Задание 9

1. Написать функцию Power3(A), вычисляющую третью степень числа A, без использования функций математической библиотеки. У функции должен быть один вещественный аргумент, и она должна возвращать вещественный результат.

Пример вызова функции:

#include <stdio.h>

//Определение функции

**int** main()

{

**double** A,P;

...

P=Power3(A);

...

**return** 0;

}

2. Напишите функцию gip, которая вычисляет длину гипотенузы прямоугольного треугольника. Аргументы функции два вещественных числа – катеты треугольника, функция возвращает вещественное значение.

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции

**int** main()

{

**double** x, y, r;

...

r=gip(x,y);

...

**return** 0;

}

3. Напишите функцию cor, которая определяет есть ли вещественные корни у уравнения *a*⋅*x*2 +*b*⋅*x* +*c*

. Аргументы функции три целых числа, функция возвращает 1, если существует хотя бы один корень, или 0, если корней нет.

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции

**int** main()

{

**int** a,b,с,r;

...

r=cor(a,b,c);

...

}

4. Напишите функцию Ploshad, которая считает площадь треугольника по трем ее сторонам. Аргументы функции три целых числа, функция возвращает вещественное значение.

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции

**int** main()

{

**int** a,b,c;

**double** r;

...

r=Ploshad(a,b,c);

...

}

5. Напишите функцию fullsq, которая определяет, является ли число полным квадратом. Аргумент функции целое число, функция возвращает 1, если число является полным квадратом, или 0, если не является.

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции

**int** main()

{

**int** a, r;

...

r=fullsq(a);

...

}

6. Напишите функцию fact, которая вычисляет факториал заданного числа. Аргумент функции целое число, функция возвращает целое значение.

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции

**int** main()

{

**int** a, r;

...

r=fact(a);

...

}

7. Напишите функцию sum, которая находит сумму цифр числа. Аргумент функции целое число, функция возвращает целое значение.

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции

**int** main()

{

**int** a, r;

...

r=sum(a);

...

}

8. Напишите функцию palindr, которая определяет, является ли число палиндромом (читается одинаково слева напрво и справа налево). Аргумент функции целое число, функция возвращает 1, если число является палиндромом, или 0, если не является.

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции

**int** main()

{

**int** a, r;

...

r=palindr(a);

...

}

9. Напишите функцию prost, которая определяет, является ли число простым. Аргумент функции целое число, функция должна вернуть 1, если оно простое, или 0, если число не является простым.

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции

**int** main()

{

**int** a, r;

...

r=prost(a);

...

}

10. Даны два числа *N* и *M* (1 ≤ *N*, *M* ≤ 109).

Вывести их наибольший общий делитель.

11. Берляндские ученые вот уже несколько лет занимаются раскопками руин древней цивилизации, существовавшей за века до образования Берляндии и ее соседей и достигшей, по косвенным сведениям, невероятно высокого уровня технологий.

Недавно археологи обнаружили странную находку, предположительно летопись некоторых событий, которая может раскрыть берляндским историкам причину исчезновения столь могущественного общества — стопку из дюжины блестящих тонких дисков из неизвестного материала, нанизанных на алмазный стержень. На верхнем диске ученые обнаружили три числа, каждое из которых состоит из двух цифр. Ученые предположили, что на диске записана дата конца великой цивилизации.

После анализа дисков было установлено, что они использовались в XXI веке по летоисчислению, использовавшемуся древней цивилизацией — так называемому "григорианскому" календарю — год продолжительностью 365 дней, разделялся по нему на двенадцать месяцев. Второй месяц в году имел продолжительность двадцать восемь дней, первый, третий, пятый, седьмой, восьмой, десятый и двенадцатый — тридцать один день, остальные — тридцать дней. В особые года, номер которых делился на четыре и не делился на сто, либо делился на четыреста, второй месяц длился двадцать девять дней. Веком номер *i* назывался период с 100⋅(*i*−1) + 1 года по 100⋅*i*.

Так как достоверно не известно, в каком порядке представители древней цивилизации записывали даты, вам, как главному специалисту по григорианскому календарю, поручили провести исследование — установить, каким датам в XXI веке могла соответствовать надпись, в предположении, что одно из чисел соответствует дню в месяце (дни в каждом месяце нумеровались с единицы), еще одно из чисел — номеру месяца (месяцы также нумеровались с единицы), и еще одно число — последним двум цифрам года в XXI веке григорианского календаря.

По заданной надписи на диске выясните, каким датам в XXI веке она могла соответствовать.

Во входном файле в формате aa/bb/cc записаны числа с диска.

В выходной файл в произвольном порядке выведите все корректные даты dd/mm/yy в XXI веке, где dd соответствует номеру дня, mm -- номеру месяца, yy — номеру года, причем числа, соответствующие dd, mm и yy являются перестановками чисел с диска.

В случае, если никакая перестановка исходных чисел не является корректной датой XXI века, выведите "No such date".

12. Функция *A*(*n*) для целых неотрицательных *n*

определена так:

*A*(0)=0, *A*(1)=1, *A*(2*n*)=*A*(*n*), *A*(2*n*+1)=*A*(*n*)+*A*(*n*+1).

Для данного *n* найти и напечатать *A*(*n*).

13. Определим множества *K**i* рекуррентно. Пусть *K*0 = [0,1]. Разделим сегмент [0,1] на три части точками 1/3 и 2/3 и удалим из него интервал (1/3,2/3). Получим множество *K*1, состоящее из двух оставшихся сегментов [0,1/3] и [2/3,1]. Каждый из них разделим на три части (точками 1/9 и 2/9 для первого сегмента, и точками 7/9 и 8/9 – для второго) и удалим средние интервалы (1/9,2/9) и (7/9,8/9). Таким образом получаем множество *K*2, и т.д. Пусть мы построим множество *K**i*. Поделим каждый оставшийся сегмент из *K**i* на 3 части и удалим из этих сегментов средние интервалы. Получим, таким образом, из *K**i* множество *K**i*+1.

Вводятся 3 целых числа *n*, *a*, *b*.

Необходимо определить, принадлежит ли точка с координатой *a**b* множеству *K**n*.

Вход содержит 3 целых числа диапазоне от 1 до 100 – *a* *b* *n*.

Вывести единственное число – 1, если принадлежит, иначе 0.

14. Загадано некоторое английское слово, случайно выбранное из словаря. Слово содержит от 1 до 20 букв. Первоначально все буквы слова заменены символом '\*'. Можно назвать одну из 26 букв алфавита от 'a' до 'z' и все буквы слова, совпадающие с названной, будут открыты. Необходимо, сделав как можно меньшее количество попыток, открыть все буквы слова.

Вы должны написать подпрограмму с именем solve, угадывающую загаданное слово.

void solve(const char s[]); // С/С++

Этой подпрограмме передается строка, первоначально состоящая из символов '\*'. Количество символов '\*' в строке совпадает с длиной загаданного слова. По мере угадывания символы '\*' в строке будут заменяться на буквы.

Для угадывания буквы подпрограмма должна вызывать функцию guess.

int guess(char b); // С/С++

В качестве аргумента необходимо указать строчную английскую букву от 'a' до 'z'. Функция возвращает количество открытых букв.

Пересылаемое на проверку решение должно содержать только подпрограмму solve, вспомогательные функции, команды #include (в C/C++) и глобальные переменные. В функциях не должен выполняться ввод или вывод.

Пример программы на С/C++

**void** solve(**const** **char** s[])

{ **char** ch;

**for**(ch='a';ch<='z';++ch)

{ guess(ch);

**if**(strchr(s,'\*')==NULL) **break**;

}

}

15. Сколько есть способов сосчитать от *N* до 0, пропуская не более одного числа между соседними числами? Отчет обязательно должен начинаться с числа N и заканчиваться 0.

Например, 5, 4, 3, 2, 1, 0, и 5, 3, 1, 0 и 5, 4, 2, 1, 0 являются правильными обратными отсчетами, а 5, 4, 1,0 – неправильным.

16. Вывести все правильные скобочные выражения длиной *N*, состоящие из круглых и квадратных скобок.

17. Из прямоугольного листа клетчатой бумаги (*M* строк, *N* столбцов) удалили некоторые клетки. На сколько кусков распадётся оставшаяся часть листа? Две клетки не распадаются, если они имеют общую сторону.

18. Главным украшением рабочего стола Прокурора является кукаляка – набор красивых шкатулок, вложенных одна в другую как матрешки. Однажды Прокурор достал все шкатулки друг из друга, чтобы ими полюбоваться, но когда он попробовал их сложить друг в друга обратно, у него всё время оставались лишние шкатулки. Причиной этой неудачи могла быть глупость Прокурора или шутка Странника, который заходил к Прокурору и мог незаметно подменить несколько шкатулок.

Напишите программу, которая определит, как шкатулки можно сложить друг в друга, использовав максимальное количество из имеющихся. Шкатулки имеют форму прямоугольных параллелепипедов. В каждую шкатулку можно класть только одну шкатулку, но внутри неё могут быть другие вложенные друг в друга шкатулки. Шкатулки можно класть на любую грань, но так чтобы стенки вложенных шкатулок были параллельны друг другу, и размеры внутренней шкатулки были строго меньше внешней.

Первая строка ввода содержит одно целое число – количество шкатулок *N* (1 < *N* ≤ 1000). Далее следует *N* строк, каждая строка содержат по три целых числа в диапазоне от 1 до 1000 – длины ребер шкатулки в произвольном порядке.

В первой строке вывести одно целое число *K* – максимальное количество вложенных шкатулок. В следующей строке вывести *K* целых чисел – номера шкатулок, начиная с самой внутренней. Если возможно несколько вариантов, то можно вывести любой из них.

19. Краткое содержание фильма: для решающей битвы с силами зла были выбраны пять самых лучших бойцов Земли. Со стороны сил зла также будут сражаться пять бойцов. Каждый из бойцов сражается только один раз и только с одним противником. Чтобы Земля не была порабощена, каждый из бойцов Земли должен победить своего противника, а матч должен закончиться со счетом 5:0 в пользу Земли. Для каждой из возможных пар соперников известна вероятность *p**i**j* победы *i*-го бойца Земли над *j*-ым бойцом сил зла. Требуется подобрать пары бойцов таким образом, чтобы вероятность счета 5:0 (т.е. Π*p**i**j*) была максимальной.

Во входном файле пять строк по пять чисел (с точностью 3 разряда после запятой) в каждой строке – вероятности победы *p**i**j* (0 < *p**i**j* ≤ 1) для каждой пары бойцов, сначала строка с вероятностями для первого бойца Земли, затем для второго и т.д.

В выходной файл вывести пять целых чисел: номера противников *j**k* для каждого *k*-го бойца Земли, сначала номер противника для первого бойца, затем для второго и т.д. Если существует несколько вариантов, максимизирующих вероятность общей победы, вывести один (любой) из них.

20. Вывести все представления натурального числа *N* суммой натуральных чисел. Перестановка слагаемых нового способа представления не даёт.