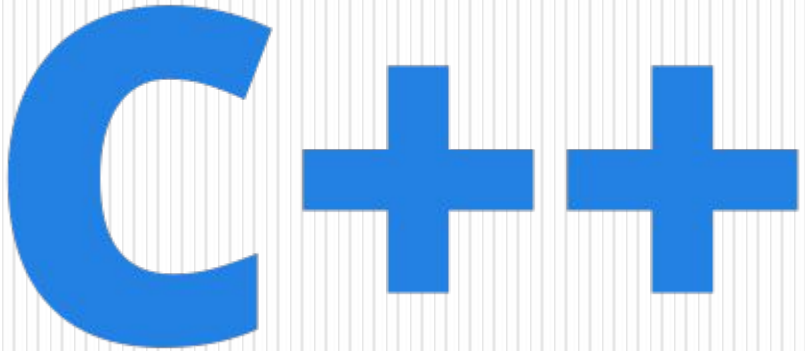
Лекция 5

Массивы, одномерные массивы, многомерные массивы, динамические массивы



Содержание

Массив

Массив – это пронумерованный набор однотипных элементов. Массивы бывают статическими и динамическими. У статического массива количество элементов известно заранее и не может быть изменено. У динамического массива количество элементов заранее неизвестно и определяется в процессе выполнения программы.

Также массивы различаются по размерности: одномерные, двумерные, трехмерные и т.д. Примером одномерного массива может послужить вектор a{1,4,3,5}. Примером двухмерного массива может послужить матрица. Примером трехмерного массива может послужить набор высот местности.

Массивы различают по типу элементов. Бывают целочисленные, вещественные (состоящие из дробных чисел), символьные массивы.

Примеры массивов:

- вектор a{1,-4,3,5} – одномерный вещественный массив из трех элементов;
- матрица – двумерный целочисленный массив из шести элементов;
- {“x”3”%”} – одномерный символьный массив;
- {“x-3.31,”%”} – не является массивом, т.к. часть элементов символы, часть элементов числа.

Индекс – это номер элемента в массиве.

У одномерного массива один индекс, обычно он обозначается .

Чтобы использовать одномерный массив в программе, необходимо:

1. объявить массив в функции main():
2. тип_данных имя_массива[количество элементов];
3. double a[3]; //статический массив a из трех дробных чисел
4. int b[7]; //статический массив b из семи целых чисел проинициализировать массив, т.е. задать каждому элементу конкретное числовое значение;
5. провести вычисления, исследования.
6. Примечание. Индексация в массиве начинается с 0, т.е. индекс у самого первого элемента в массиве i = 0. Индексация в массиве a(7)={-10;0.2;3;-4.7;0.5;-8;11} указана таблице

индекс i	0	1	2	3	4	5	6
a[i]	-10	0,2	3	-4,7	0,5	-8	11

Одномерные массивы

При использовании простых переменных каждой области памяти для хранения данных соответствует свое имя. Если с группой величин одинакового типа требуется выполнять однообразные действия, им дают одно имя, а различают по порядковому номеру. Это позволяет компактно записывать множество операций с помощью циклов. Конечная именованная последовательность однотипных величин называется массивом. Описание массива в программе отличается от описания простой переменной наличием после имени квадратных скобок, в которых задается количество элементов массива (размерность):

float a [10]; // описание массива из 10 вещественных чисел

При описании массивов квадратные скобки являются элементом синтаксиса, а не указанием на необязательность конструкции.

Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных. Инициализирующие значения для массивов записываются в фигурных

скобках. Значения элементам присваиваются по порядку. Если элементов в массиве больше, чем инициализаторов, элементы, для которых значения не указаны, обнуляются:

int b[5] = {3. 2. 1}; / / b[0]=3. b[1]=2. b[2]=1. b[3]=0. b[4]=0

Для доступа к элементу массива после его имени указывается номер элемента
(индекс) в квадратных скобках. В следующем примере подсчитывается сумма элементов массива.

```
#include "pch.h"
#include <iostream>
using namespace std;
int main(){
const int n = 10;
int i, sum;
int marks[n] = {3, 4, 5, 4, 4};
for (i = 0; sum = 0; 1<n; i++) sum += marks[i];
cout << "Сумма элементов: " << sum;
return 0;
}
```


Размерность массивов предпочтительнее задавать с помощью именованных констант, как это сделано в примере, поскольку при таком подходе для ее изменения

достаточно скорректировать значение константы всего лишь в одном месте программы. Следует обратить внимание, что последний элемент массива имеет номер, на единицу меньший заданной при его описании размерности.

При обращении к элементам массива автоматический контроль выхода индекса за границу массива не производится, что может привести к ошибкам.

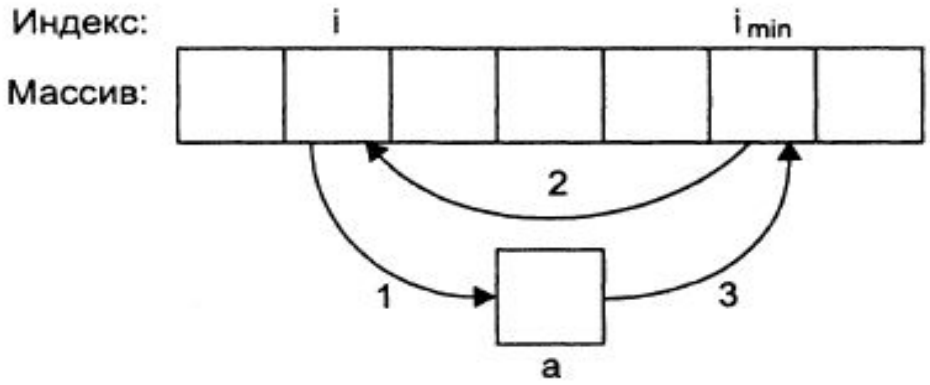
Пример

Сортировка целочисленного массива методом выбора. Алгоритм состоит в том, что выбирается наименьший элемент массива и меняется местами с первым элементом, затем рассматриваются элементы, начиная со второго, и наименьший из них меняется местами со вторым элементом, и так далее n-1 раз (при последнем проходе цикла при необходимости меняются местами предпоследний и последний элементы массива). Пример кода на следующем слайде.

Пример кода

```
#include "pch.h"
#include <iostream>
using namespace std;
int main() {
const int n = 20:                    // количество элементов массива
int b[n];                            // описание массива
int i;
for (i = 0; i<n; i++) cin >> b[i];  // ввод массива
for (i = 0; i<n-1; i++) {            // n-1 раз ищем наименьший элемент
// принимаем за наименьший первый из рассматриваемых элементов:
int imin = 1:
// поиск номера минимального элемента из неупорядоченных:
for (int j = i + 1; j<n; j++)
// если нашли меньший элемент, запоминаем его номер:
if (b[j] < b[imin]) imin = j;
int a = b[i];                        // обмен элементов
b[i] = b[imin];    // с номерами
b[imin] = a;      // i  и imin
}
// вывод упорядоченного массива:
for (i = 0; i<n; i++)cout << b[i] << ' ';
return 0:
}
```


Процесс обмена элементов массива с номерами 1 и 1m1n через буферную переменную a на 1-м проходе цикла проиллюстрирован на рисунке



Динамический массив

Динамические массивы создают с помощью операции new, при этом необходимо указать тип и размерность, например:

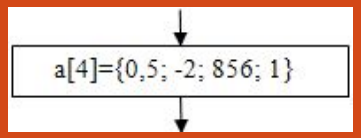
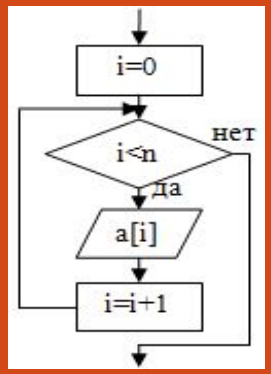
int n = 100:

float *p = new float [n];

В этой строке создается переменная-указатель на float, в динамической памяти

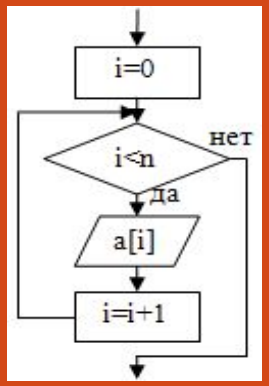
отводится непрерывная область, достаточная для размещения 100 элементов вещественного типа, и адрес ее начала записывается в указатель p. Динамические массивы нельзя при создании инициализировать, и они не обнуляются. Преимущество динамических массивов состоит в том, что размерность может быть переменной, то есть объем памяти, выделяемой под массив, определяется на этапе выполнения программы.

Способы инициализации одномерного массива

Часть блок-схемы	Часть программы
инициализация числами 	double a[4]={0.5, -2,856, 1};
с клавиатуры: 	double a[n]; int i; for(i=0; i<n; i=i+1){cout<<"a["<<i<<"]=";cin>>a[i]; }

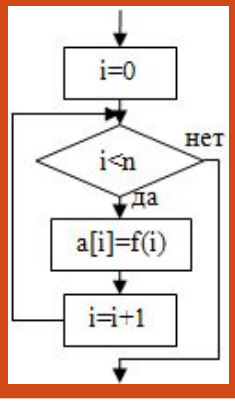
Способы инициализации одномерного массива

из файла:



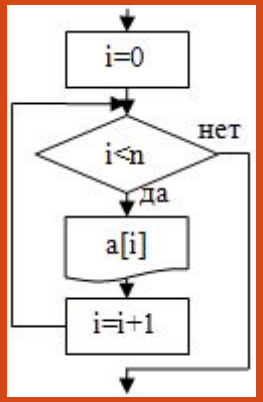

```
double a[n];
int i;
fstream file;
file.open("1.txt", ios::in);
for(i=0; i<n; i=i+1){file>>a[i]; }
file.close();
```

по заданной формуле a[i]=f(i):



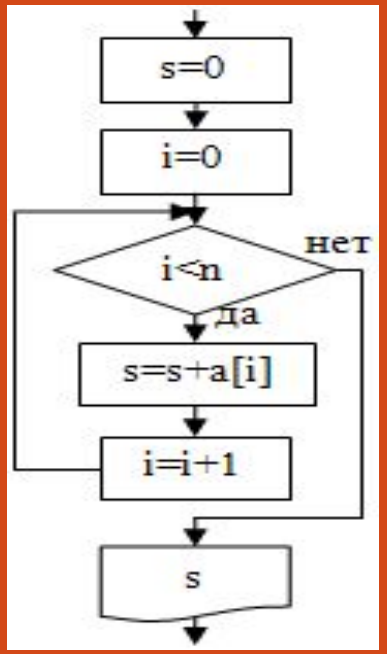

```
double a[n];
int i;
for(i=0; i<n; i=i+1){a[i]=f(i);  }
```

Вывод одномерного массива

Часть блок-схемы	Часть программы
	for(i=0; i<n; i=i+1){cout<<"a["<<i<<"]="<<a[i]<<endl;}

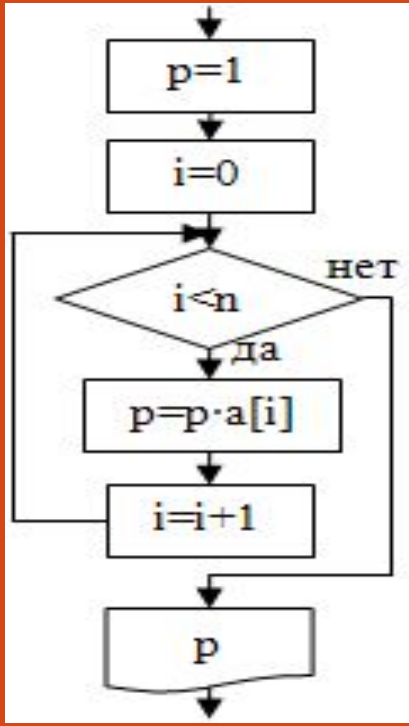
как сумма, произведение, минимальное, максимальное значение

нахождение суммы:



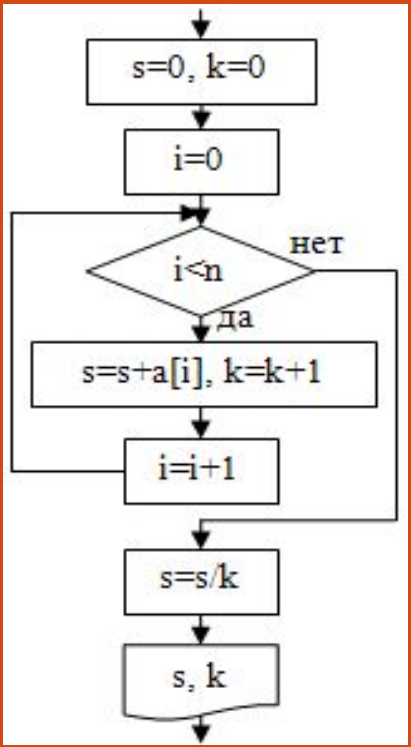

```
s=0;
for(i=0; i<n; i=i+1){      s=s+a[i];}
cout<<"s="<<s<<endl;
```

нахождение произведения:



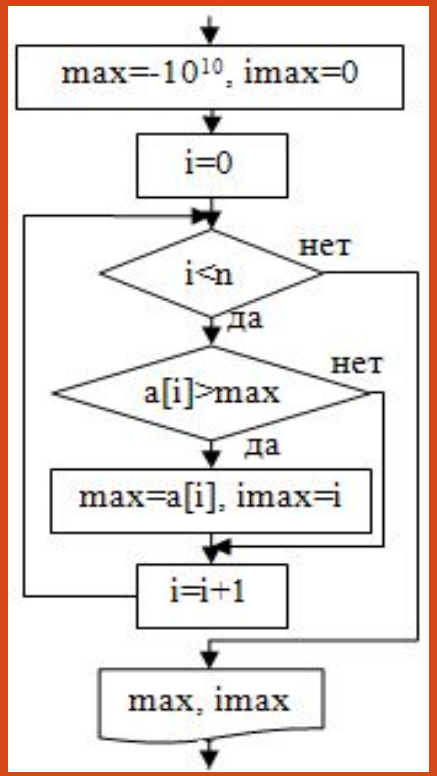

```
p=1;
for(i=0; i<n; i=i+1)
{     p=p*a[i];}
cout<<"p="<<p<<endl;
```

нахождение среднего арифметического и количества элементов:



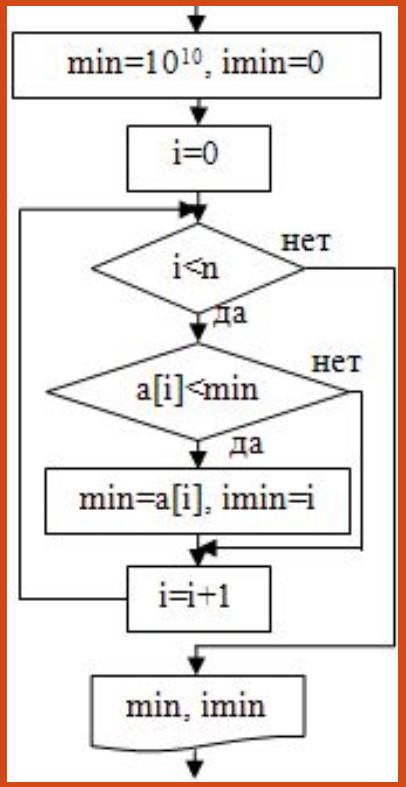

```
s=0, k=0;
for(i=0; i<n; i=i+1){
s=s+a[i]; k=k+1;}
s=s/k;
cout<<"s="<<s<<endl;
cout<<"k="<<k<<endl;
```

нахождение максимального элемента:



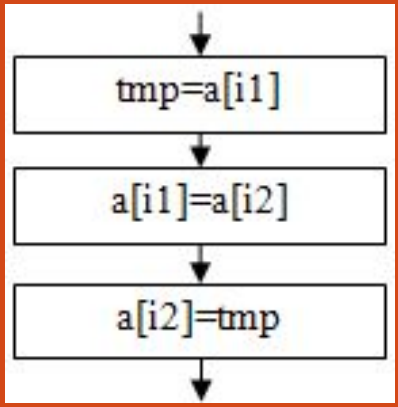

```
max=-10E10;
imax=0;
for(i=0; i<n; i=i+1)
{
if(a[i]>max)
{           max=a[i];           imax=i;  }
}
cout<<"max="<<max<<"
imax="<<imax<<endl;
```

нахождение минимального элемента:




```
min=10E10;
imin=0;
for(i=0; i<n; i=i+1){
if(a[i]<min){
min=a[i];
imin=i;}
}
cout<<"min="<<min<<"
imin="<<imin<<endl;
```

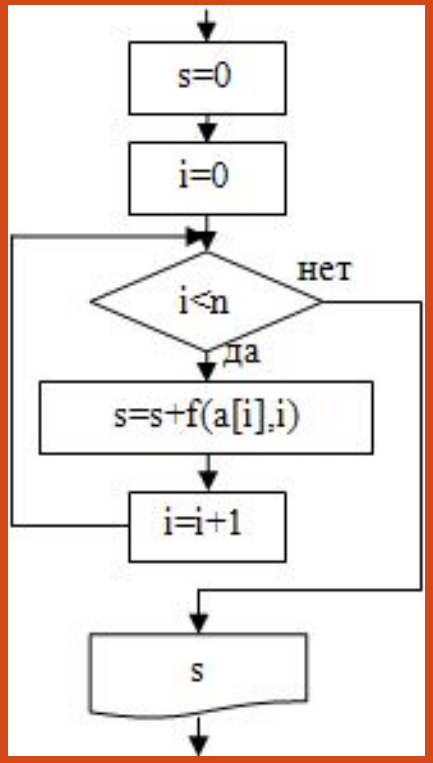
поменять местами элементы с индексами i1 и i2:




```
tmp=a[i1];
a[i1]=a[i2];
a[i2]=tmp;
```

вычисление формулы

$$S = \sum_{i=0}^{n=1} f(a[i], i)$$



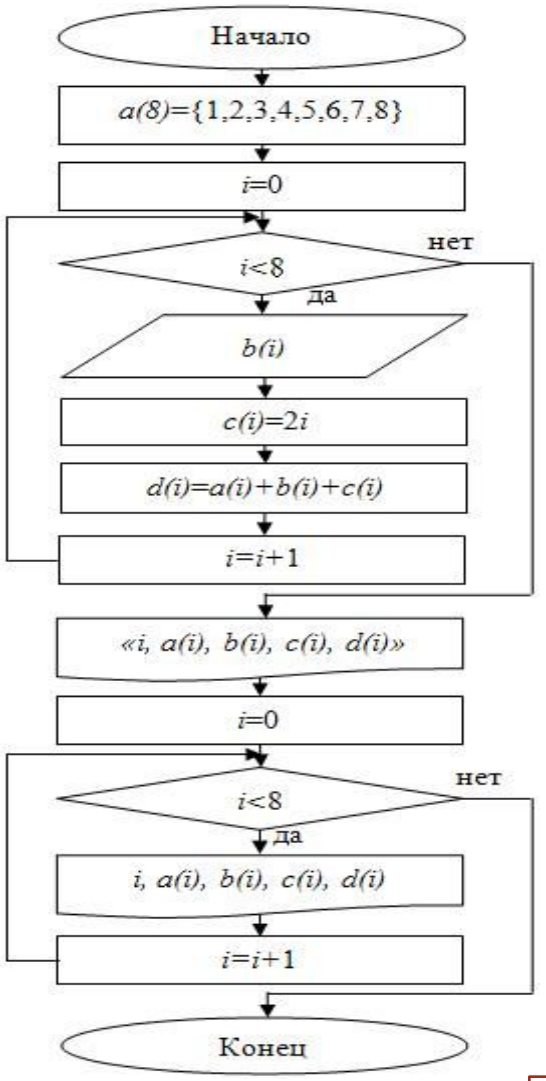

```
s=0;
for(i=0; i<n; i=i+1){
s=s+f(a[i],i);}

cout<<"s="<<s<<endl;
```

Пример

Далее представлен пример, в котором даны четыре одномерных массива: a(8) = {1,2,3,4,5,6,7,8}, b(8) вводится с клавиатуры, c(8) вычисляется по формуле $c_i=2i$, d(8) вычисляется по формуле $d_i=a_i+b_i+c_i$. Построить таблицу значений массивов.

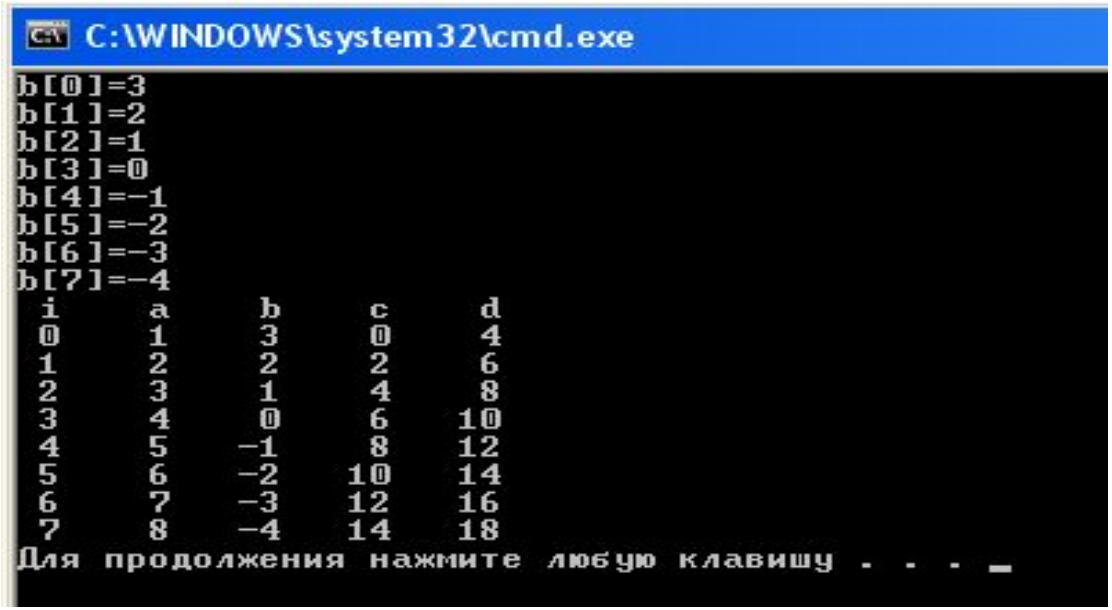
Для решения сначала необходимо проинициализировать массивы согласно условию задачи. Массив a задан числами (первый способ инициализации), поэтому он будет проинициализирован при объявлении.



Код программы (Visual Studio) с оператором for:

```
// proga.cpp: определяет точку входа для консольного приложения
#include "pch.h«
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
double a[8]={1,2,3,4,5,6,7,8}; double b[8], c[8], d[8];  int i;
for(i=0; i<8; i=i+1){
                cout<<"b["<<i<<"]=";
                cin>>b[i];
                c[i]=2.0*i;
                d[i]=a[i]+b[i]+c[i];
        }
cout<<setw(2)<<"i"<<setw(5)<<"a"<<setw(5)<<"b"<<setw(5)<<"c"<<setw(5)<<"d"<<endl;
        for(i=0; i<8; i=i+1){
cout<<setw(2)<<i<<setw(5)<<a[i]<<setw(5)<<b[i]<<setw(5)<<c[i]<< setw(5)<<d[i]<<endl;
        }
        return 0;}
```


Результат выполнения

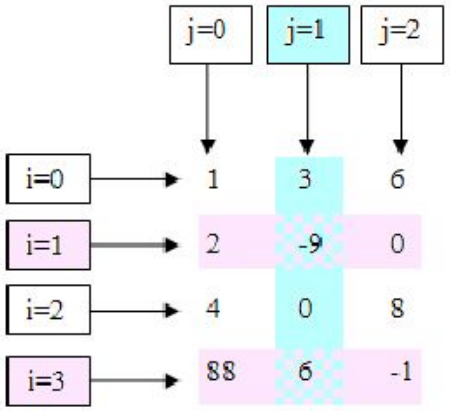


Многомерные массивы

Многомерные массивы в C++ задаются указанием каждого измерения в квадратных скобках, например, оператор

int matr [6][8]:

задает описание двумерного массива из 6 строк и 8 столбцов. В памяти такой массив располагается в последовательных ячейках построчно. Многомерные массивы размещаются так, что при переходе к следующему элементу быстрее всего изменяется последний индекс. Для доступа к элементу многомерного массива указываются все его индексы, например, matr[i][j], или более экзотическим способом: *(matr[i]+j) или *(*(matr+i)+j). Это возможно, поскольку matr[i] является адресом начала i-й строки массива.



При инициализации многомерного массива он представляется либо как массив из массивов, при этом каждый массив заключается в свои фигурные скобки (в этом

случае левую размерность при описании можно не указывать), либо задается общий

список элементов в том порядке, в котором элементы располагаются в памяти:

int mass2 [][]={ {1, 1}, {0, 2}, {1, 0} };

int mass2 [3][2]={1, 1, 0, 2, 1, 0};

Далее рассмотрен пример программы, которая определяет в целочисленной матрице номер строки, которая содержит наибольшее количество элементов, равных нулю.

Код программы

```
#include "pch.h"
#include <iostream>
using namespace std;
int main(){
const int nstr = 4. nstb = 5;
// размерности массива
int b[nstr][nstb];
// описание массива
int i, j;
for (i = 0; i<nstr; i++) // ввод массива
for (j = 0; j<nstb; j++) scanf("%d", &b[i][j]);
int istr = -1, MaxKol = 0;
for (i = 0; i<nstr; i++){ // просмотр массива по строкам
int Kol = 0;
for (j = 0; j<nstb; j++) if (b[i][j] == 0)Kol++;
if (Kol > MaxKol){istr = i; MaxKol = Kol;}
}
printf(" Исходный массив:\n");
for (i = 0; i<nstr; i++){
for (j = 0; j<nstb; j++)printf("%d ", b[i][j]);
printf("\n");}
if (istr == -1)printf("Нулевых элементов нет");
else printf("Номер строки: %d", istr);
return 0;}
```


Номер искомой строки хранится в переменной istr, количество нулевых элементов в текущей (i-й) строке - в переменной Kol, максимальное количество нулевых элементов - в переменной MaxKol. Массив просматривается по строкам, в каждой из них подсчитывается количество нулевых элементов (обратите внимание, что переменная Kol обнуляется перед просмотром каждой строки). Наибольшее количество и номер соответствующей строки запоминаются.

Динамический многомерный массив

Для создания динамического многомерного массива необходимо указать в операции new все его размерности (самая левая размерность может быть переменной), например:

```
int nstr = 5;
int ** m = (int **) new int [nstr][10];
```

Более универсальный и безопасный способ выделения памяти под двумерный массив, когда обе его размерности задаются на этапе выполнения программы, приведен ниже:

```
int nstr, nstb;
cout << " Введите количество строк и столбцов :";
cin >> nstr >> nstb;
int **a = new int *[nstr];
for(int i = 0; i<nstr; i++)
a[i] = new int [nstb];
...
```


Видеоурок «Массивы в C++»

Контрольные вопросы

1. Что такое массив?
2. Перечислите виды массивов.
3. Что такое индекс?
4. Как добиться отображения массива на экране в виде ровной таблицы?
5. Назовите способы инициализации массивов

Список литературы

- Павловская Т.А. С/С++. Программирование на языке высокого уровня / Т. А. Павловская. - СПб.: Питер, 2004. - 461 с.: ил.
- Павловская Т.А. С/С ++. Структурное программирование: Практикум / Т.А. Павловская, Ю.А. Щупак. СПб.: Питер, 2007. - 239 с.: ил.
- Павловская Т. А., Щупак Ю. А. С++. Объектно-ориентированное программирование: Практикум. - СПб.: Питер, 2006. - 265 с: ил.
- Кольцов Д.М. 100 примеров на Си. - СПб.: “Наука и техника”, 2017 - 256 с.
- 5 Доусон М. Изучаем С++ через программирование игр. - СПб.: “Питер”, 2016. - 352.
- Седжвик Р. Фундаментальные алгоритмы на С++. Анализ/Структуры данных/Сортировка/Поиск: Пер. с англ. Роберт Седжвик. - К.: Издательство “Диасофт”, 2001. - 688с.
- Сиддкхартха Р. Освой самостоятельно С++ за 21 день. - М.: SAMS, 2013. - 651 с.
- Стивен, П. Язык программирования С++. Лекции и упражнения, 6-е изд. Пер. с англ. - М.: ООО "И.Д. Вильямс", 2012. - 1248 с.
- Черносвитов, А. Visual C++: руководство по практическому изучению / А. Черносвитов . - СПб. : Питер, 2002. - 528 с. : ил.

Список литературы

- Страуструп Б. Дизайн и эволюция языка C++. - М.: ДМК, 2000. - 448 с.
- Мейерс С. Эффективное использование C++. - М.: ДМК, 2000. - 240 с.
- Бадд Т. Объектно-ориентированное программирование в действии. - СПб: Питер, 1997. - 464 с.
- Лаптев В.В. С ++. Объектно-ориентированное программирование: Учебное пособие.- СПб.: Питер, 2008. - 464 с.: ил.
- Страуструп Б. Язык программирования C++. Режим доступа: http://8361.ru/6sem/books/Straustrup-Yazyk_programmirovaniya_c.pdf.
- Керниган Б., Ритчи Д. Язык программирования Си. Режим доступа: http://cpp.com.ru/kr_cbook/index.html.
- Герберт Шилдт: C++ базовый курс. Режим доступа: https://www.bsuir.by/m/12_100229_1_98220.pdf,
- Богуславский А.А., Соколов С.М. Основы программирования на языке Си++. Режим доступа: http://www.ict.edu.ru/ft/004246/cpp_p1.pdf.
- Линский, Е. Основы C++. Режим доступа: https://www.lektorium.tv/lecture/13373.
- Конова Е. А., Поллак Г. А. Алгоритмы и программы. Язык C++: Учебное пособие. Режим доступа: https://vk.com/doc7608079_489807856?hash=e279524206b2efd567&dl=f85cf2703018eeaa2