**Лабораторная работа №3**

**Тема: Использование пользовательских функций. Передача параметров по значению. Обработка числовых массивов и матриц.**

***Цель работы:***

1. Изучить методы создания и использования функций.
2. Получить навыки работы с функциями, передачи параметров и возврата результата по значению.
3. Познакомиться с новыми понятиями: время жизни и область видимости переменных.
4. Получить навыки работы с локальными и глобальными переменными

**Лабораторное задание**

1. Изучить теоретическую часть лабораторной работы
2. Изучить примеры программирования и выполнить их на компьютере.
3. Выполнить задание (номер варианта соответствует номеру компьютера).

* Разработать алгоритм решения
* Написать программу, реализующую разработанный алгоритм, используя пользовательские функции с передачей параметров по значению

**Теоретические сведения.**

Функцией называется выделенная последовательность инструкций, предназначенных для решения определенной задачи. Ранее мы уже использовали библиотечные функции ввода-вывода printf() и scanf(), в данной лабораторной работе познакомимся с правилами создания своих (пользовательских) функций.

Есть несколько причин для использования пользовательских функций, во-первых, программа приобретает некоторую структуру и, тем самым, становится более понятной и упорядоченной, во-вторых, исключаются повторы похожих участков текста.

Функция может многократно вызываться из различных частей программы, в общем случае она выполняет следующие действия:

* получает параметры;
* выполняет инструкции, согласно заложенному алгоритму;
* может возвращать результат в вызывающую программу.

**Функции, общие понятия.**

С использованием функций в языке С связаны понятия, которые условно можно разделить на две группы.

В первую группу входят *определение, прототип и вызов функции -* все три понятия связанны с подготовкой функции к работе.

Вторая группа, *параметры и возвращаемое значение*, обеспечивает связь функции с «внешней средой». Функция может многократно вызываться из различных частей программы, при этом необходимообеспечить её связь с вызывающей программой, из вызывающей программы в функцию передать необходимые для работы данные, а по окончанию работы принять результат.

*Определение функции* – это описание действий, выполняемых функцией согласно требованиям алгоритма. Именно эта часть программы будет впоследствии многократно вызываться из других частей программы.

*Прототип функции (объявление)* используется в том случае, если вызов функции предшествует её определению или если определение и вызовы функции находятся в разных файлах.

*Вызов функции* обеспечивает связь с вызывающей программой. При вызове:

* передаются параметры из вызывающей программы в функцию
* управление передается первой инструкции в теле функции,
* после завершения работы функции в вызывающую программу передается возвращаемое значение, а управление возвращается в точку вызова.

Определение функции состоит из заголовка и тела, например:

double f1 (int a, int f) //заголовок

{ … } // тело

В данном примере определена функция **f1** с двумя параметрами **int a** и **int f** , возвращающая значение типа **double**

*Тип функции (в нашем примере double)* определяет тип значения, которое возвращает функция. Если функция не должна возвращать значение, то используется тип **void**, который в данном случае означает отсутствие значения.

В заголовке функции параметры называются *формальными* (в нашем примере **int a, int f**), и служат для её связи с вызывающей программой. *Формальные параметры создаются в начале работы функции – это локальные переменные, которые инициализируются значениями, полученными из вызывающей программы при вызове функции.* Параметры при вызове функции получают конкретные значения и называются *фактическими* параметрами, например вызов функции может выглядеть так:

**…**

double z;

int s1=10;

…

z = f1(s1, 5); //вызов функции f1, s1 и 5 – фактические параметры

**Передача параметров в функцию и возврат значений**

Параметры позволяют передать информацию из вызывающей программы в функцию. В теле функции параметрами можно пользоваться так же, как и локальными переменными.*При вызове функции:*

* для каждого формального параметрасоздаётся локальная переменная;
* начальными значениями созданных переменных являются фактические параметры, определяемые при вызове функции.

В языке С функция может возвращать только одно значение, для этого её выполнение следует завершить оператором ***return****,* содержащим некоторое выражение*.* Следует отметить, что *тип функции в определении должен соответствовать типу выражения оператора return в её теле.*

**Пример 1:**

int sum(int a, int b)

{

return a+b;

}

int main()

{

int x =5,y=10,z;

z = sum(x,y);

}

При вызове функции создаются два формальных параметра **a, b** (локальные переменные), которые инициализируются фактическими параметрами **x, y:**

**a=x**

**b=y**

Функция возвращает значение типа **int,** которое записывается в переменную **z**

**Передача параметров по значению**

Во всех предыдущих примерах вызываемой функции передавались не сами переменные, а лишь их значения. При таком вызове функция не может изменить саму переменную в вызывающей программе, она может изменить лишь её временную копию (параметр), значение которой теряется при выходе из функции.

**Пример 2:** Возведение числа в степень.

#include <iostream>

using namespace std;

#define DEC 25 // основание

// функция для возведения числа base в n-ую степень

int power (int base, int n)

{

int r; // локальная переменная функции power

for (r = 1; n > 0; --n)

r = r \* base;

return r; // возврат значения переменной r в вызывающую программу

} // переменная r - уничтожается

int main()

{

int i, p=0;

for(i=0 ; i<5 ; i++)

{

p=power(DEC , i); // вызов функции power и передача значений DEC и i

cout<<i<<':'<<p<<endl;

}

return 0;

}

При вызове функции **power(DEC , i)**: int **base= DEC,** int **n=i**

Обратите внимание, что функция **power()**, не оказывает никакого влияния на переменную **i** в вызывающей программе.

**Передача массива в функцию**

Для того, чтобы организовать обработку массива в функции, необходимо сообщить ей информацию об имени и размере массива:

int func (char s[], int n)

{ . . . }

Функция **func** позволяет обрабатывать массивы произвольного размера. При этом первый параметр – имя массива, второй – его размер.

Вызов функции:

// Пример обработки массива в 100 элементов.

int A[100];

func (A, 100);

**Использование прототипа функции**

В языке Си определения функций могут следовать за определением функции main, перед ним, или находится в другом файле.

Положение определения функции:

* за функцией main
* перед функцией main
* в другом модуле (файле).

Однако во всех случаях к моменту вызова функция должна быть определена или объявлена. Это требование обусловлено тем, что компилятор должен осуществить проверку корректности вызова функции (проверку соответствия количества и типов фактических параметров, количеству и типам формальных параметров). Когда вызов функции предшествует её определению, эта проверка выполняется по прототипу.

*Прототип* напоминает заголовок в определении функции:

* Тело функции отсутствует;
* Имена формальных параметров могут быть опущены (**типы параметров опускать нельзя!**).

Прототипы:

int power (int base, int n);

или

int power (int , int );

Если функция определена до функции main() - прототип не обязателен.

**Пример 3:** Использование прототипа функции

#include <stdio.h>

int my\_max(int , int); // прототип функции

int main()

{

int x, y, r;

printf(" x= ");

scanf("%d",&x);

printf(" y= ");

scanf("%d",&y);

r = my\_max( x, y ); // вызов предшествует определению

printf(" r= %d\t %d\n", r, my\_max( 50, x + y ));

return 0;

}/ /---------------------------------конец функции main()

// определение функции my\_max()

int my\_max (int a, int b)

{ return (a>b)? a:b; }

Функция my\_max() определяет свой максимальный параметр и возвращает его значение в вызывающую программу.

**Примеры программирования**

**Пример 4:** Функция, вычисляющая сумму массива. Работаем с двумя разными массивами.

#include <iostream>

#include <clocale>

#define N 30 // размер массива mas

#define N1 50 // размер массива mas1

using namespace std;

int summa (int array[], int n)

{

int res=0;

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

res+= array[i];

return res;

}

//функция для печати массива M[n] в col колонок

void printmas (int M[], int n, int col)

{

int i;

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

{

cout<<M[i];

if ((i+1)%col)

cout<<'\t';

else

cout<<endl;

}

cout<<endl;

}

int main()

{

setlocale(LC\_ALL, "Rus"); //русификация вывода текста на консоль

int mas[N], mas1[N1]; // определение массивов (30 и 50 элементов)

for (int i = 0; i < N; i++) // цикл для инициализации массива mas

mas[i] = 2\*i + 1;

cout<<"--------------- mas --------------"<<endl;

printmas (mas,N,5);

for (int i = 0; i < N1; i++) // цикл для инициализации массива mas1

mas1[i] = rand()%100;

cout<<"--------------- mas1 --------------"<<endl;

printmas (mas1,N1,5);

int s = summa (mas,N); // создание переменной s и вычисление суммы

// печать переменной s , вычисление и печать суммы массива mas1

cout<<"Сумма mas="<<s<<endl<<"Сумма mas1 = "<< summa (mas1,N1)<<endl;

system ("pause");

return 0;

}

**Пример 5:** С помощью датчика случайных чисел создать матрицу вещественных чисел 5Х5, вывести на экран, найти минимальные элементы в строках и максимальные элементы в столбцах.

#include <stdio.h>

#include <time.h>

#include <locale.h>

#define RAW 5 // количество строк

#define COL 5 // количество столбцов

// \_\_\_\_\_\_область определения функций\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//функция для инициализации матрицы;

void initmatr(double M[RAW][COL])

{

int i,j;

for (i=0;i<RAW; i++)

for (j=0;j<COL; j++)

M[i][j]=rand()%100/10.;

}

//функция для печати матрицы в

void printmatr(double M[RAW][COL])

{

int i,j;

for (i=0; i<RAW; i++)

{

for (j=0;j<COL; j++)

printf("%7.2f\t",M[i][j]);

printf("\n");

}

}

// поиск минимального элемента в строке.

double rawmin (double M[RAW][COL], int traw)

{

double min;

int i;

for (i=0,min=M[traw][0]; i<COL; i++)

if (min > M[traw][i])

min=M[traw][i];

return min;

}

// поиск максимального элемента в столбце

double colmax (double M[RAW][COL], int tcol)

{

double max;

int i;

for (i=0,max=M[0][tcol];i<RAW; i++)

if (max < M[i][tcol])

max=M[i][tcol];

return max;

}

//конец области определения функций\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int main()

{ //---------------------------------------- настройки

setlocale(LC\_ALL,"Rus"); // русификация консоли вывода

srand(time(0)); // инициализация датчика случайных чисел

// область определений (выделение памяти)

double matr[RAW][COL],cmax;

int i;

//----------------------------------------------------------------------

initmatr(matr);

printf("Исходная матрица:\n");

printmatr(matr);

printf("Минимальные значения в строках:\n");

for (i=0;i<RAW;i++)

printf("%d строка - %7.2f\n",i+1,rawmin (matr,i));

printf("Максимальные значения в столбцах:\n");

for (i=0;i<COL;i++)

{

cmax=colmax (matr,i);

printf("%d столбец - %4.2f\n",i+1,cmax);

}

system ("pause");

return 0;

}

Функции **initmatr()** и **printmatr()** не возвращают значений и имеют по одному параметру, определяющему матрицу. Функции **rawmin()** и **colmax()** возвращают результат и имеют по два параметра, первый определяет матрицу, а второй обрабатываемую строку (столбец). Возвращаемое значение без запоминания выводится на экран, когда вызов функций **rawmin()** является параметром в функции **printf().** Результат, возвращаемый функцией **colmax()** запоминается в переменной **cmax**, а затем выводится на экран.

Недостатком данной реализации является то, что все функции работают с матрицей фиксированного размера **M[RAW][COL]**, этот подход приемлем в том случае, если нужно работать с одной матрицей (или несколькими матрицами одинакового размера). Следующий пример демонстрирует работу с матрицей произвольного размера.

**Пример 6:** Выполнить задание примера 5, используя функции, работающие с произвольными матрицами. Продемонстрировать работу на двух матрицах разной размерности

#include <stdio.h>

#include <time.h>

#include <locale.h>

#define RAW1 6 // размерность первой матрицы: количество строк

#define COL1 5 // количество столбцов

#define RAW2 8 // размерность второй матрицы: количество строк

#define COL2 8 // количество столбцов

// \_\_\_\_\_\_область определения функций\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//функция для инициализации матрицы размера r х c

void initmatr1 (double M[], int r, int c)

{

int i,j;

for (i=0; i<r; i++)

for (j=0; j<c; j++)

M[i\*c+j]=rand()%100/10.;

}

//функция для печати матрицы размера r х c

void printmatr1 (double M[], int r, int c)

{

int i,j;

for (i=0; i<r; i++)

{

for (j=0; j<c; j++)

printf("%7.2f\t", M[i\*c+j]);

printf("\n");

}

}

// поиск минимального элемента в строке traw матрицы r х c

double rawmin1 (double M[], int r, int c, int traw)

{

double min;

int i;

for (i=0, min=M[traw\*c]; i<c; i++)

if (min > M[traw\*c+i])

min=M[traw\*c+i];

return min;

}

// поиск максимального элемента в столбце tcol матрицы r х c

double colmax1 (double M[], int r, int c, int tcol)

{

double max;

int i;

for (i=0,max=M[tcol]; i<r; i++)

if (max < M[i\*c+tcol])

max=M[i\*c+tcol];

return max;

}

//конец области определения функций\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int main()

{ //---------------------------------------- настройки

setlocale(LC\_ALL,"Rus");

srand (time(0));

//------ область определений (выделение памяти)

int i;

double matr1[RAW1\*COL1], matr2[RAW2\*COL2],cmax;

//----------------------------------------------------------------------

initmatr1 (matr1, RAW1, COL1);

printf("Исходная матрица1:\n");

printmatr1 (matr1, RAW1, COL1);

printf("Минимальные значения в строках:\n");

for (i=0; i<RAW1; i++)

printf("%d строка - %7.2f\n",i+1,rawmin1 (matr1, RAW1, COL1, i));

printf("Максимальные значения в столбцах:\n");

for (i=0; i<COL1; i++)

{

cmax=colmax1 (matr1, RAW1, COL1, i);

printf("%d столбец - %7.2f\n", i+1, cmax);

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n ");

initmatr1 (matr2, RAW2, COL2);

printf("Исходная матрица2:\n");

printmatr1 (matr2, RAW2, COL2);

printf("Минимальные значения в строках:\n");

for (i=0; i<RAW2; i++)

printf("%d строка - %7.2f\n", i+1, rawmin1 (matr2, RAW2, COL2, i));

printf("Максимальные значения в столбцах:\n");

for (i=0; i<COL2; i++)

{

cmax=colmax1 (matr2, RAW2, COL2, i);

printf("%d столбец - %4.2f\n", i+1, cmax);

}

system ("pause");

return 0;

}

**Вопросы.**

1. Поясните общие понятия, связанные с использованием функции: определение, вызов, параметры функции.
2. Что такое прототип функции, когда он используется.
3. Что такое тип функции?
4. Какую роль выполняют параметры в функции? Расскажите о формальных и фактических параметрах функции.
5. Расскажите об использовании переменных в функциях, какая разница между локальной и глобальной переменной?
6. Как передать массив в функцию?

**Общие требования для всех вариантов заданий:**

1. Все константы, задаваемые в программе должны быть именованные (N,M,S …), их значения выбираются студентом по своему усмотрению, могут быть изменены по требованию преподавателя во время защиты.
2. При выполнении задания использовать следующие функции из примеров:
   1. инициализация массива (или матрицы) случайными числами
   2. вывод массива (или матрицы) на экран терминала. Столбцы должны быть выровнены.
3. Предусмотреть зацикливание программы для многократного выполнения задания с различными данными. После очередного выполнения задания спрашивать: «Продолжить выполнение?»

|  |  |  |
| --- | --- | --- |
| Варианты задания | Условия задания | |
| 1, 16 | Создать функцию для поиска положения (индекса) первого отрицательного элемента произвольного массива целых чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение отрицательного элемента массива. Полученный индекс разделяет массив на 2 части. 3. Определить среднее арифметическое элементов второй части массива 4. Пункты 2 и 3 выполнить для массивов A и B | |
|  | Дана целочисленная матрица NxN  Создать функцию, которая возвращает количество нулей в заданной строке заштрихованной области. Отразить на экране результаты для всех строк. |
| 2, 17 | Создать функцию для поиска положения (индекса) максимального элемента произвольного массива вещественных чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение максимумов в обоих массивах 3. Определить в каком массиве максимум находится дальше от начала массива | |
|  | Дана целочисленная матрица NxM, содержащая как положительные, так и отрицательные элементы. Создать функцию, которая возвращает среднее арифметическое отрицательных элементов в заданном столбце заштрихованной области. Отразить на экране результаты для всех столбцов. |
| 3, 18 | Создать функцию для поиска положения (индекса) первого положительного элемента произвольного массива целых чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение положительного элемента массива. Полученный индекс разделяет массив на 2 части. 3. Определить количество элементов в первой и второй частях массива 4. Пункты 2 и 3 выполнить для массивов A и B | |
|  | Дана целочисленная матрица NxN, содержащая как положительные, так и отрицательные элементы. Создать функцию, которая возвращает количество положительных элементов в заданной строке заштрихованной области. Отразить на экране результаты для всех строк. |
| 4, 19 | Создать функцию для поиска положения (индекса) последнего положительного элемента произвольного массива целых чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение положительного элемента массива. Полученный индекс разделяет массив на 2 части. 3. Определить минимальный элемент в первой части массив 4. Пункты 2 и 3 выполнить для массивов A и B | |
|  | Дана целочисленная матрица NxM, содержащая как положительные, так и отрицательные элементы. Создать функцию, которая возвращает минимумы положительных элементов в заданном столбце заштрихованной области. Отразить на экране результаты для всех столбцов. |

|  |  |  |
| --- | --- | --- |
| 5, 20 | Создать функцию для поиска положения (индекса) минимального элемента произвольного массива вещественных чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение минимумов в обоих массивах. Полученные индексы разделяют оба массива на 2 части. 3. Определить в каком массиве среднее арифметическое второй части больше. | |
|  | Дана целочисленная матрица NxN  Создать функцию, которая возвращает минимум в заданной строке заштрихованной области. Отразить на экране результаты для всех строк. |
| 6, 21 | Создать функцию для поиска положения (индекса) максимального элемента произвольного массива вещественных чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение максимумов в обоих массивах   Определить в каком массиве максимум находится дальше от начала массива | |
|  | Дана целочисленная матрица NxM  Создать функцию, которая возвращает суммы в заданном столбце верхней заштрихованной области. Отразить на экране результаты для всех столбцов. |
| 7, 22 | Создать функцию для поиска положения (индекса) максимального элемента произвольного массива вещественных чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение максимумов в обоих массивах. Точка максимума (его положение) разделяет массив на 2 части. 3. Определить в каком массиве сумма элементов второй части меньше | |
|  | Дана целочисленная матрица NxN  Создать функцию, которые возвращает сумму элементов в заданной строке заштрихованной области. Отразить на экране результаты для всех строк. |
| 8, 23 | Создать функцию для поиска положения (индекса) последнего положительного элемента произвольного массива целых чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение положительного элемента массива. Полученный индекс разделяет массив на 2 части. 3. Определить количество отрицательных элементов во второй части массива   Пункты 2 и 3 выполнить для массивов A и B | |
|  | Дана целочисленная матрица 6x9  Создать функцию, которая возвращает сумму элементов в заданном столбце заштрихованной области. Отразить на экране результаты для всех столбцов. |
| 9, 24 | Создать функцию для поиска положения (индекса) первого нулевого элемента произвольного массива целых чисел. Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение нулевого элемента массива. Полученный индекс разделяет массив на 2 части. 3. Определить сумму элементов в первой и во второй частях массива 4. Пункты 2 и 3 выполнить для массивов A и B | |
|  | Дана целочисленная матрица NxM, Создать функцию, которая возвращает среднее арифметическое в заданном столбце заштрихованной области. Отразить на экране результаты для всех столбцов. |

|  |  |  |
| --- | --- | --- |
| 10, 25 | Создать функцию для поиска положения (индекса) последнего положительного элемента произвольного массива целых чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение положительного элемента массива. Полученный индекс разделяет массив на 2 части. 3. Определить минимальный элемент в первой части массив 4. Пункты 2 и 3 выполнить для массивов A и B | |
|  | Дана целочисленная матрица NxN, содержащая как положительные, так и отрицательные элементы. Создать функцию, которые возвращает максимальный по модулю элемент в заданной строке заштрихованной области. Отразить на экране результаты для всех строк. |
| 11, 26 | Создать функцию для поиска положения (индекса) минимального элемента произвольного массива вещественных чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение минимумов в обоих массивах 3. Определить в каком массиве минимум находится ближе к концу массива | |
|  | Дана целочисленная матрица NxN  Создать функцию для нахождения суммы элементов в заданном столбце заштрихованной области. Отразить на экране результаты для всех столбцов. |
| 12, 27 | Создать функцию для поиска положения (индекса) последнего отрицательного элемента произвольного массива целых чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение отрицательного элемента массива. Полученный индекс разделяет массив на 2 части. 3. Определить максимальный элемент в второй части массив 4. Пункты 2 и 3 выполнить для массивов A и B | |
|  | Дана целочисленная матрица NxM  Создать функцию, которые возвращает максимумы в заданном столбце верхней заштрихованной области. Отразить на экране результаты для всех столбцов. |
| 13, 28 | Создать функцию для поиска положения (индекса) минимального элемента произвольного массива вещественных чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение минимумов в обоих массивах. Полученные индексы разделяют оба массива на 2 части. 3. Определить в каком массиве среднее арифметическое второй части больше. | |
|  | Дана целочисленная матрица NxN  Создать функцию, которая возвращает сумму элементов в заданной строке правой заштрихованной области.  Отразить на экране результаты для всех строк. |
| 14, 29 | Создать функцию для поиска положения (индекса) минимального элемента произвольного массива вещественных чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение минимумов в обоих массивах. Полученные индексы разделяют оба массива на 2 части. 3. Определить в каком массиве среднее арифметическое второй части больше. | |
|  | Дана целочисленная матрица NxM  Создать функции, которая возвращает максимумы в заданном столбце нижней заштрихованной области. Отразить на экране результаты для всех столбцов. |

|  |  |  |
| --- | --- | --- |
| 15, 30 | Создать функцию для поиска положения (индекса) первого отрицательного элемента произвольного массива целых чисел.  Тестовая программа:   1. Создать и инициализировать 2 массива A[N1], B[N2] 2. Найти положение отрицательного элемента массива. Полученный индекс разделяет массив на 2 части. 3. Определить среднее арифметическое элементов второй части массива 4. Пункты 2 и 3 выполнить для массивов A и B | |
|  | Дана целочисленная матрица NxN  Создать функцию, которая возвращает минимум в заданной строке заштрихованной области. Отразить на экране результаты для всех строк. |