Концепция типов данных. Массивы

Наумов Д.А., доц. каф. ИТГД

Алгоритмические языки и программирование, 2019

Содержание лекции

1. Типы данных
   - Организация данных в программах
   - Механизм типов данных
   - Слабая и сильная типизация
   - Производные типы данных
   - Классификация типов данных в языке Pascal
2. Массив
   - Определение, описание, модель
   - Операция индексации
   - Типовые операции с массивом

Программирование языке низкого уровня требует точного знания:

- как данные представлены в виде последовательности битов;
- какие машинные команды должны применяться для реализации требуемых операций.

Язык высокого уровня обеспечивает следующие возможности:

- на объекты данных ссылаются с помощью определенных пользователем имен, а не конкретных адресов памяти;
- объекты данных связаны с типом, определяющим множество значений, которые могут приниматься объектами этого типа, и множество операций, которые могут применяться к объектам этого типа.

Хороший механизм типов является ключевым фактором при обеспечении надежности языка программирования, что имеет первостепенную важность при программировании.

Тип данных специфицирует:

- структуру объектов;
- область значений, которые могут принимать данные этого типа;
- множество операций, применимых к данным этого типа.

```
1  var
2    i, j, k: integer;
3    n: longint;
(*  знак 0..1 bit, мантисса 39..40 bit, эспонента 8 bit  *)
4    b: real;
5    c: char;
6  begin
7    j := i / 2;
8    n := succ(n);
9    c := chr(13);
10 end;
```

В общем случае в языке программирования должно быть *множество предопределенных типов данных* и *набор механизмов для спецификации типов*, определяемых пользователем.

Типы данных

1. простые;
   - нет внутренней структуры;
   - могут содержать лишь одно значение;
   - доступные операции предопределены;
2. структурные
   - состоят из других простых и (или) структурных типов;
   - могут содержать составные значения;
   - могут инкапсулировать поведение.

Особенности слабой типизации

- операция, которая может восприниматься машиной как корректная, может быть некорректной на абстрактном уровне программы;

```
1  var c: char;
2  c := 10;
```

- для сохранения корректности предусмотрено выполнение операции преобразования типа;

```
1  var x,y: real;
2  var i,j,k: integer;
3  i := x;
4  k := y - j;
```

- увеличение гибкости, обеспечиваемое слабой типизацией, является слишком дорогой ценой за резкое уменьшение ясности программ и необходимость дополнительного контроля во время работы компилятора.

Особенности сильной типизации

- каждый объект обладает уникальным типом;
- тип определяет множество значений и множество операций;
- тип присваиваемого значения и тип объекта данных, которому производится присваивание, должны быть эквивалентны;
- применяемая к объекту операция должна принадлежать множеству операций, определяемому типом объекта.

```
1  var x: real;
2      i: integer;
3      b: boolean;
4      c: char;
5  i := 'A' (*разные типы в левой и правой частях*)
6  x := i;
7  i := i or 10; (*недопустимая операция*)
```

На абстрактном уровне тип данных можно рассматривать как множество определенных свойств, являющихся общими для конкретного класса объектов.

```
1  var age, total_age : integer;
2    i: integer;
3  begin
4    total_age := 0;
4    for i := 1 to 10 do begin
5      readln(age);
6      total_age := total_age + age;
7    end;
8    writeln(total_age);
9  end.

(* ошибочная строка *)
     total_age := total_age + i;
```

Преимущество сильной типизации: программисту разрешается определять при описании типа свои собственные типы.

```
1  type
2    TAge = 0..200;
3    TIndex = 1..10;
4  var
5    age, total_age : TAge;
2    i: TIndex;
3  begin
4    total_age := 0;
4    for i := 1 to 10 do begin
5      readln(age);
6      total_age := total_age + age;
7    end;
8    writeln(total_age);
9  end.
```

Имеются две различные основы для вычисления эквивалентности типов данных:

- структурная эквивалентность: два объекта принадлежат эквивалентным типам, если у них одинаковая структура;
- именная эквивалентность: два объекта принадлежат эквивалентным типам, если они описаны с помощью одного и того же типа.

В языке Pascal принят принцип именной эквивалентности типов, устанавливающий, что два типа T1 и T2 эквивалентны, если выполняется одно из следующих условий:

- T1 и T2 — одно и то же имя типа;
- Тип T2 описан с использованием типа T1 равенством вида type T2=T1; или последовательностью подобного вида равенств.

```
1  type T1 = integer;
2     T3 = T1;
3     T2 = T3;
4     T4 = 1..10;
5     T5 = 1..10;
```

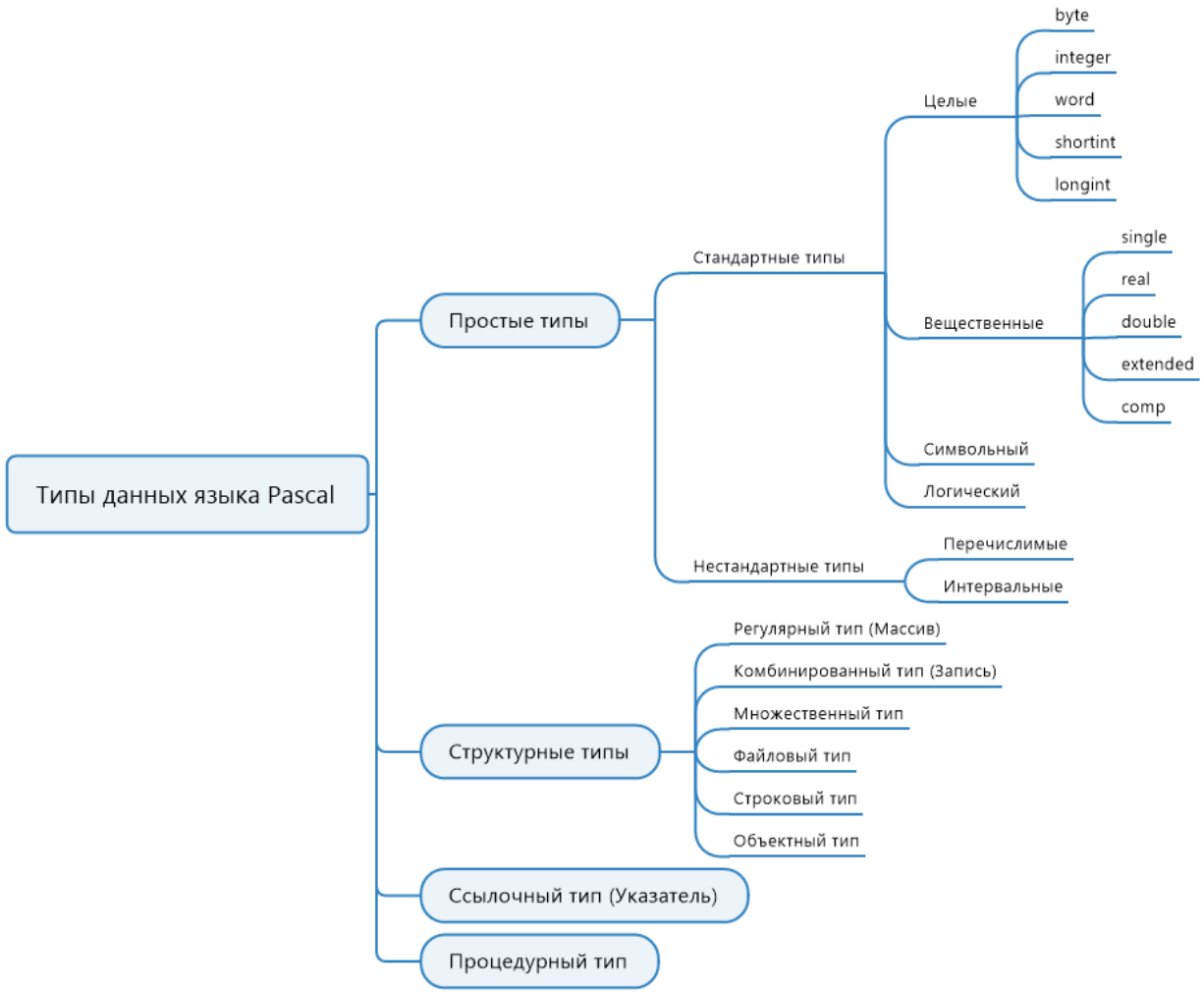


Рис.: Классификация типов языка Pascal

Перечисляемый тип данных относится к нестандартным порядковым типам и определяется набором идентификаторов, с которыми могут совпадать значения переменных этого типа.

```
1  type TDay = (MO, TU, WE, TH, FR, SA, SU);
2  var  D: TDay;
3  (* ORD (WE) = 2 *)
```

Максимальная мощность перечисляемого типа - 256 значений.

```
(* типы эквивалентны по внутреннему представлению...*)
1  type TColors = (black, red, white );
2       TOrdenal=(one, two, three);
3       TDays=(monday, thesday, wednesday);
4  var  col: TColor; num: TOrdenal; day: TDays;
(* допустимые операторы присваивания *)
5  col := black; num := two; day := monday;
(* недопустимые операторы присваивания *)
6  col := two; day := black;
```

Для перечисляемых типов определены стандартные функции PRED, SUCC и ORD, имеющие тот же смысл, что и для стандартных скалярных типов.

```
1  type TColors = (black, red, white);
2  (* SUCC(red) = white *)
3  (* PRED(red) = black *)
4  (* ORD(red) = 1 *)
```

- значения перечисляемого типа должны быть определены только идентификатором (именем);
- нельзя присваивать переменной значение из описания другого типа;
- недопустимо описание двух и более перечисляемых типов с совпадающими значениями;
- нельзя значения перечисляемого типа использовать для ввода и вывода.

```
1 type TColor1 = (red, yellow, blue);
2      TColor2 = (green, blue, gray);
```

Ограниченный тип данных относится к нестандартным порядковыми типам, образуется на основе порядковых типов, называемых базовыми, путём ограничения диапазона значений этих типов заданием минимального и максимального значений.

```
1  type TDay = (MO, TU, WE, TH, FR, SA, SU);
2       TNumber = 10..25;
3       TChars = 'c'..'x';
4       TWeekDays = SA..SU;
```

- базовым типом для создания ограниченного типа может быть любой порядковый тип;
- два символа ".."рассматриваются как один символ, поэтому между ними не допустимы пробелы;
- необходимо, чтобы левая граница диапазона не превышала его правую границу.

Массив – упорядоченный набор однотипных элементов (компонентов массива), доступ к которым осуществляется при помощи индекса.
Основные характеристики массива:

- размерность (одномерный, двухмерный и т.д.);
- тип индексов;
- тип элементов;

```
(* Описание типа одномерного массива *)
1  type ИмяТипа = array[ТипыИндексов] of ТипЭлемента;
2  var ИмяПеременной1: ИмяТипа;
(* Описание переменной-массива *)
3  var ИмяПеременной2: array[ТипыИндексов] of ТипЭлемента;
```

Доступ к элементам массива осуществляется при помощи индексов.
Тип индексов может быть любым порядковым типом:

- целым (byte, shortint, integer, word);
- символьным (char);
- логическим (boolean);
- перечисляемым;
- отрезковым.

```
1 type
2    digit = array[0..9] of char; (*одномерный массив*)
(* двухмерный массив *)
3    matrix = array[byte, 'A'..'D'] of real;
4 var
5    A, B : digit;
6    M1, M2 : matrix;
(* трехмерный массив *)
7    Cube : array[1..5, 'A'..'H', boolean] of char;
```

Массивы могут использовать для представления списка или вектора (одномерный массив), матрицы (двумерный массив), и т.д.

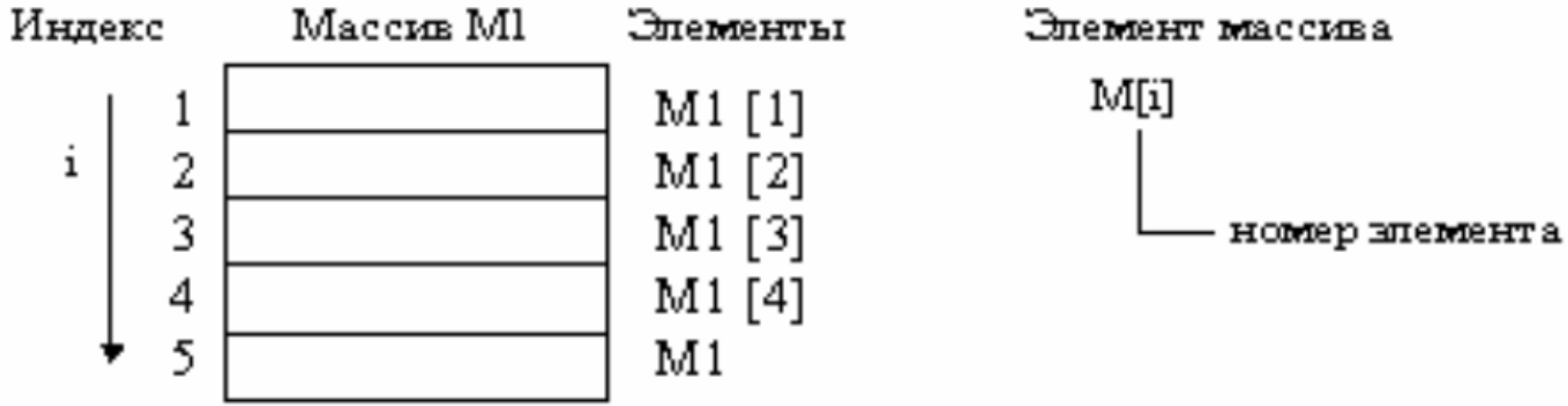


Рис.: Одномерный массив

Массивы могут использовать для представления списка или вектора (одномерный массив), матрицы (двумерный массив), и т.д.

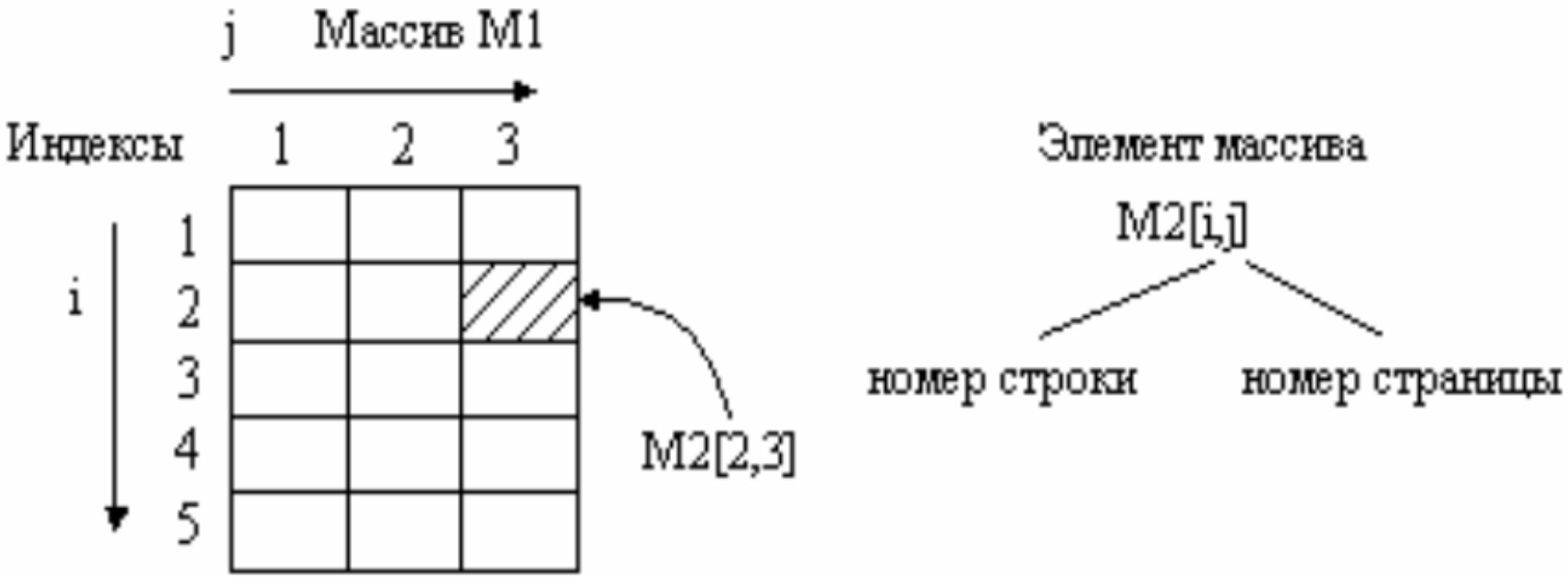


Рис.: Двумерный массив

Доступ к отдельному элементу массива осуществляется при помощи операции индексации: указанием идентификатора массива, за которым в квадратных скобках указаны выражения, тип значения и количество которых соответствуют типам индексов.

```
1 type Fam = (Ivanov, Petrov, Sidorov);
2   TMark = 2..5;
3 var
4   MarksAvg: array[Fam] of real;
5   Group741, Group748: array[1..30] of TMark;
6   i, j, k: integer;
```

К компонентам массива применимы операции и функции, допустимые для переменной базового типа.

```
1 i := 15; j := 20; k := 10;
2 MarksAvg[ Ivanov ] := 4.75;
3 Group741[i] := Group748[j - k];
(* допустимо при совпаднии типов индексов и элементов *)
4 Group748 := Group741;
```

Ввод элементов одномерного массива при помощи цикла с параметром.

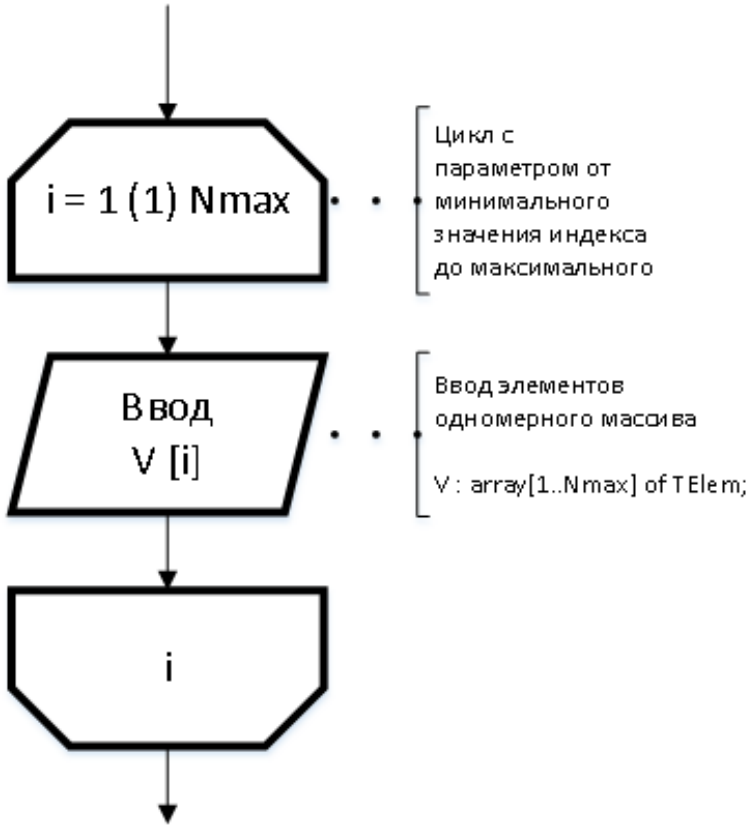


Рис.: Ввод элементов одномерного массива

Ввод элементов одномерного массива при помощи цикла с параметром.

```
const
  Nmax = 5;
var
  V: array[1..Nmax] of integer;
  i: integer;
begin
  writeln('Введите значения элементов массива V:');

  for i := 1 to Nmax do
  begin
    write('V[', i, ']=');
    readln(V[i]);
  end;
end.
```

Рис.: Ввод элементов одномерного массива

Ввод элементов двумерного массива при помощи вложенный циклов.

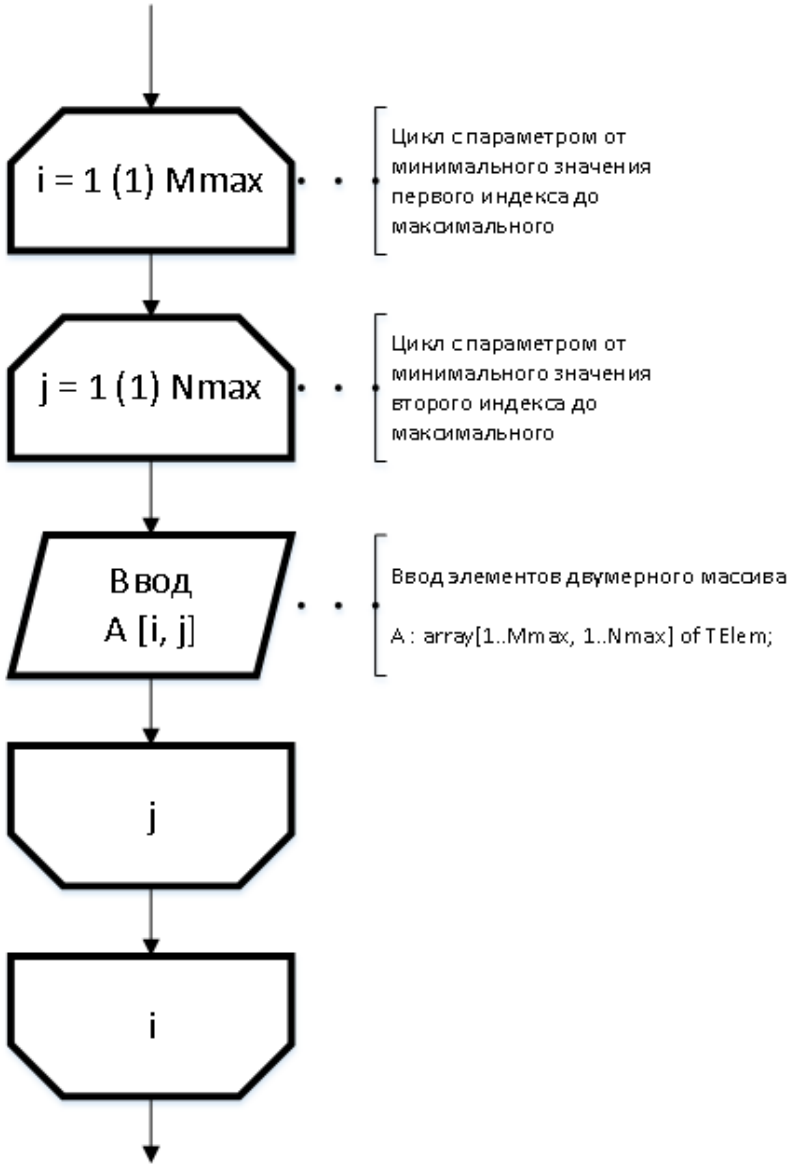


Рис.: Ввод элементов двумерного массива

Ввод элементов двумерного массива при помощи вложенных циклов.

```
const
  Mmax = 3;
  Nmax = 5;
var
  A: array[1..Mmax, 1..Nmax] of integer;
  i, j: integer;
begin
  writeln('Введите значения элементов массива A:');

  for i := 1 to Mmax do
  for j := 1 to Nmax do
    begin
      write('A[', i, ',', j, ']=');
      readln(A[i, j]);
    end;
end.
```

Рис.: Ввод элементов двумерного массива

Вывод элементов одномерного массива.

```
writeln('Значения элементов массива V:');
for i := 1 to Nmax do
  write(V[i]:7);
writeln;
```

Рис.: Вывод элементов одномерного массива

Вывод элементов двумерного массива.

```
writeln('Значения элементов массива A:');
for i := 1 to Nmax do
begin
  for j := 1 to Nmax do
    write( A[i, j]:7);
  writeln;
end;
```

Рис.: Вывод элементов двумерного массива

В одномерном массиве M, состоящем из N целых чисел, найти элементы, значения которых равны заданному числу K.

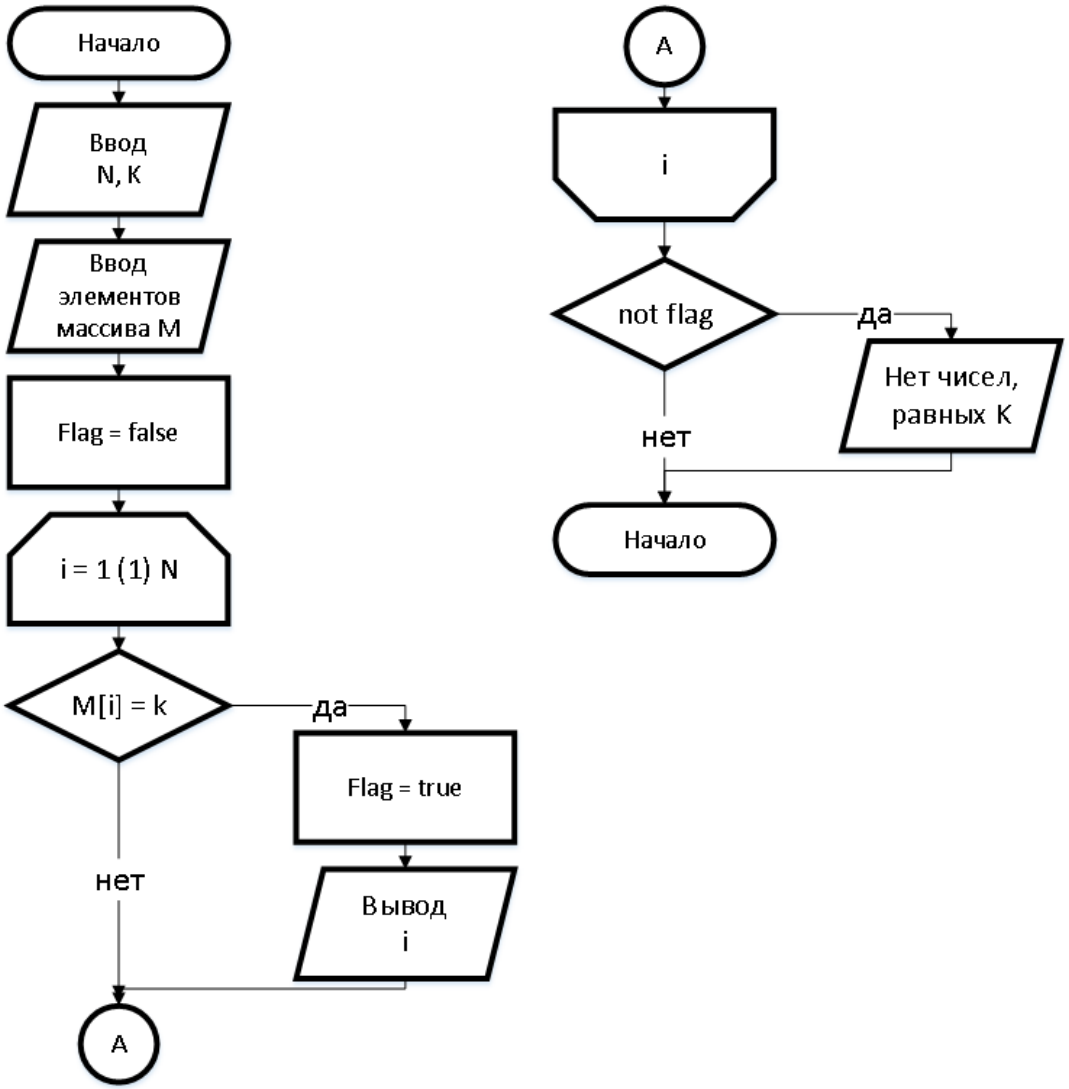


Рис.: Поиск элементов в одномерном массиве

В одномерном массиве M, состоящем из N целых чисел, найти элементы, значения которых равны заданному числу K.

```
flag:=false;

for i:=1 to N do
  if M[i] = k then
  begin
    flag:=true;
    writeln(i,'-й элемент равен ', k)
  end;

if not flag then
  writeln('Чисел, равных ', k, ' в массиве M не');
```

В одномерном массиве M, состоящем из N целых чисел, найти минимальное значение элемента.

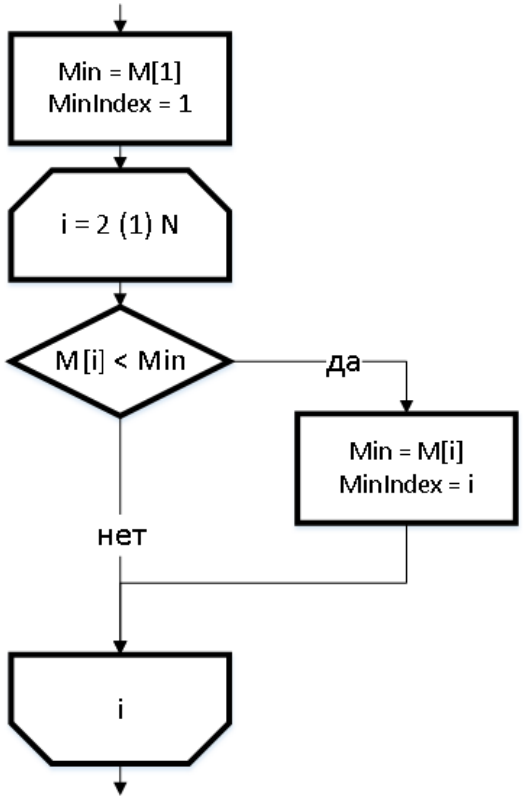


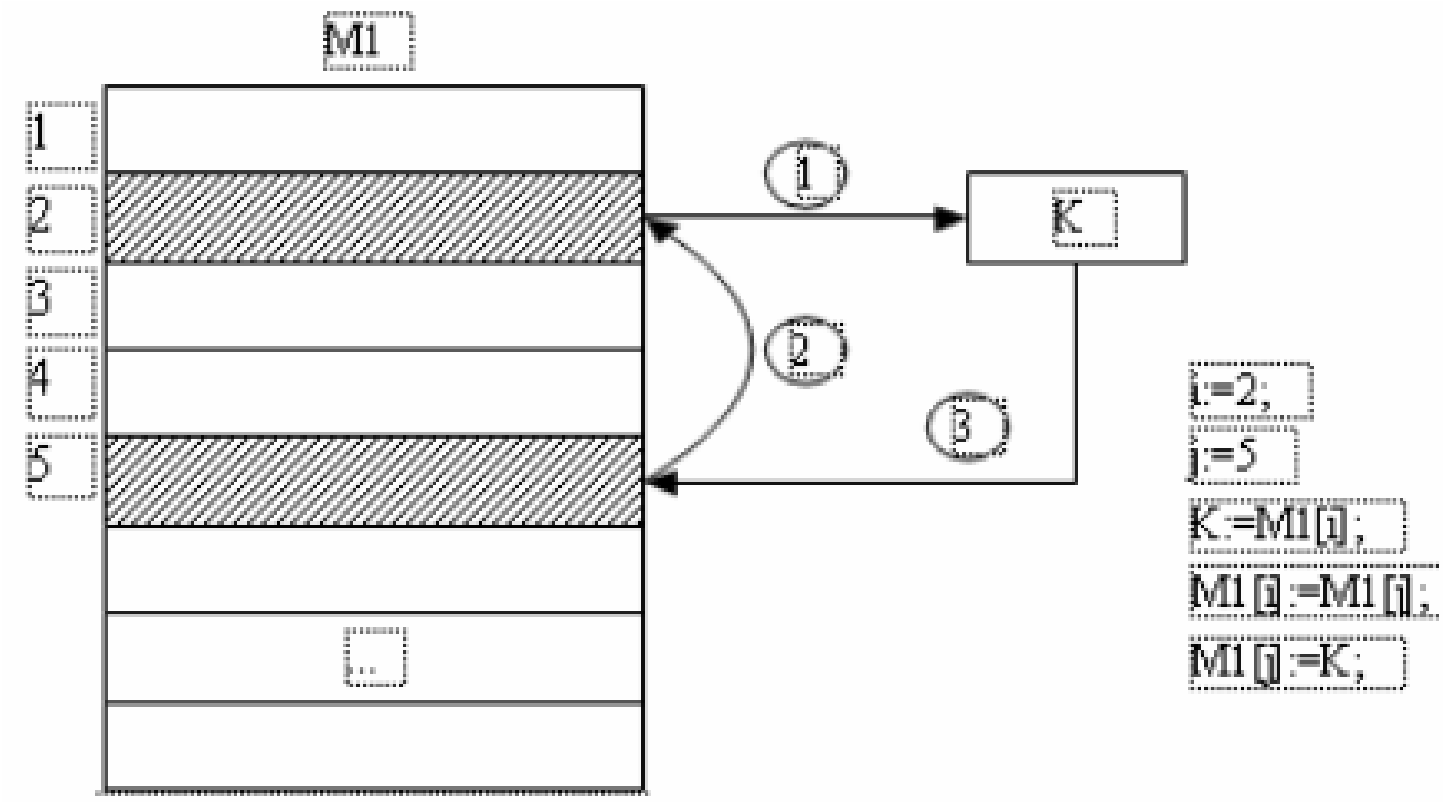
Рис.: Поиск минимального элемента в одномерном массиве

В одномерном массиве M, состоящем из N целых чисел, найти минимальное значение элемента.

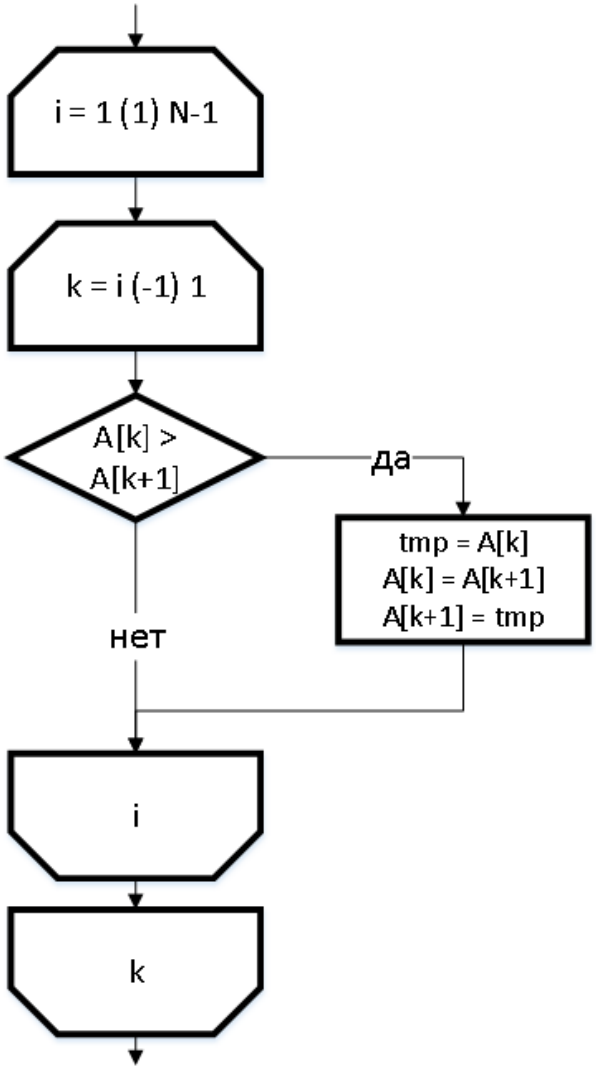
```
Min := M[1];
MinIndex := 1;

for i := 2 to N do
  if M[i] < Min then
  begin
    Min := M[i];
    MinIndex := i;
  end;
```

Для обмена значениями двух элементов массива необходимо использовать вспомогательную переменную для временного хранения одного из элементов.



Типовой задачей является сортировка элементов массива.



Фрагмент кода для сортировки методом "пузырька".

```
for i := 1 to N-1 do
  for k := i downto 1 do
    if A[k] > A[k+1] then
    begin
      tmp := A[k];
      A[k] := A[k+1];
      A[k+1] := tmp;
    end;
```

Объявление констант типа массив позволяет задать значение компонент неизменяемого массива.

```
1  Type Status = (Active, Passive, Waiting);
2    StatusMap = Array [Status] Of String[7];
3  Const StatStr : StatusMap = ('Active', 'Passive', 'Waiting');
```

Упакованные константы со строковым типом (символьные массивы) могут быть определены и как одиночные символы, и как строки.

```
4  Const Digits1 : array [0..9] Of Char
       = ('0', '1', '2', '3', '4', '5','6', '7', '8', '9');
5  Const Digits2 : array [0..9] Of Char = '0123456789';
```

Константы - многомерные массивы определяются заключением констант каждой размерности в отдельные наборы круглых скобок, разделенные запятыми.

```
6  Type Cube = array[0..1, 0..1, 0..1] Of Integer;
7  Const Array_Maze : cube =
     (((0, 1), (2, 3)), ((4, 5), (6, 7)));
```

1. Алгоритмические языки и основы программирования: Учебное пособие / В.Д.Былкин, Ю.В Блинков, Т.А. Глебова, В.В. Пикулин / Под общ ред. проф. А.Н.Кошева. - Пенза: ПГУАС, 2004. - 280 с. ISBN 5-9282-022J-0