##### Задание 1

1. Напишите функцию input, которой передаются строка с именем файла, двуменрный массив и два указателя на переменные, куда нужно поместить размеры введенной матрицы. Функция должна открыть файл с указанным именем, ввести из него размеры матрицы, затем ввести значения элементов матрицы в соответствии с введенными размерами.

Пример вызова функции:

#include <stdio.h>

// определение input

...

int main()

{ double matrix[100][100];

int n,m;

input("data.txt",matrix,&n,&m);

...

}

2. Напишите функцию inc, которой передается указатель на первый элемент подпоcледовательности из целых чисел и указатель на элемент за последним элементом последовательности. Функция должна увеличить все элементы подпоследовательности на 1.

Пример вызова функции:

#include <stdio.h>

// определение inc

...

int main()

{

int a[10]={0};

inc(a+2,a+4);

// {0,0,1,1,0, 0,0,0,0,0}

inc(a+5,a+10);

// {0,0,1,1,0, 1,1,1,1,1}

...

}

3. Напишите функцию integral, которая вычисляет интеграл заданной функции методом трапеций. Функции передается указатель на функцию *f*, границы интервала интегрирования *a* и *b* и количество отрезков *n*

, на которые разбивается интервал интегрирования.

*I*=*h*⋅(*f*(*a*)+*f*(*b*)2+∑*i*=1*n*−1*f*(*a*+*h*⋅*i*))

где *h*=*b*−*a**n*.

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции integral

...

double myfunc(double x)

{ return sin(x\*x);

}

int main()

{

printf("%.6lf\n",integral(sqrt,1.0,2.5,10));

printf("%.6lf\n",integral(myfunc,0.0,5.0,100));

...

}

4. Напишите функцию solve, которой передаются коэффициенты квадратного уравнения *a* (*a*>0), *b*, *c*

и два указателя на переменные, куда нужно поместить корни уравнения (сначала меньший корень, затем больший). Функция должна вернуть 0, если решения уравнения в действительных числах не существует, или 1, если корни найдены.

*x*1=−*b*−√*D**/*2*a*, *x*2=−*b*+√*D**/*2*a*, где *D*=*b*2−4*a**c*

Пример вызова функции:

#include <stdio.h>

#include <math.h>

// определение solve

...

int main()

{ double a,b,c,x1,x2;

scanf("%lf%lf%lf",&a,&b,&c);

if(solve(a,b,c,&x1,&x2))

printf("%.6lf %.6lf\n",x1,x2);

else

printf("No solution\n");

}

5. Напишите функцию copy, которой передается указатель на позицию в строке, куда нужно скопировать строку, заданную вторым указателем, включая нулевой байт. Функция должна вернуть указатель на нулевой байт в строке.

Пример вызова функции:

#include <stdio.h>

// определение copy

...

int main()

{

char s[1000];

copy(copy(s,"abc"),"def");

// "abcdef"

copy(copy(copy(s,"x"),"+ab"),"!");

// "x+ab!"

...

}

6. Напишите функцию root, которая вычисляет приближенное значение корня уравнения методом секущих. Функции передается указатель на функцию *f*, начальные точки приближения *x*1 и *x*2 и точность *ε*. Поиск корня выполняется до тех пор, пока |*x**i*−*x**i*−1|>*ε*. Результатом функции является *x**i*

.

*x**i*+1=*x**i*−1−*f*(*x**i*−1)⋅(*x**i*−*x**i*−1)*f*(*x**i*)−*f*(*x**i*−1)

Пример вызова функции:

#include <stdio.h>

#include <math.h>

//Определение функции root

...

double myfunc(double x)

{ return cos(x\*x);

}

int main()

{

printf("%.6lf\n",root(cos,1.0,1.1,1e-3));

printf("%.6lf\n",root(myfunc,0.0,0.1,1e-6));

...

}

7. Напишите функцию maxmin, которой передается массив, количество элементов в нем и два указатели на переменные, в которые нужно поместить индексы максимального и минимального элемента в массиве.

Пример вызова функции:

#include <stdio.h>

// определение maxmin

...

int main()

{ int arr[100];

int n;

scanf("%d",&n);

for(int i=0;i<n;i++)

scanf("%d",&arr[i]);

int imax,imin;

maxmin(arr,n,&imax,&imin);

printf("%d %d\n",imax,imin);

}

8. Напишите функцию split, которой передается строка, в которой находятся несколько слов, разделенных одним пробелом. Функция должна поместить в массив, переданный вторым аргументом, указатели на начало слов. Символы пробела (' ') в строке нужно заменить на нулевой символ ('\0'). Последним элементом в массив нужно поместить нулевой указатель (NULL).

Напишите функцию void print(char \*\*), которая печатает слова из массива указателей в обратном порядке. Для этого проверяется первый указатель в массиве. Если он равен NULL, то выполнение функции заканчивается. Иначе функция print вызывается рекурсивно для печати оставшейся части массива, а затем на отдельной строке печатается слово, на которое указывает первый элемент массива.

Пример вызова функций:

#include <stdio.h>

// определения split и print

...

int main()

{ char str[1001];

char \*words[101];

scanf("%1000[^\n]",str); getchar();

split(str,words);

print(words);

}

9. Функция быстрой сортировки имеет прототип

void qsort( void \*ptr, size\_t count, size\_t size,

int (\*comp)(const void \*, const void \*) );

Первый аргумент функции - адрес массива, далее количество элементов массиве и размер одного элемента. Последним аргументом является указатель на функцию, которой передаются адреса двух элементов массива для сравнения. Функция сравнения должна вернуть 0 в случае равенства элементов, отрицательное число, если первый элемент меньше второго, иначе положительное число.

Функция для сравнения строк имеет следующий прототип

int strcmp( const char \*lhs, const char \*rhs);

Эта функция сравнения возвращает 0 в случае равенства строк, отрицательное число, если первая строка меньше второй, иначе положительное число (т.е. соответствует требованиям qsort на функцию сравнения).

Напишите оператор вызова функции для сортировки массива строк:

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main()

{ char list[1000][100];

int n;

scanf("%d",&n);

getchar();

for(int i=0;i<n;++i) {

list[i][0]=0;

scanf("%99[^\n]",list[i]);

getchar();

}

... // вызов qsort

for(int i=0;i<n;++i)

printf("%s\n",list[i]);

}

10. Функция быстрой сортировки имеет прототип

void qsort( void \*ptr, size\_t count, size\_t size,

int (\*comp)(const void \*, const void \*) );

Первый аргумент функции - адрес массива, далее количество элементов массиве и размер одного элемента. Последним аргументом является указатель на функцию, которой передаются адреса двух элементов массива для сравнения. Функция сравнения должна вернуть 0 в случае равенства элементов, отрицательное число, если первый элемент меньше второго, иначе положительное число.

Функция для сравнения строк имеет следующий прототип

int strcmp( const char \*lhs, const char \*rhs);

Функция сравнения должна вернуть 0 в случае равенства элементов, отрицательное число, если первый элемент меньше второго, иначе положительное число.

Напишите функцию сравнения для сортировки массива структур, содержащих следующую информацию о студентах:

typedef struct student {

char name[50]; // фамилия

int group; // номер группы

} student;

Сортировка должна выполняться сначала по номеру группы, а при совпадении группы – по фамилии с помощью strcmp.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

typedef struct student {

char name[50]; // фамилия

int group; // номер группы

} student;

int studcmp(const void \*st1, const void \*st2){

...

}

int main()

{ student list[1000];

int n;

scanf("%d",&n);

for(int i=0;i<n;++i) {

scanf("%49s %d",list[i].name,list[i].group);

}

qsort(list,n,sizeof(student),studcmp);

for(int i=0;i<n;++i)

printf("%d %s\n",list[i].group,list[i].name);

}