

2023

Глава 3 Массивы, строки и структуры. Адресная арифметика

МГТУ им. Н.Э. Баумана

Факультет Информатика и системы
управления

Кафедра Компьютерные системы и сети

Лектор: д.т.н., проф.

Иванова Галина Сергеевна

3.1 Массивы

Массив – это упорядоченная совокупность *однотипных данных*.

Каждому элементу массива соответствует один или несколько *индексов порядкового типа*, определяющих положение элемента в массиве.

a					c				b							
-5	0	12	54	-8		0	1	2	A	N	D		O	R	...	T
0	1	2	3	4	0	-5	0	13	0	1	2	3	4	5	...	255
					1	46	83	-8								
					2	54	0	93								

\$ Массив =

Тип **Имя** **[Размер]** **{[Размер]}** **[[=]** **{Список_значений>}**;

где Тип – тип элемента, который может быть любым кроме файла, в том числе массивом, строкой и т.п.

Размер – натуральное значение, которое определяет количество элементов по очередному измерению. Количество измерений не ограничено, но массив в памяти не может занимать более 2 Гб.

Размерность массива - количество его измерений.

Тип индекса – порядковый – определяет доступ к элементу.

Примеры объявления массивов

- a) `int a[5];` // объявление массива
- b) `signed char c[3][3];` // объявление матрицы
- c) `typedef signed char byte;` // объявление типа элемента
`byte c[3][3];` // объявление матрицы
- d) `unsigned char b[256];` // объявление массива

Инициализация массива при объявлении

```
int a[5] = {0, -3, 7, 3, 5};
```

```
float d[3][5] = {{0.0, -3.6, 7.8, 3.789, 5.0},  
                {6.1, 0, -4.56, 8.9, 3.0},  
                {-0.35, -6.3, 1.4, -2.8, 1.9}};
```

Статические и автоматические массивы

Статические массивы:

- под внешний массив, объявленный вне подпрограмм (extern), или статический, описанный static, память выделяется *во время компиляции* программы:

```
float a[10][10];  
int main() {  
    int n,m;  
    cout << "Enter n,m:";  
    cin >> n >> m; // ввод размерности матрицы
```

Внешний массив.
Размерность указывается
константами по максимуму!

Автоматические массивы:

- под массив, локально объявленный внутри подпрограммы, память выделяется в стеке *во время выполнения* программы.

Примечание. Допускается в среде Qt Creator (компилятор Clang) указывать размер локального массива переменными:

```
int main() {  
    int n,m;  
    cout << "Enter n,m:";  
    cin >> n >> m;  
    float a[n][m];
```

Локальный массив.
Размерность указана
переменными!

Операции над массивами

А. Доступ к элементу массива:

Пример:

```
int c[3][4];  
...  
c[2][0] = 5; // прямой доступ
```

c	0	1	2	3
0	-5	0	13	11
1	46	83	-8	-2
2	54	0	93	93

```
i = 2; j = 0;  
c[i][j] = 5; // косвенный доступ: значения индексов  
              // находятся в переменных
```

Косвенный доступ к элементам массива

a	0	1	2	3	4	5
	3.5	-5.1	0	8.4	-0.3	4.9

a[2]

Задано значение индекса
задано константой -
прямой доступ

a	0	1	2	3	4	5
	3.5	-5.1	0	8.4	-0.3	4.9

a[i]

Значение индекса
хранится в переменной -
косвенный доступ

i
2

Косвенный доступ позволяет реализовать
последовательную обработку элементов массивов:

```
for (i=0; i<6; i++) a[i] = i*i;
```

ИЛИ

```
for (i=5; i>=0; i--) a[i] = i*i;
```

Б. Ввод/вывод массивов

Ввод-вывод массивов осуществляется поэлементно:

Пример 1. Ввод элементов одномерного массива

```
int a[5]; // массив на 5 целых чисел
```






```
for (i=0; i<5; i++) scanf ("%d", &a[i]);
```

ИЛИ

```
for (i=0; i<5; i++) cin >> a[i];
```

Значения вводятся через пробел, Tab() или Enter():

а) 2 -6 8 56 34 

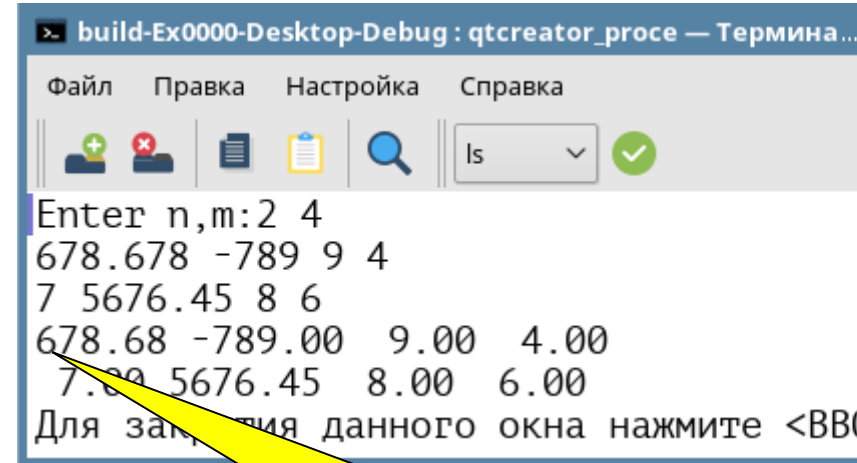
б) 2 
-6  8 
56 
34 

При вводе чисел пробелы, символы табуляции и Enter служат только разделителями и игнорируются до следующего числа

Пример 2. Ввод/вывод матрицы (функции)

```
#include <stdio.h> Ex03_01
```

```
int main() {  
    double a[10][10];  
    int n,m,i,j;  
    printf("Enter n,m:");  
    scanf("%d %d", &n, &m);  
    for(i=0;i<n;i++)  
        for(j=0;j<m;j++)  
            scanf("%lf", &a[i][j]);  
    for (int i=0;i<n;i++){  
        for (int j=0;j<m;j++)  
            printf("%5.2lf ", a[i][j]);  
        printf("\n"); // переход на следующую строку  
    }  
    return 0;  
}
```

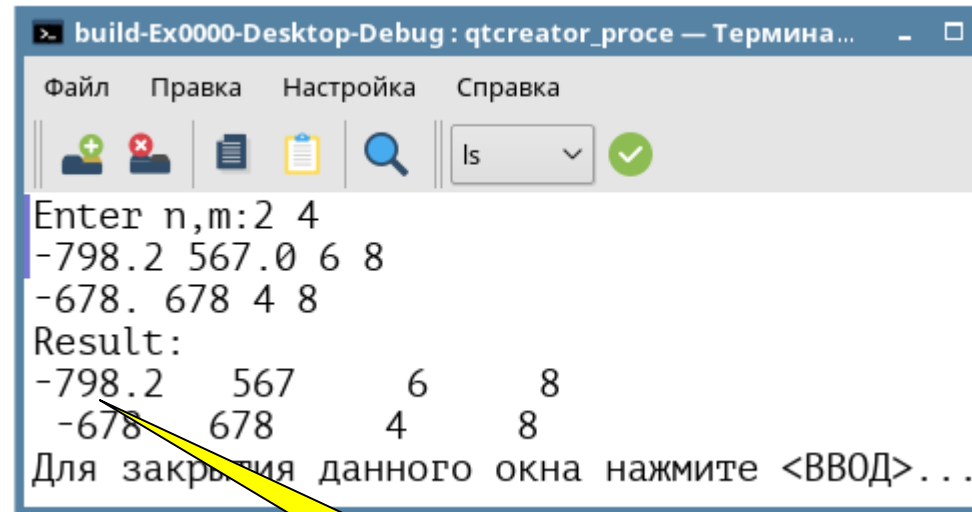


```
build-Ex0000-Desktop-Debug : qtcreeator_proce — Термина...  
Файл  Провка  Настройка  Справка  
Enter n,m:2 4  
678.678 -789 9 4  
7 5676.45 8 6  
678.68 -789.00 9.00 4.00  
7.00 5676.45 8.00 6.00  
Для завершения данного окна нажмите <BB>
```

Не уместилось!
Расширено
автоматически!

Пример Ex03.02. Ввод/вывод матрицы (потоки)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    float a[10][10];
    int n,m,i,j;
    cout << "Enter n,m:";
    cin >> n >> m;
    for(i=0;i<n;i++)
        for(j=0;j<m;j++)
            cin >> a[i][j];
    cout << "Result:" << endl;
    for (int i=0;i<n;i++){
        for (int j=0;j<m;j++)
            cout << setw(5)<<a[i][j]<<' ';
        cout << "\n"; // переход на следующую строку
    }
    return 0;
}
```



build-Ex0000-Desktop-Debug : qtcreator_proce — Терминал

Файл Правка Настройка Справка

Enter n,m:2 4
-798.2 567.0 6 8
-678. 678 4 8
Result:
-798.2 567 6 8
-678 678 4 8
Для закрытия данного окна нажмите <ВВОД>...

Не уместилось!
Расширено
автоматически!

Максимальный элемент массива и его номер

Ex03_03

A

45	34	56	2	-3
----	----	----	---	----

АМАХ

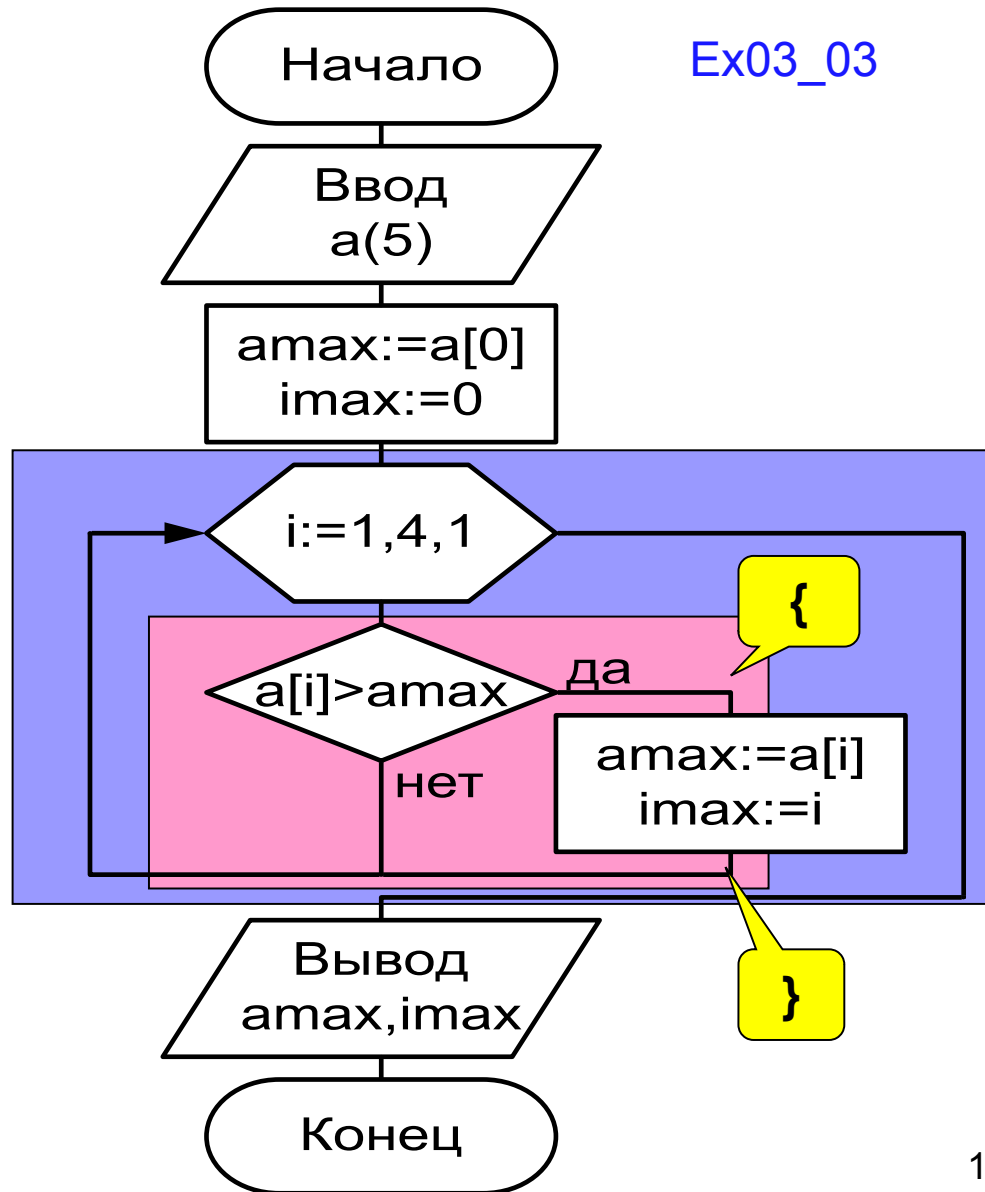
56

ІМАХ

2

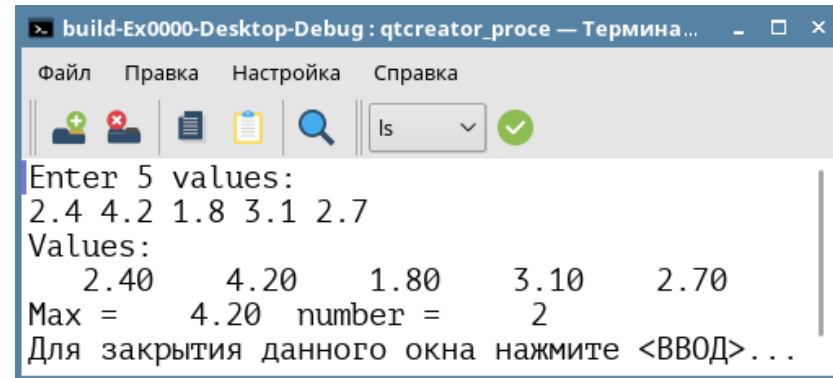
i

4



Программа поиска максимального элемента

```
#include <stdio.h>
int main()
{
    float a[5], amax; int imax;
    puts("Enter 5 values:");
    for(int i=0;i<5;i++)
        scanf("%f",&a[i]);
    amax=a[0];
    imax=0;
    for(int i=1;i<5;i++)
        if(a[i]>amax)
            { amax=a[i]; imax=i;}
    puts("Values:");
    for(int i=0;i<5;i++)
        printf("%7.2f ",a[i]);
    printf("\n");
    printf("Max = %7.2f   number = %5d\n",amax,imax+1);
    return 0;
}
```



Пример 2(Ex03_04) Поисковый цикл. Неструктурная и структурная реализации

Задача. Дан массив размером $n \approx 10000$ чисел и число. Определить номер первого элемента массива, равного заданному числу.

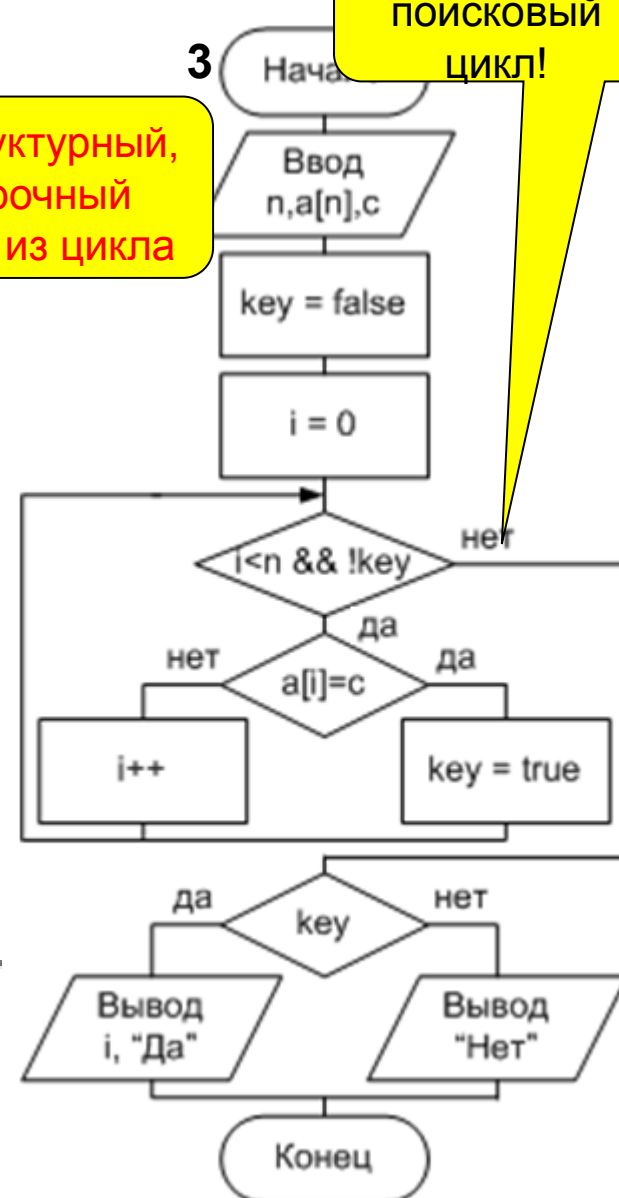
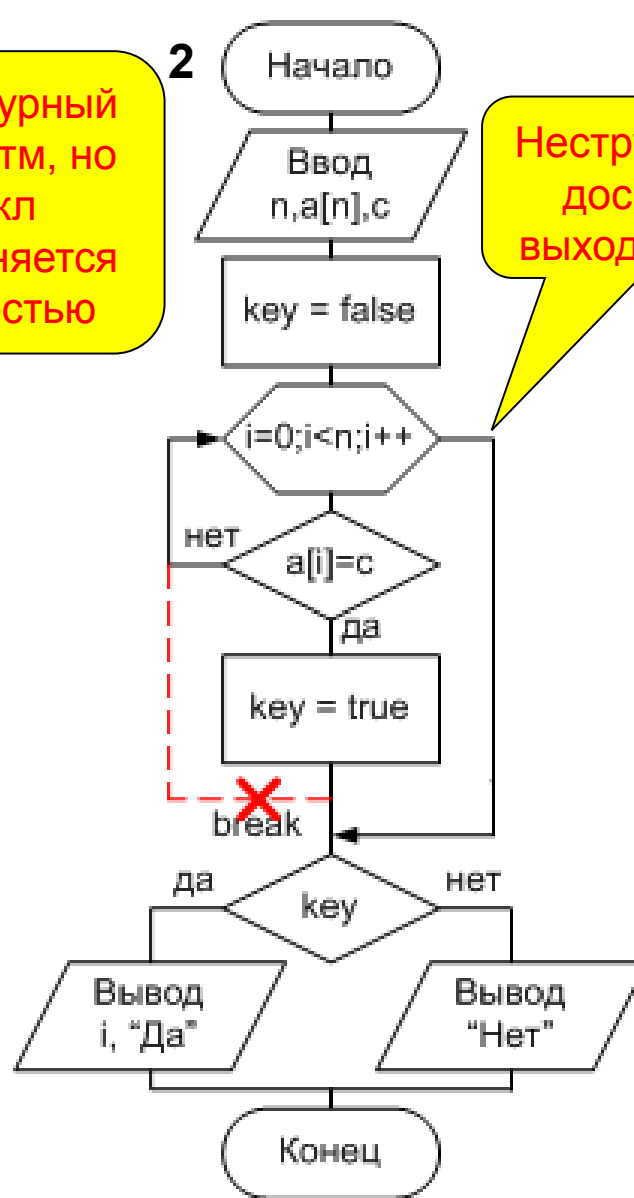
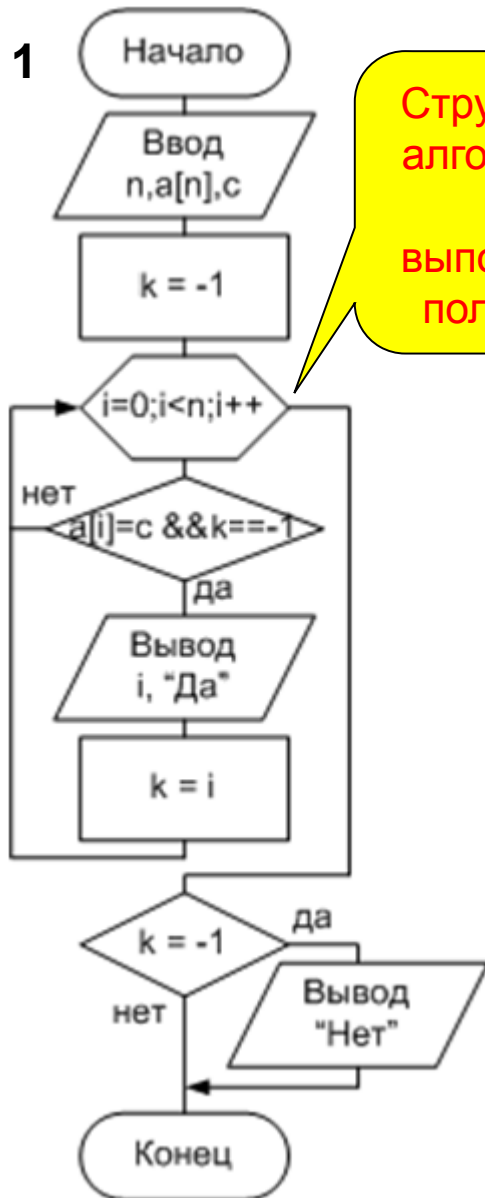
Рассмотрим три алгоритма решения задачи, которые дают одинаковые результаты.

Алгоритмы называют **эквивалентными**, если они дают **одинаковые результаты** при любых исходных данных.

Однако у первого – большая вычислительная сложность, а второй – неструктурен.

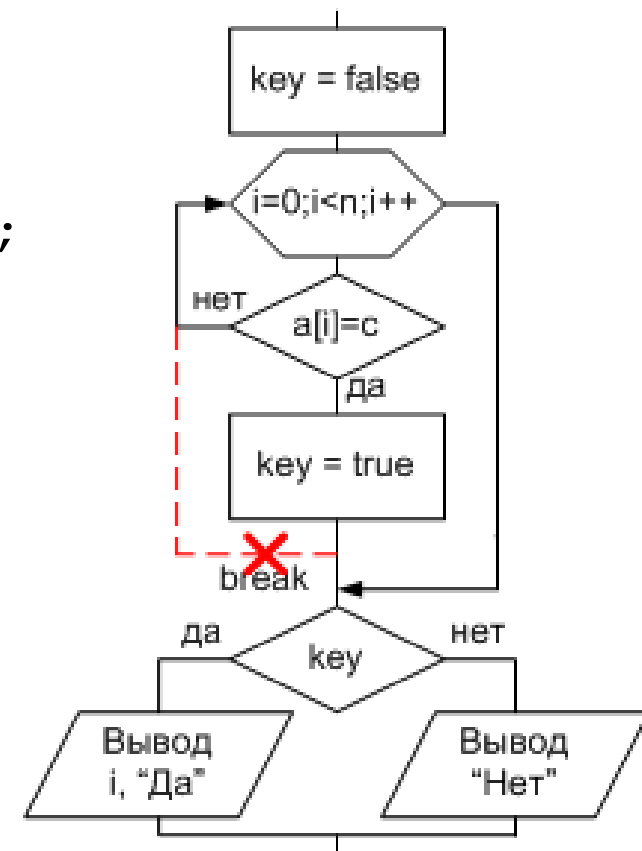
Третий вариант алгоритма реализует классический поисковый цикл с двумя выходами посредством двойного условия.

Алгоритмы поиска элемента в массиве



Реализация неструктурного варианта 2

```
#include <iostream>
using namespace std;
int a[10000];
int main() {
    int n,c,i;
    cout << "Enter n: "; cin >> n;
    cout << "Enter array:\n";
    for (int i=0;i<n;i++) cin >> a[i];
    cout << "Enter c: "; cin >> c;
    bool key = false;
    for (i=0;i<n;i++)
        if (a[i]==c) {
            key = true; break; }
    if (key) cout << i << "Yes.\n";
    else cout << "No.\n";
    return 0; }
```



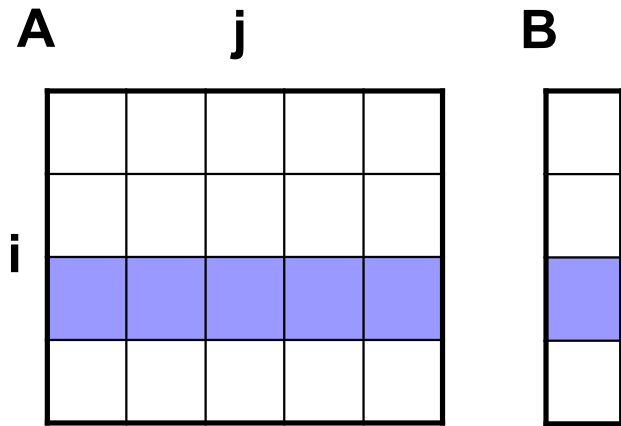
Реализация структурного варианта 3

```
#include <iostream>
using namespace std;
int a[10000];
int main() {
    int n,c;
    cout << "Enter n: "; cin >> n;
    cout << "Enter array:\n";
    for (int i=0;i<n;i++) cin >> a[i];
    cout << "Enter c: "; cin >> c;
    bool key = false; int i = 0;
    while (i < n && !key)
        if (a[i]==c) key = !key;
        else i++;
    if (key) cout << i << "Yes.\n";
    else cout << "No.\n";
    return 0; }
```

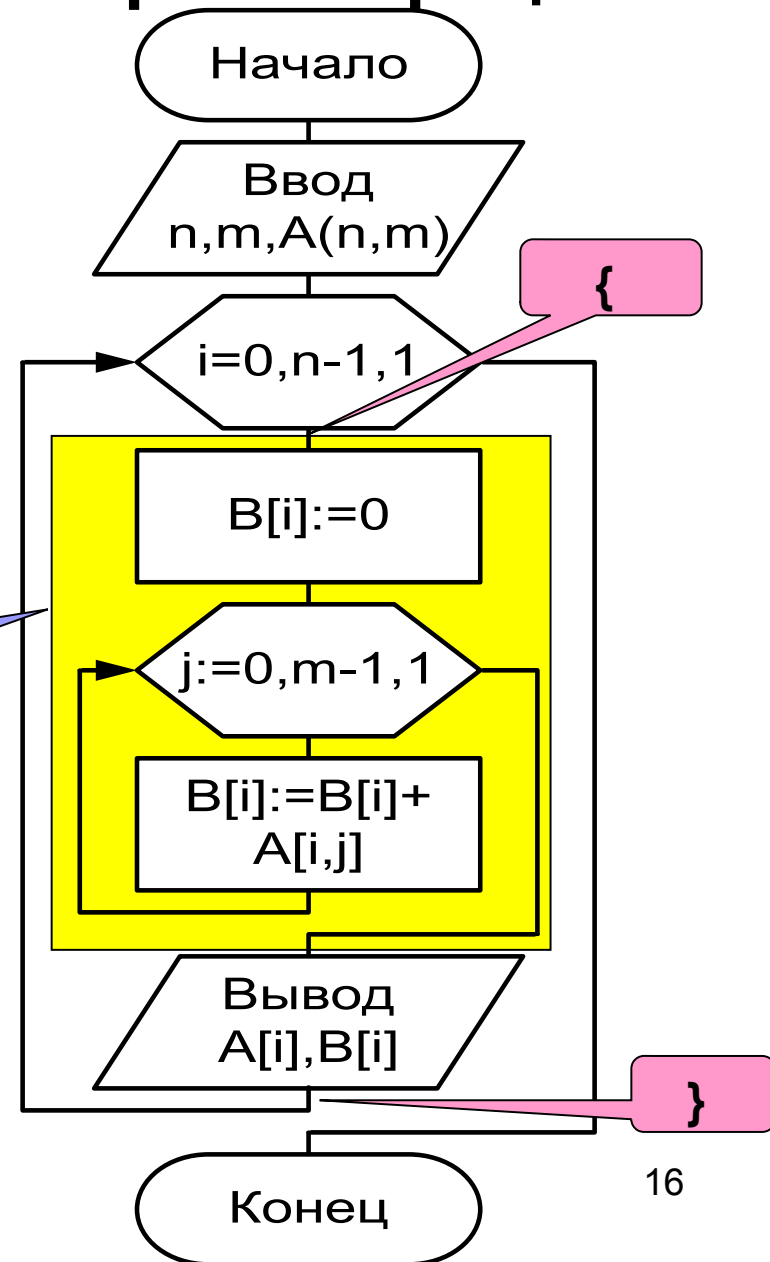


Пример 3 Сумма элементов строк матрицы

Ex03_05

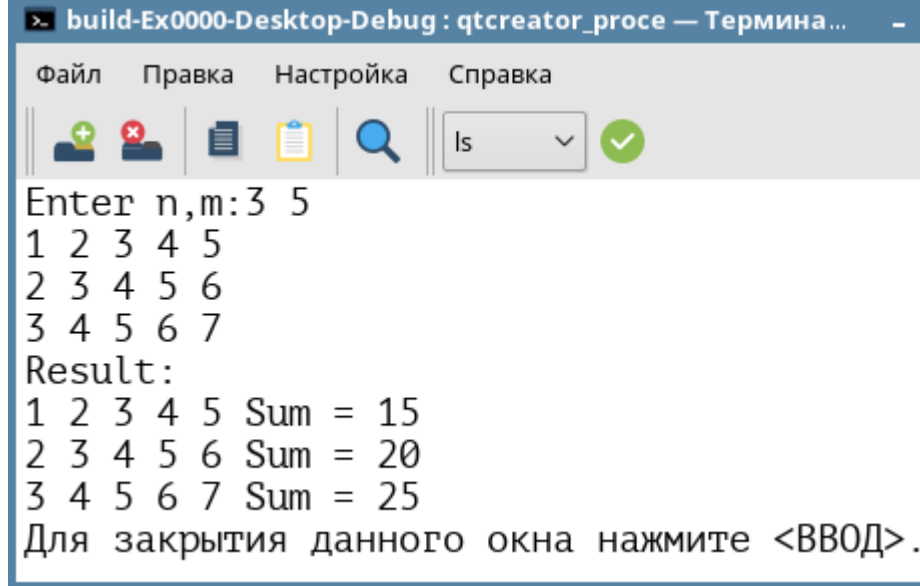


Подсчет суммы
элементов *i*-ой
строки



Программа суммирования элементов строк

```
#include <iostream>
using namespace std;
int main()
{
    float a[10][10], b[10];
    int n,m,i,j;
    cout << "Enter n,m:";
    cin >> n >> m;
    for(i=0;i<n;i++)
        for(j=0;j<m;j++)
            cin >> a[i][j];
    cout << "Result:" << endl;
    for(i=0;i<n;i++)
    {
        for(j=0,b[i]=0;j<m;j++)
        {
            cout << a[i][j] << ' ';
            b[i] += a[i][j];
        }
        cout << "Sum = " << b[i] << endl;
    }
    return 0;
}
```



The screenshot shows a terminal window titled "build-Ex0000-Desktop-Debug : qtcreeator_proce — Термина...". The terminal displays the following output:

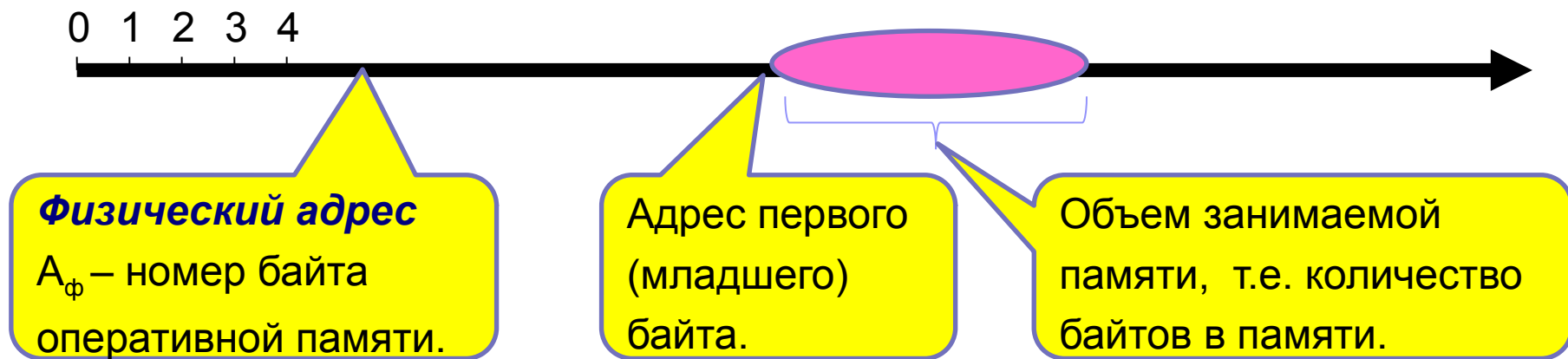
```
Enter n,m:3 5
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
Result:
1 2 3 4 5 Sum = 15
2 3 4 5 6 Sum = 20
3 4 5 6 7 Sum = 25
Для закрытия данного окна нажмите <ВВОД>.
```

The terminal window has a menu bar with "Файл", "Правка", "Настройка", and "Справка". Below the menu bar is a toolbar with icons for adding, removing, and saving files, a search icon, and a dropdown menu showing "ls" with a green checkmark icon.

3.2 Адресация оперативной памяти

Минимальная адресуемая единица памяти большинства современных процессоров – **байт**.

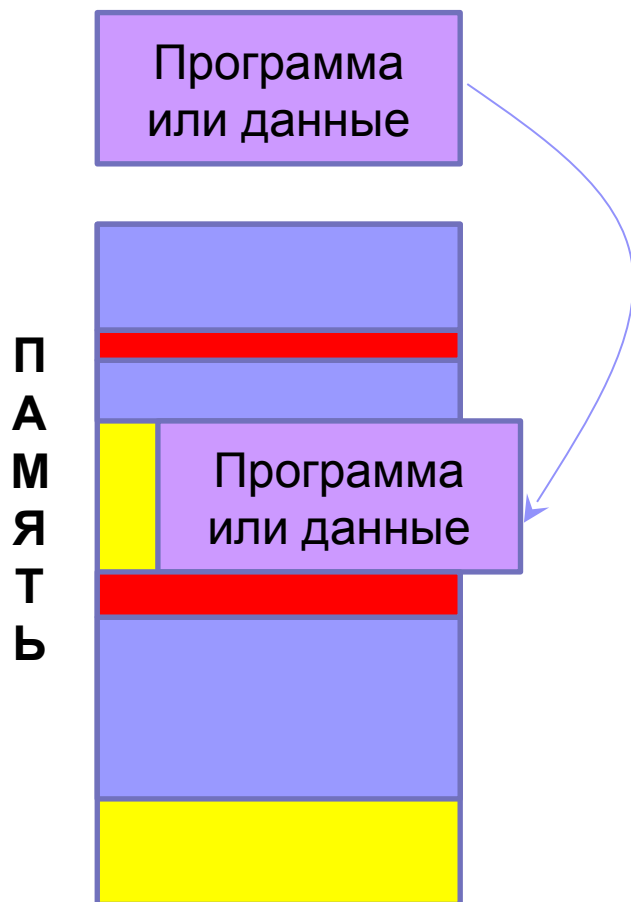
Байты памяти нумеруют, начиная с нуля. Номер байта и есть его адрес.



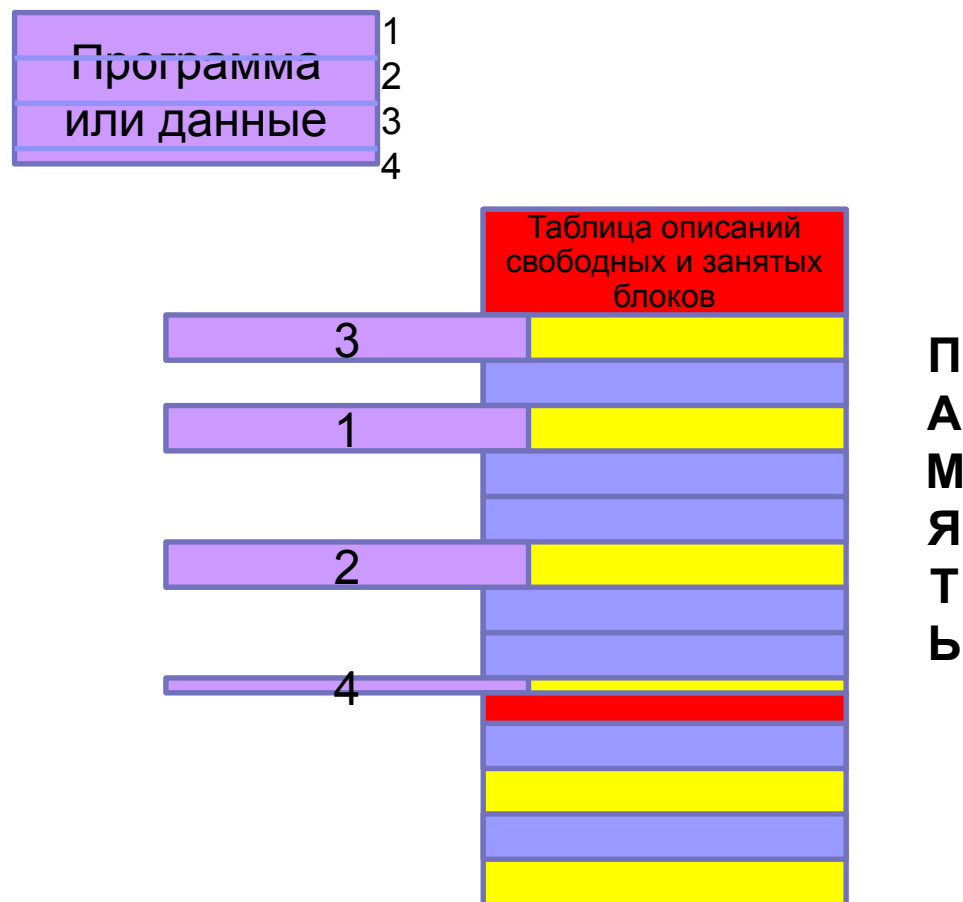
Размещаемая в памяти компьютера информация: числовые и текстовые данные, а также машинные коды программы - обычно имеют размер более 1 байта, т.е. занимают в памяти некоторое количество байтов.

Адрес программного объекта (переменной, подпрограммы и т.п.) = адрес его первого (младшего) байта.

Непрерывное и дискретное распределение памяти



Проблема – фрагментация памяти, при которой суммарный свободный объем памяти может быть достаточен для размещения данных, а непрерывного куска нужного размера нет...



Проблемы:

- часть памяти занята таблицей и возможны неполные блоки;
- фрагментация памяти, при которой части информации могут быть расположены в разных местах памяти...

3.3 Указатели

Указатель – тип данных, используемый при написании программы для хранения адресов. В памяти для 32-х разрядного приложения занимает 4 байта и адресует сегмент размером $V \cdot 2^{32} = 4\text{Гб}$.

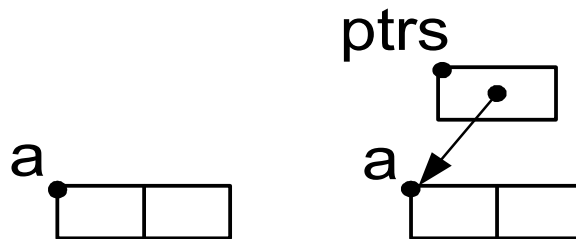
При сегментной модели памяти адреса определяются относительно начала сегмента.

\$ Указатель =

[Изменяемость_значения] Тип [Изменяемость_адреса] *Имя [=Адрес];

Примеры:

1) `short a, *ptrs = &a;`



2) `const short *ptrs;`

3) `short *const ptrs = &a;`

Неизменяемое значение:
можно `ptrs = &b;`; нельзя `*ptrs=10;`

Неизменяемый указатель
можно `*ptrs=10;`; нельзя `ptrs = &b;`

Операции над указателями

1. Присваивание

Примеры:

Типизированные
указатели

Нетипизированный
указатель

```
int a, *ptri, *ptrj;
```

```
void *b;
```

```
1) ptri = &a;
```

```
2) ptri = nullptr; // нулевой адрес (начиная с C++11)
```

```
3) ptri = ptrj;
```

```
4) b = &a;
```

```
5) ptri = b; ⇒ ptri = (int *) b;
```

Явное переопределение
типа указателя (C-style)

```
ptri = static_cast<int *> b;
```

Типизированному указателю нельзя присваивать нетипизированный!

Явное переопределение
типа указателя (C++-style)

Операции над указателями

2. Разыменование

Примеры:

```
int c, a = 5, *ptri = &a; void *b = &a;
```

1) `c = *ptri;`

2) `*ptri=125;`

3) `*b=6;` \Rightarrow `*(int *) b = 6;`

Явное переопределение
типа указателя (C-style)

`*static_cast<int*>(b) = 6;`

Нетипизированные
указатели нельзя
разыменовывать

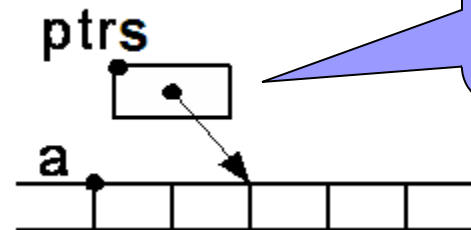
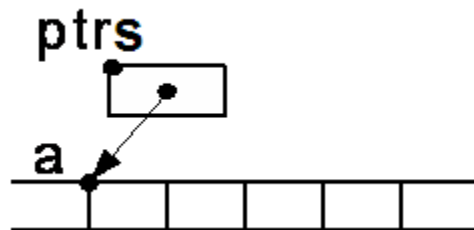
Явное переопределение
типа указателя (C++-style)

Основное правило адресной арифметики

Указатель + n \Leftrightarrow Адрес + n * sizeof(Тип_данных)

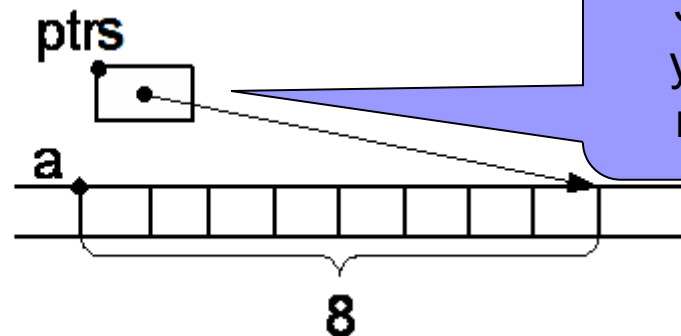
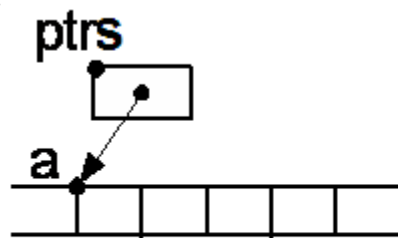
Примеры: short a, *ptrs =&a;

1) ptrs++;



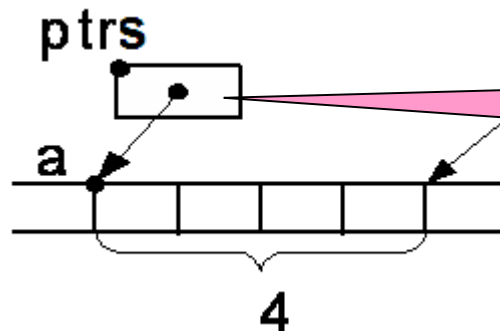
Значение
указателя
меняется

2) ptrs+=4;



Значение
указателя
меняется

3) *(ptrs+2)=2;



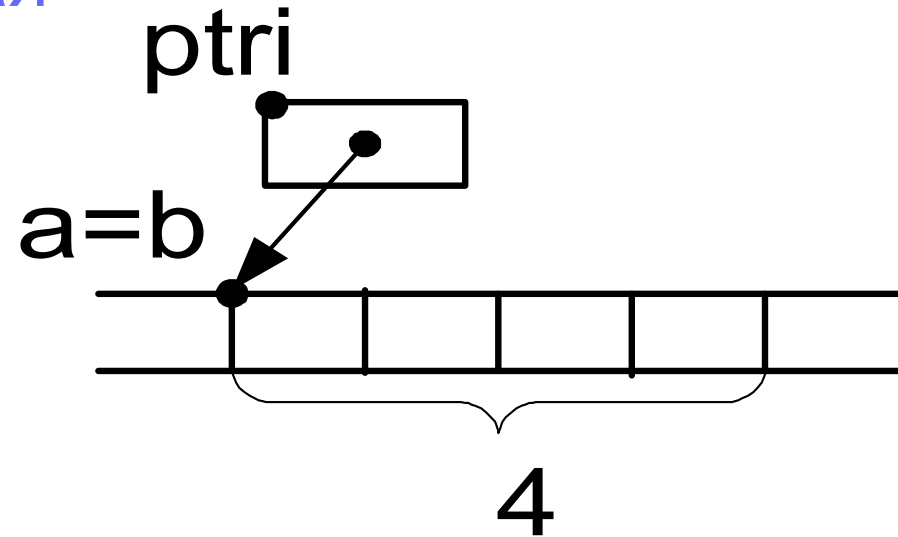
Значение
указателя
не меняется!!!

Ссылки

```
int a,      // переменная  
*ptri=&a,  // указатель  
&b=a;      // ссылка
```

...

```
a=3; ⇔ *ptri=3; ⇔ b=3;
```



Ссылка тоже физически представляет собой *адрес*, но в отличие от указателя при работе со ссылками **не используется операция разыменования**, т.е. ссылка – это второе имя (alias или псевдоним).

3.4 Управление динамической памятью

A. C-style

1. Размещение в памяти одного значения

Выделение памяти

```
void * malloc(size_t size);
```

Освобождение памяти

```
void free(void *block);
```

Пример:

```
int *a;
if ((a = (int *) malloc(sizeof(int))) == nullptr){
    printf("Not enough memory.");
    exit(1);
}

*a = -244;
free(a);
```

2. Размещение нескольких значений

Выделение памяти

```
void * calloc(size_t n, size_t size);
```

Освобождение памяти

```
void free(void *block);
```

Пример:

```
int *list;
```

```
list = (int *) calloc(3, sizeof(int));
```

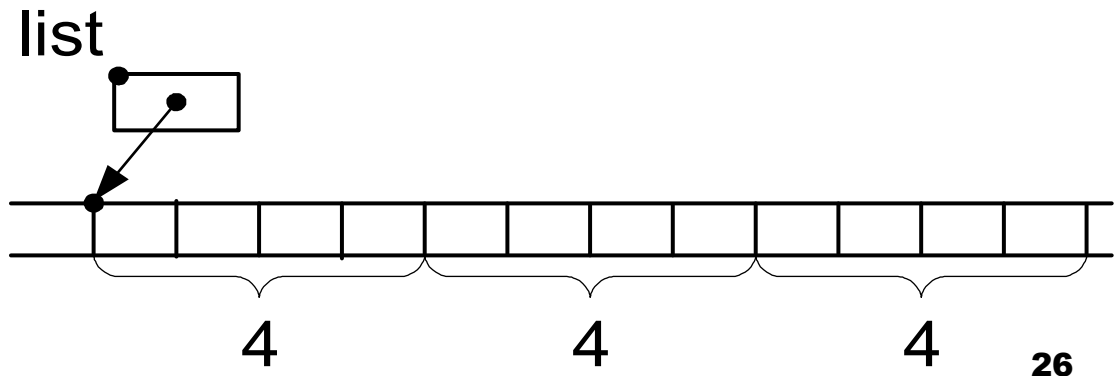
```
*list = -244;
```

```
*(list+1) = 15;
```

```
*(list+2) = -45;
```

...

```
free(list);
```



Б. C++-style

1. Размещение одного значения

Операция выделения памяти:

Указатель = **new** Имя_типа [**(<Значение>)**];

Операция освобождения памяти:

delete Указатель;

Примеры:

а) `int *k;`

`k = new int;`

`*k = 85;`

б) `int *a;`

`if ((a = new int(-244)) == nullptr) {`

`printf("Not enough memory.");`

`exit(1); }`

`delete a;`

2. Размещение нескольких значений

Операция выделения памяти для n значений:

Указатель = **new** Имя_типа **[Количество]**;

Операция освобождения памяти:

delete [] Типизированный_указатель;

Пример:

```
int *list;  
list = new int[3];  
*list=-244;  
*(list+1)=15;  
*(list+2)=-45;  
delete[ ] list;
```

Массивы

Организация массива в C++ – особый случай использования адресной арифметики.

Переменная массива – указатель на некоторое количество подряд идущих элементов одного типа, т.е. имеющих одинаковую длину.

Именно поэтому:

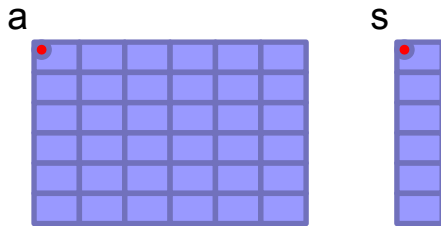
- 1) индексы массива всегда начинаются с 0;
- 2) многомерные массивы в памяти расположены построчно;
- 3) для адресации элементов массива независимо от способа описания можно использовать адресную арифметику:

$$(list+i) \Leftrightarrow \&(list[i])$$
$$*(list+i) \Leftrightarrow list[i]$$

Варианты программы подсчета сумм строк (2)

Автоматический массив (C/C++):

```
int a[n][n], s[n];
```

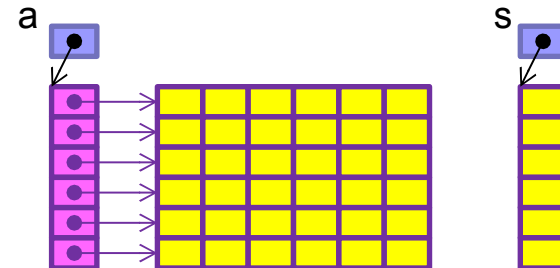


! Память будет выделена в момент выполнения программы и будет освобождена автоматически при выходе из функции, в которой объявлены массивы!

Динамический массив

Выделение памяти:

```
int **a = new int*[n],  
      *s = new int[n];  
for (int i;i<n;i++)  
    a[i] = new int[n];
```



Освобождение памяти:

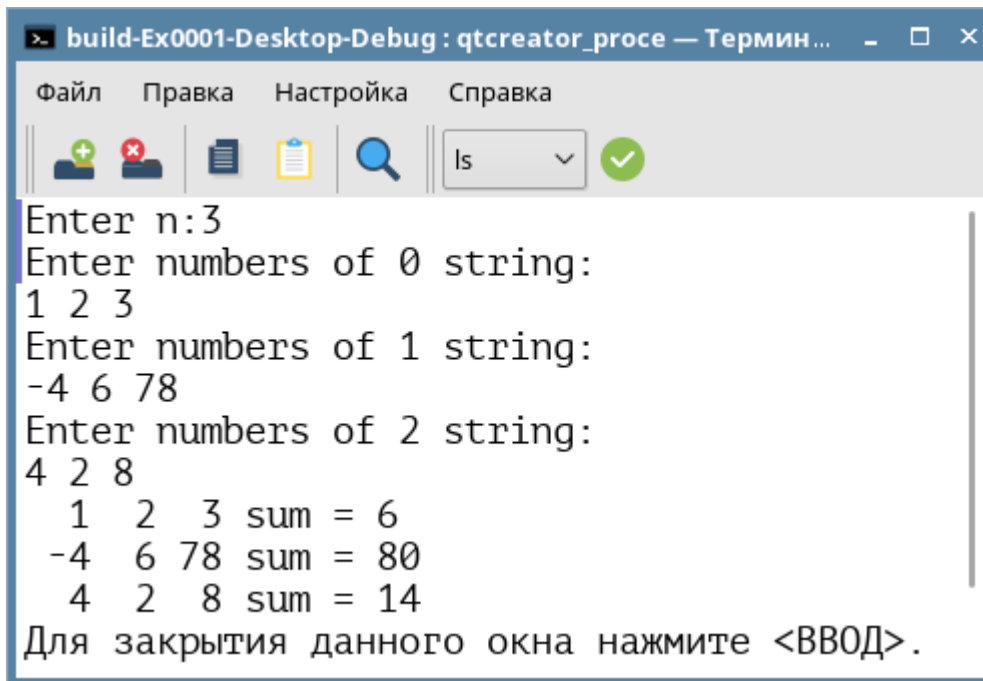
```
for (int i;i<n;i++)  
    delete [] a[i];  
delete [] a;  
delete [] s;
```

Пример программы подсчета сумм строк

Ex03_06

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    int n;
    cout<<"Enter n:";      cin >> n;
    int a[n][n], s[n]; // для компилятора C++
    for (int i = 0; i < n; i++) {
        cout << "Enter numbers of " << i << " string:\n";
        for (int j = 0; j < n; j++) cin >> a[i][j];
    }
    for (int i = 0; i < n; i++){
        s[i] = 0;
        for (int j = 0; j < n; j++) s[i] += a[i][j];
    }
    for (int i = 0; i < n; i++) {
        for (int j=0;j<n;j++) cout << setw(3)<< a[i][j];
        cout << " sum = " << s[i] << endl;
    }
    return 0;
}
```

Пример программы подсчета сумм строк



The screenshot shows a terminal window titled "build-Ex0001-Desktop-Debug : qtcreeator_proce — Термин...". The window has a menu bar with "Файл", "Правка", "Настройка", and "Справка". Below the menu is a toolbar with icons for adding/removing items, a document, a clipboard, a search icon, a dropdown menu showing "ls", and a green checkmark icon. The terminal output is as follows:

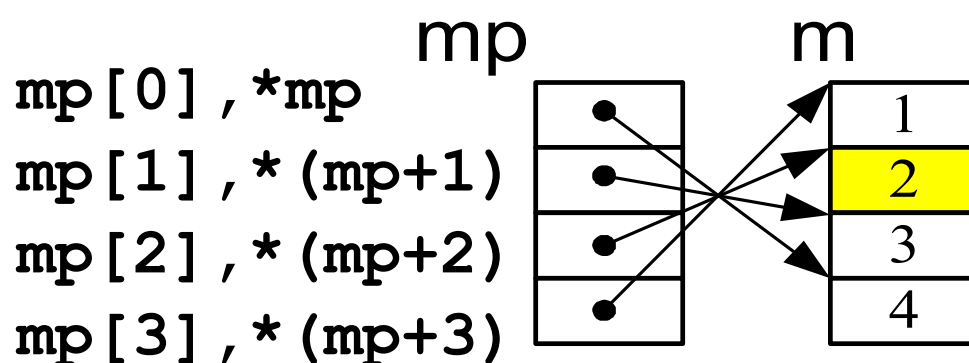
```
Enter n:3
Enter numbers of 0 string:
1 2 3
Enter numbers of 1 string:
-4 6 78
Enter numbers of 2 string:
4 2 8
    1  2  3 sum = 6
   -4  6 78 sum = 80
    4  2  8 sum = 14
Для закрытия данного окна нажмите <ВВОД>.
```


Многоуровневая адресация

```
int m[]={1,2,3,4};  
int *mp[]={m+3,m+2,m+1,m};
```

$(list+i) \Leftrightarrow \&(list[i])$

$*(list+i) \Leftrightarrow list[i]$



$m[1], *(m+1)$

ИЛИ

$mp[0][-2],$
 $*(mp[0]-2),$
 $*(*mp-2),$
 $mp[1][-1],$
 $*(mp[1]-1),$
 $*(* (mp+1) -1)$

3.5 Цикл foreach или цикл по коллекции (Ex3_07)

Цикл предложен для стандартных шаблонов коллекций, однако может использоваться в том числе и для массивов:

```
#include <iostream>
using namespace std;

int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    for(int &x : arr) {
        x*= 2;
    }
    for(auto x:arr)
        cout << x << ' ';
    for(const auto &x:arr)
        cout << x << ' ';
    return 0;
}
```

Размер массива
должен быть
известен!!!

Можно auto, тогда
тип определится
автоматически

Ссылка (&) - для
изменения чисел в
массиве

Значение – работа с
копиями чисел
массива

Константа – работа
со значениями с
запретом изменений

3.6 Строки

Строка в C и C++ – последовательность (массив) символов, завершающаяся нулевым байтом.

Примечание. Цикл `for` по коллекции для строк использовать нельзя, поскольку он не видит завершающего нуля!

4	A	B	C	D	
---	---	---	---	---	--

Длина строки
Паскаля

A	B	C	D	\0	
---	---	---	---	----	--

Признак конца
строки C(C++)

Объявление строки

Объявление строки с выделением памяти:

```
char Имя_указателя [Объем_памяти] [= Значение];
```

Объявление указателя на строку:

```
char *Имя_указателя [= Значение];
```

Примеры:

а) `char str[6];`

По умолчанию
константный
указатель!!!

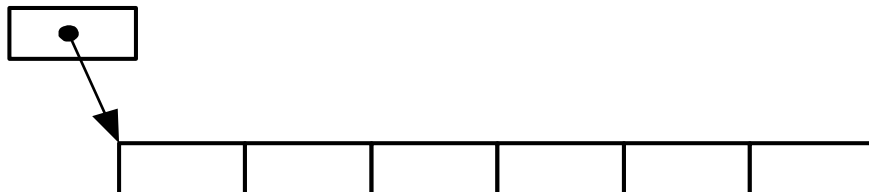


б) `char *ptrstr;`

```
ptrstr = new char[6];
```

```
delete [] ptrstr;
```

ptrstr



в) `char str1[5] = { 'A', 'B', 'C', 'D', '\0' }; // указатель константен`

г) `char str2[5] = "ABCD"; // указатель константен`

д) `char str3[] = "ABCD"; // указатель константен`

е) `const char *str4 = "ABCD"; // Важно! Иначе типы не совместимы`

Объявление и инициализация массивов строк

Массив указателей на строки

```
char * Имя [Размер] [= Значения];
```

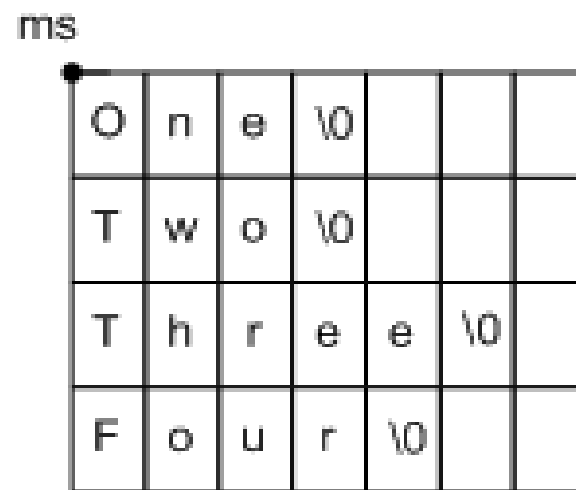
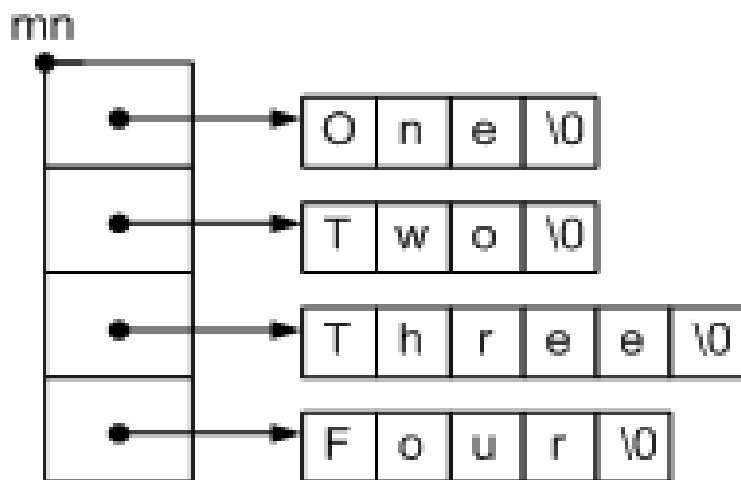
Массив строк указанной длины

```
char Имя [Размер][Размер] [= Значения];
```

Примеры:

а) `const char * mn[4] = {"One", "Two", "Three", "Four"};`

б) `char ms[4][7] = {"One", "Two", "Three", "Four"};`



Ввод-вывод строк

Ввод:

```
char str[50];
```

- 1) `gets(str);` // процедура-функция – ввод до Enter (устаревшая!)
`gets_s(str, 50);` // для VS
`fgets(str, 50, stdin);` // для Clang
- 2) `scanf("%s", str);` // ввод до пробела
- 3) `cin >> str;` // с использованием потока

Вывод:

- 1) `puts(str);` // вывод и переход на следующую строку
- 2) `printf("String = %s\n", str);` // вывод и переход на следующую
// строку
- 3) `cout << str << endl;` // с использованием потока

Функции, работающие со строками

Библиотеки: `string.h`, `stdlib.h`

1) **определение длины строки**: `size_t strlen(char *s);`

например: `int k = strlen(str);`

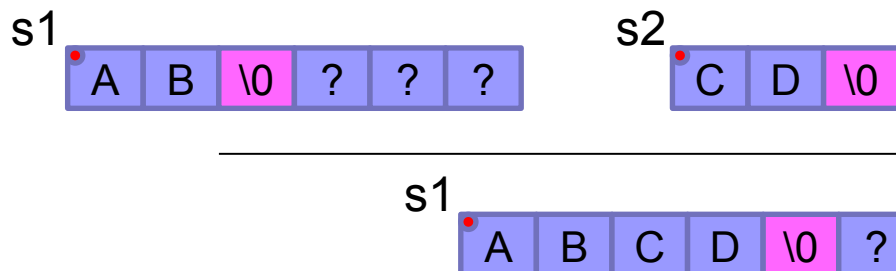
2) **конкатенация (слияние) строк**:

`char *strcat(char *dest, const char *src);`

например: `puts(strcat(s1, s2));` // **strcat** возвращает указатель

или `strcat(s1, s2);`

Процедура-функция, результат получаем по адресу первого операнда, в котором должно хватать места, и дублируется как результат функции.



Функции, работающие со строками

3) сравнение строк:

```
int strcmp(const char *s1, const char *s2);
```

например: `k = strcmp(s1, s2);`

Выполняется посимвольным вычитанием кодов символов до конца или получения отличного от нуля результата: если $k=0$, то строки равны, если $k>0$, то первая больше, иначе – вторая больше.

s1

A	B	\0	?	?	?
---	---	----	---	---	---

s2

A	D	\0
---	---	----

0 -2

Первая строка
меньше второй

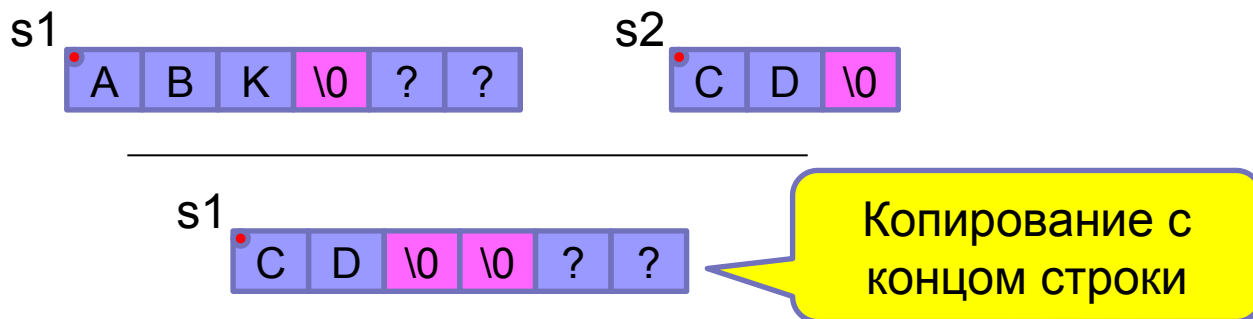
Функции, работающие со строками (2)

4) **копирование строки** `src` в `dest`:

```
char *strcpy(char *dest, const char *src);
```

например: `puts(strcpy(s1, s2));`

или `strcpy(s1, s2);`

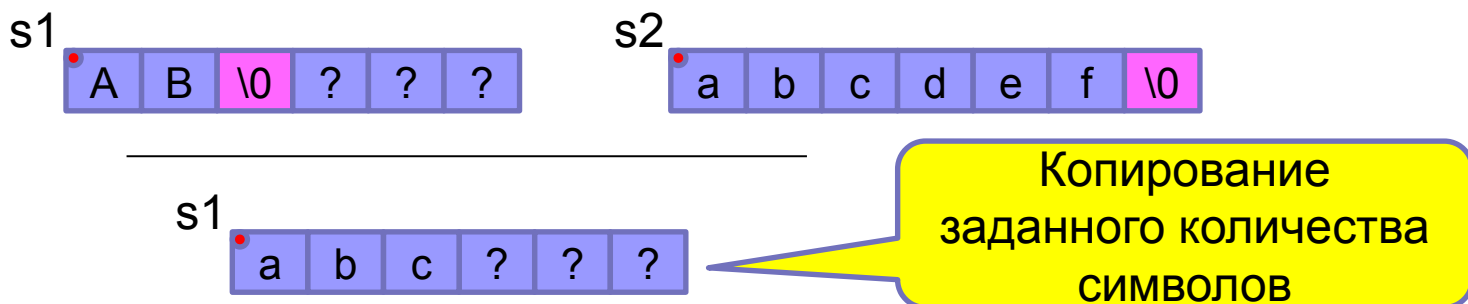


5) **копирование фрагмента**:

```
char *strncpy(char *dest, const char *src, size_t maxlen);
```

например: `strncpy(s1, "abcdef", 3);`

Копирует в строку `dest` фрагмент размера `maxlen` из строки `src`.



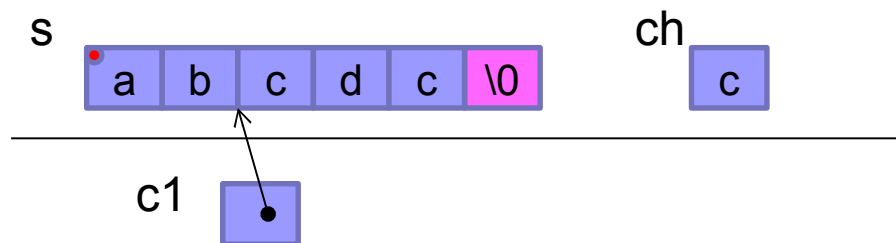
Функции, работающие со строками (2)

6) **поиск символа c в строке s:**

```
char *strchr(const char *s, int c);
```

например: `char * c1 = strchr(s, ch);`

Возвращает адрес первого вхождения символа в строку или `nullptr`.

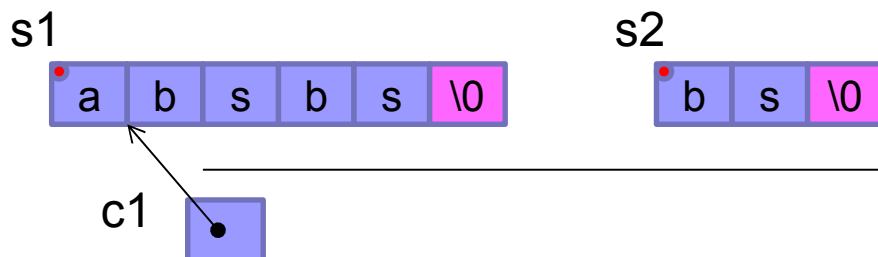


7) **поиск подстроки s2 в строке s1:**

```
char *strstr(const char *s1, const char *s2);
```

например: `char * c1 = strstr(s1, s2);`

Возвращает адрес первого вхождения подстроки `s2` в строку `s1` или `nullptr`.



Функции, работающие со строками (3)

8) ПОИСК ТОКЕНОВ В СТРОКЕ:

```
char *strtok(char *strToken, const char *strDelimit);
```

Пример:

```
#include <string.h>
```

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

```
int main() {
```

```
    char str[] = "A string\tof , , tokens\nand some more  
tokens";
```

```
    char seps[] = " , \t \n", *token;
```

```
    token = strtok(str, seps);
```

```
    while (token != nullptr)
```

```
    {
```

```
        printf("%s ", token);
```

```
        token = strtok(nullptr, seps);
```

```
    }
```

```
    return 0;
```

```
}
```

Разделители

A string of tokens and some more tokens

Поиск токенов в строке



Функции, работающие со строками (4)

9) преобразование строки в целое число:

```
int atoi(const char *s);
```

Любое число вводится и выводится на консоль в виде строки символов:

Например вводим число -45 в символьном виде '-' '4' '5' или

в шестнадцатеричном виде:

2 D	3 4	3 5	0 0
-----	-----	-----	-----

После преобразования во внутреннее представление (short) получаем:

1 1 1 1 1 1 1 1	1 1 0 1 0 0 1 1
-----------------	-----------------

или в шестнадцатеричном виде:

F F	D 3
-----	-----

10) преобразование строки в вещественное число:

```
double atof(const char *s);
```

11) преобразование числа в строку (MVS, stdlib.h):

```
char *itoa(int value, char *s, int radix);
```

radix – основание системы счисления.

Функции, работающие со строками (5)

- 12) преобразование вещественного числа в строку (MVS, stdlib.h) :

```
char *fcvt(double value, int decimals,  
            int *dec, int *sign);
```

`decimals` – количество знаков после точки;

`dec, sign` – позиции точки и знак.

- 13) преобразование вещественного числа в строку (MVS, stdlib.h) :

```
char *ecvt(double value,  
            int count, int *dec, int *sign);
```

`count` - количество преобразуемых цифр;

`dec, sign` – позиции точки и знак.

Функции, работающие со строками (6)

14) формирование строки по формату (stdio.h):

```
int sprintf(char *buf, const char *format,  
            arg-list);
```

`buf` – сформированная строка,

`format` – формат;

`arg-list` – список аргументов.

Функция возвращает длину сформированной строки.

Пример:

```
char str[80];  
sprintf (str, "%s %d %c", "one", 2, '3');  
cout << str << endl;
```

one 2 3

Пример преобразования числа в строку (Ex3_08)

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

int main()
{
    char *buf;           // буфер
    int decimal, sign;    // позиция десятичной точки и знак
    int count=10;        // количество преобразуемых разрядов
    int err;             // код ошибки

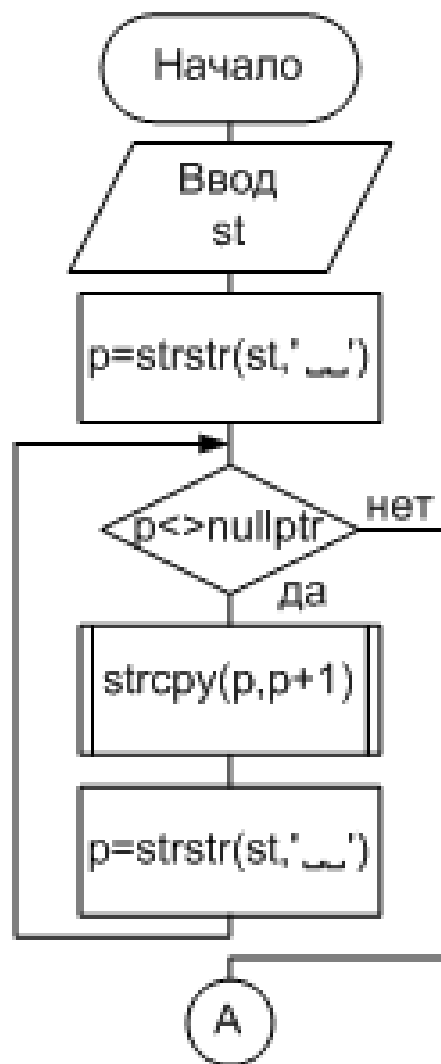
    buf = ecvt(3.1415926535, count, &decimal, &sign);
    printf("Converted value to string: %s\n", buf);
    printf("Decimal= %d, Sign= %d.", decimal, sign);

    return 0;
}
```

Converted value to string: 31415926535
Decimal= 1, Sign= 0.

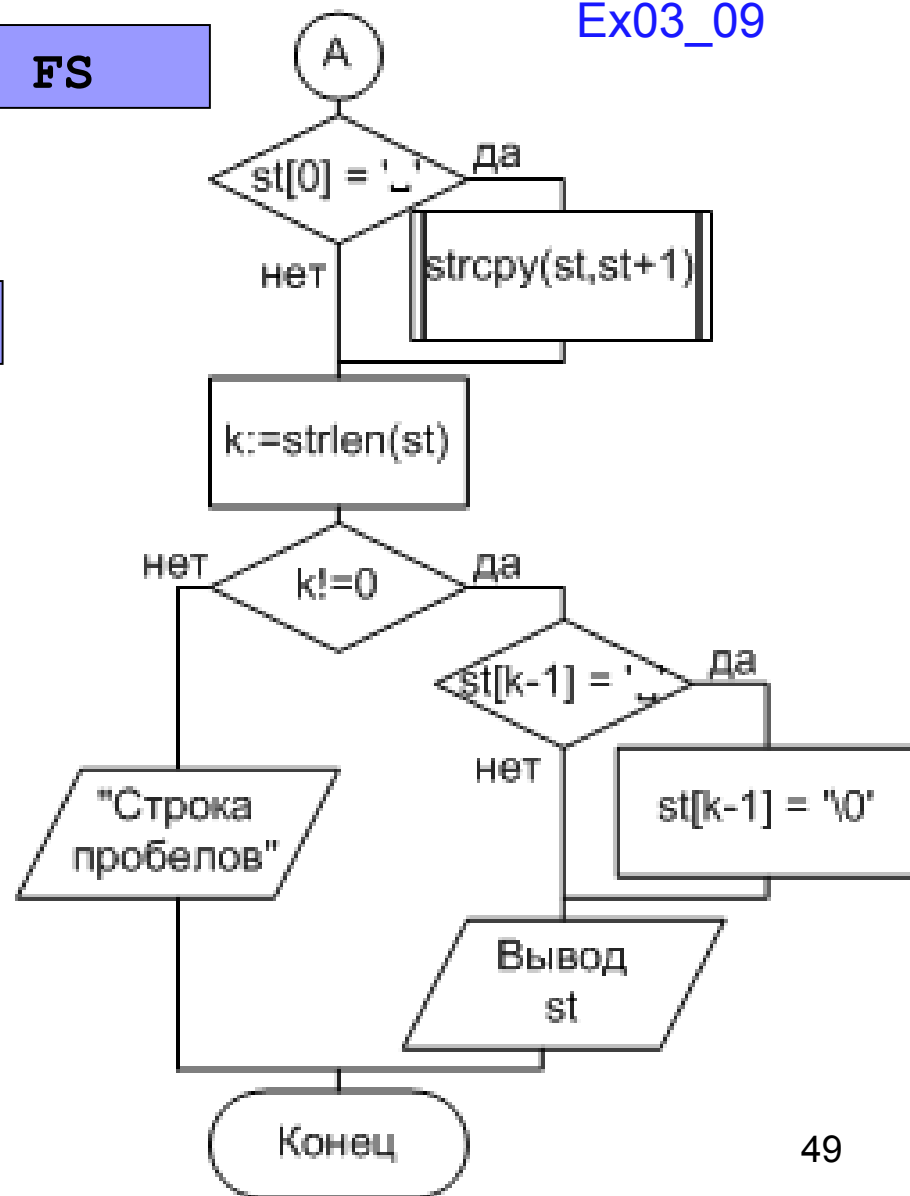
Удаление «лишних» пробелов из строки

Ex03_09



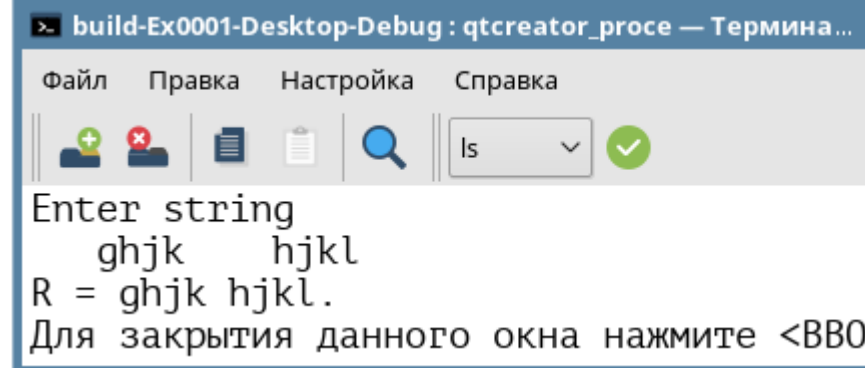
ASD FS

ASD FS



Программа

```
#include <string.h>
#include <stdio.h>
int main() {
    char st[40];
    puts("Enter string");
    fgets(st, sizeof(st), stdin); // ВМЕСТО gets(st);
    int n = strlen(st); // на последнем месте стоит \n
    st[n-1] = '\\0';
    while (char *p = strstr(st, " "))
        strcpy(p, p+1);
    if (st[0]==' ')
        strcpy(st, st+1);
    if (int k = strlen(st)) {
        if (st[k-1] == ' ')
            st[k-1]='\\0';
        printf("R = %s.\\n", st);
    }
    else puts("Empty string.");
    return 0;
```



```
}
```

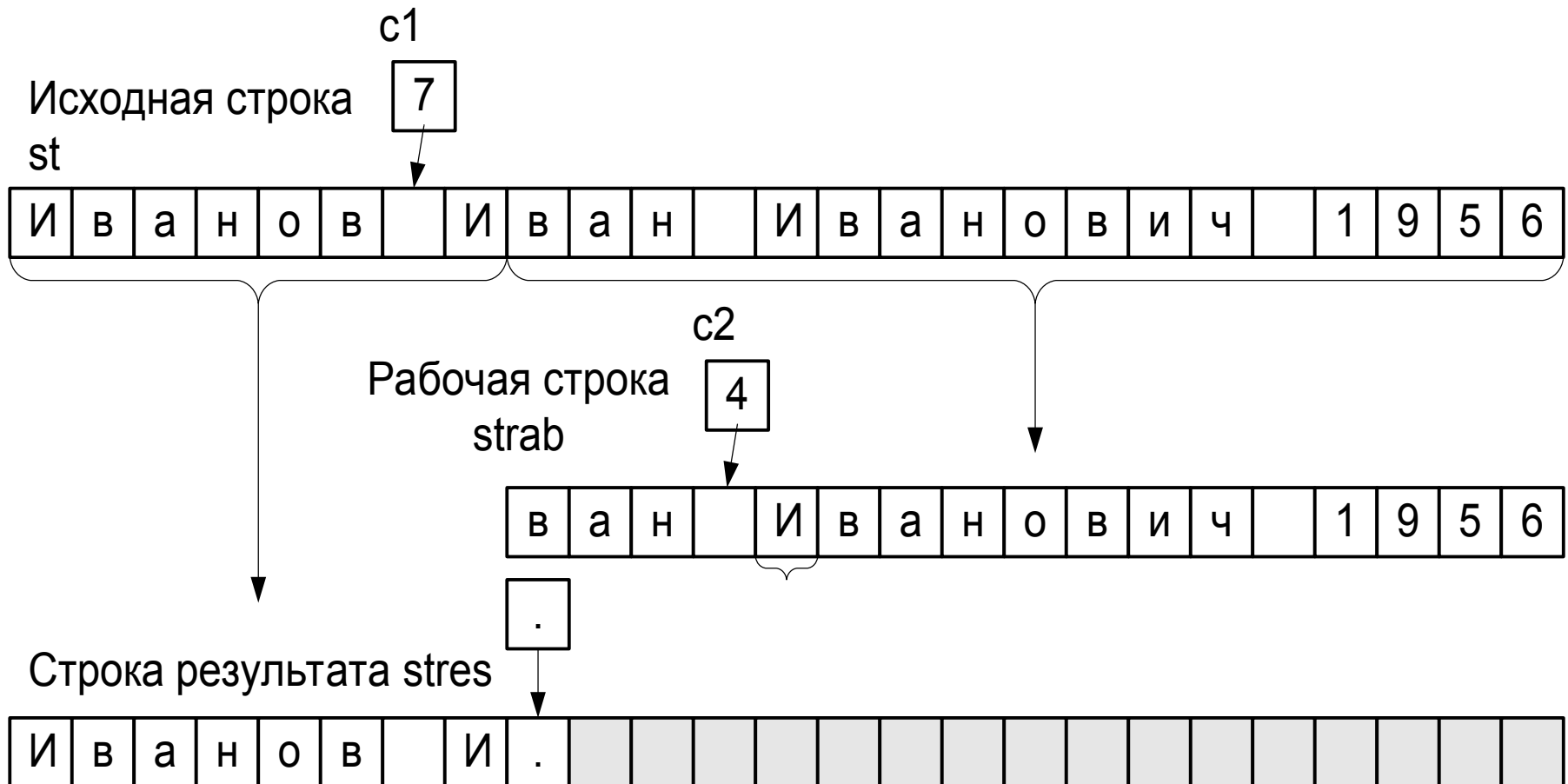
Преобразование последовательности строк

Вводится последовательность строк вида

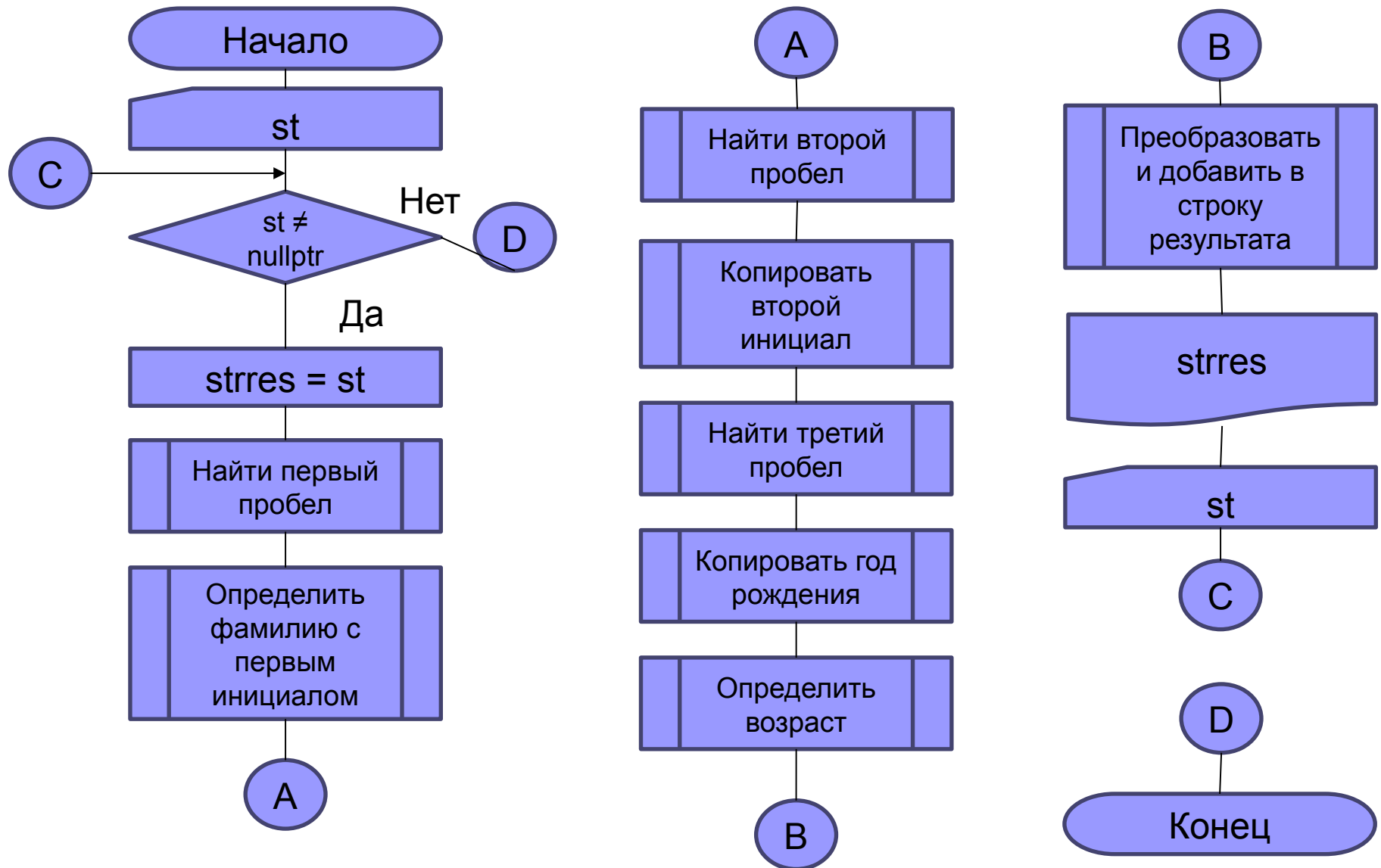
Ex03_10

Иванов Иван Иванович 1956 \Rightarrow Иванов И.И. 45

Завершение ввода – при чтении пустой строки.



Обобщенная схема алгоритма



Пример использования функций обработки строк

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

```
int main()
{ char st[80], stres[80], strab[80],
  *c1, *c2, *c3;
  int old;
  while ((puts("Enter string or end:"),
          strcmp(gets(st), "end") != 0) {
    strcpy(stres, st);
    c1 = strchr(stres, ' ');
    *(c1+2) = '.';
```

st

Petrov Petr Petrovich 1956

stres

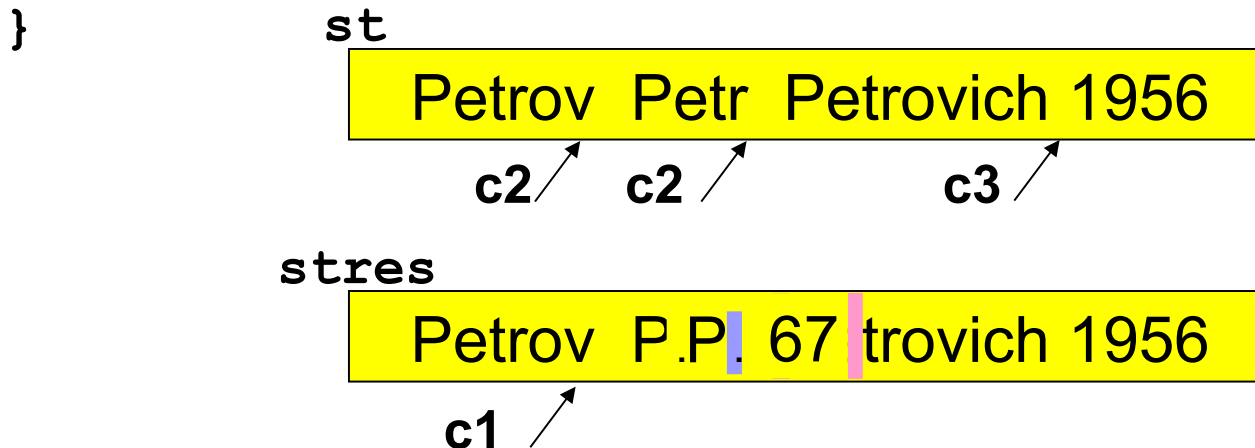
Petrov P. tr Petrovich 1956

c1



Пример использования функций обработки строк (2)

```
c2=strchr(st,' ');  
c2=strchr(c2+1,' ');  
strncpy(c1+3,c2+1,1);  
strncpy(c1+4,". \0",3);  
c3=strchr(c2+1,' ');  
old=2023-atoi(c3+1);  
sprintf(strab,"%d",old); // itoa(old,strab,10);  
strcat(stres,strab);  
puts(stres);    }  
return 0;
```



3.7 Структуры

Структура - последовательность полей, в общем случае различных типов.

А. Объявление структур и переменной (C-style)

```
struct [Имя_структуры] {{Описание_поля} }  
                                     [{Переменная [= Значение]}];  
[struct] Имя_структуры {Переменная [= Значение]};
```

Примеры:

```
а) struct student {char name[22];char family[22];int old;};  
    struct student stud1={"Petr","Petrov",19},stud[10],  
                                     *ptrstud;  
б) struct {char name[22];char family[22];int old;}  
          stud1, stud[10], *ptrstud;
```

Б. Объявление структур (C++-style)

```
typedef struct {Описание_поля} Имя_структуры;  
[struct] Имя_структуры {Переменная [= Значение]};
```

Пример:

```
typedef struct {char name[22];char family[22];int old;}  
student;  
struct student stud1={"Petr","Petrov",19},  
stud[10],*ptrstud;
```


Обращение к полям структуры

Имя_переменной.Имя_поля

Имя_массива[Индекс]. Имя_поля

(*Имя_указателя).Имя_поля или

Имя_указателя -> Имя_поля

Примеры:

`stud1.name`

`stud[i].name`

`(*ptrstud).name` \Leftrightarrow `ptrstud -> name`

Задача Массив записей

Вводится список:

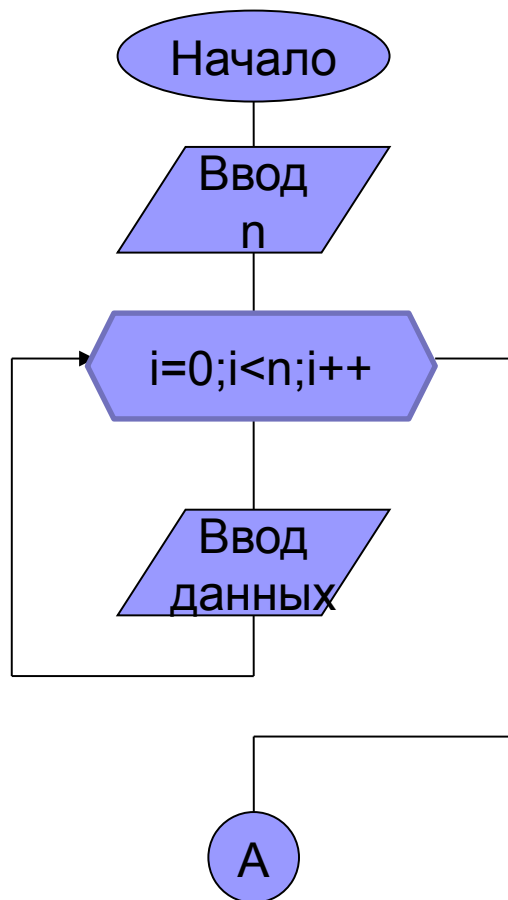
Ф.И.О.	Год р.	Месяц р.	Дата р.
Иванов Б.А.	1986	11	26
Петров М.А.	1985	5	12
Сидоров А.В.	1986	4	8

Определить дату рождения по фамилии и инициалам.

Программа

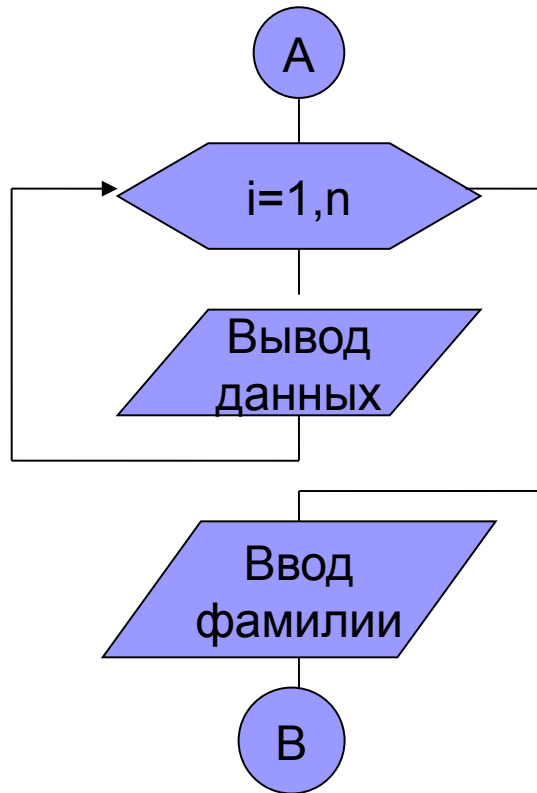
```
#include <string.h>
#include <stdio.h>
struct data {                // структура
    unsigned short year;    // год
    unsigned short month;   // месяц
    unsigned short day;     // день
};
struct record {              // структура
    char fam[22];           // фамилия
    data birthday;          // день рождения
};
int main() {
    record basa[40];         // массив из 40 структур типа record
    char name[22];          // строка для ввода фамилии
    bool key;               // переменная для реализации поиска
    int n;                  // количество записей в массиве
```

Ввод записей



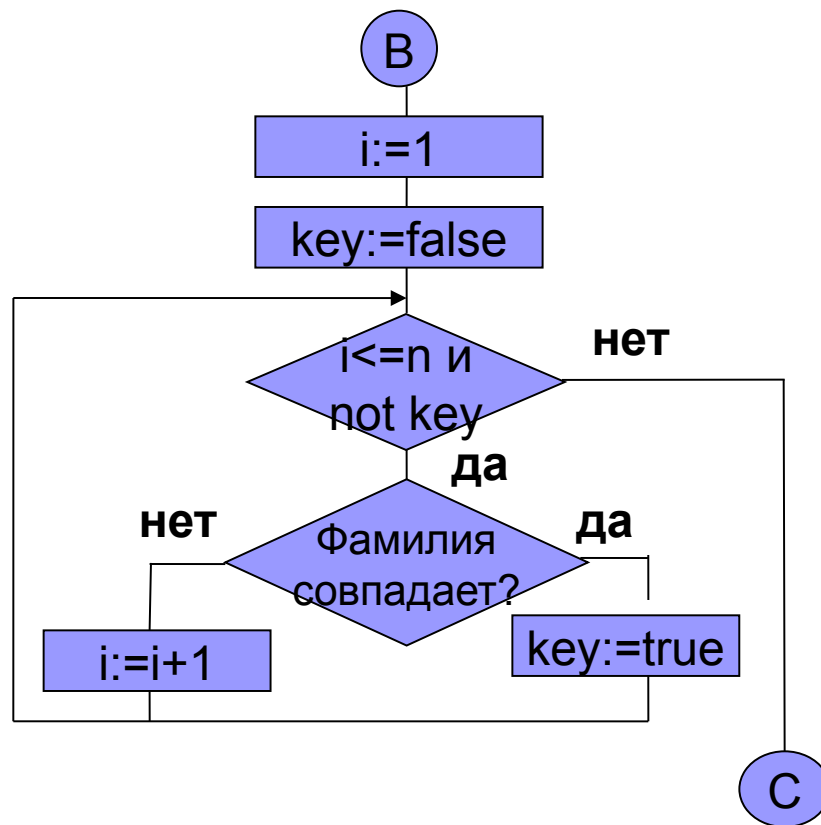
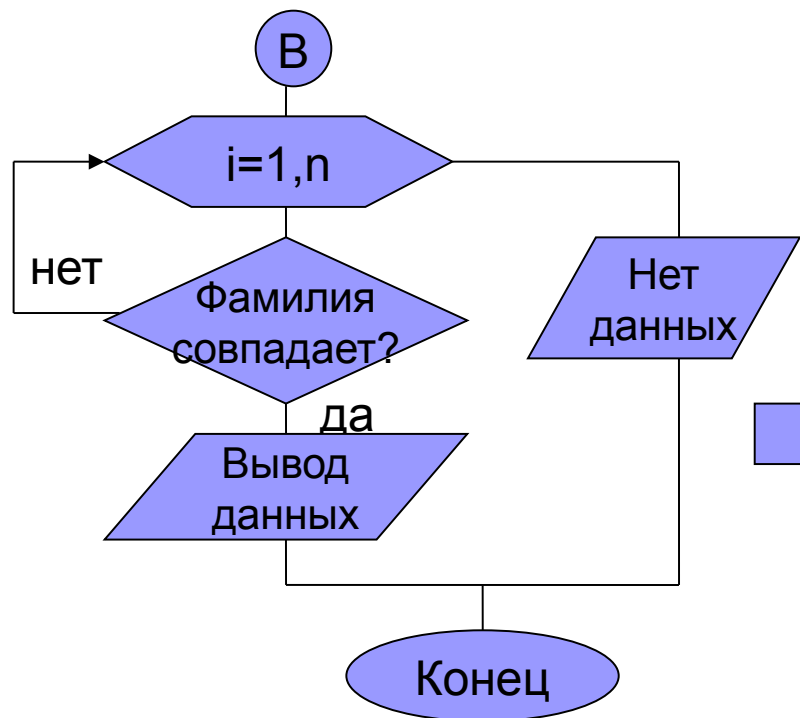
```
printf("Enter n:");
scanf("%d",&n);
for(int i=0;i<n;i++){
    printf("Enter family: ");
    scanf("%s",basa[i].fam);
    printf("Enter birthday year: ");
    scanf("%hu",
           &basa[i].birthday.year);
    printf("Enter birthday month: ");
    scanf("%hu",
           &basa[i].birthday.month);
    printf("Enter birthday day: ");
    scanf("%hu",
           &basa[i].birthday.day);
}
```

Вывод списка и ввод фамилии



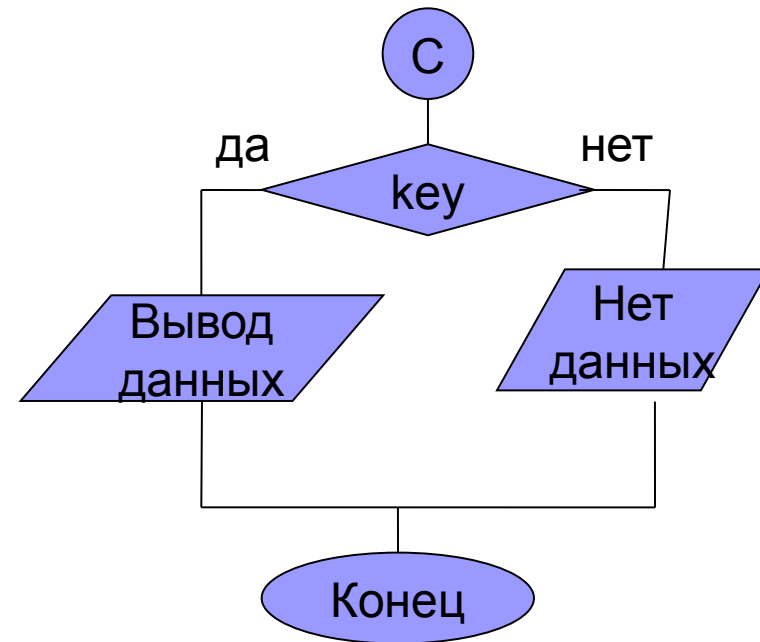
```
puts("List:");  
for(int i=0;i<n;i++){  
    printf("%s ",basa[i].fam);  
    printf("%d.",  
           basa[i].birthday.year);  
    printf("%d.",  
           basa[i].birthday.month);  
    printf("%d\n",  
           basa[i].birthday.day);  
}  
printf("Enter family: ");  
scanf("%s",name);
```

Поиск. Программирование поискового цикла



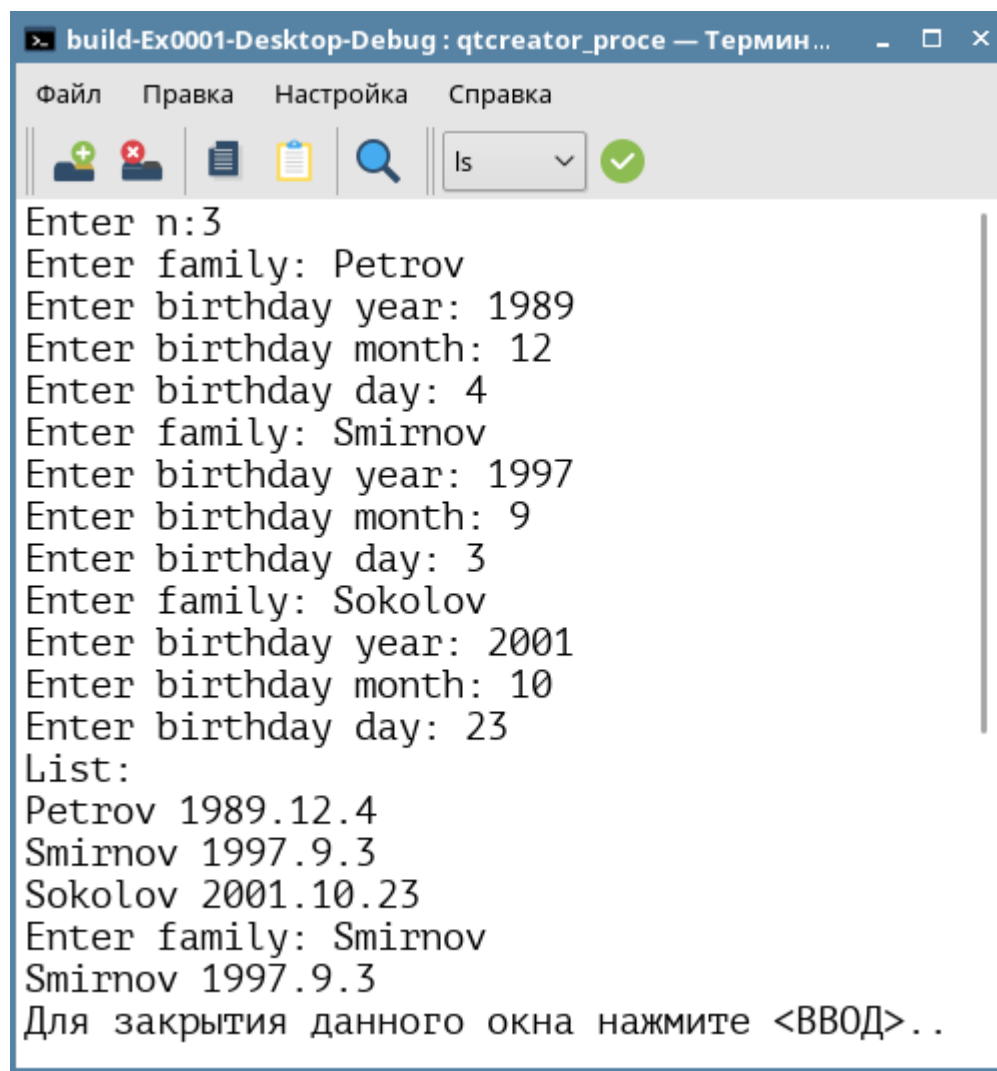
```
key = false;
int i=0;
while (i<n && !key)
    if (strcmp(basa[i].fam, name))
        i++;
    else key = true;
```

Вывод результата



```
if (key) {  
    printf("%s ",basa[i].fam) ;  
    printf("%d.",basa[i].birthday.year) ;  
    printf("%d.",basa[i].birthday.month) ;  
    printf("%d\n",basa[i].birthday.day) ;  
}  
else  
    puts("No data.") ;  
return 0 ;  
}
```

Результаты работы программы



The screenshot shows a terminal window titled "build-Ex0001-Desktop-Debug : qtcreeator_proce — Термин...". The window has a menu bar with "Файл", "Правка", "Настройка", and "Справка". Below the menu is a toolbar with icons for adding/removing files, saving, undo, redo, and search. A dropdown menu shows "ls" and a green checkmark icon. The terminal output is as follows:

```
Enter n:3
Enter family: Petrov
Enter birthday year: 1989
Enter birthday month: 12
Enter birthday day: 4
Enter family: Smirnov
Enter birthday year: 1997
Enter birthday month: 9
Enter birthday day: 3
Enter family: Sokolov
Enter birthday year: 2001
Enter birthday month: 10
Enter birthday day: 23
List:
Petrov 1989.12.4
Smirnov 1997.9.3
Sokolov 2001.10.23
Enter family: Smirnov
Smirnov 1997.9.3
Для закрытия данного окна нажмите <ВВОД>..
```


Пример использования структуры (Ex3_05)

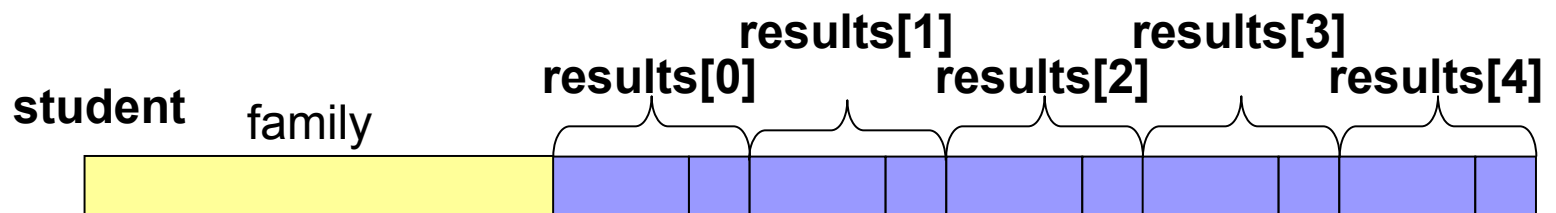
Программа определения среднего балла каждого студента и группы в целом

```
#include <stdio.h>
#include <string.h>
typedef struct {
    char name[10];
    int ball;
} test;
typedef struct {
    char family[22];
    test results[5];
} student;
```

test



name ball



Пример использования структуры (2)

```
int main()
{
    student stud[10]; int i,n=0; float avarstud,avarage=0;
    while (puts("Input names, subjects and marks or end"),
           scanf("\n%s",stud[n].family),
           strcmp(stud[n].family,"end")!=0) {
        for (avarstud=0,i=0; i<3; i++)
            { scanf("\n%s %d",stud[n].results[i].name,
                    &stud[n].results[i].ball);
              avarstud+=stud[n].results[i].ball;}
        printf("Average:%s=%5.2f\n",
               stud[n].family,avarstud/3);

        avarage+=avarstud;
        n++; }
    printf("Group average mark=%5.2f\n",avarage/n/3);
    return 0;
}
```

3.8 Объединения

```
union [Имя_объединения] {{Описание_поля} }  
    [{Переменная [= Значение]}];
```

```
[union] Имя_объединения{Переменная [= Значение]};
```

Пример:

```
union mem  
{  
    double d;  
    long l;  
    int k[2];  
};
```

