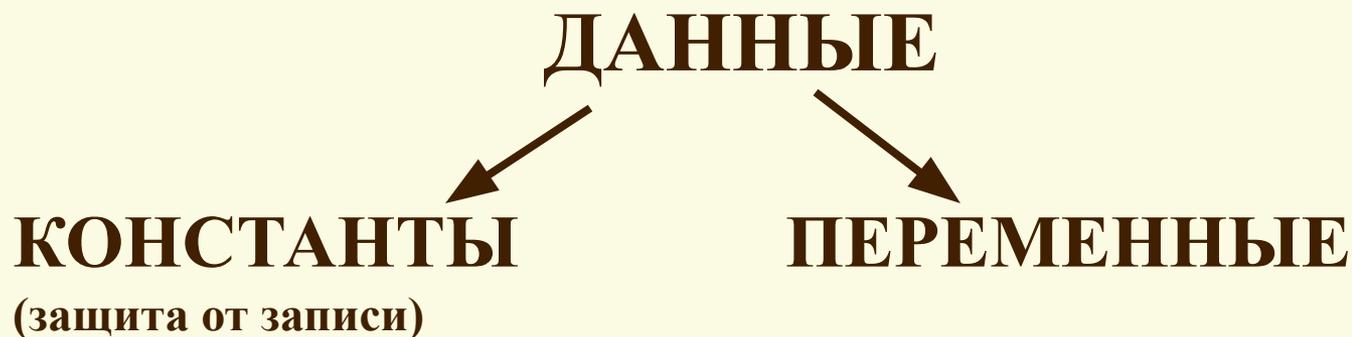


# МАССИВЫ

- **Определение**
- **Описание**
- **Обращение к элементам массива**
- **Связь массивов с указателями**
- **Примеры программ**

# КЛАССИФИКАЦИЯ ДАННЫХ ПО СТРУКТУРЕ



# ОПРЕДЕЛЕНИЕ

---

**Массив** - это сложное данное, состоящее из конечного числа упорядоченных компонент, имеющих одно имя, одинаковый тип и расположенных в последовательных ячейках памяти компьютера.

**Упорядоченность** компонент массива: компоненты пронумерованы.

**Доступ к элементу массива** - по его номерам (индексам).

**Размерность массива** - количество индексов у его элементов.

**Размер** - количество значений каждого индекса.

# МАССИВЫ В ПРОГРАММЕ

ОПИСАНИЕ

СИ

*размеры - только  
константы*

тип имя[размер\_1]...[размер\_N]

ОБРАЩЕНИЕ К  
ЭЛЕМЕНТУ  
МАССИВА

СИ

имя[индекс\_1]...[индекс\_N]

индекс\_i - целое выражение, индекс\_i = 0,1,...,N-1

*В Си элементы массивов нумеруются, начиная с нуля.*

# МАССИВЫ В СИ-ПРОГРАММЕ

---

Примеры. `float a[20];`

`a[0], a[1], ..., a[19].`

`int b[3][5];`

`b[0][0] b[0][1] ... b[0][4]`  
`b[1][0] b[1][1] ... b[1][4]`  
`b[2][0] b[2][1] ... b[2][4]`

*Первый индекс - номер строки, второй - столбца*

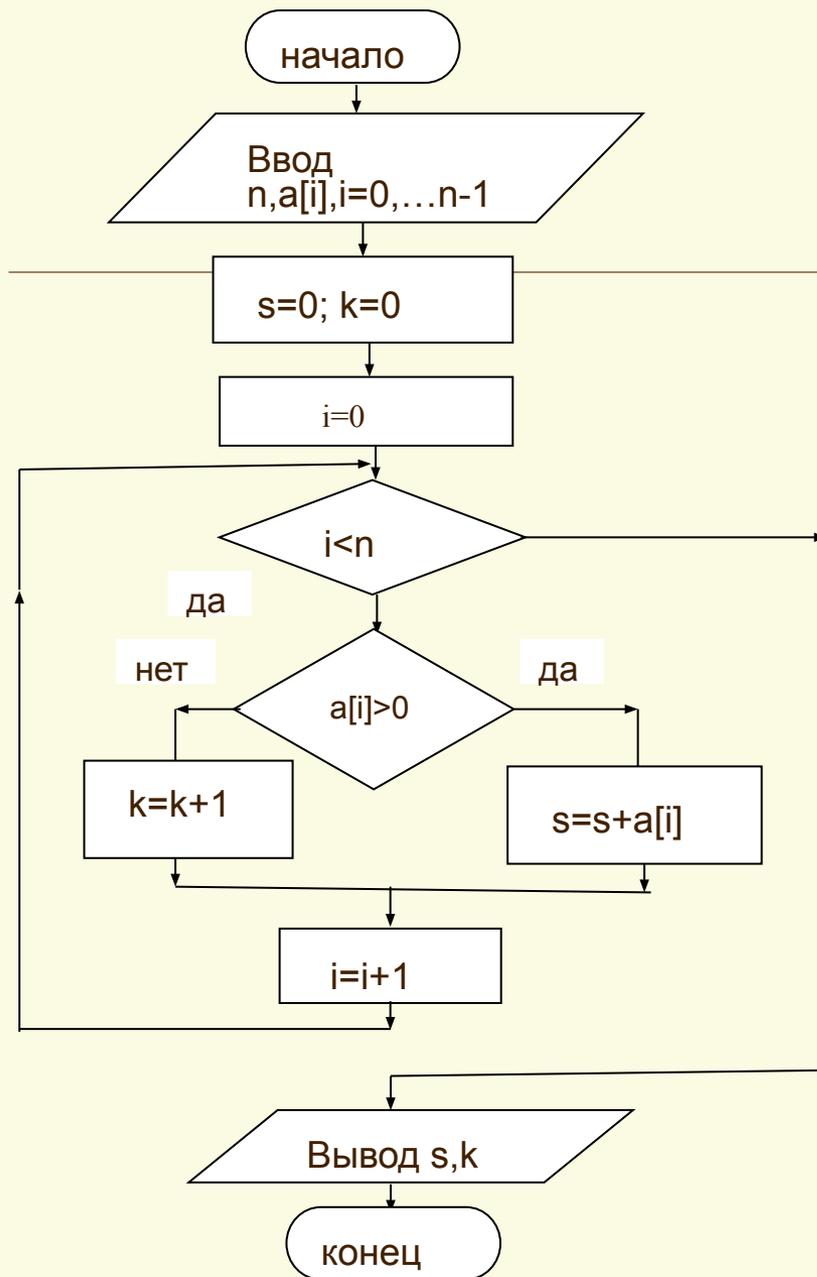
*В памяти компьютера элементы массива расположены по строкам (чаще меняется последний индекс)*

# Примеры программ с массивами

Дан массив **a** из **n** элементов,  $n \leq 20$ . Вычислить сумму положительных и количество неположительных элементов массива.

## Состав данных

Имя	Смысл	Тип	Структура
<i>Исходные данные</i>			
n	число элементов массива	целый	простая переменная
a	заданный массив	вещественный	одномерный массив из 20 элементов
<i>Выходные данные</i>			
s	сумма положительных элементов массива	вещественный	простая переменная
k	количество неположительных элементов	целый	простая переменная
<i>Промежуточные данные</i>			
i	счетчик элементов массива	целый	простая переменная



Блок-схема алгоритма

```

#include <iostream.h>
void main()
{float a[20],s; int k,i,n;
cout<<"Vvedite n\n";
cin>>n;
cout<<"Vvedite massiv
iz"<<n<<"elementov\n";
/* Далее цикл для поэлементного ввода
массива*/
for (i=0; i<n; i++)
cin>>a[i];
/*Далее алгоритм по блок-схеме*/
s=0; k=0;
for (i=0; i<n; i++)
if (a[i]>0)
s=s+a[i];
else
k=k+1;
cout<<" s= "<<s;
cout<<" "<<"k="<<"\n";
}
  
```

Программа

# Инициализация массивов при описании в Си

Инициализация - задание начальных значений.

## Одномерные массивы

```
char a[6]={'A', 'B', 'C', 'D'};
```

0	1	2	3	4	5
A	B	C	D	н/о	н/о

*если a - локальная переменная*

```
char a[ ]={'A', 'B', 'C', 'D'};
```

0	1	2	3
A	B	C	D

*Размер массива определяется количеством инициализирующих значений*

# Локальные и глобальные данные

---

**ДАННЫЕ**

```
graph TD; A[ДАННЫЕ] --> B[ЛОКАЛЬНЫЕ: описаны в функции (в том числе в main); по умолчанию не инициализируются.]; A --> C[ГЛОБАЛЬНЫЕ: описаны вне функций; при описании обнуляются.];
```

**ЛОКАЛЬНЫЕ:**  
описаны в функции  
(в том числе в  
main); по  
умолчанию не  
инициализируются.

**ГЛОБАЛЬНЫЕ:**  
описаны вне  
функций; при  
описании  
обнуляются.

# Инициализация массивов при описании в Си

## Двумерные массивы

*Присваивание перечисленных значений происходит по строкам (в соответствии с расположением массивов в памяти компьютера).*

```
int m[2][3]={0,1,2,5,6,7}; int m[ ][3]={0,1,2,5,6,7};
```

0	1	2
5	6	7

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

0	н/о	н/о
1	2	н/о

# Инициализация массивов при описании в Си

---

**Вывод:** при объявлении массива количество его элементов должно быть задано или явным указанием константы в квадратных скобках или количеством значений при инициализации.

*Исключение: массивы-аргументы функций.*

*Снятие ограничения: динамические массивы*

# Указатели в Си

**Указатель** - это специальное данные, которая содержит адрес другого данного.

**Основные операции для работы с указателями:**

\* - взятие содержимого по адресу (\*i - содержимое переменной с адресом i)

& - взятие адреса (&a - адрес переменной a).

**Описание имеет вид:**

*тип \*имя\_указателя;*

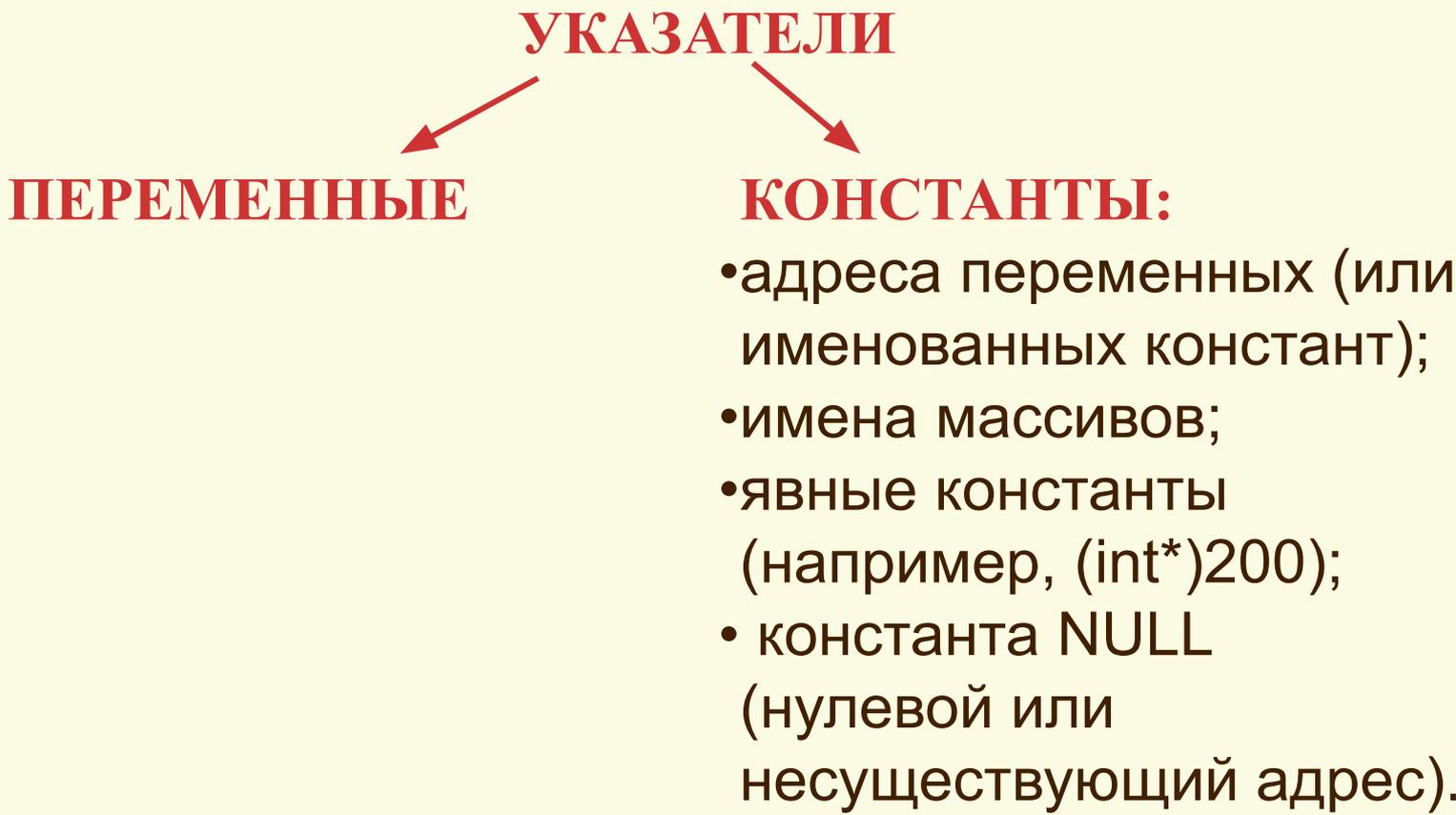
*При описании указателя задается тип значения, на которое он указывает.*

**Примеры описаний:** int \*i, j, \*pointj;  
int v1, \*pointv1=&v1, \*p=(int\*)200;

# Указатели в Си

---

## УКАЗАТЕЛИ



```
graph TD; A[УКАЗАТЕЛИ] --> B[ПЕРЕМЕННЫЕ]; A --> C[КОНСТАНТЫ:];
```

### ПЕРЕМЕННЫЕ

### КОНСТАНТЫ:

- адреса переменных (или именованных констант);
- имена массивов;
- явные константы (например, (int\*)200);
- константа NULL (нулевой или несуществующий адрес).

# Указатели в Си

---

## **ВНИМАНИЕ!**

- нельзя брать содержимое от константы без приведения типа; запись `*200` является некорректной в отличие от `*(int*)200`;
- нельзя брать адрес явной константы (например, некорректна запись `&200`), в Си адрес явной константы считается недоступным;
- нельзя определять адрес выражения.

# Указатели в Си

---

**Размер памяти**, отводимой под указатель, зависит:

- от разрядности адресной шины;
- от модели памяти.

# Указатели в Си

---

## Операции над указателями:

\*

**сравнения (<, <=, >, >=, ==, !=)** - с указателями такого же типа или с NULL;

**присваивания** - значений указателей того же типа или NULL;

**арифметические операции сложения, вычитания (с константой)**

**инкремента и декремента**

# Указатели в Си

Результат арифметической операции над указателями зависит не только от значения операндов, но и от типа, с которым связан указатель.

$p = p + k$ ,  $\Leftrightarrow$   $p$  увеличивается на  $k * \text{sizeof}(\text{тип})$

Пример. `int *p; long int *pp; ... //MS DOS`  
`p++; /*p увеличилось на 2*/`  
`pp++; /*pp увеличилось на 4*/`

# Связь массивов с указателями в Си

## Одномерные массивы

Имя одномерного массива является *указателем-константой*, равной адресу начала массива, т. е. адресу элемента с индексом 0 (первого элемента).

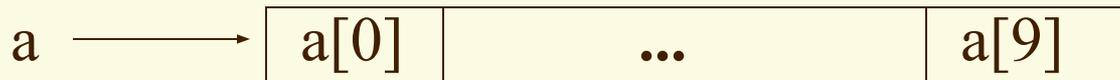
```
int a[10];
```

**&a[0]** эквивалентно **a**,

**a[0]** эквивалентно **\*a**,

**&a[i]** эквивалентно **a+i** ( $i=0,1,\dots,9$ ),

**a[i]** эквивалентно **\*(a+i)**.

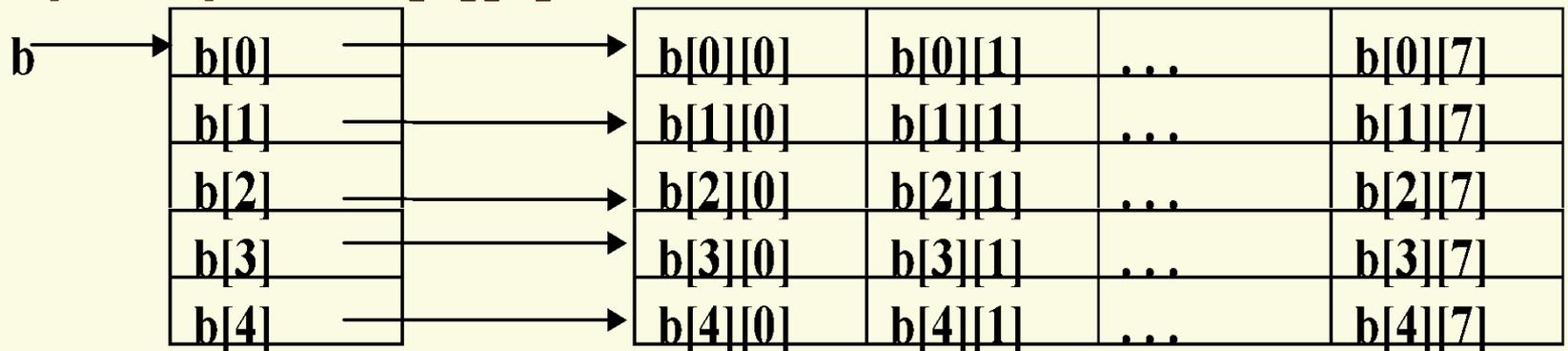


# Связь массивов с указателями в Си

## Двумерные массивы

Имя двумерного массива является *указателем-константой на начало* (элемент с индексом 0) массива *указателей-констант*, *i*-й элемент этого массива - указатель -константа на начало (элемент с индексом 0) *i*-й строки двумерного массива.

Пример: `int b[5][8];`



# Связь массивов с указателями в Си

---

## Двумерные массивы

$b[i][j] \Leftrightarrow *(b[i]+j) \Leftrightarrow (*(b+i)+j);$

$\&b[i][j] \Leftrightarrow b[i]+j \Leftrightarrow *(b+i)+j$

*Для любого из трех обозначений элемента двумерного массива программа в кодах получается практически одинаковой по производительности, хотя при использовании арифметики указателей вместо квадратных скобок несколько более короткой.*

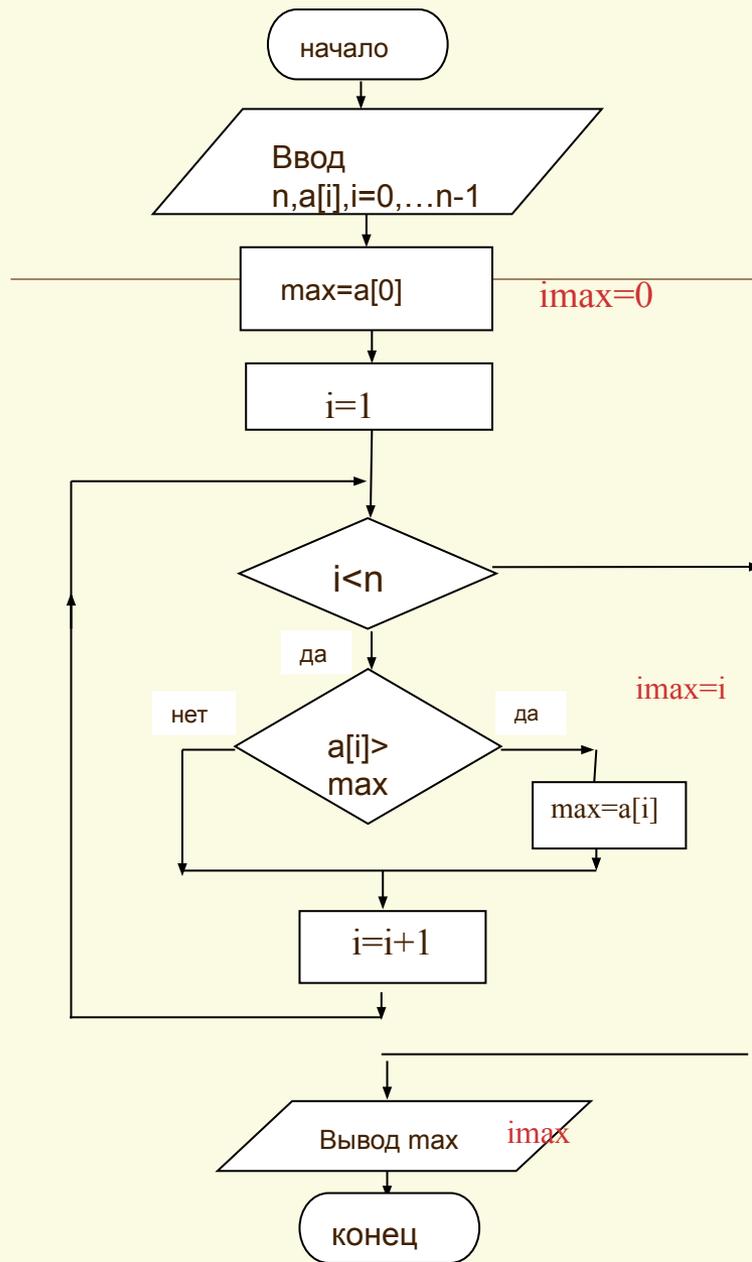
*Хороший стиль программирования предполагает употребление в пределах одной программы одного (из трех) обозначений.*

# Примеры программ с массивами

Дан массив  $a$  из  $n$  элементов,  $n \leq 20$ . Найти максимальное значение элементов массива.

## Состав данных

Имя	Смысл	Тип	Структура
<i>Исходные данные</i>			
$n$	число элементов массива	целый	простая переменная
$a$	заданный массив	вещественный	одномерный массив из 20 элементов
<i>Выходные данные</i>			
$\max$	Максимальное значение элементов массива	вещественный	простая переменная
<i>Промежуточные данные</i>			
$i$	счетчик элементов массива	целый	простая переменная



Блок-схема алгоритма

```
#include <iostream.h>
void main()
```

```
{float a[20],max; int i,n;
cout<<"Vvedite n\n";
```

```
cin>>n;
```

```
cout<<"Vvedite massiv iz"<<n<<"elementov\n";
/* Далее цикл для поэлементного ввода
массива*/
```

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

```
cin>>a[i];
```

```
/*Далее алгоритм по блок-схеме*/
```

```
max=a[0]; imax=0;
```

```
for (i=1; i<n; i++)
```

```
if (a[i]>max)
```

```
{max=a[i]; imax=i;
```

```
}
```

```
cout<<" max= "<<max<<"\n";
```

```
cout<<" imax= "<<imax<<"\n";
```

```
}
```

Программа