Практическое задание 7
Массивы в C++. Функции обработки массивов

Цель. Изучение синтаксических правил описания массивов и механизмов их реализации. Разработка алгоритмов обработки массивов в функциях. Массивы как параметры функций. Параллельные массивы. Перегрузка функций.

ПЛАН

1. Краткое теоретическое введение

Массив как конструируемый тип

Напомним определение массива. Массив, это упорядоченное множество данных одного типа, для которых в оперативной памяти процесса распределяется непрерывная область адресного пространства, адресуемая именем массива.

В отличие от базовых типов, массив нужно перед употреблением конструировать. Это означает, что в описании переменной типа «массив» необходимо указать, как распределить память для хранения его элементов.

Синтаксис описания массива:

```
Тип Имя_массива [Кол_во_элементов];  // Возможно, список инициализации
```

`Кол_во_элементов` – константная величина, то есть константа в чистом виде или `define` константа.

`Тип` – практически любой тип данных.

Семантика:

компилятор распределяет память для хранения элементов массива в указанном количестве. `Имя_массива` сопоставлено всей совокупности данных и адресует первый байт в этой области. Элементы массива размещаются подряд в соответствии с ростом номера элемента внутри массива (индекса). Контроля выхода за границу массива нет. Нумерация элементов в массиве начинается с нуля.

Операции с массивами

Единственная операция, разрешеннная для массива, это операция обращения к элементу массива [], синтаксис которой:

```
Операнд_1 [Операнд_2]
```

Фактически, это составное имя:

```
Имя_массива [Индекс]
```

Индекс, это выражение целого типа, определяющее номер элемента внутри массива (нумерация начинается с 0). Семантика операции [] позволяет получить доступ к одному элементу с указанным номером.

Инициализация массивов

Инициализация, это присвоение значения при объявлении. Для массива можно задать список инициализирующих значений так, как в примере:

```
int Arr[] = {1,2,3,4,5};
```

Так как длина массива не указана, ее можно определить так:

```
int len_a;
len_a = sizeof(a)/sizeof(int);
```

В этом примере длина массива равна 5, и ровно 5*4=20 байт памяти распределено для хранения элементов массива.

Управление в массивах

Как правило, при обработке данных массива действия применяются ко всем элементам массива по очереди. Следовательно, управление выполняется в циклах, где управляющей переменной является индекс элемента массива.

В примере для массива `Arr`, длина которого `len_a`, индекс меняется от 0 до `len_a-1`, с приращением = 1, следовательно, управление, это цикл:

```
for (int i=0; i<len_a; i++)
{
   // обращение к Arr[i]
}
```

2. Функции работы с массивами

Функции работы с массивами:

1) решают задачу обработки массива «в общем виде»;
2) получают массив через параметры «в общем виде»;
3) могут обрабатывать массив любой длины.

Следовательно, функция работы с массивом, это алгоритм, написанный всерьез и надолго. Он будет работать с любым массивом любой длины при условии, что имя массива и длину получит при обращении.

Синтакс формального параметра, которым является массив, показан в прототипе функции, вычисляющей среднее арифметическое значение элементов массива:

```
float Avg(float a[], int len);
```

Первый параметр, это массив, второй – его фактическая длина.

Мы видим, что к имени `a` присоединен суффикс `[]`, который и означает, что параметром является массив.

Семантика: масивы всегда передаются в функцию по ссылке, то есть функция будет изменять массив, если это заложено в ее алгоритме.

Тип функции может быть `void` или любой другой, зависит от задачи.

3. Описание, инициализация массива

Заголовочный файл `Array.h` скопируйте в папку проекта и присоедините к проекту, откройте и ознакомьтесь с содержимым. Все описания функций выполняются здесь, а управления вызовами – в `Source` файле проекта.

<u>Упражнение 1. Описание и инициализация массива, передача в функцию</u>

1.1. Опишите и инициализируйте массив вещественных чисел:

```
float Arr[]={1.5,2.5,3.,12.};
```

Найдите его длину:

```
int len;
len = sizeof(a)/sizeof(int);
```

Выведите массив функцией `Print_Arr.`

Передайте массив в функцию `Avg:`

```
float A = Avg(Arr,len);
```

Выведите решение.

1.2. Опишите массив вещественных чисел, емкость которого = 10.

```
float My_Arr[10];
```

Опишите переменную, обозначающую фактическую длину массива:

```
int Len_a = 5;
```

Введите `Len_a` элементов функцией `Input_Arr`, выведите на экран функцией

Print_Arr. Найдите и выведите значение наибольшего элемента массива.

Упражнение 2. Задание для самостоятельной разработки.

Опишите функцию Transform, получающую вещественный массив, которая заменит все элементы, которые больше, чем среднее арифметическое, его значением.

Обратитесь к ней, например, с тем же массивом My_Arr.

Функция изменит элементы массива, это будет известно в main. Выведите измененный массив функцией Print_Arr.

4. Параллельные массивы

Пусть нужно разработать алгоритмы, данными для которых являются точки на координатной плоскости. Как хранить данные о точках, если каждая точка, это пара координат: A(*x,y*), B(*x,y*) и так далее?

Естественное решение, это разработать структуру данных типа «точка», и создать массив точек. Но есть и иное решение, примитивное, но знать о котором полезно.

Описывают два массива одинаковой длины, например:

```
float x[], float y[];   // Координаты точек.
int Count;              // Количество точек.
```

Управление в массивах осуществляется как обычно, но в теле цикла обращение происходит к паре координат, тем самым, к очередной точке.

```
for (int i=0; i<len_a; i++)
{
   // обращение к x[i], y[i]      // это i-тая точка.
}
```

Упражнение 3. Исследование принципов работы с параллельными массивами.

Функция Point получает два массива координат точек: x[] и y[] и находит номер точки, наиболее удаленной от начала координат.

Опишите исходные данные для функции (достаточно трех точек для проверки), обратитесь к функции, и затем выведите координаты искомой точки, номер которой вернула функция.

Упражнение 4. Задание для самостоятельной разработки.

Дано некоторое произвольное количество точек на плоскости. Опишите, отладьте и протестируйте функцию, которая удалит из массива самую дальнюю точку. Пример того, как это делается в одномерном массиве, приведен в функции Del.

Бонус

Проинициализируйте два массива, которые задают n точек координатами (X,Y) в двумерном пространстве.

Опишите функцию, которая находит расстояние между двумя любыми точками.

Опишите функцию, которая найдет расстояние между всеми точками и выведет их на экран в виде таблицы.