Массивы

Базовое решение всех этих проблем в С++ обеспечивают массивы*.*

Массив- это просто некоторое множество мест в памяти, называемых *элементами массива ,* или просто элементами, каждый из которых может хранить единицу дан­ных определенного типа, и к которым можно обращаться по одному имени перемен­ной.

Имена сотрудников в программе платежной ведомости могут быть помещены в один массив, суммы выплат - в другой, а суммы налогов каждого сотрудника - в третий.

Индивидуальные элементы массива определяются значением индекса, кото­рый представляет собой просто целое число - порядковый номер элемента в масси­ве, причем первый имеет индекс 0, второй - 1 и так далее.

Вы можете представлять индекс массива как смещение от его первого элемента. Первый элемент имеет смещение 0, а потому — индекс 0, а индекс 3 ссылается на четвертый элемент в массиве.

Что касается платежной ведомости, вы можете упорядочить массивы таким образом, что если имя определенного сотрудника помещено в элемент массива *Name* с определенным значением индекса, то массивы *pay*и *tax* также должны хранить данные, ассоциированные с этим сотрудником, в элементах с тем же индексом.



На рисунке показан массив. Имя *height* содержит шесть элементов, каждый содержит отдельное значение. Это могут быть значения роста членов семьи, измеренные в дюймах. Поскольку имеется шесть элементов, значение индекса находится в пределах от 0 до 5.

Чтобы сослаться на определенный элемент, вы пишете имя массива, за которым идет значение индекса определенного элемента, заключенное в квадратные скобки. Так, например, чтобы сослаться на третий элемент, нужно написать *height*[2]. Если вы представите индекс в виде смещения от первого элемента, то легко увидеть, что значение индекса четвертого элемента будет равно 3.

Общий объем памяти, необходимый для хранения каждого элемента, определяется его типом, и все элементы массива сохраняются в одном непрерывном блоке памяти.

Объявление массивов

По сути, вы объявляете массив точно так же, как до сих пор объявляли переменные. Единственным отличием является то, что рядом с именем массива нужно указать количество его элементов. Например, вы можете объявить массив целых чисел *height*, показанный на рисунке, используя следующий оператор объявления:

*long height[6];*

Поскольку каждое значение типа *long* занимает 4 байта памяти, весь массив потребует 24 байт. Массивы могут иметь любой размер - количество элементов ограничено лишь объемом доступной памяти компьютера, на котором выполняется ваша программа.

Вы можете объявить массив любого типа. Например, чтобы объявить массив, предназначенный для хранения рабочего объема и выходной мощности серии двига­телей, вы можете написать следующее:

*double cubic\_inches[10]; // Объемы двигателей в кубических дюймах*

*double horsepower[10]; // Выходные мощности*

Таким образом можно хранить рабочие объемы и мощности до 10 двигателей, ссылаясь на них по индексам от 0 до 9. Как вы уже видели это на примере других переменных, можно объявлять несколько массивов одного и того же типа в одном операторе, но на практике почти всегда лучше объявлять эти переменные в отдельных операторах.

*double power[10];*

*int mas[10];*

Инициализация массивов

Чтобы инициализировать массив при его объявлении, нужно поместить начальные значения элементов в ограниченный фигурными скобками список, разделенный запятыми, который находится справа от знака равенства, следующего за именем массива.

Пример

*int array[5] = {200,250,300,350,400};*

Массив называется *array*, содержит 5 элементов, каждый из которых хранит значение типа *int*. Элементы массива инициализированы списком значений в фигурных скобках, соответствующим последовательности значений индекса массива, поэтому в данном случае *array[0]* получит значение 200, *array[1]* получит значение 250, *array[2]* получит значение 300 и т.д.

Нельзя поместить в скобках больше значений, чем объявлено элементов в массиве, но меньше указать можно. Если задано меньше значений, то они присваиваются последовательно элементам массива, начиная с первого – т.е. элемента с индексом 0. Элементы массива, для которых не указаны начальные значения, инициализируются нулями. Это не то же самое, что происходит, если вообще не указать списка инициализации. Без списка инициализации элементы массива получают произвольные случайные значения. Кроме того, если вы включаете список инициализации, в нем должен присутствовать хотя бы один элемент, иначе компилятор генерирует сообщение об ошибке.

// Листинг 16

//Демонстрация инициализации массива

#include <iostream.h>

#include <iomanip.h>

int main()

{

int value[5] = {1,2,3};

int junk[5];

int i;

cout << endl;

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

cout << setw(12) << value[i];

cout << endl;

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

cout << setw(12) << junk[i];

cout << endl;

return 0;

}

Результат

1 2 3 0 0

-858993460 -858993460 -858993460 -858993460 -858993460

В этом примере объявляются два массива. Массив *value* инициализирован частично, а второй- *junk*, не инициализируется вообще.

Первые три элемента массива *value* инициализируются указанными значениями, а остальные два – по умолчанию нулями. В случае *junk* все значения случайны, поскольку никаких начальных значений не указано вообще. Элементы этого массива содержат мусор.

Удобный способ инициализации всего массива нулевыми значениями – это просто указать единственное начальное значение, равное 0.

*int data[100] = {0};*

Этот оператор объявляет массив *data* размером 100 элементов, нициализированных нулями. Первый элемент инициализируется значением, указанным в фигурных скобках , а остальные инициализируются нулями по умолчанию, поскольку значения для них не указаны явно.

Можно также опустить размер массива числового типа, если указать список значений инициализации. Количество элементов в массиве определяется количеством указанных инициализирующих значений.

Объявление массива

*int value [] = {2,3,4};*

определяет массив из трех элементов с начальными значениями 2,3 и 4.

Конечно, если размер массива велик, то таким способом выполнять инициализацию неудобно. В таком случае для ввода начальных значений элементов массива можно воспользоваться циклом.

*// Листинг 17*

*//Демонстрация инициализации массива*

*//с использованием цикла*

*#include <iostream.h>*

*#include <iomanip.h>*

*int main()*

*{*

*int value[10];*

*int i;*

*for (i=0;i<10; i++)*

*{*

*cout<<"input element"<<endl;*

*cin>> value[i];*

*}*

*for (i = 0; i <10; i++)*

*cout << setw(4) << value[i];*

*cout << endl;*

*return 0;*

*}*

*Результат*

*input element*

*1*

*input element*

*2*

*input element*

*3*

*input element*

*4*

*input element*

*5*

*input element*

*6*

*input element*

*7*

*input element*

*8*

*input element*

*9*

*input element*

*10*

*1 2 3 4 5 6 7 8 9 10*

В этом примере объявлен массив *value* , инициализация которого происходит в цикле с использованием оператора *cin.* Значения элементам мы задаем сами , различные при каждом запуске программы.

Так инициализировать массив при его значительных размерах также не очень удобно. Часто возникает необходимость инициализации массива случайными числами.

*// Листинг 18*

*//Демонстрация инициализации массива*

*//случайными числами*

*#include <iostream.h>*

*#include <iomanip.h>*

*#include <stdlib.h>*

*#include <time.h>*

*int main()*

*{*

*// Инициализация генератора случайных чисел*

*srand((unsigned)time(NULL));*

*int value[25];*

*int i;*

*for (i=0;i<25; i++)*

*{*

*value[i]=rand()%50;*

*cout << setw(4) << value[i]<< endl;*

*return 0;*

*}*

*Результат*

*17 37 46 8 33 15 20 18 9 15 31 24 41 19 1 22*

*17 8 44 8 5 30 22 17 39*

В этом примере массив инициализируется случайными числами , с помощью генератора случайных чисел, функции *rand()* .Перед использованием, его необходимо проинициализировать. Для этого используется функция *srand((unsigned)time(NULL));*

Как видно, инициализация выполнена таймером, соответствующая функция, является аргументом функции *srand()* .

Теперь каждому элементу массива присваивается значение равное остатку от деления случайного числа, возвращенного генератором случайных чисел на 50.

Понятно, что элементы массива в таком случае получат значения в диапазоне от 0 до 49 включительно. Для использования таймера и генератора случайных чисел необходимо присоединить заголовочные файлы *time.h и stdlib.h.*

**С++ для огородника или**

Многомерные массивы

Массивы, которые мы определяли до сих пор, имеют один индекс и называются одномерными массивами. Но массив может иметь и более одного индексного значения- в этом случае он называется многомерным массивом.

Пример.

Предположим, что у вас есть поле, на котором расположена плантация бобов, по 10 растений на грядке, и это поле содержит 12 таких грядок (то есть всего имеется 120 единиц растений). Вы можете объявить массив для записи веса бобов, собранных от каждого растения, используя следующий оператор:

*double beans[12] [10];*

В данном случае объявляется двумерный массив *beans*, где первый индекс — номер грядки, а второй - номер растения на грядке. Чтобы обратиться к любому

конкретному элементу массива, необходимо указать два индекса. Например, вы можете установить значение элемента, соответствующего пятому растению в третьей грядке, следующим оператором:

*beans[2][4] = 10.7;*

Напомним, что значения индексов начинаются с нуля, поэтому значение индекса третьей грядки- 2, а индекс пятого растения - 4.

Будучи успешным бобовым фермером, вы можете иметь несколько таких полей, засеянных бобами по тому же шаблону. Предполагая, что у вас 8 полей, можно определить трехмерный массив для записи данных об урожае следующим образом:

*double beans[8][12] [10];*

Такая запись позволяет организовать учет каждого растения, растущего на всех этих полях, причем самый левый индекс ссылается на определенное поле. Если вы когда-нибудь займетесь выращиванием бобов в международном масштабе, то сможете использовать четырехмерный массив, где дополнительное измерение будет обозначать страну. Если предположить, что вы настолько же успешный продавец, как и фермер, то выращивание такого количества бобов, в конце концов, станет оказывать влияние на озоновый слой.

Массивы хранятся в памяти так, что самый правый индекс растет быстрее всего. То есть массив data [ 3 ] [ 4 ] это двумерный массив, состоящий из массивов по четыре элемента в каждом. Организация такого массива показана на рисунке.

Элементы массива располагаются в непрерывном блоке памяти, как показано стрелочками на рисунке. Первый индекс выбирает определенную строку внутри массива, а второй индекс выбирает элемент внутри строки.

Обратите внимание, что двумерный массив в родном С++ на самом деле одномер­ный массив, состоящий из одномерных массивов. Массив с тремя измерениями в родном С++ это одномерный массив элементов, в котором каждый элемент представляет собой одномерный массив одномерных массивов.



Инициализация многомерных массивов

Для того чтобы инициализировать многомерный массив, используется расширенный метод инициализации одномерных массивов.

Например, можно инициализи­ровать двумерный массив *data* с помощью следующего объявления:

*long data[2][4] {*

*{ 1, 2, 3, 5 },*

*{ 7, 11, 13, 17 }*

*};*

То есть, инициализация значений каждой строки массива содержится внутри собственной пары фигурных скобок. Поскольку в каждой строке четыре элемента, в каждой группе присутствует по четыре значения инициализации, и поскольку строк всего две, внутри скобок находится две группы инициализирующих значение, разде

ленных запятой. Вы можете пропустить инициализацию значений в любой строке, в этом случае остальные элементы массива инициализируются нулевыми значениями, например:

*long data[2] [4] = {*

*{ 1, 2, 3 },*

*{ 7, 11 }*

*};*

Списки инициализации дополнены пробелами, чтобы показать, где пропущены значения. Элементы *data [0][3]*, *data [1][2]* и *data* *[1][3]* не получают инициали­зирующих значений и потому равны нулю.

Если вы хотите инициализировать весь массив нулевыми значениями, можно написать просто:

*long data[2] [4] = {0};*

*// Листинг 19*

*//Демонстрация инициализации двумерного массива*

*//случайными числами*

*#include <iostream.h>*

*#include <iomanip.h>*

*#include <stdlib.h>*

*#include <time.h>*

*int main()*

*{*

*// Инициализация генератора случайных чисел*

*srand((unsigned)time(NULL));*

*int value[10][10];*

*int i,j;*

*for (i=0;i<10; i++)*

*{*

*for (j=0;j<10; j++)*

*{*

*value[i][j]=rand()%50;*

*cout << setw(5) << value[i][j];*

*}*

*cout << endl;*

*}*

*cout << endl;*

*return 0;*

*}*

*Результат*

*31 2 43 7 28 44 12 47 21 47*

*31 47 27 25 8 8 31 22 21 44*

*25 37 33 34 32 8 40 33 9 3*

*23 21 37 3 26 45 14 42 34 19*

*32 30 38 22 18 8 48 1 8 26*

*8 41 44 25 34 46 4 43 44 1*

*29 2 17 31 0 49 39 35 1 39*

*20 48 45 20 20 8 2 13 43 45*

*44 36 34 7 33 38 43 37 1 5*

*29 23 47 7 39 43 26 38 23 23*

В этом примере двумерный массив инициализируется случайными числами, аналогично задаче в предыдущем примере. Только для двумерного массива требуется два цикла, внешний цикл- по строкам, внутренний цикл –по столбцам.