ЛЕКЦИЯ 9.

Тема: Массивы

Вопросы

1. Объявление массива.
2. Хранение элементов массива в памяти.
3. Инициализация массивов.
4. Инициализация массивов случайными числами.
5. Ввод-вывод массивов.
6. Операции над элементами массива.

Массив – это структурированный тип данных, состоящий из элементов (компонентов) одного и того же типа, называемого ***базовым.***

Массив является структурой данных с так называемым прямым доступом, все элементы массива могут выбираться произвольно и являются одинаково доступными .

В математике массив - это последовательность элементов
A_1 A_2 A_3 ….A_n, где n>=0 - количество элементов массива.

Элементы массива линейно упорядочены в соответствии с их позицией в массиве.

Элемент a_i, *предшествует* a_{i+1}
для i = 1,2,..., n - 1
и a_i *следует за* a_{i-1} , для i = 2, 3, ..., n,
элемент a_i *имеет позицию i (номер i, индекс i).*

Массивы делятся на ***статические*** и ***динамические.***

Статические массивы

Количество элементов в статическом массиве известно при написании программы и никогда не меняется. Память под такой массив выделяет компилятор.

Описание статического массива

```
тип имя_массива
[размер1][размер2]...[размерN];
```

Количество элементов этого массива равен
`размер1*размер2*...*размерN;`

Объем памяти, необходимой для размещения массива, определяется на этапе компиляции и равен
`количество_элементов_массива*размер_типа_элементов.`

Одномерный массив:

```
тип имя_массива [размер1];
```

Двумерный массив:

```
тип имя_массива   [размер1][размер2];
```

В языке С и С++ нумерация элементов массива всегда начинается с 0. Поэтому можно сказать, что индекс элемента задает его смещение относительно начала массива (1-ый элемент имеет смещение 0, 2-ой – 1 и т.д.). Чаще всего используются одномерные, двумерные и трехмерные массивы.

В языке С и С++ нумерация элементов массива всегда начинается с 0.

Иначе можно сказать, что индекс элемента задает его смещение относительно начала массива (1-ый элемент имеет смещение 0, 2-ой – 1 и т.д.). Чаще всего используются одномерные, двумерные и трехмерные массивы.

Выход индекса за пределы массива никак не контролируется, последствия такого выхода непредсказуемы!

Хранение элементов массива в памяти

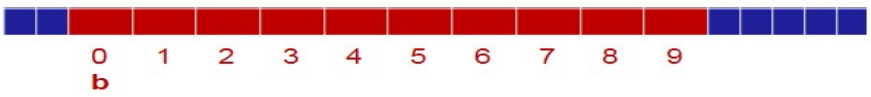
В памяти ПЭВМ массивы хранятся как сплошные последовательности компонентов, причем быстрее всего изменяется самый «дальний» (правый) индекс, если их несколько.

Адрес начала массива в памяти соответствует адресу его первого элемента (элемента с минимальными значениями индексов).

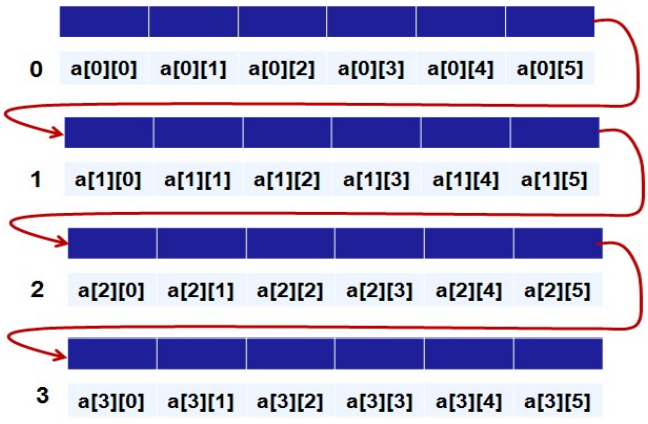
Размещение одномерных массивов в памяти

```
int b[10];
```

будут расположены в памяти следующим образом:



Размещение двумерных массивов




```
int  a[4][6];
```

Объявим массивы:

```
double Vector [100];
float Matric[10][12];
int Cub[5][5][5];
char Carray [99];
int Page[10];
char Line[81];
int sales[10][5][8];
unsighed arr[40],
long double al[1000],
char ch[80];
```

Указывая только первые p индексов, можно ссылаться на подмассивы массива

```
int sales[10][5][8];

sales[i] /* ссылка на двумерный  подмассив массива sales
*/
sales[i][j]  /* ссылка на одномерный подмассив */
sales[i][j][k]   /* ссылка на элемент  массива*/
```

Организация массивов с использованием указателей

```
int Vector [10];
```

Реализация этого описания выполнена так: **Vector** представляет собой константный указатель, содержащий адрес первого из элементов массива. Память под элементы выделяется подряд:

```
*Vector– обращение к начальному элементу(с индексом 0)
*(Vector+1) – обращение к элементу с индексом  1
*(Vector+i) – обращение к элементу с индексом i
```

Таким образом, запись **Vector[i]** – синоним записи ***(Vector+i)**

Кроме того **Vector** – это тоже самое, что **&Vector[0]**

Для доступа к элементу многомерного массива указываются все его индексы:

```
float Matric[10][12];
…
Matric[1][0]
Matric[i][j]
```

или более экзотично
***(Matric[i]+j)** или
***(*(Matric+i)+j)**

Инициализация массивов

При описании статического массива можно инициализировать сам массив. Для этого используется конструкция

```
тип имя_массива[[размерность]]=
{инициализатор[{,инициализатор}…]}
```

Для одномерного массива:

```
int a[3]={3,5,7};
int days[12]=
{31,28,31,30,31,30,31,31,30,31,30,31};
char  Zifra[]=
     {'0','1','2','3','4','\0'};
```

При наличии инициализаторов, размерность массива может быть не указана.
Для двумерного массива:

```
int matr [2][2] = {{1,0},{0,1}};
char                                                    m[3][3]=
{{'1','2','3'},{'4','5','6'},{'7','8','\0'}};
double b[2][3]={{2.0,4.6,6.2},{9.9,7.66,5.34}};
```

Правила для инициализации массивов:

- Значения элементам присваиваются по порядку. Количество элементов в списке инициализации должно соответствовать размеру массива.
- Если список меньше размера массива, то элементы массива, для которых не хватило значений, обнуляются.
- Если же список больше массива, то компилятор выдаст синтаксическую ошибку.
- Можно использовать пустые скобки при инициализации массива. Компилятор сам определит количество элементов в списке и выделит для него массив нужного размера.
- Выход массива за свои границы в С не проверяется. То есть, если массив описан как a[100], то при обращении к элементу a[200] язык С не дает

программе средств для контроля того факта, что имеется выхода за пределы памяти, отведенной под массив. Программа на языке С не выдаст сообщение об ошибке.

- При инициализации многомерных массивов есть возможность не задавать размеры всех измерений массива, кроме самого последнего.

Можно написать:

```
double temp[][3] = { { 3.2, 3.3, 3.4 }, { 4.1, 3.9, 3.9 }
};
```

Размер пропущенной размерности вычисляется так:

```
const int size_first = sizeof(temp)/sizeof(double[3]);
```

Пример 9.1.1.

```
/* дни месяца */
 int days[12]={31,28,31,30,31,30,31,31,30,31,30,31};
 int index;
 for(index = 0; index<12; index++)
   printf("Месяц %d имеет %d дней.\n",
index+1,days[index]);
```

Пример 9.1.2.

Предыдущий пример можно переписать так:

```
{int days[ ] = {31,28,31,30,31,30,31,31,30,31,30,31};
int index;
for (index=0;index<sizeof(days)/(sizeof(int));index++)
     printf("Месяц %d имеет %d дней.\n",
            index+1, days[index]);
```

Пример 9.1.3.

```
/* дни месяца */
#include <stdio.h>
int main( )
{
 setlocale(LC_ALL,".1251")
 char text1[]="Месяц";
 char text2[]="имеет";
 char text3[]="дней";
 char name [][11]= {
    {" январь "},
```

```
     {" февраль "},
     {" март "},
     {" апрель "},
     {" май "},
     {" июнь "},
     {" июль "},
     {" август "},
     {" сентябрь "},
     {" октябрь "},
     {" ноябрь "},
     {" декабрь "}   };
int days[12]={31,28,31,30,31,30,31,31,30,31,30,31};
int index;
for(index = 0; index<12; index++)
 {
   printf("%s %s %s %s %d \n",
   text1,name[index],text2,
   text3,days[index]);
  }
 }
```

Результат выполнения:

```
Месяц январь имеет дней 31
Месяц февраль имеет дней 28
Месяц март имеет дней 31
Месяц апрель имеет дней 30
Месяц май имеет дней 31
Месяц июнь имеет дней 30
Месяц июль имеет дней 31
Месяц август имеет дней 31
Месяц сентябрь имеет дней 30
Месяц октябрь имеет дней 31
Месяц ноябрь имеет дней 30
Месяц декабрь имеет дней 31
```

Инициализация массивов случайными числами

Пример 9.2.

```
#include <stdib.h>
#include <time.h>
int main()
{int  A[30],n;
 srand(time(0));
```

```
 printf(”Dimension? ”);
 scanf(”%d”,&n);
 for (int i=0; i<n; i++)
 {  A[i] = rand()%100;
    printf(”%d \n”,A[i]);
 }
 return 0;
}
```

Ввод-вывод массивов

Ввод элементов одномерного массива

Пример 9.3.1.

```
int A[20],n;
printf(" введите n? ");
scanf("%d",&n);
for (int i=0; i<n; i++)
{
     printf ("введите элемент массива");
     scanf("%d",&A[i]);
     //scanf("%d\n",&A[i]);Сравните!
}
```

Пример 9.3.2.

```
int A[20],n;
cout << "введите размер массива\n";
cin >> n;
cout << "введите элементы массива\n";
for (i=0; i<n; i++)
      cin >> A[i];
```

Вывод элементов одномерного массива

Пример 9.4.1.

```
  for ( i=0; i<n; i++)
        printf("%d \n",A[i]);//в столбец

  for ( i=0; i<n; i++)
        printf("%d ",A[i]);//в строку
```

Проверьте, что будет, если?

```
      printf("*%d ", A[i]);
      printf("%5d ", A[i]);
      printf("%x \n", A[i]);
      printf("%10.5 \n", A[i]);
      printf("%05d", A[i]);
```

Пример 9.4.2.

```
for ( i=0; i<n; i++)
    cout << A[i] << endl;
```

или

```
for ( i=0; i<n; i++)
    cout << A[i];
```

В чем разница?

Пример 9.5.

```
// вывод в 2 столбика
for ( i=0;i<n-1;i+=2)
{
  printf("%-5d%-5d\n",A[i],A[i+1]);
  //или
  //printf("%d\t%d\n",A[i],A[i+1]);
}
```

Пример 9.6.1.

```
/*  сумма элементов массива*/
int a[10];
int s = 0,i;
cout << " элементы массива\n";
for (i=0; i<10; i++)
     cin >> a[i];
for (i=0; i<10; ++i)
     s += a[i];
```

Пример 9.6.2.

```
/*  сумма элементов массива*/
const int n=10;
int a[n];
int s = 0;
cout << " элементы массива\n";
for (i=0; i<n; i++)
     cin >> a[i];
for (int i=0; i<n; i++)
    s += a[i];
```

Пример 9.6.3.

```
/*  сумма элементов массива*/
const int nmax=10;// max размер
```

```
int n;// реальный размер массива
int a[nmax];
int s = 0;
cout << "размер массива\n";
cin >> n;
cout << "элементы массива\n";
for (i=0; i<n; i++)
     cin >> a[i];
for (i=0; i<n; i++)
    s += a[i];
```

Пример 9.7. ***Вычисление значения многочлена по схеме Горнера.***

Многочлен от одной переменной x имеет вид:

$P_n(x)=a_n.x^n+a_{n-1}.x^{n-1}+...+a_1.x+a_0$.

Схема Горнера:

$y=(...((a_n.x+a_{n-1})\cdot x+a_{n-2})\cdot x+...+a_1)\cdot x+a_0$

Например: $P(x)=3\cdot x^4+5\cdot x^3-2\cdot x+6=$

$(((3\cdot x+5)\cdot x+0)\cdot x-2)\cdot x+6$

```
include <iostream>
using namespace std;
int main()
{
 //Вычисление значения многочлена
 //по схеме Горнера
 setlocale(LC_ALL,".1251");
 const int nmax=20;
 float a[nmax],x,y;
 int n,k;
 cout << "введите степень\n";
 cin >> n;
 cout<<"введите коэффициенты\n";
 for (k=n; k>=0; k--)
 {
   cout << "A[" << k << "] = ";
   cin >> a[k];
 }
 cout << "введите x\n";
 cin >> x;
 y=a[n];
```

```
 for (k=n-1; k>=0; k--)
    y=y*x+a[k];
 cout << "значение полинома= " << y
      << endl;
 return 0;
}
```

Пример 9.8. Нахождение количества повторений каждой цифры числа.

```
#include <iostream>
using namespace std;
int main()
{
 setlocale(LC_ALL,".1251");
 int a[10];
 int n,i;
 cout << "Введите число ";
 cin >> n;
 for (i=0; i<=9; i++)
    a[i]=0;

 while (n!=0)
 {
    i = n % 10;
    a[i] = a[i]+1;
    n = n / 10;
 }
 for (i=0; i<10; i++)
  if (a[i]!=0)
     cout << i << "-" <<a[i]<<endl;
  return 0;
}
```

Пример 9.9.

Поиск элемента, равного X в массиве A.

```
int Flag = -1;
int X;
…
for (int i=0; i<n; i++)
      if (A[i] == X)
      {   Flag = i;
          break;
      }
```

Пример 9.10.

```
/*найти наиболее часто встречающийся элемент в массиве*/
const int nmax=30;
int  b[nmax],n;
int max_el, max_kol, i, j, kol;
cout << "введите размер массива\n";
cin >> n;
cout << "введите элементы массива\n";
for (i=0; i<n; i++)
   cin >> b[i];
max_el=b[0];
max_kol=1;

for (i=0; i<n; i++)
{
  kol=1;
  for (j=i+1; j<n; j++)
     if (b[i]==b[j])
         kol++;
  if (kol>max_kol)
   {
     max_kol=kol;
     max_el=b[i];
   }
}
if (max_kol==1)
   cout << "в массиве нет повторяющихся\n";
else
   cout << max_el <<
     " встречается " << max_kol << " раз\n";
}
```

Операции над элементами массива

Над элементами массива можно выполнять любые операции, допустимые типом элементов (базовым типом): поиск, генерация (построение), преобразование, сортировки .

К двум совместимым статическим массивам A и B нельзя применять операция присваивания: так как это константный указатель.

A = B; //так нельзя

КОНЕЦ ЛЕКЦИИ