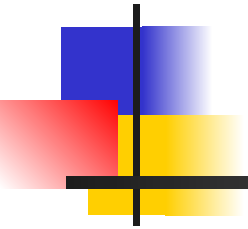
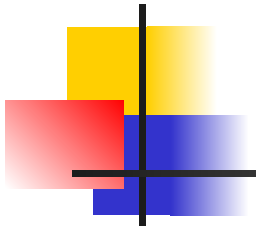


Структурный тип данных МАССИВ

Лекция №6



Одним из важных инструментов программиста является возможность работы с массивами переменных.

Массив - набор однотипных данных, хранящихся вместе и имеющих общее имя.

Возможность объединения групп элементов в массив позволяет, с одной стороны, облегчить массовую обработку данных, а с другой - упростить идентификацию элементов массива.

Основные понятия

Массив обозначается одним именем.

Например, всю совокупность действительных чисел
1.6, 14.9, -5.0, 8.5, 0.46

можно считать массивом и обозначить одним именем, например A.

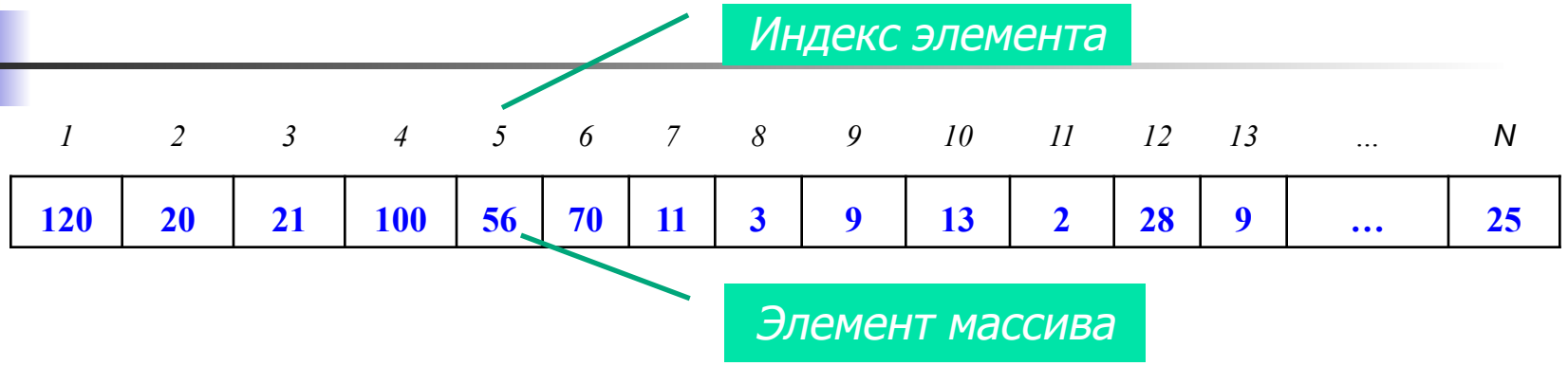
Каждый элемент массива обозначается именем массива с индексом, заключенным в квадратные скобки.

A[1], A[2], A[3], ..., A[n].

Индекс определяет положение элемента массива :

A[1]=1.6, A[2]=14.9, A[3]=-5.0, A[4]=8.5, A[5]=0.46

Доступ к элементу массива



Чтобы обратиться к какому-либо элементу массива, необходимо назвать его имя и индекс (позицию).

```
A[7]:=11;
Writeln (A[10]);
```

Двумерный массив Temp

,	1	2	3	4	5	6	7	8					31
1	-21	-22	-30	-31	-25	-20	-21	-24	…	….	…	…	-26
2													
3													
4								15					
…													
12													

Тогда для доступа к элементу этого массива необходимо указать имя массива и два индекса – номер строки и номер столбца.

```
Temp[4,8]:=15;
Temp[2,31]:=0;
```

Описание массива

Var

<перемен>:array[тип индекса] of <тип элементов>;

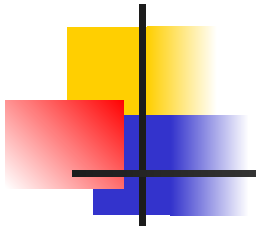
Например:

```
M: array[1..4] of integer;
MAS: array[1..3,1..5] of real;
T:array['a'..'z'] of byte;
A:array[1..1000] of string;
```

Поэтому стандартного идентификатора для типа массив нет. Программист сам конструирует тип массив:

Type

<идентиф.типа>=array[тип индекса] of <тип элемент>;



Сначала конструируется тип массив в разделе типов **Type**, где описывается структура массива. Затем нужно выделить память под переменные этого типа.

TYPE

<идентиф. типа>=**Array[**<тип индекса>**] of**<тип элементов>;

В качестве **типа индекса** может быть указан любой порядковый тип, а также диапазоны этих типов: 1..10, ‘a’…’z’.

Тип элементов массива (базовый тип) – любой допустимый в Pascal (в том числе и массив), кроме файла.

```
Type
    Mas=array[1..10] of integer;
    Mass_Char=array [byte]of char;
    Matr=array[‘A’..’C’,-5..-3]of Mas;
Var
    A:Mas; B1,B2:Mass_Char; C:Matr;
```

Двумерный массив можно описать двумя способами

Первый способ заключается в прямом указании диапазонов строк и столбцов в квадратных скобках:

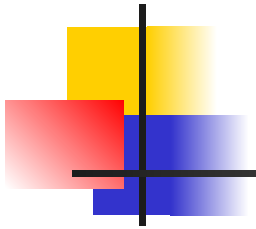
```
Type Matr=array[1..3,1..5] of byte;
Var A:Matr;
{Сначала указывается диапазон строк, затем диапазон столбцов}
```

Второй способ представляет собой описание одномерного массива, содержащего элементы типа массив.

```
Type Matr=array[1..3] of array [1..5] of byte;
```

или

```
Type Stroka=array [1..5] of byte;
     Matr=array[1..3] of Stroka;
Var A:Matr;
```



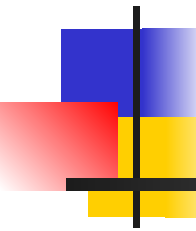
Второй способ описания двумерного массива определяется его внутренним представлением.

В памяти компьютера массив любой размерности, в том числе и двумерный, хранится в виде одномерных (линейных) массивов.

Т.е. двумерный массив представляет собой массив массивов, т.е. линейный массив элементов, которые в свою очередь также являются линейными массивами.

Поэтому массив A в памяти ЭВМ имеет следующую структуру: ((0,0,0,0,0), (0,0,0,0,0), (0,0,0,0,0)).

Длина массива составляет 3*5=15 байт.

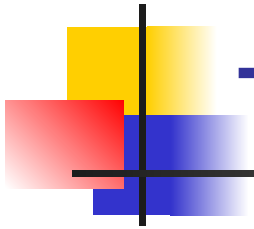


Инициализация массива

(заполнение массива элементами)

Значения элементов массива в программе можно определить тремя способами:

- Массив может быть инициализирован с помощью типизированных констант или просто присваиванием значений элементам;
- Элементы массива могут быть введены с клавиатуры или из файла;
- Элементы массива могут быть определены в программе:
  - С использованием датчика случайных чисел;
  - Заданным образом, в том числе и скопированы из другого массива.

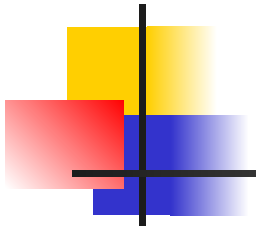


Типизированные константы

CONST

<имя константы>:<описание структуры>=(<список значений>);

```
CONST
D:array[1..10]of integer=(9,-2,0,0,-5,6,2,-13,76,9);
A:Mass=((0, -3.6, 7),( 8.3, 0.4,52.0),(-9,7.2,-13));
B:array[1..4]of char=('+', '-', '*', '/');
```



Значения элементов многомерных массивов перечисляют в порядке возрастания индексов справа налево, заключая в скобки каждый подмассив.

```
CONST
B: array [1..2, 1..6] of real=
      ((0, -3, 7, 2.3, 5.0, -9), (0,0,0,0,0,0));

S: array [1..3, 0..1, 1..4] of byte=(((0,0,0,0),(1,1,1,1)),
                                     ((2,2,2,2),(3,3,3,3)),
                                     ((4,4,4,4),(5,5,5,5)));
```

Заполнение массива с помощью генератора случайных чисел

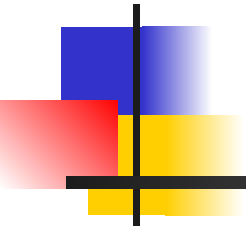
```
program Mas;
Uses CRT;
Const n=10;
Type  MyMass=array[1..n] of integer;
Var A:MyMass;     i:Integer;

Begin
randomize; {подключение генератора случайных чисел}
 for i:=1 to n do
   A[i]:=random(20);

{вывод массива А в одну строку}
 for I:=1 to n do
   write(A[i]:3);
end.
```

задания

1) Заполните двумерный массив из 5 строк и 3 столбцов случайными числами в диапазоне -50…50.
2) Заполнить массив A(20) числами Фибоначчи.
3) Сформируйте двумерный массив N*N по следующему правилу: элементы главной диагонали равны 1, элементы ниже главной диагонали – 0, а выше – сумме индексов соответствующего элемента.
4) Дан линейный массив A(15) случайных чисел. Подсчитать, сколько в нем различных чисел.
5) Заполните массив M(20) случайными числами, при этом чтобы все элементы были различны.



Основные операции над массивами

Присваивание массивов

Данную операцию можно применять только к массивам одного типа.

```
TYPE Mas=array [1..2] of real;
CONST A:Mas=(3.0, 0.12);
VAR B:Mas;
C,D: array [1..6] of char;
Begin
    B:=A;
    C:=D;
```

Данная операция не выполнима для массивов различных типов, даже, если у них одинаковая структура.

```
Var A:array [1..10]of integer;
    B:array[1..10]of integer;
Begin
   …
   A:=B;
```

Основные операции с массивами:

- Заполнение массива данными,
- Вывод элементов массива на экран или в файл,
- Поиск элемента (элементов),
- Упорядочивание элементов массива,
- Вставка/удаление элемента массива.

Все эти операции требуют последовательной обработки всех элементов массива, а поэтому требуют использования **циклов**.

Вывод элементов массива на экран монитора

- Вывод линейного массива:

```
for  i:=1 to N do
       Write (A[i]:3);
```

Формат вывода

- Вывод двумерного массива в виде таблицы:

```
for  i:=1 to N do
  begin
    for j:=1 to n do
       Write (A[i,j]:4);
    Writeln;          {Переход на новую строку}
  End;
```

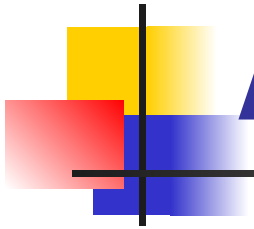
Формат вывода

Переход на новую строку

Поиск элемента массива по заданному критерию

Существует несколько методов поиска:

- **Линейный поиск** (поиск максимума, поиск нулевого элемента и т.д.),
- **Двоичный поиск**.



Алгоритм линейного поиска

Алгоритм заключается в последовательном просмотре элементов массива на определение искомого значения.

Последовательный просмотр элементов выполняется в цикле. Условием выхода из цикла будет либо обнаружение искомого элемента, либо конец массива (т.е. все элементы проверены).

Пример. Составить программу обработки массива размерностью n, заполненного целыми числами, введенными с клавиатуры. Вывести индексы и значения положительных элементов массива.

```
CONST N=10;
   VAR  A:ARRAY[0..N] of integer;
   i:byte;
   begin
   { заполнение массива }
   for i:=1 to n do begin
   write('введите ',i,' элемент массива ');
    readln(a[i]);
   end;

   { обработка элементов массива }
   for i:=1 to n do
      if a[i]>0 then writeln('положительный элемент = ',a[i],' его
   индекс = ',i);
   end.
```

Разработать программу, определяющую первый отрицательный элемент массива A(20).

```
Const n=20;
var A:array [1..n]of integer;
    i:byte;
begin
  //Заполнение массива
 //Поиск первого отрицательного элемента

 i:=0;
    repeat
        inc(i);
    until (i>n) or (a[i]<0);

 if  i>N then writeln ('NO')
       else writeln (a[i]);
end.
```

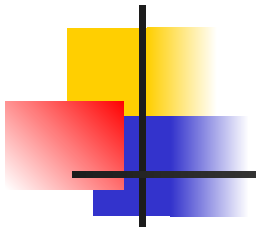
Алгоритм двоичного поиска

Допустим имеется отсортированный по возрастанию массив целых чисел. Необходимо найти значение b=23

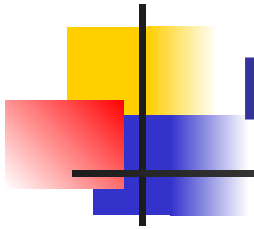
2, 5, 8, 11, 13, 14, 19, 23, 33, 45, 49

19, 23, 33, 45, 49

19,23



```
Const n=10;
Type mass=array[1..n]of integer;
Var m:mass;
    key:integer;   i,j,d:byte;  fl:boolean;
begin
//Заполнение массива упорядоченными по возрастанию числами
//Ввод ключа поиска
    Writeln('Какой элемент найти');
    Readln(key);
//поиск элемента:
    i:=1;j:=n;
    fl:=false; //элемент не найден
    repeat
      d:=(i+j)div 2; //Серединный элемент
      if m[d]=key then fl:=true
                  else
                   if m[d]<key then i:=d+1 //ищем справа
                                    else j:=d-1; //ищем слева
    until  fl or (i>j);
    if fl then
       writeln('искомый элемент ',key,'занимает позицию',d)
          else writeln('элемент не найден');
    end;
End.
```



Переформирование массива

Переформирование массивов предполагает изменение порядка элементов массива посредством их перемещения, удаления или вставки.

Вставка или удаление элементов осуществляется за счет сдвига всех элементов той части массива, которая расположена после удаляемого или до вставляемого элемента.

Задание

Дан массив A (n), где N<=10. Разработать программу удаления элемента с индексом k.

Задание

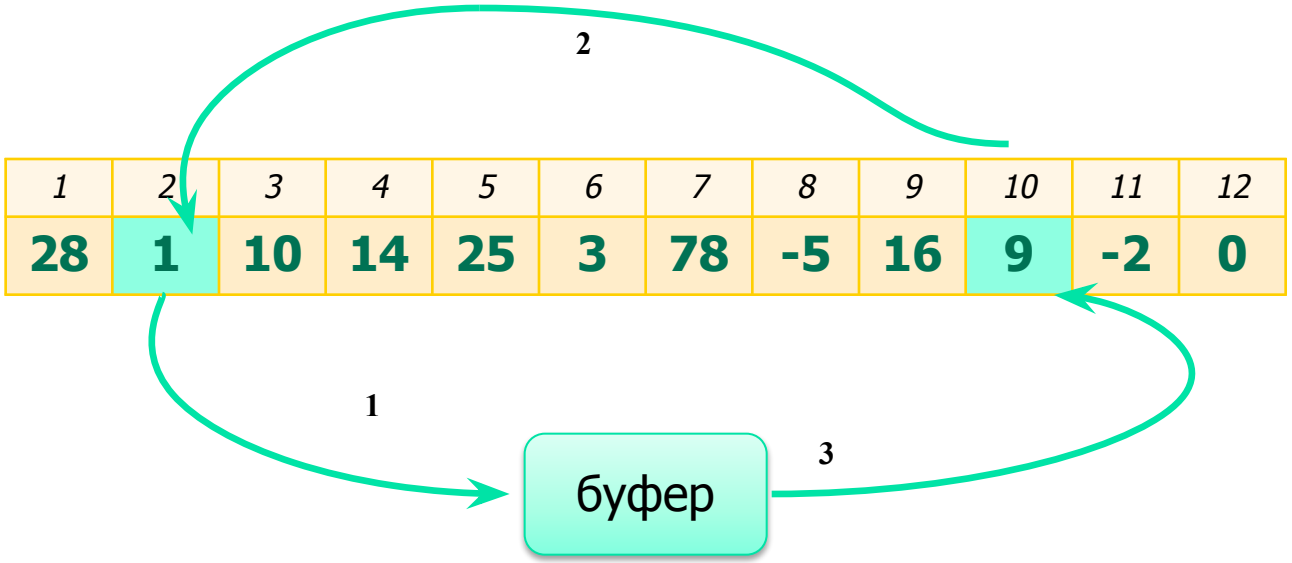
Дан массив A (n), где N<=10. Разработать программу удаления элемента с индексом k.

```
Write('Введите индекс удаляемого элемента:');
    Readln(k);
    For i:=b To N-1 Do
        A[i]:=A[i+1];
```

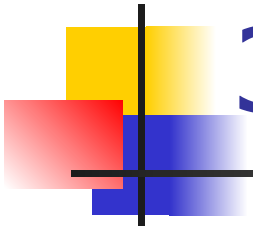
Задание

Заполнить массив A(20) случайными числами. Удалить из массива минимальный элемент массива.

Перестановка элементов массива




```
Buf:=A[2];
A[2]:=A[10];
A[10]:=buf
```

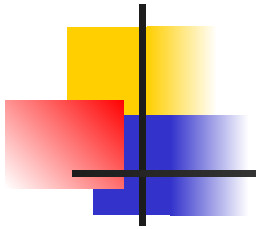


Задача

1) Напишите программу, которая переставляет первый и последний элементы массива.
2) Напишите программу, которая максимальный элемент массива ставит в конец массива.

Дана матрица A(n,n), n<=20. Выполнить перестановку строк, номера которых вводятся с клавиатуры.

```
Type Matr=array[1..20,1..20]of Integer;
Var A:Matr;  i,j,n,n1,n2:byte;
     buf:integer;
begin
 //Заполнение массива и вывод
//Перестановка строк
    Writeln('строка 1, строка 2?');
    readln(n1,n2);
    for j:=1 to n do  //цикл по столбцам
      begin
       buf:=a[n1,j];
       a[n1,j]:=a[n2,j];
       a[n2,j]:=buf;
      end;
```

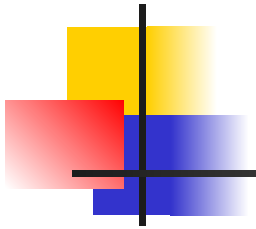


Из многомерных массивов допускается выделять подмассивы, отбрасывая индексы, записанные справа, и оставляя индексы, определяющие данный подмассив.

Например, из двумерного массива можно выделять строки, но нельзя – столбцы.

```
Writeln('строка 1, строка 2?');
  readln(n1,n2);
     buf:=a[n1];
     a[n1]:=a[n2];
     a[n2]:=buf;
```

Поэтому задачу перестановки строк можно решить одним оператором. При этом необходимо учесть совместимость массивов по присваиванию.



Сортировка массивов

Сортировка – это процесс упорядочивания информации по определенному признаку.

- *По возрастанию,*
- *По убыванию,*
- *По алфавиту*

Цель сортировки – повышение скорости **поиска** данных в больших объемах.

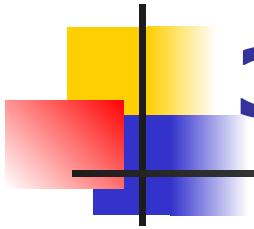
Сортировка простым обменом (Пузырьковая сортировка)

Сортировка простым обменом основана на многократном обходе массива $a_1, a_2, \ldots a_n$., при котором сравниваются два соседних элемента a_i и a_{i+1}. Если элементы неупорядочены, то они меняются местами.

7, 5, 8, 4, 1, 3, 2.

Как можно повысить эффективность алгоритма?

```
For j:=N downto 2 do
    for i:=1 to j-1 do
     if A[i]>A[i+1] then begin
                        buf:=A[i];
                        A[i]:=A[i+1];
                        A[i+1]:=buf;
                       end;
```



Задание

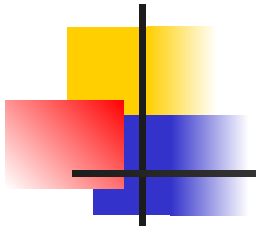
Упорядочить последовательность чисел 3, 0, 5, 1, 2 по возрастанию.

Как можно повысить эффективность алгоритма?

Как можно повысить эффективность алгоритма?

```
Repeat
     fl:=false;
     for i:=1 to j-1 do
      if A[i]>A[i+1] then begin
                          fl:=true;
                          buf:=A[i];
                          A[i]:=A[i+1];
                          A[i+1]:=buf;
                         end;
Until fl=false;
```

Сортировка простым выбором



Идея сортировки простым выбором заключается в поиске граничного элемента, например, максимального и водворении его на «свое место» в конец массива. Далее последний элемент «исключаем» (не просматриваем) и повторяется поиск максимума. Процесс повторяется до тех пор, пока длина массива не станет равна одному элементу.

Сортировка простыми вставками

В исходном состоянии считают, что сортируемая последовательность элементов состоит из двух частей: *отсортированной* (она на первом шаге состоит из единственного элемента – первого) и *неотсортированной*.

На каждом шаге из неотсортированной части последовательности берется очередной элемент и вставляется в уже отсортированную часть.

Поиск места вставки осуществляется с правого конца отсортированного подмассива путем сравнения вставляемого элемента (назовем его ключом *key*) с элементами a_j. Если ключ меньше элемента a_j (при сортировке по возрастанию), то ключ *key* и элемент a_j меняются местами, и ключ продолжает сравнение с последующим элементом подмассива a_{j-1}. Поиск места вставки завершается, если элемент вставлен или если достигнут левый конец массива.

Дана последовательность чисел:
5 2 3 8 6 1

5 ↔ 2 3 8 6 1
2 5 ↔ 3 8 6 1
2 3 5 8 6 1
2 3 5 8 ↔ 6 1
2 3 5 6 8 ↔ 1
2 3 5 6 ↔ 1 8
2 3 5 ↔ 1 6 8
2 3 ↔ 1 5 6 8
2 ↔ 1 3 5 6 8
1 2 3 5 6 8

Сортировка простыми вставками

```
for i:=2 to N do
begin
      key:=M[i];          {определяем ключ}
      j:=i-1;  {определяем правый элемент отсорт. подмассива}
      while (j>0)and(M[j]>key) do    {поиск места для вставки}
         begin
            M[j+1]:=M[j];
            dec(j);
          end;
      M[j+1]:=key;      {вставка ключа}
    end;
```