Лабораторная работа №5

Функции и массивы в C/C++

1. Цель работы:

1) Получение практических навыков при работе с функциями
2) Получение практических навыков при передаче массивов в функции.

2. Теоретические сведения

Функция – это именованная последовательность описаний и операторов, выполняющая законченное действие, например, формирование массива, печать массива и т. д.

Любая функция должна быть объявлена и определена.

- Объявление функции (прототип, заголовок) задает имя функции, тип возвращаемого значения и список передаваемых параметров.
- Определение функции содержит, кроме объявления, тело функции, которое представляет собой последовательность описаний и операторов.

```
тип имя_функции([список_формальных_параметров])
{ тело_функции}
```

- Тело_функции – это блок или составной оператор. Внутри функции нельзя определить другую функцию.

В теле функции должен быть оператор, который возвращает полученное значение функции в точку вызова. Он может иметь 2 формы:

1) return выражение;
2) return;

Первая форма используется для возврата результата, поэтому выражение должно иметь тот же тип, что и тип функции в определении. Вторая форма используется, если функция не возвращает значения, т. е. имеет тип void. Программист может не использовать этот оператор в теле функции явно, компилятор добавит его автоматически в конец функции перед }.

- Тип возвращаемого значения может быть любым, кроме массива и функции, но может быть указателем на массив или функцию.
- Список формальных параметров – это те величины, которые требуется передать в функцию. Элементы списка разделяются запятыми. Для каждого параметра указывается тип и имя. В объявлении имена можно не указывать.

Для того, чтобы выполнялись операторы, записанные в теле функции, функцию необходимо вызвать. При вызове указываются: имя функции и фактические параметры. Фактические параметры заменяют формальные параметры при выполнении операторов тела функции. Фактические и формальные параметры должны совпадать по количеству и типу.

Объявление функции должно находиться в тексте раньше вызова функции, чтобы компилятор мог осуществить проверку правильности вызова. Если функция имеет тип не void, то ее вызов может быть операндом выражения.

2.1. Параметры функции

Основным способом обмена информацией между вызываемой и вызывающей функциями является механизм параметров. Существует два способа передачи параметров в функцию: по адресу и по значению.

- При передаче по значению выполняются следующие действия:
  - вычисляются значения выражений, стоящие на месте фактических параметров;
  - в стеке выделяется память под формальные параметры функции;

- каждому фактическому параметру присваивается значение формального параметра, при этом проверяются соответствия типов и при необходимости выполняются их преобразования.

```
void Change(int a,int b)//передача по значению
{
     int r=a;
     a=b;
     b=r;
}
int main()
{
int x=1,y=5;
Change(x,y);
cout<<"x="<<x<<" y="<<y; //выведется: x=1 y=5

return 1;
}
```

- При передаче по адресу в стек заносятся копии адресов параметров, следовательно, у функции появляется доступ к ячейке памяти, в которой находится фактический параметр и она может его изменить.

```
void Change(int *a,int *b)//передача по адресу
{
     int r=*a;
     *a=*b;
     *b=r;
}
int main()
{
int x=1,y=5;
Change(&x,&y);
cout<<"x="<<x<<" y="<<y; //выведется: x=5 y=1

return 1;
}
```

Для передачи по адресу также могут использоваться ссылки. Ссылка – это синоним имени объекта, указанного при инициализации ссылки.

Формат объявления ссылки

```
тип & имя =имя_объекта;
```

Ссылка не занимает дополнительного пространства в памяти, она является просто другим именем объекта.

При передаче по ссылке в функцию передается адрес указанного при вызове параметра, а внутри функции все обращения к параметру неявно разыменовываются.

```
void Change(int &a,int &b)
{
     int r=a;
     a=b;
     b=r;
}
int main()
{
```

```
int x=1,y=5;
Change(x,y);
cout<<"x="<<x<<" y="<<y; //выведется: x=5 y=1

return 1;
}
```

Использование ссылок вместо указателей улучшает читаемость программы, т. к. не надо применять операцию разыменовывания. Использование ссылок вместо передачи по значению также более эффективно, т .к. не требует копирования параметров. Если требуется запретить изменение параметра внутри функции, используется модификатор const. Рекомендуется ставить const перед всеми параметрами, изменение которых в функции не предусмотрено (по заголовку будет понятно, какие параметры в ней будут изменяться, а какие нет).

2.2. Локальные и глобальные переменные

- Переменные, которые используются внутри данной функции, называются локальными. Память для них выделяется в стеке, поэтому после окончания работы функции они удаляются из памяти. Нельзя возвращать указатель на локальную переменную, т. к. память, выделенная такой переменной, будет освобождаться.
- Глобальные переменные – это переменные, описанные вне функций. Они видны во всех функциях, где нет локальных переменных с такими именами.

2.3. Передача одномерных массивов как параметров функции

При использовании массива как параметра функции, в функцию передается указатель на его первый элемент, т. е. массив всегда передается по адресу. При этом теряется информация о количестве элементов в массиве, поэтому размерность массива следует передавать как отдельный параметр.

```
void print(int a[100],int n) //вывод массива на печать
{
     for(int i=0;i<n;i++)
          cout<<a[i]<<" ";
     cout<<"\n";
}
```

Так как в функцию передается указатель на начало массива (передача по адресу), то массив может быть изменен за счет операторов тела функции.

2.5. Передача многомерных массивов в функцию

Многомерный массив – это массив, элементами которого служат массивы. Например, массив int a[4][5] – это массив из указателей int*, которые содержат имена одноименных массивов из 5 целых элементов:

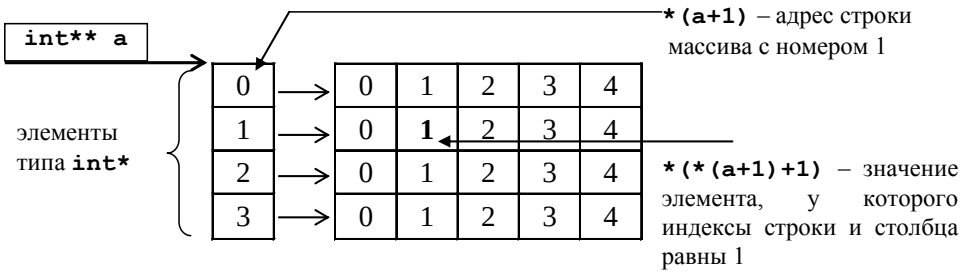

Рис. Выделение памяти под массив, элементами которого являются массивы.

При передаче многомерных массивов в функцию все размерности должны передаваться в качестве параметров.

```
const int N=4;//глобальная переменная
void transp(int a[][N],int n)// транспонирование матрицы
{
     int r;
     for(int I=0;I<n;I++)
     for(int j=0;j<n;j++)
     if(I<j)
     {
     r[a[I][j];
     a[I][j]=a[j][I];
     a[j][I]=r;
     }
}
```

3. Постановка задачи

1. Используя функции сформировать с помощью ДСЧ одномерный массив и вывести его на консоль.
2. Выполнить обработку одномерного массива (задача 1) в соответствии с вариантом, используя функции, результат вывести на консоль.
3. Сформировать двумерный массив – задача 2, заполнить его случайными числами и вывести на консоль, используя функции.

4. Варианты

Таблица 1

Вариант	Одномерный массив	Двумерный массив
1	Удалить первый четный элемент	Найти минимальный элемент в каждой строке матрицы.
2	Удалить первый отрицательный элемент	Найти максимальный элемент в каждой строке матрицы.
3	Удалить элемент с заданным ключом (значением)	Найти количество положительных элементов в каждой строке матрицы.
4	Удалить элемент равный среднему арифметическому элементов массива	Найти количество отрицательных элементов в каждой строке матрицы.
5	Удалить элемент с заданным номером	Найти среднеарифметическое значение элементов в каждой строке матрицы.
6	Удалить N элементов, начиная с номера K	Найти количество элементов, больших заданного значения, в каждой строке матрицы.
7	Удалить все четные элементы	Найти количество элементов, меньших заданного значения, в каждой строке матрицы.
8	Удалить все элементы с четными индексами	Перевернуть все четные строки матрицы.
9	Удалить все нечетные элементы	Все строки матрицы сдвинуть циклически на K элементов вправо.
10	Удалить все элементы с нечетными индексами	Все нечетные строки матрицы сдвинуть циклически на K элементов влево.
11	Добавить элемент в начало массива	Перевернуть все нечетные строки матрицы.
12	Добавить элемент в конец массива	Перевернуть все четные столбцы матрицы.
13	Добавить K элементов в начало массива	Перевернуть все нечетные строки матрицы.

14	Добавить К элементов в конец массива	Перевернуть все нечетные столбцы матрицы.

6. Содержание отчета

1. Постановка задачи (общая и для конкретного варианта).
2. Определения функций, используемых для формирования, печати и обработки массивов/строк (для каждой задачи).
3. Определение функции main().
4. Результаты тестов