

# СТРУКТУРЫ ДАННЫХ

## Основные понятия

**Структуры данных** – составные типы данных – это некоторым образом организованная совокупность данных, состоящая из данных простых типов или других структур данных.

Напомним, что данные – это информационные объекты, над которыми выполняются действия (операции) алгоритмов для получения требуемого результата. Тип данных определяет объём памяти для хранения данных, диапазон значений данных и допустимые операции (действия) над данными.

**Структуры данных – это сложные (составные) типы данных.**

К сложным структурам данных относят **статические, полустатические, динамические и файловые структуры данных.**

Основой для построения Сложных структур данных служат Базовые структуры данных (примитивные, простые структуры). В языках программирования простые структуры описываются простыми (базовыми) типами. К основным базовым типам данных относятся: числовые, логические, символьные.

**Числовые данные** – с помощью целых чисел может быть представлено количество объектов, являющихся дискретными по своей природе (т.е. счетное число объектов); значение вещественных чисел определяется лишь с некоторой конечной точностью, зависящей от внутримашинного формата вещественного числа и от разрядности процессора ЭВМ.

**Логические** – данные в виде одной из констант false (ложь) или true (истина).

**Символьные** – символы из некоторого predetermined множества. В большинстве современных ПЭВМ этим множеством является ASCII (American Standard Code for Information Interchange – американский стандартный код для обмена информацией).

# СТРУКТУРЫ ДАННЫХ

## Основные понятия

**Статические структуры** представляют собой структурированное множество базовых структур; они отличаются отсутствием изменчивости.

**Массивы** – структура данных, которая характеризуется:

- ❖ фиксированным набором элементов одного и того же типа;
- ❖ каждый элемент имеет уникальный набор значений индексов;
- ❖ количество индексов определяют мерность массива, например, три индекса - трехмерный массив, один индекс - одномерный массив или вектор;
- ❖ обращение к элементу массива выполняется по имени массива и значениям индексов для данного элемента.

Иначе: массив – это вектор, каждый элемент которого скаляр или вектор.

**Записи** – конечное упорядоченное множество полей, характеризующихся различным типом данных (базовыми, статическими и др.).

**Полустатические структуры данных** характеризуются следующими признаками: (1) они имеют переменную длину и простые процедуры ее изменения; (2) изменение длины структуры происходит в определенных пределах, не превышая какого-то максимального (предельного) значения. Доступ к элементу *может* осуществляться по его порядковому номеру, но с некоторыми ограничениями.

**Строки** – это линейно упорядоченная последовательность символов, принадлежащих конечному множеству символов, называемому алфавитом. Их важные свойства:

- ❖ длина, как правило, переменна, хотя алфавит фиксирован;
- ❖ обычно обращение к символам строки идет с какого-нибудь одного конца последовательности, т.е. важна упорядоченность этой последовательности, а не ее индексация; в связи с этим свойством строки часто называют также цепочками;
- ❖ чаще всего целью доступа к строке является не отдельный ее элемент (хотя это тоже не исключается), а некоторая цепочка символов в строке.

## Принципы обработки и типовые алгоритмы

**МАССИВ** – это структура данных, представляющая собой конечную совокупность элементов одного типа, которая определяется одним именем, например **Mas**.

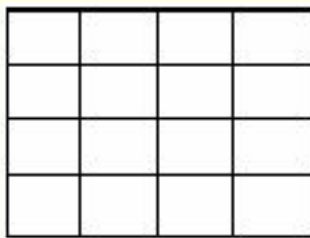
Массив – это внутренняя структура данных алгоритма, т.е. массивы используются только во время выполнения алгоритма, в котором они определены. Во внешнем представлении задачи массивы могут быть (см. рисунки):

- Линейной последовательностью значений однотипных элементов -- вектором или одномерной матрицей,
- Совокупностью двух векторов – таблицей или двумерной матрицей,
- Совокупностью трёх векторов – трёхмерной матрицей,

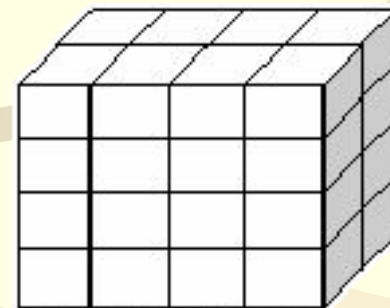
и т.д.



ОДНОМЕРНЫЙ МАССИВ



ДВУМЕРНЫЙ МАССИВ



ТРЕХМЕРНЫЙ МАССИВ

**Количество векторов в массиве называется измерением массива, количество элементов в измерении называется размерностью данного измерения.** Измерение и размерность массива данных фиксируется при создании массива и остаются неизменными при выполнении различных операций над массивами данных.

Размерность массива определяет количество элементов массива и вычисляется как произведение размерностей всех измерений массива.

Размерность измерения массива задаётся или через именованную константу, или напрямую – числом.

# МАССИВЫ

## Принципы обработки и типовые алгоритмы

**Элементы массива** располагаются в последовательных ячейках памяти и обозначаются **именем массива** и **индексом**.

Элементы массива нумеруются от 1 до N (например, в Pascal) или от 0 до N-1 (в C/C++), номер элемента массива называется *индексом*.

Для обозначения отдельных элементов массива используются переменные представленные **именем массива с индексом(-ами)**:

**M[3], FR[3,6], S[K+1].**

Пример: одномерный массив M из пяти элементов, содержащий пять нечётных чисел (ряд арифметической прогрессии с шагом 2):

**Именованние элементов массива  
через индекс**

M[1] M[2] M[3] M[4] M[5]

**Значения элементов массива**

1	3	5	7	9
---	---	---	---	---

При обращении к элементам массива необходимо контролировать выход индекса элемента за объявленные границы (размерность) массива и *избегать такой ситуации*, так это обычно приводит к ошибкам при выполнении алгоритма.

# МАССИВЫ

## Принципы обработки и типовые алгоритмы

Структура данных **Массив** позволят **выполнять операции как со всем массивом в целом, так и с отдельным элементом массива.**

С отдельными элементами массива можно работать как в **режиме последовательного доступа**, переходя от элемента с индексом  $i$  к элементу с индексом  $i+1$  (или  $i-1$ ), так и в **режиме прямого доступа**, обращаясь прямо к элементу с нужным индексом.

**Последовательная обработка массивов реализуется в цикле (ДЛЯ, ПОКА, ПОВТОРЯТЬ-ПОКА).**

Элементам массива присваиваются некоторые значения, при этом в общем случае количество элементов массива, которым заданы какие-либо значения, может быть меньше размерности массива.

## Инициализация элементов массива

Прежде чем выполнять какие либо действия с массивом, необходимо занести начальные данные в элементы массива – такая операция называется **инициализацией элементов массива**.

Начальные значения элементов могут быть заданы как константы или считаны извне (с клавиатуры и из файла).

### Инициализация массива константами

Такая инициализация обычно выполняется в том случае, когда данные массива не планируется изменять.

**Пример: Массив, хранящий названия месяцев**

Это двумерный массив символьных элементов – `month-mas`, его размерностью `12x8`, `12` – количество месяцев, `8` – максимальное количество символов в названии месяца (сентябрь).

**Инициализация  
таблицы месяцев:**

```
month-mas[1,1] = 'я'  
month-mas[1,2] = 'н'  
month-mas[1,3] = 'в'  
month-mas[1,4] = 'а'  
month-mas[1,5] = 'р'  
month-mas[1,6] = 'ь'  
month-mas[1,7] = ' '  
month-mas[1,8] = ' '
```

```
month-mas[2,1] = 'ф'  
month-mas[2,2] = 'е'  
month-mas[2,3] = 'в'  
month-mas[2,4] = 'р'  
month-mas[2,5] = 'а'  
month-mas[2,6] = 'л'  
month-mas[2,7] = 'ь'  
month-mas[2,8] = ' '
```

.....

```
.....  
month-mas[12,1] = 'д'  
month-mas[12,2] = 'е'  
month-mas[12,3] = 'к'  
month-mas[12,4] = 'а'  
month-mas[12,5] = 'б'  
month-mas[12,6] = 'р'  
month-mas[12,7] = 'ь'  
month-mas[12,8] = ' '
```

**конец**



# МАССИВЫ

## Инициализация элементов массива

### Загрузка в массив начальных значений извне

Если значения элементов массива в ходе выполнения алгоритма изменяются, то инициализировать их константами нельзя.

**Начальные значения элементов массива необходимо получить извне – ввести с клавиатуры или прочитать из файла.**

Стандартный способ ввода начальных значений элементов массива – **организовать цикл ПОКА (ПОВТОРЯТЬ-ПОКА) или ДЛЯ, в каждой итерации цикла получить очередное значение и записать его в очередной элемент массива.**

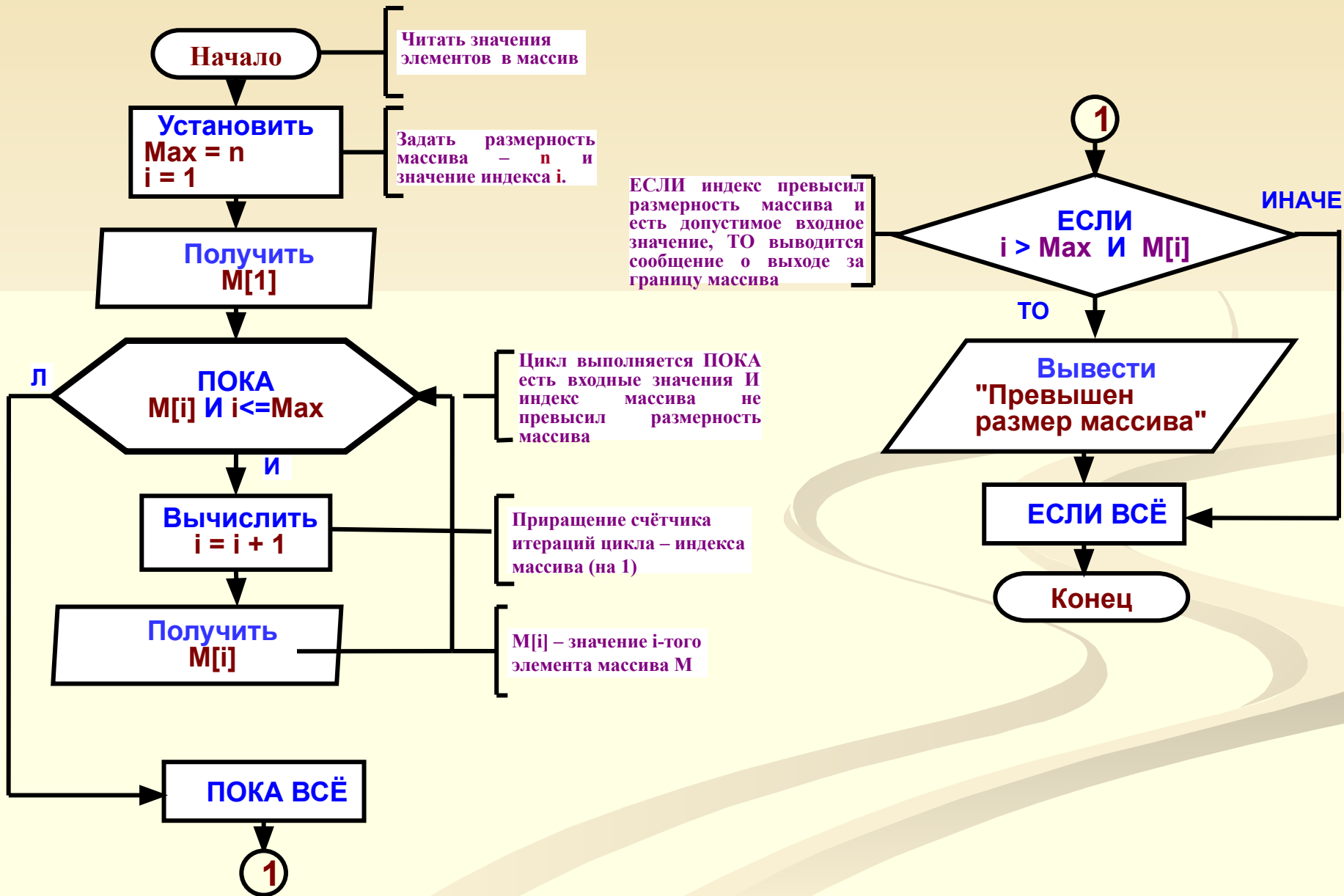
**Инициализируем одномерный массив.**

**Используем переменные:**

- **M** – имя массива,
- **i** – индекс массива,
- **Max** – размерность массива.

# МАССИВЫ

## Алгоритм заполнения массива с клавиатуры:





# МАССИВЫ

## Инициализация элементов массива

### Загрузка начальных значений в массив непостоянного размера

Если требуется массив непостоянного размера (динамический), в котором количество элементов меняется в ходе выполнения алгоритма, то задаётся массив заведомо большего размера, а для маркировки последнего элемента массива используется сигнальная метка (например, 9999).

Во время начального ввода элементов массива сигнальная метка укажет конец входных записей (завершение ввода), а во время последующей обработки массива будет указывать последний элемент массива. Если массив содержит символьные элементы, то и сигнальная метка должна быть символьная (например, точка).

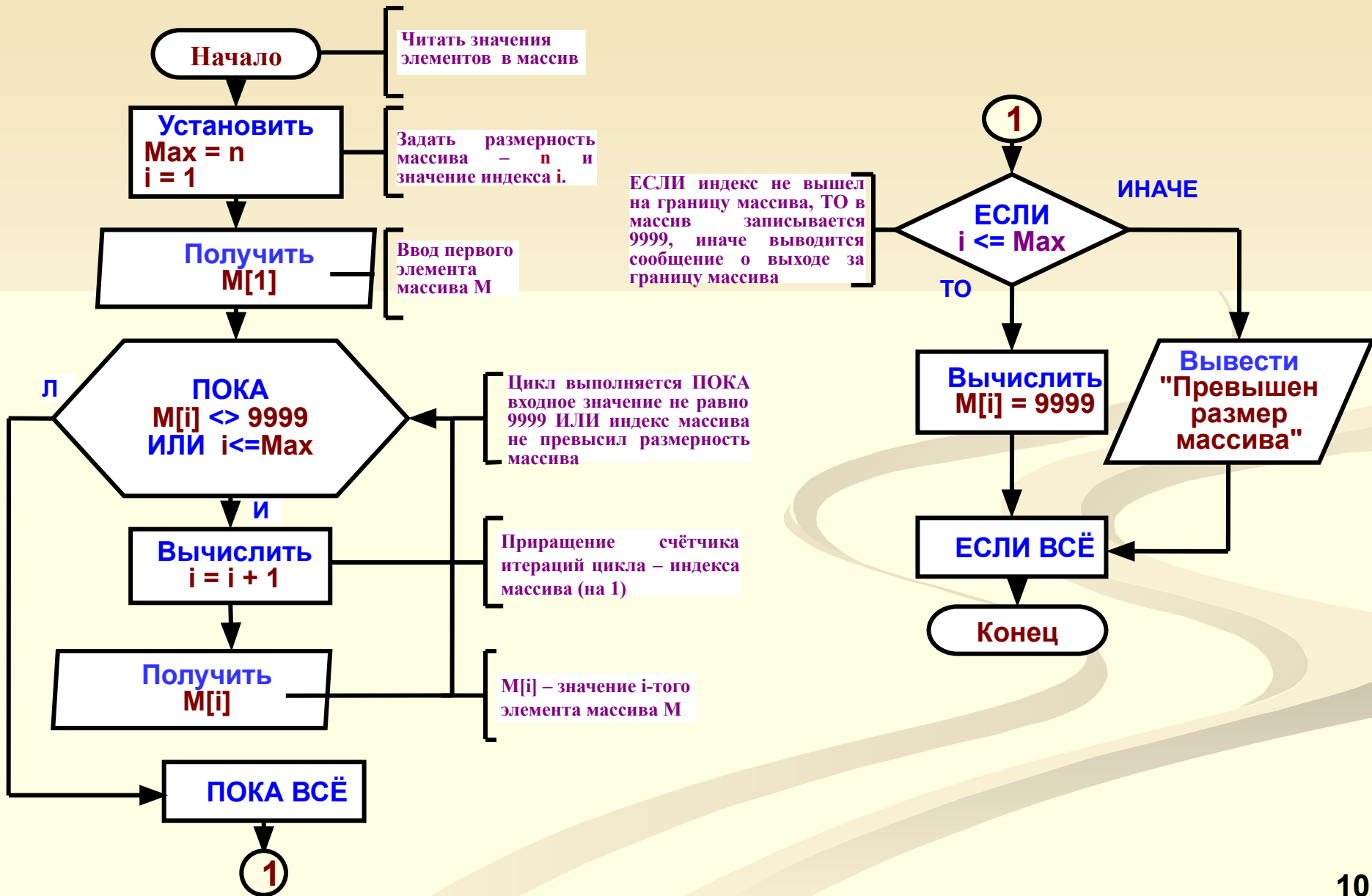
Если сигнальную метку записывать в последний элемент массива, то по ней можно определять завершение содержательной части массива.

Если надо записать в массив заранее известное количество элементов (и оно меньше размера массива), то рационально использовать цикл с параметром ДЛЯ.

При заполнении элементов массива данными из файла не требуется использовать сигнальную метку – специальное значение, обозначающее окончание вводимых данных (в приведенном алгоритме – это 9999). Работая с файлом, можно проверять специальный символ окончания файла – eof (end of file).

# МАССИВЫ

## Алгоритм заполнения массива непостоянного размера:



## Массивы

(определяемый программистом составной тип данных)

**МАССИВ** – это структура данных, представляющая собой совокупность элементов одного типа, которая определяется одним именем, например **M**. Для обозначения отдельных элементов массива используются переменные с индексом(-ами) типа **M[3]**, **FR[3,6]** – в Pascal и **FR[3][6]** – в C, **S[K+1]**.

### Объявление массива

Pascal

Type

NameM = array [F<sub>1</sub>\_Ind..L<sub>1</sub>\_Ind [,...F<sub>n</sub>\_Ind..L<sub>n</sub>\_Ind]] of TypeM;

где - NameM – имя типа массива, - TypeM – тип элементов массива,

- F<sub>x</sub>\_Ind..L<sub>x</sub>\_Ind – начальный и конечный индексы массива (интервальный тип данных).

Var ID<sub>1</sub>, ID<sub>2</sub> : NameM; где - ID<sub>1</sub> и ID<sub>2</sub> – идентификаторы объявляемых массивов.

C

TypeM NameM [E1] [[E2]...] = [{{Init\_11,Init\_12,...},{Init\_21,Init\_22,...},...}];

где - TypeM – тип элементов массива, - NameM – имя массива,

- E<sub>n</sub> – к-во элементов по измерению массива – размерность массива,

- Init<sub>n</sub> – инициализаторы (нач.значен.) каждого элемента массива. Если инициализаторов меньше, чем элементов, то начальное значение остальных элементов равно 0.

Элементы массива нумеруются от 1 до N в Pascal и от 0 до N-1 в C; номер элемента массива называется *индексом*.

Pascal

Массивы

C

## Примеры

Type

```
MS = array[1..100] of Real;
```

Var

```
t1, Z, De : MS;
```

**Можно описывать массивы прямо в разделе Var:**

Var

```
t1, Z, De : array[1..100] of Real;
```

**Описание двумерного массива:**

```
n: array [1..12, 1..14] of Integer;
```

**или**

Type

```
a = array [1..6] of integer;
```

```
b = array [1..12] of a;
```

Var

```
m : b;
```

```
int d[3] = {3,5}; // d[0]=3, d[1]=5, d[2]=0
```

```
float mask[12][5];
```

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

**или**

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

**Размерность массива разумно задавать в виде именованных констант:**

```
const int n = 12;
```

```
int mark[n] = {3,3,4,5,4,4};
```

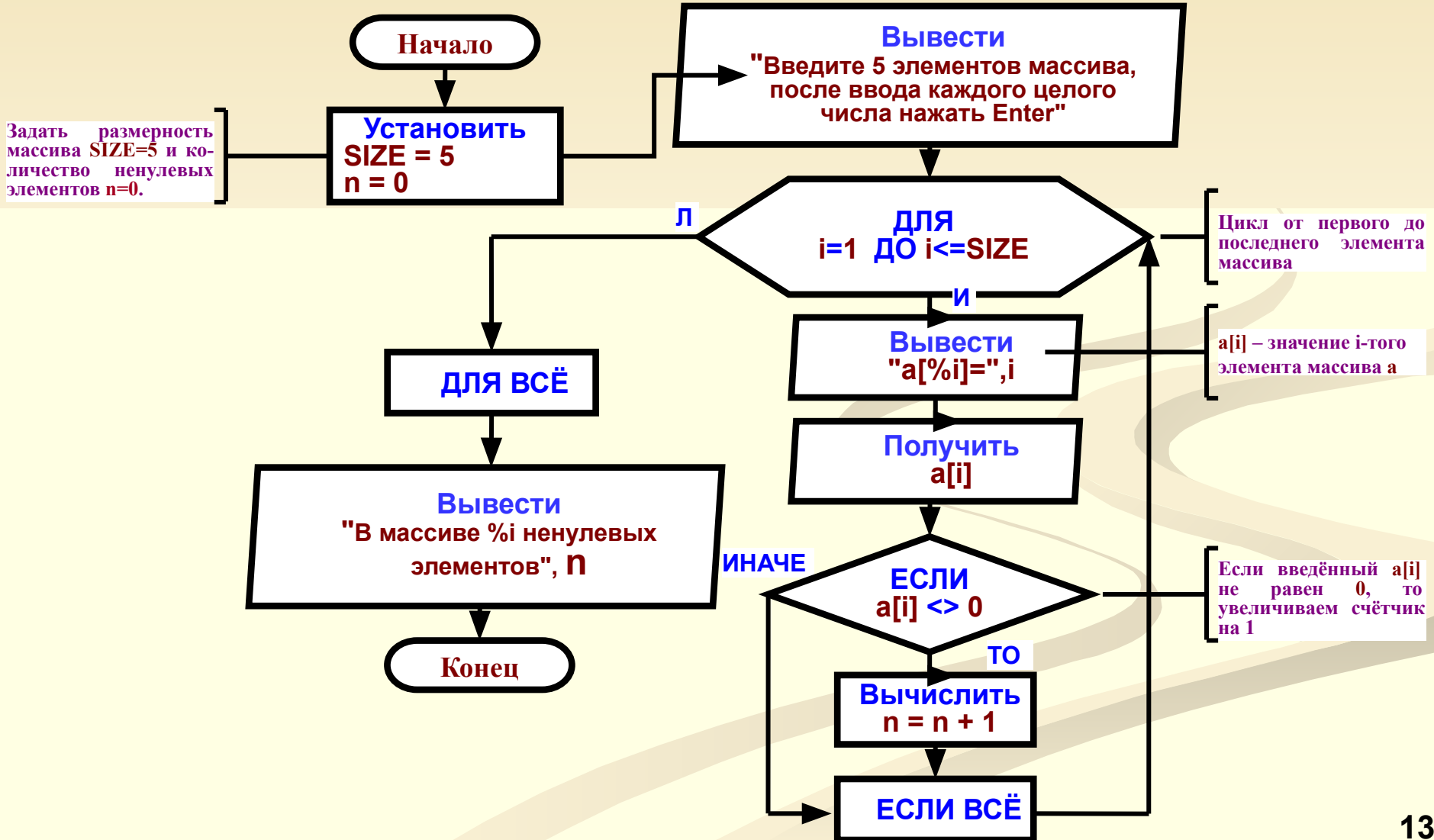
**Или через директиву препроцессора define:**

```
#define SIZE 5 // размерность массива
```

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

# МАССИВЫ

**ЗАДАНИЕ.** Нарисовать блок-схему алгоритма: ввести с клавиатуры одномерный массив из 5 элементов и вывести количество ненулевых элементов в этом массиве.



Pascal

Массивы

C

**ЗАДАНИЕ:** ввести с клавиатуры одномерный массив из 5 элементов и вывести количество ненулевых элементов в этом массиве.

```
Program MASSIV;
  Const SIZE = 5; (* Размер массива*)
  Var  a:array[1..Size] of integer;
       n: integer; i: integer;
Begin
  WriteLn('Введите 5 элементов массива');
  Write('После ввода числа');
  WriteLn('нажимайте Enter');
  n:=0;
  For i:=1 To SIZE do
  Begin
  Write('a['i,'] -> ');
  ReadLn(a[i]);
  If a[i] <> 0 Then n:=n+1;
  End;
  WriteLn('В массиве',n,' ненулевых элементов. ');
  ReadLn;
End.
```

```
#include<stdio.h>
#define SIZE 5 // размер массива
void main ()
{
  int a[SIZE]; // массив
  int n = 0, i; //ненулевые элементы, индекс
  printf("\nВведите элементы массива\n");
  Printf ("После ввода числа – Enter\n");
  for (i=0; i<SIZE; i++)
  {
    printf("a[%i] ->", i);
    scanf("%i",&a[i]);
    if (a[i] != 0) n++;
  }
  printf("В массиве %i ненулевых элемента
  \n", n);
}
```

# МАССИВЫ

## Двумерные массивы

Когда для решения задач не хватает вектора – одномерного массива, тогда возникает необходимость увеличить количество измерений массива.

Рассмотрим некоторые особенности работы с двумерными массивами.

**Двухмерный массив** – это фактически массив одномерных массивов.

Объявление двумерного массива **d**, состоящего из 10 строк и 20 столбцов:

`int d[10][20];` – в ЯП C/C++ и `d: array [1..10, 1..20] of Integer;` – в ЯП Pascal .

**Осторожно!** В Pascal размерности векторов массива отделяются запятыми, в C/C++ размерность каждого вектора массива заключена в квадратные скобки. Двумерный массив хранится в виде матрицы, в которой **первый индекс задает номер строки**, а **второй – номер столбца**. Принято, что при обходе элементов в порядке их размещения в памяти правый (второй) индекс изменяется быстрее, чем левый.

Гр

Правый (второй) индекс нумерует столбцы

Левый  
(первый)  
индекс  
нумерует  
строки

	Mas [0] [0]	Mas [0] [1]	Mas [0] [2]
	Mas [1] [0]	Mas [1] [1]	Mas [1] [2]
	Mas [2] [0]	Mas [2] [1]	Mas [2] [2]
	Mas [3] [0]	Mas [3] [1]	Mas [3] [2]





## Массивы

C / C++

### Двумерные массивы. Загрузка содержимого массива с клавиатуры

**// Заполнение двумерного массива**

```
#include<stdio.h>
```

```
// Количество строк матрицы
```

```
#define SIZE_i 5
```

```
// Количество столбцов матрицы
```

```
#define SIZE_j 5
```

```
void main ()
```

```
{  
int a[SIZE_i][SIZE_j]; // Двумерный массив
```

```
int i, j; // индексы массива
```

```
printf("\nВведите элементы массива\n");
```

*см. продолжение*

*Продолжение*  
**// Ввод элементов массива с клавиатуры**

```
printf("После ввода числа - Enter\n");
```

```
for (i=0; i<SIZE_i; i++)
```

```
for (j=0; j<SIZE_j; j++)
```

```
{  
printf ("a[%i][%i] = ",i,j);
```

```
scanf ("%i",&a[i][j]);
```

```
}
```

```
}
```

Pascal

Массивы

## Двумерные массивы. Загрузка содержимого массива с клавиатуры

```
Program Matrica;  
Const  
  N=2; (*Количество строк матрицы *)  
  M=2; (*Количество столбцов матрицы*)  
  K=2; (* Размерность массива *)
```

```
Type  
  (* Тип данных – двумерный массив *)  
  Matr=array [1..N,1..M] of integer;
```

```
Var  
  (* массив двумерных матриц*)  
  x : array [1..K]of Matr;  
  i,j,p : integer; (* индексы массивов *)
```

*см. продолжение*

*продолжение*

```
Begin  
  (* Загрузка данных *)  
  writeln ( 'Ввод данных в  
            массив двумерных матриц ');  
  for i:=1 to K do  
    for j:=1 to N do  
      for p:=1 to M do  
        write ('X[' , i , ',' , j , ',' , p ] = ');  
        readln (x[i,j,p]);  
  
  readln;  
  end.
```

Pascal

Операторы

C

**Задание на дом:**

**решить задачи обработки массивов данных:  
2-а индивидуальные задания.**

**Нарисовать блок-схемы алгоритмов решения и написать программы на Pascal и C.**