с каждым. Сведения о результатах соревнований записали в файл в виде турнирной таблицы, где номера строк и столбцов, – это номера разбойников. Найти самого сильного и самого слабого разбойника. Использовать функцию обработки турнира.

Задание 9. Учебное заведение проводит приемные экзамены на новый учебный год. Информация об абитуриентах занесена в текстовый файл в виде матрицы. Каждый должен сдать три экзамена. Те, кто, получил двойку за первый экзамен, ко второму экзамену не допускаются, а кто получил двойку за второй экзамен, не допускаются к третьему. Обработать данные об абитуриентах, выполняя удаление после первого, второго и третьего экзаменов с использованием функции обработки матрицы.

Задание 10. В зале кинотеатра  $n$  рядов, в каждом из которых  $m$  мест. При бронировании и продаже билетов формируется карта занятости мест, которая сохраняется в текстовом файле после очередной операции. Нужно помочь кассиру и зрителю выполнить процедуру бронирования, для чего организовать диалог:

- показать на экране карту занятости мест в зале;
- показать зрителю выбранное им место;
- зафиксировать выбор места по договоренности со зрителем;
- сохранить измененную карту;
- подсчитать стоимость билетов.

Задание 11. Царевна Несмеяна, принимая претендентов на ее руку и сердце, задает каждому  $M$  вопросов. Если ответ очень понравился, она присуждает 2 балла, если не очень понравился – 6 баллов, если очень не понравился – 8. Данные опроса она сама записывает в текстовый файл. В конце дня выбирается лучший претендент. Использовать функцию обработки матрицы, чтобы определить самого понравившегося претендента. Этот кандидат приписывается в файл, хранящий многолетнюю историю испытаний, по которому, возможно, когда-то будет принято окончательное решение.

Задание 12.  $n$  коротышек собирают урожай  $m$  видов различных овощей. Знайка подводит итоги их трудовой деятельности и выполняет расчет. Он завел текстовый файл, и написал программу, которая ежедневно в диалоге позволяет ввести данные об итогах трудового дня. По количеству собранных овощей в конце дня производится оплата. Сбор одного овоща каждого вида Знайка хранит в отдельном файле. Алгоритм оплаты труда сложный. Вычисляется среднее

арифметическое собранных всеми за день овощей по каждому виду. Если коротышка собрал больше, чем среднее арифметическое этого вида, то сбор каждого овоща сверху оплачивается в 2 раза дороже. Тот, кто собрал больше всех, получает премию в размере стоимости сбора трех овощей этого вида. Ежедневно Знайка подсчитывает, сколько денег получает каждый коротышка за собранный урожай. Эти данные он приписывает в итоговый текстовый файл, чтобы впоследствии подвести общий итог уборочной кампании.

Задание 13. Кот Матроскин владеет стадом  $n$  коров. Ежедневно он записывает в текстовый файл дневной удой каждой коровы. Кот желает приобрести программу, которая позволит подвести итоги производительности его коров. В конце каждого месяца Матроскин желает найти общий удой каждой коровы, а также наибольший и наименьший. Эти данные должны быть программно приписаны в новый текстовый файл, по данным которого Матроскин подведет годовой итог. Использовать функцию (функции) обработки результатов.

Задание 14. Информация о развитии лабораторных животных хранится в файле в виде матрицы, где в строках записаны номера животных, а в столбцах даты (номер недели сначала эксперимента):

	1	2	3
Крыса1	500	400	400
Крыса2	700	700	800
...			

На перекрестье строки и столбца записан вес крыски. Вывести в отдельный файл информацию о крысах, имеющих положительную динамику (вес только возрастает). Вывести в отдельный файл информацию о крысах, имеющих нестабильную динамику. Вывести в отдельный файл информацию о крысах, имеющих отрицательную динамику. Использовать функцию обработки одномерного массива, которой передавать по очереди строки матрицы.

Задание 15. Отец Федор открыл небольшой свечной заводик под Самарой. На заводе льют свечи малые, средние, большие, очень большие и особые. Ежедневный итог по продажам каждого вида продукции в стоимостном выражении матушка дописывает в текстовый файл. По завершении месяца нужно подвести итог, какая суммарная прибыль получена от каждого вида продукции, какой вид приносит наибольшую и наименьшую прибыль.

Задание 16. Сорок разбойников провели соревнование по брейк дансингу. Судили пять судей. Запись о результатах записали в файл в виде матрицы  $40 \times 5$ . Чтобы результат был точнее, самый низкий и самый высокий баллы судей решили не учитывать. Составить оценочную таблицу в виде матрицы, найти победителя соревнований. Использовать функцию обработки результатов соревнований, возвращающую номер победителя.

Задание 17. Дана матрица размером  $7 \times 7$ . Это семеро козлят устроили турнир по копытрестлингу. Сведения о результатах соревнований записали в файл в виде турнирной таблицы, где номера строк и столбцов, это номера козлят, принявших участие в турнире. Турнирная таблица представлена в виде симметричной матрицы. Найти распределение козлят по силе. Использовать функцию обработки

турнира.

Задание 18. Семеро гномов копают золотой песок шесть дней в неделю, а в седьмой отдыхают. Сколько нарыл за день каждый гном, Белоснежка вечером записывает в файл. В воскресенье подводятся итоги. Самый трудолюбивый гном награждается поцелуем красавицы, самый ленивый мытьем посуды. Найти самого старательного и самого ленивого гномов. Использовать функцию (функции) обработки результатов.

Задание 19. Информация о развитии лабораторных животных хранится в файле в виде матрицы, где в строках записаны даты (номер дня с начала эксперимента), а в столбцах номера животных:

	Кролик1	Кролик 2	Кролик 3
1	1500	1400	1300
2	1800	1700	1800
...			

На перекрестье строки и столбца записан вес животного. Для каждого животного проверить, имеет ли место положительная динамика. Для животных с положительной динамикой проверить, у кого она равномерна.

Задание 20. Коротышки провели психологическое тестирование «Узнай себя». Запись о результатах записали в файл в виде таблицы, где на перекрестье строк и столбцов записана оценка в виде итогового балла.

	Умный	Смелый	Добрый
Знайка	10	5	8
Незнайка	2	10	5
Пончик	5	3	10
Шурпчик	8	8	7
...			

Составить оценочную таблицу в виде матрицы, найти самого умного, самого смелого, самого доброго из коротышек. Использовать функции.

Задание 21. Имеется  $n$  городов, и между некоторыми из них летают самолеты. Авиатрассы проложены так, что из любого города можно перелететь в любой другой (возможно с пересадкой). В каждом городе есть только один аэропорт. Эти данные хранятся в текстовом файле. Определить, как можно перелететь из одного любого города в любой другой наилучшим образом, то есть с наименьшим числом пересадок. Использовать функции обработки данных. Исходные данные и результат печатать.

Задание 22. Коротышки устроили турнир по футболу. Было четыре команды: Знайки, Незнайки, Пончики, Сиропчики. Результаты игр записали в файл в виде таблицы соревнований.

Знайчики	Незнайчики	3	2
Знайчики	Пончики	3	1
Знайчики	Сиропчики	1	2
Незнайчики	Пончики	4	1
Незнайчики	Сиропчики	2	2
Пончики	Сиропчики	2	3

Составить турнирную таблицу в виде матрицы игр, определить победителя. Использовать функцию обработки турнира.

Задание 23. В зале кинотеатра 10 рядов по 15 мест. С 3-го по 7 ряд – VIP места. Стоимость билетов на них в два раза дороже, чем на другие места. Стоимость билета зависит от многих факторов и может изменяться. После продажи билетов данные занесены в текстовый файл в виде матрицы занятости мест. Найти сумму, на которую продано билетов. Использовать функцию обработки матрицы.

Задание 24. В зале кинотеатра 10 рядов по 15 мест. С 3-го по 7 ряд – VIP места. Стоимость билетов на них в 2 раза дороже, чем на другие места. 1 и 2-й ряды в полтора раза дешевле прочих. На обычный сеанс билет стоит  $K_1$  руб., на премьерный  $K_2$  руб., на льготный  $K_3$  руб. При продаже билетов данные заносятся в текстовый файл в виде матрицы занятости мест. Найти сумму, на которую продано билетов. Использовать функции формирования и обработки матрицы.

Задание 25. Три эксперта оценивают накануне выборов популярность десяти кандидатов по двадцатибалльной шкале.

	Эксперт1	Эксперт2	Эксперт3
Кандидат1	15	10	10
Кандидат2	18	12	10
...			

Составить оценочную таблицу в виде матрицы. Подвести итог по сумме баллов каждого кандидата, найти трех самых популярных кандидатов. Использовать функции обработки матрицы.

Задание 26. Информация о развитии лабораторных мышей хранится в файле в виде матрицы, где имена столбцов (даты), это начало каждого месяца, а строки пронумерованы номерами животных:

	Дата1	Дата2	Дата3
Мышь_1	150	140	130
Мышь_2	180	170	180
...			

На перекрестье строки и столбца записан вес животного. Для каждого животного проверить, развивается ли оно равномерно в течение месяца, для этого вес должен не убывать. Вывести номера мышей, для которых имелось падение веса. Использовать функцию обработки одномерного массива, которой передавать по очереди строки матрицы.

Задание 27. Для контроля состояния трубопровода ежечасно в течение суток измеряется давление на некоторых точках, где установлены датчики. Для снижения погрешностей датчиков перед обработкой исходные данные сглаживаются путем замены каждого элемента значением среднего арифметического трех стоящих рядом значений (для первого и последнего двух). Результат формируется в новом массиве. Выполнить подготовку данных для  $n$  датчиков, если снятые ими данные за сутки записаны в текстовый файл.

Задание 28. Белоснежка назначила каждому из семерых гномов день недели, когда тот должен дежурить по кухне. Каждый день она ставит оценку за дежурство по пятибалльной шкале, а в конце месяца подводит итог, чтобы наградить лучшего гнома. Сведения о результатах проверки Белоснежка



записывает в файл в виде матрицы, где номера строк, это номера гномов, а столбцов – номера недель. Найти самого старательного и самого ленивого гномов. Использовать функцию (функции) обработки результатов.

Задание 29. Чипполино проводит ежедневный мониторинг погоды, записывая в текстовый файл информацию о дневной температуре, влажности, давлении, количестве выпавших осадков и скорости ветра. Пытаясь дать прогноз на завтра, он всего лишь усредняет все предыдущие данные за месяц. Текстовый файл прогнозов сохраняет все данные. Выяснить, насколько прогнозирование ошибочно, сравнивая данные файла прогнозов с данными файла мониторинга.

Задание 30. Фермер снял план своего участка прямоугольной формы и сформировал матрицу размером  $n \times m$  (размер сетки один метр), в которой на пустых местах записаны нули, а на занятых цифры, например 1 – жилой дом, 2 – зеленые насаждения, 3 – хозяйственные постройки. Матрица занесена в текстовый файл. Теперь фермер желает выяснить:

- каков процент занятости его территории;
- можно ли полностью огородить участок живой изгородью шириной 1 метр, а если нет, то какие будут препятствия;
- на каком месте можно построить сарайчик размером  $k$  на  $l$ , если это возможно.

Использовать функции обработки матрицы.

Задание 31. Кукольный театр поставил  $n$  спектаклей. Ежедневно театр дает несколько различных спектаклей. Мальвина продает билеты, и перед каждым сеансом знает, сколько детских и взрослых билетов продано. Чтобы знать репертуарную политику, Мальвина желает выяснить, какие спектакли имеют наибольшую популярность. Для этого данные о посещаемости каждого спектакля она сохраняет в текстовом файле следующего вида:

Спектакль	Показов	Детских билетов	Цена	Взрослых билетов	Цена
Золушка	20	25	10	30	20
Король Лир	15	10	10	50	30
...					

Данные о спектакле вводятся в диалоге, и обновляют матрицу данных, причем данные о числе билетов суммируются, а данные о ценах усредняются. В любой момент времени можно получить информацию о том, каков интерес к спектаклям в порядке убывания отдельно для взрослых и для детей, а также узнать в порядке убывания общую прибыль от спектакля.

Задание 32. Кафе специализируется по бизнес-ланчам для тех, кто следит за своей фигурой, поэтому в меню помимо цены указана и калорийность каждого блюда. Предлагается  $N$  комплексных ланчей, состоящих из  $Q$  блюд каждый. Стоимость и калорийность каждого блюда записаны в текстовых файлах в виде матрицы стоимостей и матрицы калорийностей. Выбрать все ланчи, калорийность которых ниже, чем указанное значение. Подсчитать стоимость. Выбрать все ланчи, стоимость которых ниже, чем указанное значение. Подсчитать калорийность.

Задание 33. В аптечном складе хранятся лекарства. Сведения о лекарствах содержатся в специальной ведомости: номер лекарства, количество (в шт.), цена. Ведомость хранится в текстовом файле. Провизор время от времени развлекается, решая следующие нестандартные задачи:

- сколько стоит самый дешевый и самый дорогой препарат;
- каково количество всех препаратов на складе;
- какова общая стоимость всех препаратов.

Использовать функции для нахождения минимального, максимального элементов массива и функцию для нахождения суммы.

Задание 34. Кафе специализируется по бизнес-ланчам для тех, кто следит за своей фигурой, поэтому в меню указаны цена и калорийность каждого блюда. Стоимость и калорийность каждого блюда записаны в текстовых файлах в виде матрицы стоимостей и матрицы калорийностей. Упорядочить исходные данные по возрастанию калорийности. Упорядочить исходные данные по возрастанию цены. Выбрать все блюда, калорийность которых ниже, чем указанное значение, и подсчитать их стоимость.

Задание 35. В музее регистрируется в течение дня время прихода и ухода каждого посетителя. Таким образом, за день получены  $N$  пар значений, где первое значение в паре показывает время прихода посетителя и второе значения – время его ухода. Найти промежуток времени, в течение которого в музее одновременно находилось максимальное число посетителей, минимальное число посетителей.

Задание 36. Дана матрица размером 5×3. Это жители Средиземья организовали курсы информатики. По окончании курсов всем выдали свидетельства. Похвальные листы нужно вручить трем лучшим слушателям. Найдите лучших, если данные записаны в текстовом файле в форме таблицы следующего вида:

	MS Word	MS Excel	MS Access
Фродо Беггинз	5	5	3
Бильбо Беггинз	3	4	5
Гэндальф	3	5	5
Двалин	5	4	3
Гвалин	3	5	5
Бифур	3	4	3
Бофур	5	4	3
Бомбур	3	4	4
Торин Оукеншильд	5	4	4

Использовать функции обработки данных.

## Тема 11. Работа со строками символов. Использование файлов

Строка представляет собой массив символов, однобайтовых данных целого типа, объявленных как `char` или `unsigned char`. Внутреннее представление символа, это его код (ASCII кодом является число в пределах от 0 до 256 согласно кодовой таблице), где первые 32 символа управляющие. Синтаксически признаком символьной константы являются кавычки, например, `'?'`, `'f'`, `'Я'`, `'#'`, `'l'`.

Строковая константа, это последовательность символов, заключенная в двойные кавычки `" "`, например `"строка символов"`.

При представлении строки в памяти она рассматривается как массив символов, где каждый символ хранится в отдельном байте, включая специальные символы. В конце строки должен быть нулевой байт '\0' как признак конца строки. Он добавляется автоматически при инициализации строки и при вводе строки с помощью специальной функции gets. При формировании строки вручную необходимо заботиться о том, чтобы этот символ был добавлен. Число символов в строке (длина строки) всегда больше на 1 (для нулевого символа).

Объявление строковых переменных можно выполнять двумя способами.

Как массив:           char Str [80];

Это плохой стиль, так как длина строки может изменяться ограниченно.

Как указатель:       char \*Str;

Это хороший стиль, но в этом случае строка является динамической, и для указателя необходимо выделение памяти конструкцией Str = new char [80];

При инициализации строковых переменных происходит автоматическое выделение памяти под запись значения строки.

```
char *Str1 = "строка 1 "; // Длина 11 байт, включая пробелы.
```

```
char Str2[9] = "строка 2"; // Длина 9 байт.
```

```
char Str1[] = "строка 3"; // Специальный инициализатор.
```

```
char Err[4] = "ошибка"; // Число символов больше, чем объявлено.
```

```
char Err[10] = "не ошибка"; //
```

**Пример №1.** Символьные константы в строках. Управляющие символы (Esc-последовательности) в тексте строки предваряются знаком слэш «\». Такой символ, встреченный в строке, приводит к выполнению управляющего воздействия, например, "\n" вызовет перевод строки. Чтобы особые символы были видны при выводе, в записи строки они удваиваются, например, " \ ".

```
#include <stdio.h>
```

```
void main(void)
```

```
{
```

```
// Выведем на экран несколько строк.
```

```
printf ("%s\n","Строка символов"); // Без особенностей.
```

```
printf ("%s\n","Слэш \ запишем как \\ \n "); // Символ слэш удваивается.
```

```
printf ("%s\n","Символ \"'\" удваивается"); // Апостроф удваивается.
```

```
// Здесь слэш не только не влияет на управление строкой, но и не виден.
```

```
printf ("%s\n","длинные строки могут\
```

```
быть разбиты на части\
```

```
произвольно, на самом\
```

```
деле это одна строка");
```

```
// Здесь сочетание \n управляет выводом строки, разбивая ее на части.
```

```
printf ("%s\n","длинные строки могут\n быть разбиты на части\n произвольно, на\n самом\n деле это несколько строк.\n");
```

```
} // End of main
```

Для ввода и вывода текстовых строк используются специальные функции gets() и puts(), аргументом которых является строка.

**Пример №2.** Инициализация строк. Ввод и вывод текстовых строк. Для ввода и вывода строк можно использовать функции `printf`, `scanf` с управляющим признаком формата `%s`. При вводе строки ее текст будет введен только до первого пробела.

```
#include <stdio.h>
void main(void)
{
// Примеры объявления и инициализации строк.
    char    Str_1 [10] ;           // Выделено 10 байт,
// значения не присвоены.
    char    Str_2 [] = {'с','и','м','в','о','л','\0'}; // Выделено 7 байт,
// инициализация как массива символов.
    char    Str_3 [] = "primer & primer"; // Выделено 16 байт,
// инициализация как текстовой строки.
    char    * Str_4 = "primer string"; // Выделено 14 байт,
// инициализация как указателя.
    char    * Str_5;               // Указатель, память не выделена.
// Вывод строк может иметь варианты:
//1). Как строка по формату %s
    printf ("%s\n", Str_2);
//2). Как последовательность символов по формату %c
    int    i = 0;
    while ( Str_2[i] ) // Пока не встречен последний символ '\0'.
        printf("%c", Str_2 [i ++]);
    printf("\n");
//3). Как строка без формата
    puts (Str_3);
// Особенности ввода строк. Для Str_1 можно ввести не более 9-ти символов.
    scanf ("%s", Str_1); // Текст будет введен до пробела.
    printf ("%s", Str_1);
    gets (Str_4);        // Текст будет введен до Enter.
    puts (Str_4);
} // End of main
```

**Пример №3.** Строки и указатели. Строку нельзя считать обычным массивом из-за наличия нулевого байта `'\0'` в конце строки. Символьная строка, встреченная в выражении, это адрес массива символов. Может использоваться везде, где можно использовать указатель, но не в левой части операции присваивания. Удобно использовать косвенную адресацию, при этом текущий указатель будет адресовать один символ строки, но в отладчике будет видна оставшаяся часть строки до нулевого байта `'\0'`. Возможные ошибки связаны с механизмами выделения памяти для динамических строк. Если при объявлении указателя текстовая строка проинициализирована, то выделена память, равная длине этой строки. Нельзя даже пытаться записать в эту строку больше символов, чем выделено. Если же при объявлении указателя текстовая строка не

проинициализирована, то память под запись строки не выделена, и этот указатель можно использовать только как рабочую переменную для косвенной адресации при работе с какой-либо строкой.

```
# include <stdio.h>
```

```
void main(void)
```

```
{
```

```
// Объявлена и проинициализирована строка длиной 57 байт.
```

```
char * Str = "Пример строки текста. Память выделена при инициализации.";
```

```
char    pts;                // Указатель на строку.
```

```
int      i = 0;
```

```
// gets, puts – функции ввода-вывода строк.
```

```
    gets (Str);             // Нельзя вводить больше, чем 57 символов.
```

```
    puts (Str);
```

```
// Ошибкой будет запись gets (pts); т.к. память для pts не выделена.
```

```
// Присваивание значения указателю:
```

```
    pts = Str ;             // pts показывает на начало строки.
```

```
    pts += 14;              // pts показывает на 14-й символ строки Str.
```

```
// Присваивание значения pts вносит изменения в строку.
```

```
    * pts = 0;              // Теперь здесь будет конец строки
```

```
// (15-й символ).
```

```
    pts = Str;              // pts снова показывает на начало строки.
```

```
// Обычный механизм перемещения по строке, это смещение указателя.
```

```
    pts ++;                 // Возможная ошибка – выход за пределы
```

```
    pts ++;                 // строки.
```

```
// Для полного просмотра строки используется циклический алгоритм, в котором
```

```
// управляющей переменной является указатель на очередной элемент (символ)
```

```
// строки, который изменяется от адреса начала строки до достижения нулевого
```

```
// байта в конце строки.
```

```
pts = Str;                  // pts указывает на начало строки.
```

```
    while ( * pts )         // Пока его значение отлично от '\0'.
```

```
    {
```

```
        // Вывод очередного символа и смещение указателя.
```

```
        printf("%c", *pts ++);
```

```
    };
```

```
    printf("\n");
```

```
// Такой же циклический алгоритм используется и для изменения содержимого
```

```
// строки. Управление циклом не изменяется, а изменение символов строки
```

```
// достигается изменением значения указателя на очередной элемент (символ)
```

```
// строки.
```

```
pts = Str;                  // pts указывает на начало строки.
```

```
    while ( * pts++)
```

```
    {
```

```
        *pts++ = '@'; // Каждый второй символ будет заменен на '@'.
```

```
    };
```

```
    puts (Str);
} // End of main
```

**Пример №4.** Внутренние коды символов. Данные символьного типа при хранении имеют внутренние коды. ASCII кодом является число в пределах от 0 до 256. Внутри кодовой таблицы символы расположены в определенном порядке. Значения символов по их кодам можно увидеть, выполнив следующий пример. Здесь переменные *i*, *j* пробегают значения так, чтобы все значения выражения  $(16*i + j)$ , которые изменяются в пределах от 0 до 255, могли быть выведены в виде матрицы. Символы с кодами от 7 до 14 при выводе пропущены.

```
#include <stdio.h>
#include <conio.h>
void main (void)
{
    int      i, j;
    clrscr ();
    for (i = 0; i < 16; i ++)
    {
        for (j = 0; j < 16; j ++)
        {
            if (16*i + j > 7 && 16*i + j < 14) continue;    // Управляющие.
            // Символьное представление кода символа.
            printf ("%c ", 16*i + j);

        }
        printf("\n");
    }
} // End of main
```

Использовать внутренний порядок следования кодов символов можно при сравнении символов и при выполнении арифметических операций над символами. Эти операции выполняются над значениями внутренних кодов символов.

```
char      c1, c2;
// Пусть эти символы имеют значения.
// Как сравнить два символа:
int      k;
    k = c1 == c2;                // Или c1 > c2 и прочие условия.
    if (c1 >= '0' && c1 <= '9')
        printf ("Этот символ цифра\n");    // Символ c1 – цифра.
// Для перебора в алфавитном порядке управляющая переменная цикла
// может быть символьной.
char      c;
    for (c = 'a'; c <= 'z'; c ++)
    {
        ...
    }
```

**Пример №5.** Адресация в строках. К способам адресации в строках еще раз вернемся на примере решения задачи копирования одной строки в другую. Адресация для строк, как и для обычных массивов, может быть прямой и косвенной.

```
#include <stdio.h>
#include <conio.h>
void main (void)
{
    char    *Str1 = "Первая строка."; // Исходная строка. Длина 15 байт.
    char    *pts;                      // Рабочая переменная для косвенной адресации.
    char    *Str2;                      // Новая строка будет динамической.
    puts (Str1);                        // На экран выведена исходная строка.
    // Определение длины строки len выполняется в цикле.
    // Управляет циклом переменная pts. Ее начальное значение равно адресу строки.
    // *pts, это значение очередного символа строки. *(pts++), – следующего за ним.
    // Выход из цикла происходит, когда найден признак конца строки.
    int      len = 0;
        pts = Str1;                      // Рабочая переменная указывает на начало.
        do
            len ++;
        while ( *(pts++) != '\0'); // Поиск нулевого байта и смещение указателя.
        printf ("Длина строки %d\n", len);
    // Выделение памяти для строки Str2 динамическое.
        Str2 = new char [len];           // Выделено столько же, сколько в Str1.
    // Прямая адресация, это посимвольное копирование из Str1 в Str2.
    int      i = 0;                      // Начиная нулевого символа строки.
        while (Str1[i] != '\0')          // Пока не встречен нулевой байт в конце.
        {
            Str2[i] = Str1[i];           // Посимвольно присваивать,
            i++;                          // и переходить к следующему символу.
        }
    // Цикл while заканчивается до переноса нулевого байта в новую строку,
    // поэтому по завершении цикла нужно его дописывать программно.
        Str2[i] = '\0';                  // Формируется конец строки.
        puts (Str2);
    // В следующем примере прямой адресации тоже посимвольно копируется строка,
    // но запись цикла короче. Использование оператора do удобнее тем, что
    // нулевой байт переносится в новую строку, поэтому его не нужно приписывать.
        *Str2 = '\0';                    // Так нужно очистить строку от содержимого.
        i = 0;
        do
            Str2[i] = Str1[i];           // Присваивание.
        while ( Str1[i++] != '\0');      // Проверка достижения конца строки.
    // Нулевой символ переносится в новую строку.
```

```

    puts (Str2);
// Косвенная адресация при посимвольном копировании требует использования
// указателя на char в качестве рабочей переменной.
// В первом случае используем только один указатель на строку, которая будет
// копией. Меняя адрес первой строки, покажем, как легко можно изменить
// значение адреса, которое приведет к потере данных.
// Очистка строки.
    *Str2 = '\0';
// Управляет циклом переменная Str1, адресующая исходную строку.
    pts = Str2;                // pts – указатель на новую строку.
    while (*Str1 != '\0')
        *pts++ = *Str1++; // Str1 изменяется. Адрес, выделенный ранее
                           // для этой переменной, изменен операцией ++,
                           // строка потеряна, ее адрес неопределен.
// Признак конца строки должен быть перенесен в строку – копию.
    *pts = '\0';                // Получена вторая строка.
    puts (Str2);
// Вернуться к первоначальному значению адреса первой строки можно, но мы
// введем новую переменную, чтобы показать в следующем цикле, что нужно
// сделать, чтобы косвенная адресация не изменяла бы адреса исходных строк.
// Для этого требуется две рабочие переменные, два указателя на обе
// строки, играющие роль синонимов строк.
char    * Str3 = "Новая строка"; // Ее длина не больше, чем у копии.
char    *pts_2;                  // Для работы со второй строкой.
char    *pts_3;                  // Для работы со третьей строкой.
    pts_2 = Str2;                // Знает адрес строки.
    pts_3 = Str3;                // Знает адрес строки.
    *Str2 = '\0';                // Очистка строки.
// Управляет циклом переменная pts_3, адресующая исходную строку.
    do
        *pts_2++ = *pts_3;        // Оба указателя смещаются одинаково.
    while (*pts_3++ != '\0');    //
    puts (Str2);
// Пример короткой записи цикла управления показывает, как синтаксис C++
// позволяет сократить текст программы. Тонкости алгоритма в этом случае
// не видны. Выполняются все те же действия, что и в предыдущем примере.
    *Str2 = '\0';
    pts_2 = Str2;
    pts_3 = Str3;
    while (( *pts_2++ = *pts_3++) != '\0');
    puts(Str2);
} // End of main

```

**Пример №6.** Функции и строки. При передаче строк в функции особенностей нет. Передача выполняется так же, как и для массивов. Строка должна быть



параметром функции. Отдельно передавать длину строки нет необходимости, так как, во-первых, строка имеет признак конца, а во-вторых, ее длину всегда можно определить, используя функцию `strlen` библиотеки `<string.h>`, подробнее о которой далее по тексту. Рассмотрим пример функции преобразования строки, которая удалит из строки «лишние» пробелы, то есть те, что встречаются подряд друг за другом. Эта операция может быть названа сжатием строки. Приведем пример двух функций, первая из которых использует прямую адресацию, вторая косвенную.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
// Сжать строку, это означает удалить из нее лишние пробелы.
void compress_1 (char []);    // Прототип функции с прямой адресацией.
void compress_2 (char *);    // Прототип функции с косвенной адресацией.
void main(void)
{
    char *str = new char [80];    // Указатель на строку и выделение памяти.
    printf ("\nВведите строку символов с пробелами.\n");
    gets (Str);
    printf ("Наша строка: \n");    puts (Str);
    compress_1 (Str);    // Обращение к функции.
    printf ("Наша строка: \n");    puts (Str);
    printf ("\nВведите строку символов с пробелами.\n");
    gets (Str);
    printf ("Наша строка: \n");    puts (Str);
    compress_2 (Str);    // Обращение к функции.
    printf ("Наша строка: \n");    puts (Str);
    getch();
} // End of main
// Описание функции, использующей прямую адресацию строки.
// Длину строки Str возвращает функция strlen (Str).
void compress_1 ( char Str[] )
{
    int  i, mid;    // i – рабочая переменная для поиска рядом стоящих пробелов.
                // mid – рабочая переменная для сдвига.
    i = 0;    // Внешний цикл поиска рядом стоящих пробелов.
    while (i < strlen (Str) – 1)
        if ((Str [i] == ' ') && (Str [i+1] == ' '))    // Два пробела рядом.
        {    // Внутренний цикл сдвига.
            for (mid = i; mid < strlen(Str); mid ++)
                Str [mid] = Str [mid+1];    //символ '\0'тоже сдвигается.
        }
        else // Движение по строке, только если не было сдвига.
            i++;
}
```

```

}
// Описание функции, использующей косвенную адресацию строки.
void compress_2 (char * Str)
{
char *ip, *mid;           // Рабочие указатели для поиска и для сдвига.
ip = Str;                 // Внешний цикл поиска рядом стоящих пробелов.
while ( *ip!='\0')
    if ( *ip==' ' && *(ip+1) == ' ')           // Два пробела рядом.
        { // Внутренний цикл сдвига.
            mid = ip;                           // Запомнили адрес.
            while (*mid != '\0')
            {
                *mid =*(mid+1);           // Копирование символа.
                mid++;                     // Смещение указателя.
            }
        }
    else // Движение по строке, только если не было сдвига
        ip ++;
}

```

**Пример №7.** Функция, возвращающая строку. Строка, передаваемая в функцию как параметр, будет изменена, потому что передается адрес строки. Многие алгоритмы требуют сохранения исходной строки, многие алгоритмы должны породить новые строки. Следовательно, функция, порождающая новую строку, должна ее вернуть. Сравним две функции, одна из которых возвращает новую строку через параметр, другая через указатель. Функция сформирует новую строку на основе заданной, вставляя в нее указанный символ через один.

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
// Основная задача, это сохранить строку – образец. Новая строка будет содержать
// все символы исходной, но после каждого символа исходной строки будет
// добавлен новый символ.
// Функция Transform_1 получает и возвращает строки только через параметры,
// поэтому ее тип void. Память для строк должна быть выделена в main.
void Transform_1 (char *Str, char *Ind, char Symbol)
{
    // Память для Ind выделена в main
    char *pts;           // Рабочая переменная
    pts = Str;           // Теперь pts, это исходная строка.
    char *pti;           // Рабочая переменная
    pti = Ind;           // Теперь pti, это новая строка Ind
    do
    { // Циклом управляет pts
        *pti = *pts;      // Символ приписывается к Ind (*pti) из Str (*pts)

```

```

        pti++;
        *pti = Symbol;      // Symbol приписывается к Ind (*pti)
        pti++;
    }
    while ( *pts++ != '\0');    // Выход при достижении конца строки
                                // '\0' перенесен в строку назначения.
}
// Функция Transform_2 получает исходную строку через параметры, а новую
// возвращает через указатель, поэтому ее тип char *. Память для новой строки
// должна быть выделена в теле функции.
char * Transform_2 (char *Str, char Symbol)
{
    in    len;
        len = strlen (Str);      // Длина строки Str
    char  * Ind;
        Ind = new char [len*2];  // Выделена память для новой строки
    char *pts;
        pts = Str;               // Рабочая переменная
        pts = Str;               // Теперь pts, это исходная строка
    char *pti;
        pti = Ind;               // Рабочая переменная
        pti = Ind;               // Теперь pti, это новая строка Ind
    // Алгоритм тот же самый, но запись короче.
    do
    {
        *pti ++ = *pts;
        *pti ++ = Symbol;
    }
    while ( *pts++ != '\0');
    return Ind;                  // Возвращается адрес новой строки.
}
// Обращение к этим функциям выполняется различным образом. Вызов
// Transform_1, это оператор-функция, вызов Transform_2, это
// оператор-выражение, его значение должно быть присвоено переменной
// такого же типа, как и тип возвращаемого функцией значения.
void main(void)
{
    clrscr ();
    char  *Str = new char[80];
    char  *Str_new_1 = new char[160];
    Str = "Преобразование строки без изменения образца.";
    puts (Str);
    Transform_1 (Str, Str_new_1, '!');    // Для Str_new_1 память выделена.
    puts (Str_new_1);
    char  *Str_new_2;                    // Просто адрес.
                                           // память выделит функция.
}

```

```

    Str_new_2 = Transform_2 (Str, '*');
    puts (Str_new_2);
} //End of main

```

**Пример №8.** Использование библиотек функций работы с символами и строками.

Библиотека функций <ctype.h> используется для работы с символами, имеет много полезных функций. Например, isalpha определит, является ли символ буквой латинского алфавита, isascii определит, имеет ли символ код ASCII (лежит в диапазоне от 0 до 127), tolower и toupper позволят преобразовать символ латинского алфавита соответственно к нижнему и верхнему регистру, и так далее.

Библиотека <stdlib.h> имеет функции преобразования, которые используются для преобразования строки в число (atof, atoi) или числа в строку (itoa) и так далее.

Библиотека <string.h> содержит функции преобразования строк. Например, strlen возвращает целочисленную длину строки, включая '\0', strcpy возвращает указатель на копию строки, strcat выполняет конкатенацию строк, strchr поиск первого вхождения символа, strcmp сравнение строк, strstr поиск подстроки в строке и так далее.

Рассмотрим несколько простых примеров использования библиотечных функций.

```

#include <stdio.h>
#include <ctype.h>
#include <string.h>
// Функция подсчитает, сколько символов в строке являются цифрами, используя
// функцию isdigit, которая получает как входное данное символ (char), и
// возвращает значение 0 или 1.
int cou_dig (char *str)
{
    int count = 0;
    while ( *str != '\0' )
    {
        if (isdigit (*str) !=0 ) count++;
        str ++;
    }
    return count;
}
// Используем функцию cou_dig, а также функции strcpy, strcat, strstr.
void main(void)
{

```

```

char    * my_str = "1 2 3 4 5.";
printf ("Цифр=%d",cou_dig (my_str));
char    * Old_str = "Строка текста";
char    New_str [80];
// Для копирования строки Old_str по адресу New_str используем функцию strcpy.
strcpy (New_str, Old_str);      // New_str = "Строка текста"
puts (New_str);
// Для слияния строк (конкатенации) используем функцию strcat.
strcat (New_str, Old_str);      // New_str = "Строка текста Строка текста"
puts (New_str);
// Для поиска вхождения подстроки в строку используем функцию strstr.
char    * found = "ока";        // Строка поиска, это текст "ока".
char    * Yes;
Yes = strstr (Old_str, found);   // Yes = "ока текста"
if (Yes==NULL)                  // Если бы вхождения не было.
    printf ("Нет");
    else
        printf ("Есть \"%s\\n\", Yes);
} // End of main

```

**Пример №9.** Работа с текстом. Текст может быть представлен двумерными массивами строк. В качестве примера покажем реализацию функции сортировки строк текста в алфавитном порядке (по возрастанию первых символов строки). Используем функции из библиотеки string.h.

```

#include <stdio.h>
#include <string.h>
#define      LEN80      // Наибольшая длина строки.
#define      SIZE      20  // Наибольшее число строк.
// Функция в качестве параметра использует указатель на массив строк.
// Массив фактически двумерный. Разбивку на строки определяет константа LEN.
void SortStr (char *Str [LEN], int num)    // Массив строк, num – его длина.
{
    int      i;
    int      flag;
    char     buf [LEN];                    // Для перестановки строк местами.
// Обычная пузырьковая сортировка
flag = 1;
while (flag != 0)
    for (i = 0, flag = 0; i < num - 1; i++)
    {
        if ( strcmp (Str[i], Str[i+1]) > 0) // Символ Str[i] > символа Str[i+1]
        { // Перестановка строк
            strcpy (buf, Str[i]);
            strcpy (Str[i], Str[i+1]);
            strcpy (Str[i+1], buf);
        }
    }
}

```

```

        flag = 1;                // Запоминает факт перестановки.
    }
}

// Покажем подготовку данных для функции и выполним управление вызовами.
void main(void)
{
    char    Text [SIZE][LEN];    // Массив вводимых строк
    char    *ps [SIZE];          // Массив указателей на строки массива.
    int     i, num;               // num – число вводимых строк
    printf ("Вводи строки до пустой строки. \n");
    num = 0;
    while(( gets (Text[num]) != NULL ) && num <= SIZE && strcmp (Text [num],"") != 0)
        // Условие завершения ввода сложное:
        // • пока gets выполняет удачный ввод (возвращает не пустое значение),
        // • пока число введенных строк меньше, чем наибольшее разрешенное SIZE,
        // • пока строка не пуста, при этом strcmp позволит сравнить значения
        // символов, а не указатели:
        {
            ps [num]= Text [num];    // Указатель на очередную
                                    // прочитанную строку
            num ++;
        }
    // Так накоплен массив строк символов, его длина num.
    printf ("Вывод перед обращением к функции сортировки\n");
    for (i = 0; i < num; i ++)
        puts (Text [i]);            // Введенный массив строк
    printf ("\n");
    SortStr (ps, num);
    printf ("Вывод после обращения к функции сортировки\n ");
    for (i = 0; i < num; i ++)
        puts (Text [i]);            // Преобразованный массив строк
    printf ("\n");
    for (i = 0; i < num; i ++)
        puts (ps[i]);               // Указатель на сортированные строки
    printf("\n");
} // End of main

```

### Варианты заданий

В заданиях этой темы предлагается две задачи. Первая простая задача, это алгоритм прямого поиска в строке. Обязательно следует использовать функции. Строка может быть статическим массивом, но для работы со строкой следует применить оба способа адресации, прямой и косвенный, как мы это делали в примерах №6 и №7. Вторая задача предполагает изменение массива. Важны механизмы выделения и перераспределения памяти для строк, когда их

длина меняется. Здесь предпочтительнее использование косвенной адресации.

#### Задание 1.

1. Описать функцию работы со строкой символов, которая найдет, сколько раз входит в строку некоторый произвольный символ (задан как параметр функции).

2. Дан текст (несколько строк) в файле F1. Описать функцию преобразования строки, которая вносит изменения в строку текста, повторяя дважды каждую букву строки (знаки препинания и прочие символы не изменять). Преобразовать все строки текста, новый текст записать в файл F2.

#### Задание 2.

1. Описать функцию работы со строкой символов, которая найдет, сколько раз в строке встречаются знаки препинания '!', '?', '!' и другие.

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга пробелами. Описать функцию, удаляющую из строки слово с номером  $M$  (номер слова  $M$  один из параметров функции). В вызывающей программе номер слова лучше всего вводить в диалоге. Преобразовать все строки текста, новый текст записать в файл F2.

#### Задание 3.

1. Описать функцию работы со строкой символов, которая найдет первое вхождение некоторого слова (задан как один из параметров функции).

2. Дан текст (несколько строк) в файле F1. Описать функцию преобразования строки, которая после каждого десятого символа вставит в текст сочетание символов ":", в начало строки – ":", в конец – ":". Преобразовать все строки текста, записать новый текст в файл F2.

#### Задание 4.

1. Описать функцию работы со строкой символов, которая найдет первое вхождение некоторого символа и последнее его вхождение (символ задан как один из параметров функции).

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга одним пробелом. Описать функцию поиска в тексте некоторого слова (включить в список параметров функции, а в вызывающей программе вводить в диалоге). Описать функцию, удаляющую слово из текста. Преобразовать все строки текста, новый текст записать в файл F2.

#### Задание 5.

1. Описать функцию работы со строкой символов, которая найдет, сколько всего слов в строке, если известно, что слова отделены друг от друга пробелами или сочетанием пробела со знаком препинания.

2. Дан текст (несколько строк) в файле F1. Описать функцию, преобразующую строку текста следующим образом: если в строке есть подстроки, заключенные в скобки, удалить их вместе со скобками. Если есть вложения скобок, удалить всю внешнюю скобку. Преобразовать все строки текста, записать в файл F2.

#### Задание 6.

1. Описать функцию работы со строкой символов, которая найдет, есть ли в строке стоящие рядом одинаковые символы, и определит число таких вхождений.

2. Дан текст (несколько строк) в файле F1. Описать функцию преобразования

строки, которая заменяет все одинаковые идущие подряд символы одним вхождением этого символа. Преобразовать все строки текста, записать новый текст в файл F2.

Задание 7.

1. Описать функцию работы со строкой символов, которая проверит, не является ли эта строка палиндромом. Для проверки придется удалить все пробелы, отделяющие слова, и перевести символы в один регистр. Пример палиндрома «А роза упала на лапу Азора».

2. Дан текст (несколько строк) в файле F1. Описать функцию преобразования строки, которая заменяет все повторные вхождения символа пробел одним символом пробела, в начало строки добавляет слово "НАЧАЛО", в конец слово "КОНЕЦ". Преобразовать все строки текста, записать новый текст в файл F2.

Задание 8.

1. Описать функцию работы со строкой символов, которая найдет, сколько в строке символов, которые являются цифрами, и сформирует из них новую строку.

2. Дан текст (несколько строк) в файле F1. Описать функцию обработки строки, которая удаляет из строки все знаки препинания. Преобразовать все строки исходного текста, записать новый текст в файл F2.

Задание 9.

1. Описать функции работы со строкой символов, которые найдут самое длинное и самое короткое слово строки.

2. Дан текст (несколько строк) в файле F1. Описать функцию, которая удалит из строки символы, находящиеся рядом (справа и слева) с символом '#'. Преобразовать все строки текста, записать новый текст в файл F2.

Задание 10.

1. Описать функцию работы со строкой символов, которая найдет вхождение в строку самой длинной последовательности пробелов, и определит ее длину.

2. Дан текст (несколько строк) в файле F1. Описать функцию, которая удалит из исходной строки все слова, имеющие окончание "КАЯ" или "КОЕ". Преобразовать все строки текста, записать новый текст в файл F2.

Задание 11.

1. Описать функцию работы со строкой символов, которая найдет первое вхождение подстроки, заключенной в скобки.

2. Дан текст (несколько строк) в файле F1. Описать функцию, которая после каждого пробела вставит в текст восклицательный знак. Преобразовать все строки текста, записать новый текст в файл F2.

Задание 12.

1. Описать функцию работы со строкой символов, которая удалит из строки первое и последнее слова.

2. Дан текст (несколько строк) в файле F1. Описать функцию, которая формирует из исходной строки новую, записывая в нее только подстроки, заключенные в круглые скобки (вместе со скобками). Преобразовать все строки текста, записать новый текст в файл F2.

Задание 13.



1. Описать функцию работы со строкой символов, которая найдет, сколько слов строки начинаются с некоторой буквы, заданной произвольно (задан как один из параметров функции).

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга одним пробелом. Описать функцию преобразования строки: из исходной строки в новую строку переписать только те слова, длина которых больше трех, отделяя их пробелами. Преобразовать все строки текста, записать новый текст в файл F2.

Задание 14.

1. Описать функцию работы со строкой символов, которая найдет, сколько из слов строки начинаются и заканчиваются одной буквой, например, "абракадабра", "сервис", "кулак" и пр.

2. Дан текст (несколько строк) в файле F1. Описать функцию, которая удаляет из строки все сочетания букв "АЯ" и "ОЕ". Преобразовать все строки текста, записать новый текст в файл F2.

Задание 15.

1. Описать функцию работы со строкой символов, которая получит массив длин слов этой строки.

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга произвольным числом пробелов. Описать функцию преобразования строки, которая каждое слово отделяет от другого слова сочетанием трех символов "\*\*\*". Преобразовать все строки текста, записать новый текст в файл F2.

Задание 16.

1. Описать функцию работы со строкой символов, которая найдет, есть ли в строке символы, отличные от букв, пробела и точки.

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга одним символом пробела. Описать функцию, которая меняет местами первое слово с последним словом. Преобразовать все строки текста, записать новый текст в файл F2.

Задание 17.

1. Для заданного натурального числа  $n$  получить его правильное символьное представление в виде строки, где триады цифр отделены друг от друга пробелами. Например, для  $n = 1753967$  запись должна иметь вид: "1 753 976".

2. Дан текст (несколько строк) в файле F1. Описать функцию преобразования строки, которая заменяет все последовательности цифр, стоящих подряд, одним символом решетки '#'. Преобразовать все строки текста, записать новый текст в файл F2.

Задание 18.

1. Для заданного натурального числа  $n$  ( $n \leq 20$ ) получить его правильное текстовое представление в виде строки текста, например, "три", "девятнадцать", "одиннадцать" и пр.

2. Дан текст (несколько строк) в файле F1. Описать функцию, которая формирует новую строку, заносая в нее символы исходной строки по 10 через

каждые десять. То есть, с 1-го по 10-тый, с 21-го по 30-й и т.д. Преобразовать все строки текста, записать новый текст в файл F2.

Задание 19.

1. Описать функцию работы со строкой символов, которая проверит, начинается ли каждое слово с большой буквы.

2. Дан текст в файле F1 в виде:

ИМЯ ОТЧЕСТВО ФАМИЛИЯ\_1

ИМЯ ОТЧЕСТВО ФАМИЛИЯ\_2

...

Описать функцию, которая формирует текстовую строку в формате:  
ФАМИЛИЯ И.О.

Сохранить преобразованный текст в файле F2.

Задание 20.

1. Дано натуральное число  $n$  ( $n \leq 100$ ), означающее возраст человека в годах, например, 1, 11, 51 и т.д. Получить текстовую строку, выражающую возраст человека. Например, "Один год", "Одиннадцать лет", "Пятьдесят один год".

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга пробелами. Описать функцию, которая находит самое длинное слово в строке, и удаляет его. Преобразовать все строки текста и записать новый текст в файл F2.

Задание 21.

1. Описать функцию работы со строкой символов, которая найдет, есть ли в строке правильное соответствие открывающих и закрывающих скобок.

2. Дан текст (несколько строк) в файле F1. Описать функцию преобразования строки, которая заменяет все подряд идущие пробелы одним вхождением пробела, и удаляет все символы, отличные от букв (знаки препинания и прочие). Записать новый «чистый» текст в файл F2.

Задание 22.

1. Описать функцию работы со строкой символов, которая найдет количество пар открывающих и закрывающих скобок разного вида. Функции вернет 1, если есть соответствие пар, и 0, если соответствия нет.

2. Дан текст (несколько строк) в файлах F1 и F2. Тексты состоят из слов, отделенных друг от друга одним пробелом. Описать функцию, которая определяет, насколько совпадают данные в файлах (по словам). Все несовпадающие слова вывести в файл G, сначала слово из первого файла, затем слово из второго файла.

Задание 23.

1. Описать функцию работы со строкой символов, которая найдет, сколько в строке знаков препинания.

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга пробелами. Описать функцию, которая каждый одиночный пробел заменяет пятью пробелами, идущими подряд. Преобразовать все строки текста, записать новый текст в файл F2.

Задание 24.

1. Описать функцию работы со строкой символов, которая проверит, есть ли ровно один пробел после каждого знака препинания.

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга пробелами или сочетанием символов «точка пробел». Описать функцию, формирующую словарь данного текста в виде массива слов. Повторяющиеся слова не включать. Вывести словарь в файл F2.

Задание 25.

1. Описать функцию работы со строкой символов, которая возвращает первое слово строки. Описать функцию работы со строкой символов, которая возвращает последнее слово строки.

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга пробелами. Описать функцию, которая найдет все вхождения некоторого слова в строке, и заменит его некоторым другим словом (эти слова задать как параметры функции). Преобразовать все строки текста, записать новый текст в файл F2.

Задание 26.

1. Описать функцию работы со строкой символов, которая находит указанное слово, а если его нет, то возвращает первое слово строки.

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга пробелами или знаками препинания. Описать функцию, формирующую частотный алфавит данного текста в виде массива символов (букв). Вывести частотный алфавит в файл F2.

Задание 27.

1. Описать функцию работы с текстом, представленным в виде строки символов, которая найдет среднее число слов в предложениях этого текста.

2. Дан текст (несколько строк) в файле F1. Текст состоит из потока предложений, отделенных друг от друга знаками препинания. Описать функцию, которая читает текст, выделяет одно предложение, и выводит в текстовый файл F2 по одному предложению в строке.

Задание 28.

1. Описать функцию работы с текстом, представленным в виде строки символов, которая найдет среднюю длину всех слов этого текста.

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга пробелами. Описать функцию, которая находит самое длинное слово в тексте, а затем формирует в новом файле форматированный текст, в котором каждое слово дополнено пробелами до длины самого длинного слова. Записать новый текст в файл F2.

Задание 29.

1. Описать функцию работы с текстом, представленным в виде строки символов, которая нумерует строку, добавляя в начало строки текстовое представление номера, переданного в функцию как параметр.

2. В файле F1 даны натуральные числа ( $n < 1000$ ), означающие некоторую цену в копейках, например, 317, 5005, 100 и пр. Описать функцию, которая формирует текстовую строку, выражающую цену в рублях и копейках. Например,

"3 руб. 17 коп.", "50 руб. 05 коп.", "1 руб. 00 коп.". Записать числа исходного файла и новый текст в файл F2.

Задание 30.

1. Описать функцию работы со строкой символов, которая возвращает значение слова, номер которого задан как параметр функции.

2. Дан текст (несколько строк) в файле F1. Текст состоит из слов, отделенных друг от друга пробелами. Описать функцию, которая в строке текста находит все слова, однокоренные некоторому заданному корню, и формирует из них новую строку. Преобразовать все строки текста, записать новый текст в файл F2.

Задание 31.

1. Описать функцию работы со строкой символов, которая найдет, сколько в строке символов, которые являются прописными или строчными буквами русского алфавита.

2. Дан текст (несколько строк) в файле F1. Текст состоит из предложений. Описать функцию, которая, игнорируя первоначальное деление файла на строки, получает строку, а потом переформатирует ее так, чтобы каждая строка содержала бы ровно 60 символов (добавлять '\n'). Записать новый текст в файл F2.

Задание 32.

1. Описать функцию работы со строкой символов, которая найдет, сколько в строке символов, которые являются знаками препинания.

2. Дан текст (несколько строк) в файле F1. Текст состоит из строк. Описать функцию, которая меняет местами строки, упорядочивая их в алфавитном порядке. Записать новый текст в файл F2.

Задание 33.

1. Описать функцию работы со строкой символов, которая по значению некоторого слова (параметр функции) строит все его анаграммы (возможно, бессмысленные) с сохранением в массиве слов.

2. Дан текст (несколько строк) в файле F1. Текст состоит из строк, представляющих даты в формате "Число. Месяц. Год". Описать функцию, которая меняет местами строки, упорядочивая их в порядке возрастания дат. Записать новый текст в файл F2.

Задание 34.

1. Описать функцию работы со строкой символов, которая после каждого слова вставит в строку символ табуляции.

2. Дан текст (несколько строк) в файле F1. Описать функцию работы с текстом, представленным в виде строки символов, которая «шифрует» строку путем прибавления к коду символа некоторого постоянного числа, а также функцию обратного преобразования. Шифрованные строки записывать в файл F2.

Задание 35.

1. Описать функцию работы со строкой символов, которая удалит из строки все Esc-последовательности.

2. Дан текст (несколько строк) в файле F1. Описать функцию преобразования строки символов, которая выделяет лексемы, представляющие последовательность цифр, подсчитывает число символов в лексеме, и вписывает

это число в строку после лексемы в виде числа, заключенного в скобки. Например, если фрагмент строки "*в целом 115 человек*", то результат будет "*в целом 115 (3) человек*". Преобразованные строки записывать в файл F2.

## Тема 12. Работа со структурами и объединениями

Структура позволяет объединить в единое целое произвольное количество данных различных типов. Поэтому все задачи, которые требуют логического объединения каких-то данных, могут и должны использовать структурный тип. Данные, входящие в структуру, достаточно независимы. Они имеют собственные имена и возможность выполнить над ними любые операции, разрешенные их типом.

Рассмотрим пример использования структур в некоторой модельной задаче, то есть такой, которая не имеет особенного практического смысла, но позволяет продемонстрировать достоинства структурного типа, его возможности и механизмы. Пусть структура объединяет некую информацию об одном абстрактном объекте (человеке): имя, фамилия, и какие-то данные числового характера, например, возраст, рост, средняя зарплата и т.д., назовем их показателями. Пусть имеются сведения о некотором множестве объектов, то есть о некотором сообществе людей, например, студентов какого-либо факультета, сотрудников некоторой организации, случайной статистической выборки.

Объект называется *среднестатистическим*, если на нем достигается минимум модуля разности среднего арифметического значений его показателей со средним арифметическим всей группы объектов. Аналогично определяется уникальный объект (на нем достигается максимум).

Объект может быть назван *среднестатистическим по  $k$ -му параметру*, (уникальным по  $k$ -му параметру), тогда рассматриваются данные о значении  $k$ -го показателя этого объекта в сравнении со средним значением  $k$ -го показателя по всей совокупности объектов.

Выясним, кто в группе объектов является:

- а) среднестатистическим;
- б) среднестатистическим по отдельным показателям;
- в) просто поставим задачу найти объект в группе по ключевому значению.

Для описания одного объекта используется структура. Попытаемся отразить в примере возможности этого типа данных. В качестве полей структуры используем простые типы данных, массив и строки, причем одна строка будет статической, а вторая динамической. Для хранения имени используем статическую строку, для хранения фамилии динамическую, для хранения показателей используем массив (ограничимся пятью показателями). Информация о среднем арифметическом показателей каждого объекта может быть вычислена при получении объектом данных. Ее можно сохранить, для этого введем в структуру поля, имеющие смысл «Сумма показателей» и «Среднее арифметическое показателей», соответственно, данные целого и вещественного типов.

Поскольку в группе объектов несколько данных, следует объединить их в массив, где один элемент массива описывает один объект.

Выполним функциональную декомпозицию задачи, то есть, определим, какие алгоритмы обработки понадобятся для ее решения.

1. Сначала подумаем, откуда взять данные. Например, их можно ввести.
2. Для работы с одним объектом пригодятся функции ввода данных об одном объекте и вывода данных об одном объекте. Имя формального параметра пусть будет Map, это полные данные об одном объекте.
3. Чтобы иметь возможность визуально сравнить данные, требуется вывод общей информации, идеально подходит таблица.
4. Для поиска среднего арифметического по группе объектов и для поиска среднего по какому-то показателю требуются функции, которые возвращают значение среднего по всей группе объектов и по одному показателю. Эти функции похожи внешне и по смыслу, но первая будет работать со средним данным всего объекта, а вторая, получив в качестве параметра номер показателя, будет работать только с этим элементом массива показателей.

Функция поиска может вернуть номер элемента в массиве структур или указатель на объект. Покажем это на примере функций поиска. Функция поиска среднестатистического объекта вернет его номер в массиве структур. Имея функцию вывода, мы легко выведем его данные на экран с использованием операции разыменования []. Функция поиска среднего по  $k$ -му показателю вернет указатель на структуру. Имея функцию вывода, мы легко выведем его данные на экран с использованием операции разыменования \*.

Все эти функции требуют полные данные обо всей группе объектов. Формальный параметр, используемый в их описании, тоже имеет имя Map, но теперь он имеет совсем другую смысловую нагрузку, обозначая имя всей совокупности данных обо всех объектах, то есть о массиве объектов.

Поиск по ключевому значению, например, по фамилии, не внесет ничего нового, просто добавим его, чтобы лишний раз показать механизмы работы со строками символов.

```
#include <stdio.h>
#include <conio.h>
#include <io.h>
#include <math.h>
#include <string.h>
// Описание абстрактного типа «Объект» с именем Person использует инструкцию
// typedef, потому что объявлений объектов указанного типа будет несколько.
typedef struct
{
    char    Name [10];    // Поле «Имя», статический символьный массив.
                        // Символов в имени не более 10-ти.
    char    *Surname;    // Поле «Фамилия»,
                        // динамический символьный массив.
                        // Для него должна быть выделена динамическая
                        // память перед присваиванием значения.
    int     Data [5];    // Поле «Показатели», статический массив данных.
```

```

int      Sum;          // Поле «Сумма» накапливает и сохраняет
                        // сумму баллов.
float    Var;          // Поле «Среднее» вычисляет и сохраняет
                        // средний показатель.
} Person; // Имя типа данного "Person"
// Функция ввода данных об одном объекте возвращает объект.
void In (Person & Man) // Одно данное типа Person возвращается по ссылке
{
    puts ("Введите имя: ");
    scanf ("%s", &Man.Name);
    puts ("Введите фамилию: ");
    Man.Surname = new char [10]; // Обязательно выделить память.
                                // Число символов фамилии не более 10-ти.
    scanf ("%s", Man.Surname); // & при вводе значения указателя не пишем.
    puts ("Введите данные: ");
    Man.Sum = 0;
// Будет накоплена сумма показателей при вводе.
    for (int i = 0; i < 5; i++)
    {
        scanf ("%4d", &Man.Data [i]);
        Man.Sum += Man.Data[i];
    }
    Man.Var = (float) Man.Sum / 5.; // Вычислен средний показатель.
}
// Функция вывода полных данных об одном объекте.
void Out (Person Man) // Передается данное типа Person
{
    printf ("%s ", Man.Name);
    printf ("%s ", Man.Surname);
    for (int i = 0; i < 5; i++)
        printf ("%4d", Man.Data[i]); // Показатели выводятся одной строкой.
    printf ("\nСумма баллов: %d ", Man.Sum);
    printf ("Среднее: %6.2f\n", Man.Var);
}
// Функция вывода данных обо всех объектах в форме таблицы.
void Out_All (Person Man [], int n) // Передается n данных типа Person
{
    printf ("=====\n");
    printf ("ИМЯ \t ФАМИЛИЯ \t ДАННЫЕ \t СУММА \t СРЕДНЕЕ\n");
    printf ("=====\n");
    for (int k = 0; k < n; k++)
    {
        printf ("% -10s ", Man[k].Name);
        printf ("% -10s ", Man[k].Surname);
    }
}

```

```

        for (int i = 0; i < 5; i++)
            printf ("%4d", Man[k].Data[i]);
        printf ("%4d ", Man[k].Sum);
        printf ("%6.2f \n", Man[k].Var);
    }
    printf ("===== \n");
}
// Функция вычисляет средний показатель по всем объектам, суммируя данные
// полей Var по всему массиву Man.
float Sred (Person Man [], int n)      // Передается n данных типа Person
{
    float Var = 0;                      // Переменная Var не имеет отношения к полю
                                        // структуры с именем Var.
    for (int i = 0; i < n; i++)
        Var += Man[i].Var;
    return Var / (float) n;              // Среднее по всем средним.
}
// Функция вычисляет средний показатель по k-тому параметру.
// Номер показателя (от 0 до 5), это один из параметров функции.
float Sred_k (Person Man [], int n, int k)  // Передается n данных типа Person
{                                           // и номер показателя k.
    float Var_k = 0;                      // Var_k – средний по одному показателю.
    for (int i = 0; i < n; i++)
        Var_k += (float) Man[i].Data[k]; // Складывает один k-тый показатель по
                                        // всем объектам.
    return Var_k / (float) n;              // Среднее по k-тому показателю.
}
// Функция определяет номер среднестатистического объекта в массиве.
int Sred_Stat (Person Man [], int n)      // Передается n данных типа Person
{
    float Var_All = Sred (Man, n);         // Средний показатель по всем объектам.
    float Var_min = fabs (Var_All – Man[0].Var);
    int Nom = 0;                          // Наименьший имеет номер 0.
    for (int i = 0; i < n; i++)
        if ( fabs (Var_All – Man[i].Var) < Var_min)
            Nom = i;                      // Запоминаем номер самого среднего.
    return Nom;
}
// Функция определяет объект, среднестатистический по k-тому показателю,
// и возвращает указатель на найденный объект.
Person * Sred_Stat_k (Person Man [], int n, int k) // Передается n данных типа Person
{                                           // и номер показателя.
    float Var_k = Sred_k (Man, n, k);      // Находим среднее значение k-того
                                        // показателя по всем объектам.
}

```



```

float    Var_min = fabs (Var_k – Man[0].Data[k]);
Person   * Nom = Man;                                // Указатель Nom запоминает адрес
                                                    // объекта.

    for (int i = 0; i < n; i++)
        if ( fabs (Var_k – Man[i].Data[k]) < Var_min)
            Nom = Man + i;                            // Запоминаем адрес самого среднего.
    return Nom;
}
// Функция поиска объекта по фамилии. Передается n данных типа Person, и
// Who – строка, содержащая фамилию искомого объекта. Выполняется прямой
// поиск по всей группе объектов путем последовательного сравнения искомой
// строки с полем Surname каждого элемента массива Man.
// Функция возвращает указатель на найденный объект или NULL.
Person * Found_Fam ( char *Who, Person Man [], int n )
{
    for (int i = 0; i < n; i++)
        if ( strcmp (Who ,Man[i].Surname) == 0)        // ==0, значит, найдено.
            return Man+i;                                // Возвращает адрес.
    return NULL;                                         // Возвращает NULL, если поиск неудачен.
}
// Продemonстрируем текст программы, которая объявляет данные и
// осуществляет управление ими путем вызова функций обработки данных.
void main (void)
{
    int      n;
    // Объявлен массив структур, то есть данных типа Person.
    Person   All_Person [20];
    printf ("Введите количество\n");                    // Реальное количество данных n.
    scanf ("%d", &n);
    // Ввод данных выполняется в цикле вызова функции In ввода одного объекта.
    for (int i = 0; i < n; i++)
    {
        In (All_Person[i]);
        // Out (All_Person[i]);
    }
    // Вывод на экран полной таблицы.
    Out_All(All_Person,n);
    // Для поиска номера среднестатистического объекта вызывается функция
    // Sred_Stat. Ей передается весь массив структур.
    int      Found = Sred_Stat (All_Person, n);
    printf ( "Самый средний имеет номер: %d\n", Found);
    printf ( "Его данные: \n");
    Out (All_Person [Found]);
    // Снова выведем таблицу на экран.

```

```

    Out_All(All_Person,n);
// Для поиска значения объекта, среднестатистического по k-му показателю
// вызывается функция Sred_Stat_k. Ей передается весь массив структур.
// Она возвращает указатель на объект.
Person    *Found_k;
    for (int k = 0;k < 5;k ++)
    {
        printf ("Средний по %d -му показателю\n", k);
        Found_k = Sred_Stat_k (All_Person, n, k);
        Out (*Found_k);
    }
// Поиск объекта по фамилии.
char      Surname [10];      // Тоже фамилия, но не та, что в Person.
puts ("Введите строку для поиска\n");
scanf ("%s", Surname);      //
Person    * Who;             // Чтобы сохранить адрес, возвращенный функцией.
    Who = Found_Fam (Surname, All_Person, n);
    if (Who != NULL)         // Проверяем, найден объект или нет.
        Out (*Who);
    else
        puts("Такого нет.\n");
} // End of main

```

### **Варианты заданий**

Задание 1. Определить структуры, описывающие шар и точку в трехмерном пространстве.

Написать и протестировать функции для ввода и вывода данных, и для проверки, находится ли точка внутри заданного шара. Объявить массив точек, и выполнить проверку для каждой из них.

Задание 2. Определить структуру для описания понятия алгебраический полином.

Написать и протестировать функции для ввода полинома, вывода полинома, сложения и вычитания полиномов, дифференцирования полинома.

Задание 3. Определить структуру для регистрации автомашин. Она должна иметь следующие поля: фамилия владельца, дата регистрации, марка машины, год выпуска, цвет, регистрационный номер.

Написать и протестировать функции для регистрации новой машины, удаления машины из регистрационного списка, поиска машины по марке и по цвету, а также по сочетанию признаков «марка и цвет».

Задание 4. Определить структуру для регистрации автомашин. Она должна иметь следующие поля: фамилия владельца, дата регистрации, марка машины, год выпуска, цвет, регистрационный номер.

Написать и протестировать функции для регистрации новой машины, удаления машины из регистрационного списка, поиска машины по фамилии владельца, поиска по году выпуска.

Задание 5. Определить две структуры, описывающие точку в полярной системе координат и декартовой системе координат.

Написать и протестировать функции для ввода и вывода данных, для получения декартовых координат точки по заданным полярным координатам, для получения полярных координат точки по заданным декартовым координатам, и для вычисления расстояния между двумя точками, заданными в любой системе координат.

Задание 6. Определить структуру, описывающую прямоугольник со сторонами, параллельными осям координат (прямоугольник задаётся двумя точками – левой нижней и правой верхней).

Написать и протестировать функции для ввода и вывода прямоугольников, и для определения, пересекаются ли два произвольных прямоугольника. Написать функцию, которая вернет указатель на новый прямоугольник, или, если пересечения нет, NULL.

Задание 7. Определить структуру, описывающую багаж пассажира, с полями: количество вещей и общий вес вещей. Пусть имеются данные о багаже нескольких пассажиров, где информация о багаже каждого отдельного пассажира представляет собой соответствующую пару чисел.

Написать и протестировать функции для ввода и вывода общей информации о багаже. Найти число пассажиров, имеющих не более двух вещей. Найти число пассажиров, количество вещей которых превосходит среднее число вещей.

Задание 8. Определить структуру, описывающую багаж пассажира, с полями: количество вещей и общий вес вещей. Пусть имеются данные о багаже нескольких пассажиров, где информация о багаже каждого отдельного пассажира представляет собой соответствующую пару чисел.

Написать и протестировать функции для ввода и вывода общей информации о багаже. Определить, имеются ли два пассажира, багажи которых совпадают по числу вещей и различаются по весу не более чем на 0,5 кг. Выяснить, имеется ли пассажир, багаж которого превышает багаж каждого из остальных пассажиров и по числу вещей и по весу.

Задание 9. Определить структуру, описывающую багаж пассажира, с полями: количество вещей и общий вес вещей. Пусть имеются данные о багаже нескольких пассажиров, где информация о багаже каждого отдельного пассажира представляет собой соответствующую пару чисел.

Написать и протестировать функции для ввода и вывода общей информации о багаже. Выяснить, имеется ли пассажир, багаж которого состоит из одной вещи весом не менее 30 кг. Дать сведения о багаже, число вещей в котором не меньше, чем в любом другом багаже, а вес вещей не больше, чем в любом другом багаже с этим же числом вещей.

Задание 10. Определить структуру, описывающую сведения об ученике, с полями: имя, фамилия, название класса, в котором он учится (год обучения и буква).

Написать и протестировать функции для добавления нового ученика, для вывода информации об одном ученике и вывода общей информации. Выяснить,

имеются ли в школе однофамильцы. Выяснить, имеются ли однофамильцы в каких-нибудь классах.

Задание 11. Определить структуру, описывающую сведения об ученике, с полями: имя, фамилия, название класса, в котором он учится (год обучения и буква).

Написать и протестировать функции для добавления нового ученика, для вывода информации об одном ученике и вывода общей информации. Выяснить, в каких классах больше чем  $n$  учащихся. Выяснить, сколько человек учится в каждой параллели.

Задание 12. Определить структуру, описывающую сведения об ученике, с полями: имя, фамилия, название класса, в котором он учится (год обучения и буква), а также отметки, полученные учениками в последней четверти.

Написать и протестировать функции для добавления нового ученика, для вывода информации об одном ученике и вывода общей информации. Выяснить, сколько учеников имеют отметки не ниже 4. Выяснить, сколько учеников имеют тройки в четверти.

Задание 13. Определить структуру, описывающую дату, с полями число, название месяца, год. Написать и протестировать функции для ввода даты, вывода в формате «ЧЧ.ММ.ГГ» и «ЧЧ Месяц ГГ». Пусть есть некоторое количество дат, связанных с некоторым событием, например, исторической датой. Написать и протестировать функции для сортировки этих данных по дате и по названию события.

Задание 14. Определить структуру, описывающую сведения о книге, с полями: фамилия автора, название и год издания.

Написать и протестировать функции для добавления новой книги, для вывода информации об одной книге и вывода общей информации. Написать и протестировать функции для поиска по фамилии автора, по году издания, а также по совокупности этих признаков.

Задание 15. Определить структуру, описывающую сведения о книге, с полями: фамилия автора, название, год издания.

Написать и протестировать функции для добавления новой книги, для вывода информации об одной книге и вывода общей информации. Написать и протестировать функции для поиска по названию книги, и по ключевому слову названия. Например, если название книги «Информатика и программирование», то она должна быть найдена и по ключевому слову «Информатика», и по ключевому слову «Программирование». Если таких книг несколько, то сообщить сведения обо всех книгах.

Задание 16. Определить структуру, описывающую сведения о сотрудниках учреждения, с полями: фамилия, имя, отчество, занимаемая должность, номер телефона.

Написать и протестировать функции для добавления нового сотрудника, для вывода информации об одном человеке и вывода общей информации. Написать и протестировать функции для поиска сотрудника по фамилии и по должности.

Задание 17. Определить структуру, описывающую кубики: размер (длина

ребра в сантиметрах), цвет (красный, желтый, зеленый или синий) и материал (деревянный, металлический, картонный).

Написать и протестировать функции для ввода и вывода данных, а также функции поиска количества кубиков каждого из перечисленных цветов и их суммарного объема.

Задание 18. Определить структуру, описывающую кубики: размер кубика (длина ребра в сантиметрах), его цвет (красный, желтый, зеленый или синий) и материал (деревянный, металлический, картонный).

Написать и протестировать функции для ввода и вывода данных, а также функции поиска количества кубиков с указанными значениями «материал» и «длина ребра». Например, найти количество деревянных кубиков с ребром 3 см и количество металлических кубиков с ребром 5 см.

Задание 19. Определить структуру, описывающую сведения о веществах: название вещества, его удельный вес и проводимость (проводник, полупроводник, изолятор).

Написать и протестировать функции для ввода и вывода данных. Написать и протестировать функции для поиска по названию вещества и по проводимости. Найти удельные веса и названия всех проводников.

Задание 20. Определить структуру, описывающую сведения о веществах: название вещества, его удельный вес и проводимость (проводник, полупроводник, изолятор).

Написать и протестировать функции для ввода и вывода данных. Написать и протестировать функции для сортировки данных по любому из признаков.

Задание 21. Определить структуру, описывающую сведения об игрушках: название игрушки, ее стоимость и возрастные границы детей для которых она предназначена (например, для детей от двух до пяти лет).

Написать и протестировать функции для ввода и вывода данных. Написать и протестировать функцию поиска, чтобы получить сведения о названиях игрушек, цена которых не превышает  $k$  руб. и которые подходят детям  $n$  лет. Написать и протестировать функцию поиска, чтобы получить сведения о самом дорогом, например, конструкторе.

Задание 22. Определить структуру, описывающую сведения об игрушках: название игрушки, ее стоимость и возрастные границы детей для которых она предназначена (например, для детей от двух до пяти лет).

Написать и протестировать функции для ввода и вывода данных. Написать и протестировать функцию поиска по наименованию. Написать и протестировать функцию поиска, чтобы получить сведения о названиях наиболее дорогих игрушек (цена которых отличается от цены самой дорогой игрушки не более чем на 10 руб.).

Задание 23. Определить структуру, описывающую сведения об игрушках: название игрушки, ее стоимость и возрастные границы детей для которых она предназначена (например, для детей от двух до пяти лет).

Написать и протестировать функции для ввода и вывода данных. Написать и протестировать функцию поиска по диапазону цены. Написать и протестировать

функцию поиска, чтобы подобрать игрушку любую, кроме мяча, подходящую ребенку 3 лет, и дополнительно мяч так, чтобы суммарная стоимость игрушек не превосходила 100 руб.

Задание 24. Определить структуру, описывающую сведения об игрушках: название игрушки, ее стоимость и возрастные границы детей для которых она предназначена (например, для детей от двух до пяти лет).

Написать и протестировать функции для ввода и вывода данных. Написать и протестировать функцию поиска, чтобы подобрать игрушку по названию, по цене, и по возрастному диапазону. Например, все мячи ценой  $n$  руб., предназначенный детям от 3 до 8 лет.

Задание 25. Имеются сведения об изделиях, выпускаемых малым предприятием: наименование изделия, годовой план выпуска изделий в штуках, фактический поквартальный выпуск каждого изделия.

№п/п	Наименование изделия	Годовой план выпуска	Фактический выпуск по кварталам			
			I	II	III	IV
1	Стул	100	10	20	35	35

Определить структуру для представления этих данных.

Написать и протестировать функции для ввода данных и вывода в виде таблицы. Написать и протестировать функции для определения фактического выпуска изделий и процента выполнения плана по каждому виду изделий. Написать функцию поиска по наименованию с выводом общей информации.

Задание 26. Имеются сведения о сотрудниках учреждения в следующем виде: табельный номер сотрудника, его фамилия, имя, отчество, оклад. Определить структуру для представления этих данных.

Написать и протестировать функции для ввода данных и вывода в виде таблицы. Написать и протестировать функции для поиска по табельному номеру, по фамилии. Вычислить среднюю заработную плату всех сотрудников.

Задание 27. Имеются данные о выпускаемых изделиях: наименование изделия, артикул, себестоимость изделия и его цена. Определить структуру для представления этих данных. Написать и протестировать функции для ввода данных и вывода в форме полной ведомости выпускаемых изделий следующего вида:

№ п/п	Наименование изделия	Артикул	Себестоимость	Цена

Написать и протестировать функции для поиска изделий по наименованию и по диапазону цен. (Например, цена не превышает  $K_1$  руб., но более  $K_2$  руб.)

Задание 28. Известны сведения о себестоимости некоторых видов продукции: наименование,  $J_m$  – индекс изменения норм расхода на данный вид материальных затрат,  $J_c$  – индекс изменения оптовых цен на данный вид,  $J_p$  – индекс выпуска продукции,  $D_o$  – отношение стоимости основных и вспомогательных материалов к выпуску товарной продукции по отчету,  $T_{пп}$  – объем товарной продукции по плану. На основании этих данных может быть вычислена экономия от снижения себестоимости, которая вычисляется по формуле

$$\mathcal{E} = (1 - J_m \cdot J_c) \cdot D_o \cdot J_p \cdot T_{pp}.$$

Определить структуру для представления этих данных. Разработать функции для ввода данных и вывода в форме выходной ведомости следующего вида:

Наименование	$J_m$	$J_c$	$J_p$	$D_o$	$T_{pp}$	$\mathcal{E}$

Здесь  $\mathcal{E}$  – экономия, которая должна быть вычислена для каждого наименования. Написать и протестировать функции для добавления новых данных, а также для поиска наименований, для которых достигнута наибольшая и наименьшая экономия.

Задание 29. В магазине ведется учет продажи товаров. По каждому товару известны наименование товара, цена, торговая надбавка, количество проданного товара. Определить структуру для представления этих данных. Разработать функции для ввода данных и вывода в форме отчетной ведомости следующего вида:

Наименование товара	Цена (руб.)	Торговая надбавка (%)	Количество	Общая стоимость

Написать и протестировать функции для добавления данных о новом товаре, для поиска по наименованию, для вычисления общей итоговой стоимости продаж.

Задание 30. Известны данные о выпускаемой продукции по цехам предприятия: название цеха, отношение стоимости основных и вспомогательных материалов к выпуску продукции по отчету  $D_o$  и по плану  $D_p$ , объем товарной продукции по плану  $T_p$ .

Определить структуру для представления этих данных. Разработать функции для ввода данных и вывода в форме выходной ведомости следующего вида:

Цех	Отношение стоимости основных и вспомогательных материалов к объему выпускаемой продукции		Объем товарной продукции по плану	Экономия от снижения себестоимости
	$D_o$	$D_p$	$T_p$	$\mathcal{E}$

Здесь экономия от снижения себестоимости по каждому цеху вычисляется по формуле:  $\mathcal{E} = (D_o - D_p) \cdot T_p$ .

Написать и протестировать функции, чтобы найти цеха, для которых достигнута наибольшая и наименьшая экономия, а также для поиска информации по названию цеха.

Задание 31. В выпуске продукции принимают участие несколько бригад, для которых ведется помесечный учет. Бригады имеют номера. Определить структуру для представления этих данных. Разработать функции для ввода данных и вывода в форме выходной ведомости следующего вида:

Месяц	Номер бригады				Итоговая выработка по месяцам
	1	2	3	4	

Написать и протестировать функции, чтобы найти среднюю выработку по бригадам за весь отчетный период, а также для поиска сведений о выработке за указанный месяц по названию месяца.

Задание 32. Предприятие нанимает работников на сдельную работу. Для каждого из них ведется ежедневный учет: фамилия, имя, отчество, разряд, расценка за единицу продукции, количество произведенных изделий. Определить структуру для представления этих данных. Разработать функции для ввода данных и вывода в форме расчетной ведомости следующего вида:

Ф. И. О.	Разряд	Расценки	Количество	Сумма

Написать и протестировать функции, чтобы найти общую выработку за день и общую сумму к оплате. Сохранить эти данные в виде таблицы.

Дата	Количество произведено	Оплачено

Задание 33. Для каждого участка цеха завода имеются сведения о количестве отпущенных для производства материалов: наименование материала, единица измерения, объем отпущенного участку материала. Определить структуру для представления этих данных. Разработать функции для ввода данных и вывода с подведением итогов в форме выходной ведомости следующего вида:

Наименование	Ед. измерения	Участок 1	Участок 2	Участок 3	Итого

Написать и протестировать функции, чтобы найти общий расход материалов каждым участком, и всего по цеху, а также для поиска по наименованию.

Задание 34. На складе ведется учет наличия товаров. Для подведения итогов месяца есть сведения: наименование товара; количество поступившего и реализованного товара за месяц. Определить структуру для представления этих данных. Разработать функции для ввода данных и вывода в форме оборотной ведомости следующего вида:

Наименование товара	Поступило за месяц	Реализовано за месяц	Разница

Написать и протестировать функции, чтобы найти общий приход товара и общий расход, а также для вывода отчета об остатках товара на складе по форме:

Наименование товара	Остаток

Задание 35. На предприятие поступает сырье разного вида. Учет поступления сырья ведется по следующим позициям: вид сырья (наименование), плановое поступление, фактическое поступление. Определить структуру для представления этих данных. Разработать функции для ввода данных и вывода в форме отчетной ведомости следующего вида:

Вид сырья	Плановое поступление	Фактическое поступление	Процент выполнения плана

Написать и протестировать функции, чтобы найти общий объем плановых поступлений и общий объем фактических поступлений, а также для поиска по наименованию.

### Тема 13. Работа в графическом режиме



Данная тема включена как дополнительная. Работа в графическом режиме связана с использованием графических библиотек, принципы и содержание которых зависят от среды разработки. Задания этой темы предлагались для разработки в среде программирования Borland C++ 3.1 с использованием BGI графики, которая на данный момент времени устарела. Именно поэтому ее принципы в данном пособии не рассматриваются. Однако, задачи представляют интерес с точки зрения их алгоритмической разработки, и могут решаться в других графических средах. Они интересны еще и потому, что интегрируют знания различных разделов данного курса.

### Варианты заданий

#### Задание 1.

1. Нарисовать и оживить изображение автомобиля.
2. Столбиковая диаграмма представляет собой набор прямоугольников, основания которых равны, а высоты пропорциональны числовым величинам, взятым из некоторой совокупности исходных данных. Для большей наглядности прямоугольники диаграммы обычно закрашивают в разные цвета. Построить такую диаграмму для  $n$  исходных данных. Применять автоматическое масштабирование.
3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение точек относительно горизонтали, проходящей через середину экрана. Выделить различными цветами точки, лежащие выше и ниже этой линии.

#### Задание 2.

1. Нарисовать и оживить изображение «цыпленок, клюющий зерно».
2. Круговая диаграмма – это круг, площади секторов которого пропорциональны соответствующим числовым величинам, взятым из некоторой совокупности исходных данных. Для большей наглядности секторы диаграммы закрашивают в разные цвета. Построить такую диаграмму для  $n$  исходных данных. Применять автоматическое масштабирование.
3. В текстовом файле хранятся координаты центров  $n$  окружностей на координатной плоскости с заданным радиусом. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение окружностей. Выделить цветом концентрические окружности.

#### Задание 3.

1. Нарисовать и оживить изображение «взлет ракеты».
2. Построить графики функций:  $Y_1(x) = x^2$ ,  $Y_2(x) = x^3$ , где  $x \in [-2.5, 2.5]$ . Шаг по  $x$  принять равным 0,1, но предусмотреть возможность изменения. Применять автоматическое масштабирование.
3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране точки, а также линию, соединяющую эти точки в порядке обхода. Каждый отрезок линии выделить другим цветом.

#### Задание 4.

1. Нарисовать и оживить изображение «извержение вулкана».

2. Построить график функции:  $y(x)=e^{\sin(x)}$ , где  $x = [0.2, \pi]$ , шаг =  $\pi / 12$  . Применять автоматическое масштабирование.

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране точки. Выделить цветом точки, удовлетворяющие условию  $x > y$ .

Задание 5.

1. Нарисовать и оживить изображение физиономии клоуна.

2. Построить график функции:  $y(x) = x \cdot \cos(x) + x \cdot \sin(x)$ , где  $x = [0.2, \pi]$ , шаг  $x = \pi / 12$  . Применять автоматическое масштабирование.

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение точек относительно вертикали, проходящей через середину экрана. Выделить различными цветами точки, лежащие левее и правее этой линии.

Задание 6.

1. Нарисовать и оживить изображение «часовой циферблат» (будильник).

2. Построить график функции:  $y(x) = A \cdot x^3 + B \cdot x^2 + C$ , где  $x = [-5, +5]$  с шагом = 1. Значения  $A, B, C$  вводить. Применять автоматическое масштабирование.

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Построить на экране множество точек, а также отрезки, соединяющие точки с центром экрана. Каждый отрезок линии выделить другим цветом.

Задание 7.

1. Нарисовать и оживить изображение «домик». Имитировать зажигание света в окне, дым из трубы.

2. Построить графики функций:  $y_1(x) = x + \sin(x)$ ,  $y_2(x) = x + \cos(x)$ , где  $x = [-\pi, +\pi]$  с шагом  $\pi / 8$ . Применять автоматическое масштабирование.

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Построить на экране множество точек, а также отрезки, соединяющие каждую точку со всеми другими. Множество отрезков каждой точки выделить другим цветом.

Задание 8.

1. Нарисовать и оживить изображение поросенка.

2. Построить улитку Паскаля, которая параметрически задается следующим образом:

$$x = a \cdot \cos^2(t) + b \cdot \cos(t),$$

$$y = a \cdot \cos(t) \cdot \sin(t) + b \sin(t),$$

где  $a > 0, b > 0, t = [0, 2\pi)$ .

Рассмотреть случаи, когда  $b > 2a, a < b < 2a, a > b$  .

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Это координаты центров пересечения диагоналей квадратов со стороной указанной длины. Прочитать данные из файла в массивы. Построить на экране множество квадратов. Выделить цветом квадраты, пересекающиеся друг с другом.

Задание 9.

1. Нарисовать и оживить изображение «Чебурашка».
2. Изобразить кардиоду – кривую, заданную параметрически следующим образом:

$$x = a \cdot \cos(t) \cdot (1 + \cos(t)),$$

$$y = a \cdot \sin(t) \cdot (1 + \cos(t)),$$

где  $a > 0$ ,  $t = [0, 2\pi]$ .

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости, где  $n$  – четное число. Это координаты концов отрезков на плоскости. Задана окружность указанного радиуса с центром в середине экрана. Прочитать данные из файла в массивы. Изобразить взаимное расположение на плоскости отрезков и окружности. Выделить цветом отрезки, полностью лежащие внутри окружности.

Задание 10.

1. Нарисовать и оживить изображение «НЛО на фоне звездного неба».
2. Изобразить эпициклоиду – кривую, заданную параметрически следующим образом:

$$x = (a+b) \cdot \cos(t) - a \cdot \cos((a+b) \cdot t / a),$$

$$y = (a+b) \cdot \sin(t) - a \cdot \sin((a+b) \cdot t / a),$$

где  $a > 0$ ,  $b > 0$ .

Рассмотреть два случая:

- 1) Если  $b/a$  – целое положительное число,  $t = [0, 2\pi]$ .
- 2) Если  $b/a = p/g$ , где  $p$  и  $g$  – положительные целые взаимно простые числа,  $t = [0, 2g\pi]$ .

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение точек относительно диагонали, проходящей через середину экрана. Выделить различными цветами точки, лежащие выше и ниже этой линии.

Задание 11.

1. Нарисовать и оживить изображение летящей птицы.
2. Изобразить астроида – кривую, заданную параметрически уравнениями:

$$x = b \cdot \cos^3(t),$$

$$y = b \cdot \sin^3(t),$$

где  $t = [0, 2\pi]$ .

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Построить на экране множество точек, а также отрезки, соединяющие точки с верхним левым и правым нижним углами экрана. Каждую пару отрезков выделить другим цветом.

Задание 12.

1. Нарисовать и оживить изображение «стакан с лимонадом».
2. Построить строфоиду – кривую, которая задана параметрически следующим образом:

$$x = a \cdot \frac{t^2 - 1}{t^2 + 1}$$

$$y = a \cdot t \cdot \frac{t^2 - 1}{t^2 + 1}$$

где  $t = (-\infty, +\infty)$ ,  $a > 0$ .

3. В текстовом файле хранятся радиусы  $n$  окружностей на координатной плоскости. Прочитать данные из файла в массив. Изобразить на экране взаимное расположение окружностей, если центр находится в середине экрана. Выделить цветом каждую окружность.

Задание 13.

1. Нарисовать и оживить изображение «парусник в море».

2. Построить спираль вокруг начала координат с  $n$  витками и внешним радиусом  $r$ , начальное направление спирали образует с осью  $Ox$  угол 1 градус, если параметрическое представление спирали:

$$x = r \cdot \cos(t),$$

$$y = r \cdot \sin(t),$$

где  $r = t/2$ ,  $1 \leq t \leq 2\pi n$ .

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Построить на экране множество окружностей с центрами в указанных точках определенного радиуса. Выделить разными цветами окружности, центры которых лежат выше и ниже горизонтали, проходящей через середину экрана.

Задание 14.

1. Нарисовать и оживить изображение «вращение колеса».

2. Изобразить Циссоиду – кривую, заданную параметрически уравнениями:

$$x = \frac{a \cdot t^2}{1 + t^2}$$

$$y = \frac{a \cdot t^3}{1 + t^2}$$

где  $t = (-\infty, +\infty)$ ,  $a > 0$ .

3. В текстовом файле хранятся координаты центров и радиусы  $n$  окружностей на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение окружностей. Выделить цветом окружности, радиус которых равен некоторому указанному числу.

Задание 15.

1. Нарисовать и оживить изображение мыши или хомячка.

2. Изобразить конхоиду Никомеда – кривую, заданную параметрически уравнениями:

$$x = a + L \cdot \cos(t),$$

$$y = a \cdot \operatorname{tg}(t) + L \cdot \sin(t),$$

где правая ветвь получена для  $t = (-\pi/2; +\pi/2)$ ,

левая ветвь получена для  $t = (\pi/2; \frac{3\pi}{2})$ , при  $a > 0, L > 0$ .

Рассмотреть случаи, когда  $L < a, L > a, L = a$ .

3. В текстовом файле хранятся координаты центров и радиусы  $n$  окружностей на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение окружностей. Показать линию, соединяющую центры окружностей. Выделить разным цветом каждый отрезок.

Задание 16.

1. Нарисовать и оживить изображение «ветреный день».

2. Изобразить овалы Кассини по их уравнению в полярных координатах:

$$\rho^2 = c^2 \cos 2\varphi \pm \sqrt{c^4 \cos^2 2\varphi + (a^4 - c^4)}$$

Рассмотреть случаи, когда  $a > c\sqrt{2} > 0; 0 < c < a < c\sqrt{2}; 0 < a < c$ .

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости, где  $n$  – четное число. Это координаты концов отрезков на плоскости. Прочитать данные из файла в массивы. Изобразить взаимное расположение на плоскости отрезков. Выделить цветом те отрезки, которые пересекаются.

Задание 17.

1. Нарисовать и оживить изображение «пляшущие человечки».

2. Построить график функции:  $y(x) = \cos(x-1) + |x|$ , где  $x = [-\pi, +\pi]$  с шагом  $\pi/8$ . Применять автоматическое масштабирование.

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости, где  $n$  четное число. Это координаты концов отрезков на плоскости. Прочитать данные из файла в массивы. Изобразить взаимное расположение на плоскости отрезков. Выделить цветом отрезки, лежащие полностью правее вертикали, проходящей через середину экрана.

Задание 18.

1. Нарисовать и оживить изображение «растущее дерево».

2. Изобразить Спиорограф, зубчатый диск радиуса  $b$ , расположенный внутри колеса радиуса  $a$ , который вращается и находится в зацеплении с внешним колесом. В диске имеется метка на расстоянии  $d$  от центра диска, имеющая перо, которое и вычерчивает узор. Уравнение вычерчиваемой кривой в параметрических координатах имеет вид:

$$x = (a - b) \cdot \cos(t) + d \cdot \cos(\varphi),$$

$$y = (a - b) \cdot \sin(t) - d \cdot \sin(\varphi),$$

$$\text{где } \varphi = \frac{a}{b}t; \text{ и } d < b < a.$$

Угол  $t$  меняется от 0 до  $2\pi n$ , где  $n$  равно  $b$ , поделенному на наибольший общий делитель чисел  $b$  и  $a$ .

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране множество окружностей с центрами в указанных точках и случайными значениями радиусов, выбранных из диапазона 1–200. Выделить различными цветами окружности, радиусы которых больше или меньше 100.

#### Задание 19.

1. Нарисовать и оживить изображение «дерево на ветру».
2. Изобразить усеченный конус, если заданы координаты центров оснований и радиусы оснований. Имитировать вращение вокруг центральной оси.
3. В текстовом файле хранятся координаты центров и радиусы  $n$  окружностей на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение окружностей. Выделить цветом окружности, центры которых лежат выше диагонали, проходящей через середину экрана. Выделить различными цветами окружности, лежащие выше и ниже этой линии.

#### Задание 20.

1. Нарисовать и оживить изображение – рекламу газированной воды.
2. Построить треугольную пирамиду по заданным координатам вершин. Две видимые грани закрасить. Имитировать вращение вокруг высоты.
3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране точки. Выделить цветом точки, принадлежащие вертикальной полосе, делящей экран на три части.

#### Задание 21.

1. Нарисовать и оживить изображение – рекламу жевательной резинки.
2. Построить параллелепипед по заданным координатам вершин. Две видимые грани закрасить. Имитировать вращение вокруг высоты.
3. Изобразить  $n$  усеченных конусов, если координаты центров оснований и радиусы оснований заданы в текстовом файле. Предварительно данные прочитать из файла в массивы.

#### Задание 22.

1. Нарисовать и оживить изображение «бегущие облака».
2. Изобразить усеченный конус, в основаниях которого эллипсы с радиусами  $r_1$  и  $r_2$ , где  $r_1 < r_2 \leq 60$ . Имитировать вращение вокруг центральной оси.
3. Построить  $n$  прямоугольников на плоскости по координатам вершин, заданным в текстовом файле. Предварительно данные прочитать из файла в массивы.

#### Задание 23.

1. Нарисовать и оживить изображение «крушение самолета».
2. Построить в координатной плоскости ( $xOy$ ) куб, ребро которого  $n \leq 100$ , закрасить две грани куба. Имитировать вращение вокруг одного из ребер.
3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Построить на экране множество точек, а также отрезки, соединяющие точки с центром экрана. Каждую пару отрезков выделить другим цветом.

#### Задание 24.

1. Нарисовать и оживить изображение «поздравление с новым годом».
2. Построить плоский прямоугольник по заданным координатам вершин. Имитировать вращение вокруг любой из сторон.
3. В текстовом файле хранятся радиусы  $n$  окружностей на координатной

плоскости. Прочитать данные из файла в массив. Изобразить на экране взаимное расположение окружностей, если центр первой окружности – в середине экрана, а последующие смещены вправо с шагом, равным 10. Если окружность выходит за край экрана, продолжить их рисование слева экрана. Выделить цветом каждую окружность.

Задание 25.

1. Нарисовать и оживить изображение «поздравление с днем 8 марта».
2. Изобразить равносторонний треугольник заданного размера. Имитировать вращение вокруг центра.
3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Построить на экране множество окружностей с центрами в указанных точках определенного радиуса. Выделить разными цветами окружности, центры которых лежат выше и ниже главной диагонали экрана.

Задание 26.

1. Нарисовать и оживить изображение «поздравление с днем рождения».
2. Построить синусоиду  $y(x) = A \cdot \sin(x + \varphi)$  для двух периодов. Дать возможность пользователю ввести значения  $A$ ,  $\varphi$ . Показать, как изменяется график функции при изменении параметров.
3. В текстовом файле хранятся высоты и радиусы оснований  $n$  цилиндров. Прочитать данные из файла в массивы. Изобразить на экране цилиндры в пространстве, считая координаты их оснований на плоскости  $xOy$  одинаковыми. Выделить цветом каждый цилиндр.

Задание 27.

1. Нарисовать и оживить изображение – рекламу мыла.
2. Построить синусоиду  $y(x) = A \cdot \sin(x + \varphi)$  для двух периодов. Изобразить изменение волны смещением графика функции вправо (влево) при изменении  $\varphi$  в произвольном диапазоне.
3. В текстовом файле хранятся координаты центров и радиусы  $n$  окружностей на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение окружностей. Показать линию, соединяющую центры окружностей. Выделить разным цветом каждый отрезок.

Задание 28.

1. Нарисовать и оживить изображение «олимпийские кольца».
2. Построить синусоиду  $y(x) = A \cdot \sin(x + \varphi)$  для двух периодов. Изобразить изменение волны при увеличении (уменьшении) амплитуды  $A$  в произвольном диапазоне. Выделить цветом каждый новый график.
3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости, а также координаты и центр окружности. Прочитать данные из файла в массивы. Построить на экране множество точек и окружность. Выделить цветом точки, попадающие внутрь окружности.

Задание 29.

1. Нарисовать и оживить изображение «развевающийся российский флаг».
2. Построить плоский прямоугольник по заданным координатам вершин.

Имитировать вращение относительно одной вершины.

3. В текстовом файле хранятся радиусы и координаты центров  $n$  окружностей на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение окружностей. Выделить цветом те из них, которые пересекаются.

Задание 30.

1. Нарисовать и оживить изображение «разбегающиеся по воде круги».

2. Изобразить месяц. По нажатию клавиши Enter повернуть его влево или вправо, по нажатию пробела качнуть.

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Каждую из них считать центром пересечения диагоналей некоторого квадрата, длина стороны которого равна, например, 10. Прочитать данные из файла в массивы. Построить на экране множество квадратов с центрами в указанных точках. Выделить разными цветами квадраты, лежащие выше горизонтали, проходящей через середину экрана.

Задание 31.

1. Нарисовать и оживить изображение смайликов.

2. Изобразить солнышко. По нажатию клавиш Enter и Esc увеличивать (уменьшать) длину лучей, по нажатию стрелок передвинуть на какое-то расстояние направо (налево).

3. В текстовом файле хранятся высоты и радиусы оснований  $n$  конусов. Прочитать данные из файла в массивы. Изобразить на экране конусы в пространстве, считая координаты центров нижнего основания на плоскости  $xOy$  одинаковыми. Выделить цветом каждый конус.

Задание 32.

1. Нарисовать и оживить изображение «поздравление с днем 23 февраля».

2. Изобразить НЛО на фоне звездного неба. По нажатию клавиш стрелок управлять перемещением объекта по экрану.

3. В текстовом файле хранятся координаты центров и радиусы  $n$  окружностей на координатной плоскости. Прочитать данные из файла в массивы. Изобразить на экране взаимное расположение окружностей. Показать отрезки, соединяющие центры окружностей с центром экрана. Выделить разным цветом каждый отрезок.

Задание 33.

1. Нарисовать и оживить изображение ночного неба на тему вечеров на хуторе близ Диканьки.

2. Изобразить снегопад. По нажатию клавиш стрелок вверх или вниз управлять интенсивностью снежного потока.

3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Дана прямая уравнением  $y=A \cdot x+B$ . Прочитать данные из файла в массивы. Построить на экране прямую линию в некотором масштабе, и изобразить расположение множества точек относительно прямой. Выделить разным цветом точки, лежащие по разные стороны прямой.

Задание 34.



1. Нарисовать и оживить изображение фейерверка в ночном небе.
2. Построить плоский прямоугольник по заданным координатам вершин. По нажатию стрелок вправо, влево имитировать вращение относительно прямой линии, проходящей через центр прямоугольника.
3. В текстовом файле хранятся высоты и радиусы оснований  $n$  прямых круговых конусов. Прочитать данные из файла в массивы. Найти средний объем конусов. Изобразить на экране конусы в пространстве. Те из них, объем которых больше среднего, изобразить в левой части экрана, те, объем которых меньше среднего, в правой части. Выделить цветом.

Задание 35.

1. Нарисовать и оживить изображение собачки или кошечки.
2. Изобразить змею. По нажатию клавиш стрелок управлять перемещением объекта по экрану.
3. В текстовом файле хранятся координаты  $n$  точек на координатной плоскости. Прочитать данные из файла в массивы. Построить на экране множество точек, при этом выделить цветом те из них, координаты которых удовлетворяют неравенству  $|y| + 2|x| \leq x^2 + 1$ .

## Тема 14. Полустатические структуры данных

Во всех задачах этого раздела предполагается реализация структур данных на основе массива. Необходимо пользоваться только теми операциями, которые определены для используемых структур данных.

### Стек

**Стеком** называется упорядоченный набор элементов, в котором размещение новых элементов и удаление существующих производится только с одного его конца, называемого вершиной стека. При представлении стека в статической памяти для него выделяется память, как для вектора. В дескрипторе этого вектора кроме обычных для вектора параметров должен находиться также указатель стека – адрес вершины стека. Указатель стека может указывать либо на первый свободный элемент стека, либо на последний записанный в стек элемент. При работе со стеками разрешается пользоваться следующими операциями: вталкивание элемента в стек, выталкивание элемента из стека, определение пустоты и переполнения стека.

Стек должен быть описан следующим образом:

```
struct    STACK
{
    int    data [ST_SIZE];
    int    top;
};
```

Элементы стека хранятся в массиве размером ST\_SIZE. Переменная top является указателем на вершину стека.

### Варианты заданий

Задание 1. Разработайте следующие функции для обслуживания стека:

- добавление элемента в стек;
- печать стека;
- создание стека;
- извлечение (удаление) элемента из стека.

Задание 2. Разработайте комплекс функций для обслуживания стека:

- создание стека;
- добавление элемента в стек;
- печать стека;
- упорядочивание элементов в стеке по полю данных.

Задание 3. Выполните синтаксический разбор арифметического выражения. Выражение содержит бинарные операции '+' и '-'. В качестве операндов для простоты используются цифры от 0 до 9. Выражение заканчивается знаком '='. Например,  $9-4+6=$ . Используйте два стека: стек операций, элементы которого имеют тип `char`, и стек операндов, содержащий элементы целого типа.

Задание 4. Вычислите арифметическое выражение. Выражение содержит бинарные операции '+' и '-'. В качестве операндов для простоты используются цифры от 0 до 9. Выражение является синтаксически правильным и заканчивается знаком '='. Например,  $5+3-9=$ . Используйте два стека: стек операций, элементы которого имеют тип `char`, и стек операндов, содержащий элементы целого типа.

Задание 5. Напишите программу, заполняющую стек числами, введенными с клавиатуры и затем меняющую местами верхнюю и нижнюю половины стека (т.е. из стека  $1 - 2 - 3 - 4$  должен получиться стек  $4 - 5 - 6 - 1 - 2 - 3$ ). При этом можно пользоваться только функциями добавления и выталкивания элементов.

Задание 6. Определите правильность расстановки скобок в арифметическом выражении, используя стеки. Возможно использование скобок трех типов: круглые скобки `()`, квадратные скобки `[]` и фигурные скобки `{ }`.

Задание 7. Используя стек, решить следующую задачу: напечатать содержимое текстового файла, выписывая символы каждой его строки в обратном порядке.

Задание 8. Дана строка вида  $s*w$  ( $s, w$  – строки из символов, не содержащих символа `*`). Чтение разрешено по одному символу. Проверить, является ли строка  $w$  обратной строке  $s$ . Например, для случая  $s = ABCDEF, w = FEDCBA$  ответ положительный.

Задание 9. Вычислите значение выражения, представленного в обратной польской записи (в постфиксной форме). Выражение состоит из цифр от 1 до 9 и знаков операций.

Обычная запись:

$(b + c) * d$

$a + (b + c) * d$

Обратная польская запись:

$bc + d *$

$a b c + d * +$

**Указание.** Просматривая строку, анализируем очередной символ: если это цифра, то записываем ее в стек; если это знак операции, то читаем два элемента, выполняем математическую операцию, определяемую этим знаком, и заносим результат в стек.

Задание 10. Разработайте программу преобразования выражения из

инфиксного в префиксное представление.

Задание 12. Разработайте программу вычисления выражения в префиксной форме.

Задание 12. Разработайте программу преобразования постфиксной формы в инфиксную. При этом результирующее выражение должно быть записано с необходимыми скобками. Например,  $AB + C *$  должно быть преобразовано в  $(A+B)*C$

### Очередь

**Очередью** называется упорядоченный набор элементов, которые могут удаляться с одного ее конца (называемого началом очереди) и помещаться в другой конец этого набора (называемого концом очереди).

Очередь мы будем описывать так:

```
struct    QUEUE
{
    int    data [QUE_SIZE];
    int    head;
    int    tail;
};
```

Как и ранее, мы предполагаем, что данные хранятся в массиве, размер которого равен значению переменной `QUE_SIZE`. Точно также переменная `head` указывает на голову очереди, а переменная `tail` – на первую свободную ячейку в ее хвосте.

Как мы уже знаем, над очередью выполняются следующие операции:

- создать очередь;
- добавить элемент в очередь;
- изъять элемент из очереди;
- очередь пуста (да/нет);
- очередь полная (да/нет);
- определение текущей длины очереди.

### Варианты заданий

Задание 13. Напишите программу, обрабатывающую очередь пассажиров, желающих купить авиабилет. Элементы очереди содержат следующую информацию: фамилия пассажира, пункт прибытия, дата вылета. Информация об имеющихся билетах хранится в списке, элементы которого содержат номер рейса, пункт прибытия, дату, число непроданных билетов. Пассажир, обеспеченный билетом, удаляется из очереди, остальные остаются там. Содержимое списка по мере обеспечения пассажиров билетами корректируется.

Задание 14. Напишите программу, формирующую очередь целых чисел и затем удаляющую из нее все числа, меньшие заданного числа (использовать можно только функции добавления и изъятия элементов).

Задание 15. Напишите программу, определяющую, является ли введенная строка символов палиндромом (то есть читается ли она одинаково слева направо и справа налево). Используйте для решения одну из рассмотренных структур.

Задание 16. За один просмотр файла, элементами которого являются целые числа, и без использования дополнительных файлов переписать его элементы в другой файл так, чтобы первоначально были записаны все числа, меньшие заданного, а затем все числа из отрезка  $[a, b]$  и все остальные. Взаимный порядок чисел в каждой из групп должен быть сохранен.

*Указания.* В решении задачи числа последовательно считываются из файла. Если очередное число меньше  $a$ , то оно записывается в файл, если оно принадлежит отрезку  $[a, b]$ , то заносится в первую очередь, иначе – во вторую. После завершения чтения в выходной файл записываются числа из первой и второй очередей.

Задание 17. Содержимое текстового файла  $f$ , разделенного на строки, переписать в текстовый файл  $g$ , перенося при этом в конец каждой строки все входящие в него цифры, с сохранением взаимного исходного порядка.

### Дек

**Дек**, – это особый вид очереди. Последовательная структура данных, в которой как включение, так и исключение элементов может осуществляться с любого из двух концов. Логическая и физическая структуры дека аналогичны логической и физической структуре кольцевой FIFO очереди. Однако применительно к деку, целесообразно говорить не о начале и конце очереди, а о левом и правом конце дека.

```
struct    DEQ
{
    int    data[DEQ_SIZE];
    int    left;
    int    right;
};
```

Элементы дека хранятся в массиве размером `DEQ_SIZE`, дек содержит два указателя: `left` – на левый конец дека и `right` – на правый конец.

Над деком выполняются следующие операции:

- очистка дека;
- включение элемента справа;
- включение элемента слева;
- исключение элемента справа;
- исключение элемента слева;
- определение размера дека.

### Варианты заданий

Задание 17. Вывести в порядке возрастания первые  $n$  натуральных чисел, в разложении которых на простые множители входят только числа 2, 3 и 5.

*Указание.* Вводятся три дека `Dec1`, `Dec2`, `Dec`, в которых хранятся числа, кратные 2, 3, 5, и вывод которых еще не выполнен. Из деков считываются первые элементы. Минимальный элемент ( $t$ ) выводится в результирующий файл, не минимальные возвращаются в соответствующие деки, и с другого конца деков добавляются числа  $2*t$ ,  $3*t$ ,  $5*t$ . Логика изменения содержимого деков

проиллюстрирована данными из таблицы:

Вывод элемента	Dec1	Dec2	Dec3
1	2	3	5
2	4	3, 6	5, 10
3	4, 6	6, 9	5, 10, 15
5	6, 8, 10	6, 9, 12, 15	10, 15, 20, 25
6	8, 10, 12	9, 12, 15, 18	10, 15, 20, 25, 30
8	10, 12, 16	9, 12, 15, 18, 24	10, 15, 20, 25, 40

Задание 18. Разработайте программу, выполняющую сравнение двух длинных чисел.

Задание 19. Разработайте программу, выполняющую умножение двух длинных чисел.

Задание 20. Разработайте программу вычисления 100!

## Тема 15. Динамические структуры данных

### Связанный список

**Связанный список** – это структура данных, в произвольно выбранное место которой данные могут как включаться, так и изыматься. Для обеспечения такой гибкости, в каждый элемент добавляется указатель на следующий элемент списка.

Структуру списка можно задать так:

```
struct LINK
{
    int      data;
    struct LINK *next;
};
```

В этом описании значения элементов списка хранятся в поле `data`, а поле `next` является указателем на следующий элемент списка.

Можно создать циклический список, который не имеет первого и последнего элементов.

Элементы двунаправленного связанного списка содержат три поля: поле `data`, содержащее информацию, хранимую в элементе, и левое и правое поля, содержащие указатели на соседние элементы. Такой список можно реализовать следующим образом:

```
struct LINK
{
    int      data;
    struct LINK *right;
    struct LINK *left;
};
```

Здесь `right` и `left` – правый и левый указатели на соседние элементы соответственно.

В начало каждого списка можно поместить фиктивный узел, называемый ведущим (заголовком) списка. Поле элемента заглавного узла игнорируется, но ссылка узла сохраняется в качестве указателя узла, содержащего первый элемент

списка.

### **Варианты заданий**

**Задание 1.** Разработайте комплекс функций для обслуживания однонаправленного списка, содержащего элементы с ключами. Поле ключа заполняется уникальными значениями. Правило формирования ключа выберите самостоятельно. Это может быть номер элемента в списке или номер элемента при его создании.

Необходимо разработать следующие функции:

- создание списка;
- добавление элемента после элемента с заданным ключом;
- вывод содержимого поля ключа и поля данных.

**Задание 2.** Разработайте функции для обслуживания однонаправленного списка, содержащего элементы без ключей:

- создание списка;
- добавление элемента после элемента с заданным номером;
- вывод содержимого полей данных.

**Задание 3.** Разработайте функции для обслуживания однонаправленного списка, содержащего элементы без ключей:

- создание списка;
- вывод содержимого полей данных;
- удаление элемента с заданным значением.

**Задание 4.** Разработайте функции для обслуживания однонаправленного списка, содержащего элементы без ключей:

- создание списка;
- вывод содержимого полей данных;
- запись списка в файл;
- удаление списка.

**Задание 5.** Разработайте функции для обслуживания двунаправленного списка, содержащего элементы с ключами. Поле ключа заполняется уникальными значениями. Правило формирования ключа выберите самостоятельно. Это может быть номер элемента в списке или номер элемента при его создании.

Необходимо разработать следующие функции:

- создание списка;
- добавление элемента после элемента с заданным ключом;
- вывод значения указателя на предыдущий элемент.

**Задание 6.** Разработайте функции для обслуживания двунаправленного списка, содержащего элементы с ключами. Поле ключа заполняется уникальными значениями. Правило формирования ключа выберите самостоятельно. Это может быть номер элемента в списке или номер элемента при его создании.

Необходимо разработать следующие функции:

- создание списка;
- вывод значения указателя на предыдущий элемент;
- удаление элемента с заданным значением.

**Задание 7.** Разработайте функции для обслуживания кольцевого списка:

- создание пустого списка;
- добавление элемента в начало списка;
- вывод содержимого поля данных;
- удаление элемента с заданным номером.

Задание 8. Разработайте функции для обслуживания кольцевого списка:

- создание пустого списка;
- добавление элемента в начало списка;
- вывод содержимого поля данных;
- запись списка в файл.

Задание 9. Разработайте функции для обслуживания кольцевого списка:

- создание пустого списка;
- добавление элемента в конец списка;
- вывод номеров элементов;
- удаление элемента из конца списка.

Задание 10. Разработайте функции для обслуживания кольцевого списка

- создание списка;
- добавление элемента в конец списка;
- вывод поля данных списка;
- запись списка в файл.

Задание 11. Разработайте функции для обслуживания кольцевого списка:

- создание списка;
- добавление элемента после элемента с заданным номером;
- вывод содержимого поля данных.

Задание 12. Разработайте функции для обслуживания кольцевого списка:

- создание пустого списка;
- добавление элементов в список;
- вывод содержимого поля данных;
- упорядочение элементов в списке по полю данных.

Задание 13. Разработайте функции для обслуживания кольцевого списка, содержащего элементы с ключами. Поле ключа заполняется уникальными значениями. Правило формирования ключа выберите самостоятельно. Это может быть номер элемента в списке или номер элемента при его создании.

Необходимо разработать следующие функции:

- создание пустого списка;
- добавление элемента после элемента с заданным ключом;
- вывод содержимого поля ключа и поля данных.

Задание 14. Разработайте комплекс функций для обслуживания кольцевого списка, содержащего элементы с ключами. Поле ключа заполняется уникальными значениями. Правило формирования ключа выберите самостоятельно. Это может быть номер элемента в списке или номер элемента при его создании.

Необходимо разработать следующие функции:

- создание списка;
- добавление элемента после элемента с заданным ключом;
- вывод содержимого поля ключа и поля данных;

- запись списка в файл.

Задание 15. Напишите программу обработки кольцевого списка, решающего задачу Джозефуса. Задача Джозефуса представляет собой своего рода считалку: элементы «становятся» в круг, вводится некоторое число. Необходимо, начиная с первого элемента, отсчитать  $k$ -ый элемент списка и удалить его. Далее отсчет начинается с  $(k+1)$ -го элемента и опять удаляется  $k$ -й элемент. Так продолжать до тех пор, пока в списке не останется один элемент. Выдать содержимое последнего оставшегося элемента (например, строку).

Задание 16. В кассах аэропорта 4 окна. В каждом окне на обслуживание пассажира уходит некоторое (различное для каждого пассажира) время. Вновь входящий пассажир становится в самую короткую очередь. При двух и более одинаковых по длине очередях (в том числе и пустых) он становится в ближайшую очередь (имеющую наименьший номер). Для каждого пассажира известно время его прихода в кассы и время на его обслуживание.

Напишите программу, моделирующую деятельность касс. Определите при помощи этой модели среднее время на обслуживание одного пассажира при заданном входном потоке пассажиров.

Задание 17. В кассах аэропорта 4 окна. В каждом окне на обслуживание пассажира уходит некоторое (различное для каждого пассажира) время. Вновь входящий пассажир становится в самую короткую очередь. При двух и более одинаковых по длине очередях (в том числе и пустых) он становится в ближайшую очередь (имеющую наименьший номер). Для каждого пассажира известно время его прихода в кассы и время на его обслуживание.

Напишите программу, моделирующую деятельность касс. Определите при помощи этой модели среднюю частоту появления пассажиров (среднее время на обслуживание одного пассажира задано), при которой нагрузка на кассы оптимальна (то есть во все кассы стоит не более одного пассажира и не более одной очереди пустует в каждый момент времени).

Задание 18. Дан список. Создайте функцию, которая перемещает наибольший элемент данного списка так, чтобы он стал последним узлом.

Задание 19. Дан список. Создайте функцию, которая перемещает наименьший элемент данного списка так, чтобы он стал первым узлом.

Задание 20. Создайте список, который перераспределяет связный список так, чтобы узлы в четных позициях следовали после узлов в нечетных позициях, сохраняя порядок сортировки четных и нечетных узлов.

Задание 21. Напишите программу, меняющую для связного списка местами узлы, которые следуют после узлов, указываемых ссылками  $t$  и  $u$ .

Задание 22. Дан список. Напишите программу, которая создает новый список, содержащий те же узлы в том же порядке данного списка.

Задание 23. Обратите порядок следования элементов списка, используя список с фиктивным узлом.

Задание 24. Отсортируйте связный список методом вставки в список без использования заглавного узла.

Задание 25. Напишите программу, которая удаляет все узлы связанного



списка.

Задание 26. Напишите программу, которая вставляет узлы связанного списка, находящиеся в позициях с номерами, кратными 5.

Задание 27. Напишите программу, которая удаляет узлы в четных позициях связанного списка.

Задание 28. Пусть  $L$  обозначает кольцевой (циклический) двунаправленный список с заглавным звеном. Напишите программу, которая удваивает каждое вхождение в список некоторого элемента  $E$ .

Задание 29. Пусть  $L$  обозначает кольцевой (циклический) двунаправленный список с заглавным звеном. Напишите программу, которая в списке  $L$  переставляет в обратном порядке все элементы между первым и последним вхождениями элемента  $E$ , если  $E$  входит в  $L$  не менее двух раз.

Задание 30. Пусть  $L$  обозначает кольцевой (циклический) двунаправленный список с заглавным звеном. Напишите программу, которая из списка  $L$ , содержащего не менее двух элементов, удаляет все элементы, у которых одинаковые «соседи» (первый и последний элементы считать соседями).

### **Стек и очередь на основе линейного списка**

Ранее мы построили стек и очередь на основе массива. Можно сконструировать эти структуры и на основе линейного списка.

Стек представляется как линейный список, в котором включение и исключение элементов всегда производятся из начала списка. Для представления стека нам достаточно иметь один указатель – `stack`, который всегда указывает на последний записанный в стек элемент.

Очередь на основе линейного списка реализуется несколько сложнее. Мы уже говорили о том, что элементы из очереди исключаются в порядке их поступления: первый поступивший элемент первым удаляется из очереди. Для представления очереди с помощью линейного односвязного списка достаточно иметь один указатель `tail` на конец очереди.

Стек и очередь строим на следующей структуре списка

```
struct LINK
{
    int      data;
    struct LINK *next;
};
```

Таким образом, в отличие от списка, для стека и очереди очень важна дисциплина обслуживания.

### **Варианты заданий**

Задание 1. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- заменить в списке все вхождения элемента  $E_1$  на  $E_2$ .
- удалить все вхождения элемента  $E_3$ .

Задание 2. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- удалить из непустого списка первый элемент;
- удалить из непустого списка второй элемент;
- заменить в списке за каждым вхождением элемента E1 следующий на E2, если такой есть и он отличен от E1.

Задание 3. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- удалить из непустого списка последний элемент;
- удалить из непустого списка все отрицательные элементы (тип элементов целочисленный);
- первый элемент перенести в конец непустого списка.

Задание 4. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- проверить на равенство списки L1 и L2;
- поменять местами первые элементы списков L1 и L2.

Задание 5. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- проверить, есть ли в списке хотя бы два одинаковых элемента;
- удалить из списка все вхождения некоторого элемента, заданного заранее, если таковые есть.

Задание 6. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- проверить, является ли список упорядоченным;
- вставить в упорядоченный список некоторый элемент, не нарушая упорядоченность.

Задание 7. Описать и использовать процедуры работы с линейным одно или двунаправленным списком, элементами которого являются слова:

- найти в списке все слова, которые начинаются на некоторую букву, заданную заранее;
- если такие есть, сформировать из них новый список такого же типа.

Задание 8. Описать и использовать процедуры работы с линейным одно или двунаправленным списком, элементами которого являются слова:

- найти в списке все слова, которые начинаются с той же буквы, что и следующее слово;
- если такие есть, сформировать из них новый список такого же типа.

Задание 9. Одно из возможных представлений «длинного» текста, это разбиение его на строки равной длины. Далее для обращения к строкам можно использовать ссылки. Описать и использовать процедуры работы с линейным однонаправленным списком, представляющим текст:

- определить число строк в тексте;
- найти строку по заданному ее номеру;
- поменять местами строки с номерами K1 и K2.

Задание 10. Одно из возможных представлений «длинного» текста, это

разбиение его на строки равной длины. Далее можно использовать ссылки на эти строки. Описать и использовать процедуры работы с линейным однонаправленным списком, представляющим текст:

- определить число строк в тексте;
- удалить строку с заданным номером  $K$ ;
- вставить в текст произвольную строку после строки с номером  $K1$ .

Задание 11. Одно из возможных представлений «длинного» текста, это разбиение его на строки равной длины, и использование ссылок на эти строки. Описать и использовать процедуры работы с линейным однонаправленным списком, представляющим текст:

- определить общее число строк в тексте;
- найти строку по заданному ее номеру;
- вставить в текст разделяющую строку после каждой  $K$ -той строки.

Задание 12. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- слить два списка добавлением в конец списка  $L1$  всех элементов списка  $L2$ ;
- определить число вхождений некоторого элемента  $E$  в список.

Задание 13. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- слить два списка попеременной записью в новый список элементов из исходных списков  $L1$  и  $L2$ ;
- определить число элементов в списке.

Задание 14. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- получить точную копию списка;
- удвоить каждое вхождение в список некоторого элемента  $E$ .

Задание 15. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- сжать список, оставляя из каждой группы подряд идущих одинаковых элементов один;
- заменить в списке каждое вхождение элемента  $E1$ , если такой есть, на  $E2$ .

Задание 16. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- найти первое вхождение некоторого элемента  $E$ , если такое есть;
- слить два списка  $L1$  и  $L2$  вставкой всех элементов списка  $L2$  в исходный список  $L1$  после некоторого заданного элемента  $E$ .

Задание 17. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- проверить на равенство списки  $L1$  и  $L2$ ;
- поменять местами первые элементы списков  $L1$  и  $L2$ .

Задание 18. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- проверить, есть ли в списке хотя бы два одинаковых элемента;
- удалить из списка все вхождения заданного элемента  $E$ , если таковые есть.

Задание 19. Описать и использовать процедуры работы с линейным одно или двунаправленным списком:

- проверить, является ли список упорядоченным;
- вставить в упорядоченный список некоторый элемент E, не нарушая упорядоченность.

Задание 20. Описать и использовать процедуры работы с линейным одно или двунаправленным списком, элементами которого являются слова:

- найти в списке все слова, которые начинаются на некоторую букву, заданную заранее;
- если такие есть, сформировать из них новый список такого же типа.

Задание 21. Описать и использовать процедуры работы с линейным одно или двунаправленным списком, элементами которого являются целочисленные данные:

- сжать список, удаляя из него те элементы, значения которых равны нулю;
- сформировать новый список такого же типа только из положительных элементов исходного списка.

Задание 22. Описать и использовать процедуры работы с линейным одно или двунаправленным списком, элементами которого являются слова:

- найти первое вхождение в список некоторого слова;
- слить списки L1 и L2 вставкой всех элементов списка L2 в исходный список L1 после слова, которое было найдено. Если вхождения не было, прилить L2 в конец списка L1.

Задание 23. Описать и использовать процедуры работы с линейным одно или двунаправленным списком, элементами которого являются слова:

- найти длину (число элементов) списка;
- проверить на равенство списки L1 и L2;

Задание 24. Описать и использовать процедуры работы с линейным одно или двунаправленным списком, элементами которого являются слова:

- проверить, упорядочены ли слова, входящие в список, в алфавитном порядке;
- удалить из списка все слова, начинающиеся с некоторой заданной буквы.

Задание 25. Описать и использовать процедуры работы с линейным одно или двунаправленным списком, элементами которого являются целочисленные значения:

- проверить, является ли список упорядоченным по возрастанию его элементов или по убыванию;
- определить число элементов списка.

## **Бинарное дерево**

**Бинарное дерево** – это конечное множество элементов, которое либо пусто, либо содержит один элемент, называемый корнем дерева, а остальные элементы множества делятся на два непересекающихся подмножества, каждое из которых само является бинарным деревом. Эти подмножества называются левым и правым

поддеревьями исходного дерева. Каждый элемент бинарного дерева называется узлом дерева.

Структура данных для реализации двоичных деревьев очень похожа на структуру двунаправленного списка; каждый узел дерева содержит поле данных и два указателя – на левого и правого потомков:

```
struct NODE
{
    int data;
    struct NODE *l_heir;
    struct NODE *r_heir;
};
```

### **Варианты заданий**

Задание 1. Заполните двоичное дерево и постройте его зеркальное отображение относительно вертикали, расположенной справа.

Задание 2. Создайте программу, которая подсчитывает количество листьев в бинарном дереве.

Задание 3. Используя очередь или стек, напишите программу, которая присваивает параметру E элемент из самого левого листа непустого бинарного дерева.

Задание 4. Используя очередь или стек, напишите программу, которая определяет число вхождений элемента E в бинарное дерево.

Задание 5. Используя стек или очередь, напишите программу, которая вычисляет среднее арифметическое всех элементов непустого бинарного дерева.

Задание 6. Используя стек или очередь, напишите программу, которая меняет местами максимальный и минимальный элементы непустого двоичного дерева, все элементы которого различны.

Задание 7. Используя рекурсивную функцию, напишите программу, которая присваивает параметру E элемент из самого левого листа непустого бинарного дерева.

Задание 8. Используя рекурсивную функцию, напишите программу, которая определяет число вхождений элемента E в бинарное дерево.

Задание 9. Используя рекурсивную функцию, напишите программу, которая вычисляет среднее арифметическое всех элементов непустого бинарного дерева.

## Тема 16. Прикладные алгоритмы

Под сортировкой понимают процесс перестановки объектов данного множества в определенном порядке. Порядок определяется типом сортируемых данных. Для чисел типа `int` или `float` возможны сортировки в порядке возрастания или убывания., для строк символов – в алфавитном порядке. Цель сортировки – облегчить последующий поиск элементов в отсортированном множестве.

Алгоритмы поиска занимают очень важное место среди прикладных алгоритмов. Основная задача любого алгоритма поиска – найти запись с заданным ключом.

### Варианты заданий

Задание 1. Реализуйте алгоритм сортировки выбором связного списка.

Задание 2. Разработайте программу пузырьковой сортировки для связных списков.

Задание 3. Постройте программную реализацию варианта сортировки методом Шелла, ориентированного на связные списки.

Задание 4. Используйте один из изученных методов сортировки для построения программы сортировки четных элементов массива.

Задание 5. Используйте один из изученных методов сортировки для построения программы сортировки элементов массива, записанных на нечетных местах.

Задание 6. Используйте один из изученных методов сортировки для построения программы сортировки отрицательных элементов массива.

Задание 7. Реализуйте сортировку квадратичным методом. Рассмотрим его на примере. Пусть  $N$  – размер массива является точным квадратом натурального числа, например 3. Дан массив 7 10 3 5 15 9 6 12 8. Разобьем его на группы на  $\sqrt{N}$  групп по  $\sqrt{N}$  элементов в каждой. При разбивке на группы получим (7 10 3) (5 15 9) (6 12 8). Максимальные элементы в каждой группе 10 15 12. Максимальный из них 15, он принадлежит второй группе. Если оставшиеся элементы из второй группы, а это 5 и 9, меньше 10 и 12, то мы нашли сразу три элемента, записываемые на свои места. Если нет, то заменяем максимальные элементы элементами из групп.

Задание 8. Дан массив 7 9 13 1 8 4 10 11 5 3 6 2. Мы видим возрастающий участок 7 9 13, а справа, если читать справа налево, есть участок 2 6. Слияние этих двух фрагментов массива дает 2 6 7 9 13. А затем «сливаем» 1 8 и 3 5 11, получим 1 3 5 8 11. Записываем в массив и получаем 2 6 7 9 13 4 10 11 8 5 3 1. Обратите внимание на то, как записан массив! Продолжим. Если с первыми фрагментами все ясно, то вторые 4 10 и 10 перекрываются. Необходимо учесть этот факт и не дублировать 10 при записи в массив. Получаем: 1 2 3 5 6 7 8 9 11 13 10 4. Последнее слияние дает 1 2 3 4 5 6 7 8 9 10 11 13. Массив отсортирован. Этот метод называется «сортировкой естественным двухпутевым слиянием». Реализуйте его.

Задание 9. Реализуйте алгоритм, который называется «простое двухпутевое слияние». Он в отличие от алгоритма, описанного в предыдущем примере,

использует для слияния участки фиксированной длины. Рассмотрим пример,

Исходный массив: 13 1 7 5 4 3 9 10 19 14 16 23 2 4 8.

1-й шаг (длина 1). 8 13 2 7 4 16 9 19 10 6 14 3 23 5 4 1.

2-й шаг (длина 2). 1 4 8 13 3 4 14 16 19 10 9 6 23 7 5 2.

3-й шаг (длина 4). 1 2 4 5 7 8 13 23 19 16 14 10 9 6 4 3.

4-й шаг (длина 8). 1 2 3 4 4 5 6 7 8 9 10 13 14 16 19 23.

Реализуйте данный алгоритм. Размер массива не всегда совпадает с числом, равным степени двойки.

Задание 10. Дан массив. Разделим массив на участки определенной длины. Каждый из них отсортируем с помощью алгоритма простых вставок, а затем используем «простое двухпутевое слияние». Так, в предыдущем примере слияние используется на 3-м и 4-м шагах, а первые два заменяются работой по алгоритму простых вставок.

Задание 11. Напишите программу поиска в массиве А элементов массива В. Оба массива находятся во входном файле. Используйте методы поиска, рассмотренные в этом разделе.

Задание 12. Переставьте элементы массива А таким образом, чтобы вначале были записаны элементы, меньшие некоторого числа Х, затем равные Х и, наконец, большие Х. Указанные перестановки следует делать за один просмотр А и без использования дополнительных массивов.

Задание 13. Имеется внешний файл А из целых чисел. Используя три внешних файла В, С, D в качестве рабочих, упорядочить файл по неубыванию методом внешней сортировкой сбалансированным слиянием. Метод применяется так. Назовем «отрезком» как можно более длинный упорядоченный по неубыванию участок файла. На начальном этапе сортировки определяются отрезки файла А и они попеременно переносятся в файлы С и D. На следующем этапе пары i-х отрезков файлов С и D сливаются в более длинные отрезки и попеременно переносятся в файлы А и В. Затем сливаются пары отрезков файлов А и В и переносятся в файлы С и D и т.д. Учтите, что в конце концов единая упорядоченная последовательность чисел должна находиться в файле А.

А 

1	7	9	6	3	5	1	4	8	2	7
---	---	---	---	---	---	---	---	---	---	---

      С 

1	7	9	2	7
---	---	---	---	---

В 



      D 

5	1	4	8
---	---	---	---

а)

б)

А 

1	6	7	9	2	7
---	---	---	---	---	---

С 

1	1	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

В 

1	3	4	5	8
---	---	---	---	---

D 

2	7	
---	---	--

в)

с)

Задание 14. Дан массив из N чисел. Разрешенными операциями являются сложение двух элементов массива и сравнение сумм. Найти наибольший элемент массива за минимальное количество сравнений.

Задание 15. Разработайте программу проверки наличия заданного элемента в дереве.

Задание 16. Разработайте программу проверки наличия в дереве хотя бы двух

одинаковых элементов.

Задание 17. Напишите программу удаления из созданного двоичного дерева поиска всех отрицательных элементов.

Задание 18. Напишите программу, определяющую второй максимальный элемент в двоичном дереве поиска.



## Библиографический список

### Проектирование программ. Алгоритмы и структуры данных

1. Кушнеренко А.Г., Лебедев Г.В. Программирование для математиков. – М.: Наука, 1988. – 310 с.
2. Мейер Б., Бодуэн К. Методы программирования. – М.: Мир, 1982. – 290 с.
3. Йодан Э. Структурное проектирование и конструирование программ. – М.: Мир, 1979. – 406 с.
4. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.
5. Вирт Н. Алгоритмы и структуры данных. – М.: Мир, 1989. – 428 с.
6. Кнут Д. Искусство программирования для ЭВМ. Основные алгоритмы. – М.: Мир, 1976. – Т. 1. – 720 с.
7. Кнут Д. Искусство программирования для ЭВМ. Сортировка и поиск. – М.: Мир, 1978. – Т. 2. – 706 с.
8. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 436 с.
9. Гудман С., Хидетниemi С. Введение в разработку и анализ алгоритмов. – М.: Мир, 1981. – 450 с.

### Стандарт языков С и С++

10. Подбельский В.В., Фомин С.С. Программирование на языке Си. – М.: Финансы и статистика, 1999, 2000 – 600 с.
11. Шилдт Г. Самоучитель С++. – СПб.: BHV, 1998. – 683 с.
12. Шилдт Г. Теория и практика С++: Руководство для профессионалов. – СПб.: BHV, 1996. – 412 с.
13. Павловская Т.А. С/С++. Программирование на языке высокого уровня. – СПб.: Питер, 2004. – 461 с.
14. Березин Б.И., Березин С.Б. Начальный курс С и С++. – М.: Диалог-МИФИ, 1999. – 288 с.
15. Бабэ Б. Просто и ясно о Borland C++. – М.: Бином, 1995. – 394 с.
16. Арнуш К. Borland C++ 5: Освой самостоятельно. – М.: Бином, 1997. – 720 с.
17. Голуб А.И. С и С++. Правила программирования. – М.: Бином, 1996. – 272 с.
18. Дейтел Х.М., Дейтел П.Д. Как программировать на С++. – М.: Бином, 1998. – 1021 с.
19. Хикс К. С: Руководство пользователя. – М.: Бином, 1997. – 442 с.
20. Франка П. С++: Учебный курс: 26 уроков для освоения языка. – СПб.: Питер, 1999. – 521 с.

### Сборники задач с решениями на С и С++

21. Крячков А.В., Сухинина И.В., Томшин В.К. Программирование на С и С++. Практикум. – М.: Радио и связь, 1997, 2000. – 344 с.
22. Павловская Т.А., Щупак Ю.А. С/С++. Структурное программирование: Практикум. – СПб.: Питер, 2002. – 240 с.
23. Культин Н.Б. С/С++ в задачах и примерах. – СПб.: BHV, 2004. – 288 с.

## Оглавление

Введение.....	1
Тема 1. Базовые типы данных. Вычисления по формулам. Простые программы на языке C++ .....	4
Тема 2. Простые типы данных. Управление алгоритмом с использованием условного оператора if или переключателя switch .....	11
Тема 3. Инструменты C++ для реализации циклических алгоритмов .....	21
Тема 4. Циклические алгоритмы вычисления сумм, произведений, количеств, пределов, последовательностей. Сложные циклы .....	32
Тема 5. Использование циклических алгоритмов в решении содержательных задач .....	44
Тема 6. Практическое использование механизма функций C++ .....	51
Тема 7. Массивы и указатели. Алгоритмы работы с одномерными массивами. Использование функций при работе с массивами .....	71
Тема 8. Работа с одномерными массивами. Использование массивов в решении содержательных задач .....	92
Тема 9. Работа с двумерными массивами. Использование функций для работы с двумерными массивами. Использование файлов для хранения данных .....	97
Тема 10. Использование файлов, одно и двумерных массивов в решении содержательных задач .....	107
Тема 11. Работа со строками символов. Использование файлов .....	114
Тема 12. Работа со структурами и объединениями .....	133
Тема 13. Работа в графическом режиме .....	144
Тема 14. Полустатические структуры данных .....	153
Тема 15. Динамические структуры данных .....	157
Тема 16. Прикладные алгоритмы .....	166
Библиографический список .....	169