Массивы

Алтайский государственный университет
Факультет математики и ИТ
Кафедра информатики

Барнаул 2014

Лекция 10

- План
  - Массивы
  - Массивы: типичные задачи
  - Массивы как параметры функций
  - Двумерные массивы

Пять заданий для самопроверки

Задание 1

- Что выведет программа?

```
#include <stdio.h>

void main() {
  extern int p;
  printf("%d",p);
}
int p;
```

```
0
```

Задание 2

- Что выведет программа?

```
#include <stdio.h>

void main() {
  extern int p=25;
  printf("%d",p);
}
int p;
```

Возникнет ошибка компиляции!

`extern` предшествует объявлению, а не определению переменной => нельзя инициализировать

Задание 3

- Что выведет программа?

```
#include <stdio.h>

#define char double

void main() {
  char a=9;
  printf("%d",sizeof(a));
}
```

```
8
```

Задание 4

- Что выведет программа?

```
#include <stdio.h>
#define float int

void main() {
  int x=10;
  float y=3;
  y=x%y;
  printf("%f",y);
}
```

```
0
```

Задание 5

- Что выведет программа?

```
#include <stdio.h>

void main() {
  unsigned short int a=65536;
  if(!a)
    printf("%d",a,++a);
  else
    printf("%d",a,++a);
}
```

```
1
```

Массивы

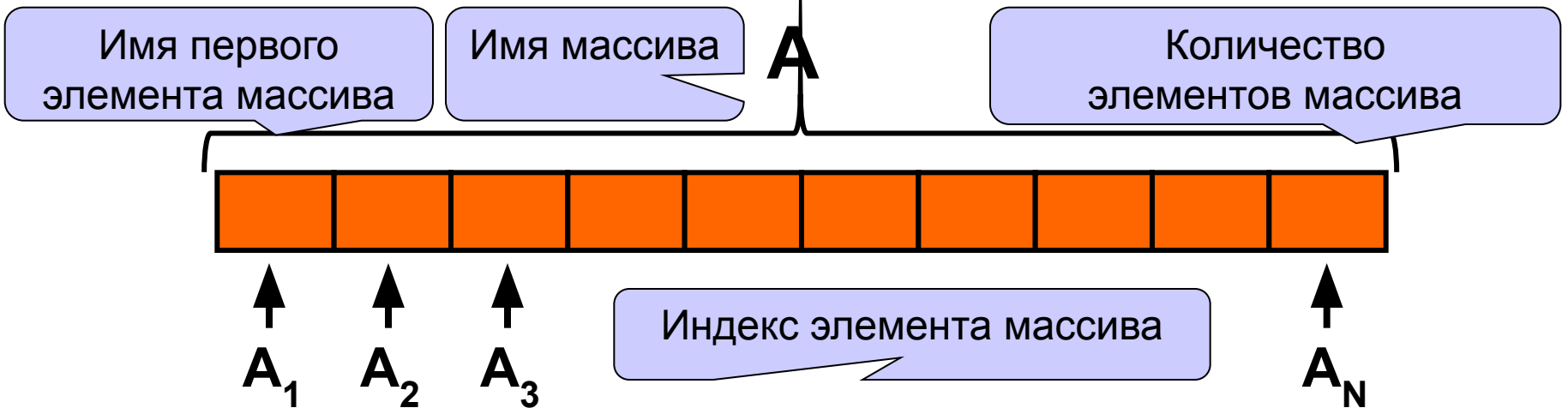
- Основные понятия
- Объявление массивов
- Ввод и вывод массивов
- Заполнение массива случайными числами
- Поэлементная обработка массивов

Массивы

Массив – последовательность из фиксированного количества однотипных величин, имеющих общее имя и расположенных в памяти подряд.

Ключевые моменты:

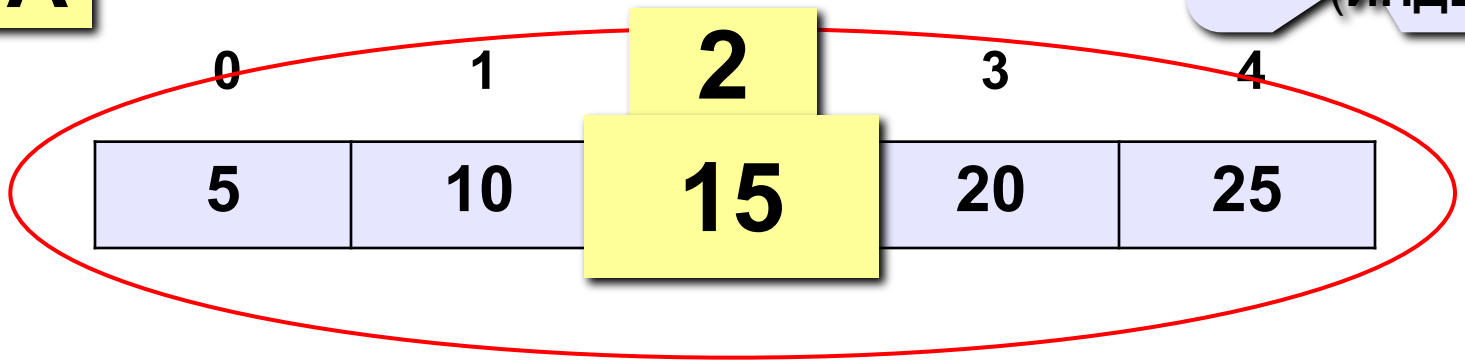
- все элементы имеют **один тип**
- количество элементов **фиксировано**
- весь массив имеет **одно имя**
- все элементы расположены в памяти **подряд**
- положение элемента определяется его **индексом**



Массивы

A **массив**

НОМЕР элемента массива (**ИНДЕКС**)



`A[0]` `A[1]` `A[4]`

ЗНАЧЕНИЕ элемента массива

ЗНАЧЕНИЕ элемента массива: 15

`A[2]`

НОМЕР (ИНДЕКС) элемента массива: 2

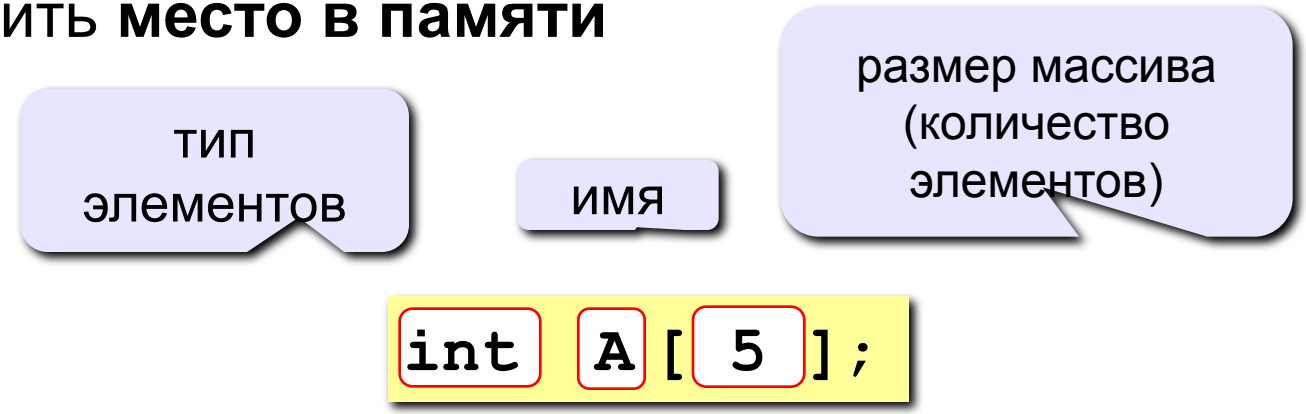
Нумерация элементов массива в Си начинается с НУЛЯ!

Объявление массивов

Зачем объявлять?

- определить **имя** массива
- определить **тип** массива
- определить **число элементов**
- выделить **место в памяти**

Пример:



Размер через константу:

```
const int N = 5;
int  A[ N ];
```

```
#define N 5
int  A[ N ];
```

Объявление массивов

Еще примеры:

```
int X[10], Y[10];
float zz, A[20];
char s[80];
```

С присвоением начальных значений:

```
int A[4] = { 8, -3, 4, 6 };
float B[2] = { 1. };
char C[3] = { 'A', '1', '\0' };
```

остальные нулевые!

Если начальные значения не заданы, в ячейках памяти находится «мусор»!

Что неправильно?

```
const int N = 10;
float A[N];
```

```
int X[4.5];
```

```
int A[10];
A[10] = 0;
```

выход за границы массива (стираются данные в памяти)

```
float X[5];
int n = 1;
X[n-2] = 4.5;
X[n+8] = 12.;
```

дробная часть отбрасывается (ошибки нет)

```
int X[4];
X[2] = 4.5;
```

```
float A[2] = { 1, 3.8 };
```

```
float B[2] = { 1., 3.8, 5.5 };
```

Массивы

Объявление:

```
const int N = 5;
int A[N], i;
```

Ввод с клавиатуры:

```
printf("Введите 5 элементов массива:\n");
for( i=0; i<N; i++ ) {
  printf ("A[%d] = ", i );
  scanf ("%d", &A[i] );
}
```

```
A[1] = 5
A[2] = 12
A[3] = 34
A[4] = 56
A[5] = 13
```

По

Вы

```
for( i=0; i<N; i++ ) A[i] =A[i]*2;
```

```
printf("Результат:\n");
for( i=0; i<N; i++ )
  printf("%4d", A[i]);
```

```
Результат:
  10   24   68 112   26
```

Заполнение случайными числами

```
#include <stdlib.h>   // случайные числа
```

RAND_MAX – максимальное случайное целое число (обычно RAND_MAX = 32767)

Случайное целое число в интервале [0,RAND_MAX]

```
x = rand(); // первое число
x = rand(); // уже другое число
```

Установить начальное значение последовательности:

```
srand ( 345 );    // начнем с 345
srand (clock()); // типичная инициализация
//или             // датчика случайных чисел.
srand (time(0)); // нужен time.h !
```

Целые числа в заданном интервале

Целые числа в интервале [0,N-1]:

```
int random(int N) {
  return rand()% N;
}
```

Примеры:

```
x = random ( 100 );   // интервал [0,99]
x = random ( z );     // интервал [0,z-1]
```

Целые числа в интервале [a,b]:

```
x = random ( z ) + a;    // интервал [a,z-1+a]
x = random (b – a + 1) + a; // интервал [a,b]
```

Заполнение случайными числами

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

int random(int N)
{ return rand() % N; }

void main()
{
  const int N = 10;
  int A[N], i;
  srand(clock());
  printf("Исходный массив:\n");
  for (i=0; i<N; i++) {
    A[i] = random(100) + 50;
    printf("%4d", A[i]);
  }
  ...
}
```

функция выдает случайное число от 0 до N-1

? Какой интервал?

Программа

Задача: ввести с клавиатуры массив из 5 элементов, умножить все элементы на 2 и вывести полученный массив на экран.

```
#include <stdio.h>

main()
{
  const int N = 5;
  int A[N], i;
  // ввод элементов массива
  // обработка массива
  // вывод результата
}
```

на предыдущих слайдах

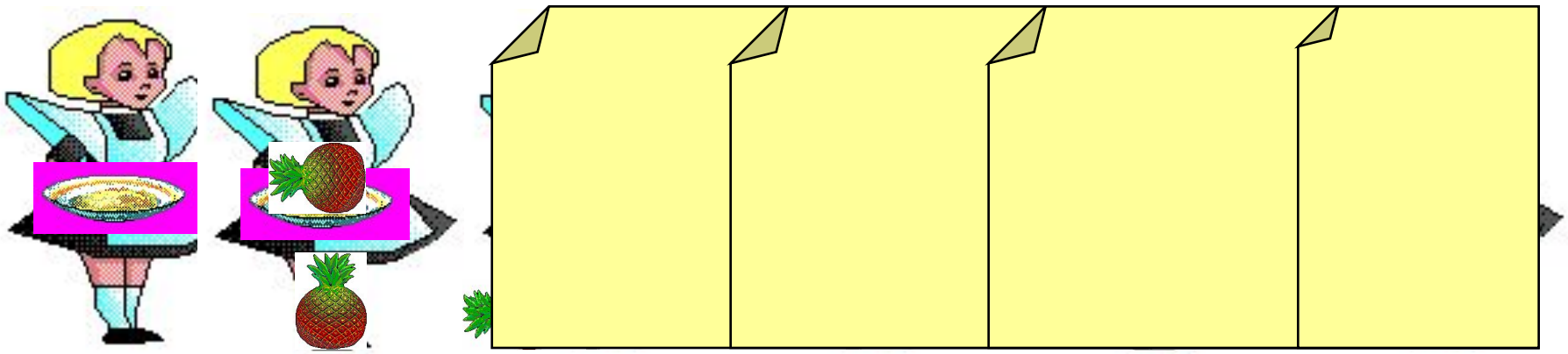
Массивы: типичные задачи

- Поиск максимального элемента
- Перестановка элементов
- Отбор элементов массива
- Линейный и двоичный поиск в массиве

Максимальный элемент

Задача: найти в массиве максимальный элемент.

Алгоритм:



Псевдокод:

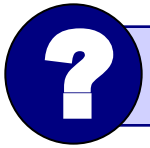
```
// считаем, что элемент A[0] – максимальный
for ( i=1; i < N; i++ )
  if ( A[i] > максимального )
    // запомнить новый максимальный элемент A[i]
```

Почему цикл от i=1?

Максимальный элемент

Дополнение: как найти номер максимального элемента?

```
                    // пока A[0]- максимальный
iMax = 0;
for ( i=1; i < N; i++ ) // проверяем остальные
  if ( A[i] > A[iMax] ) {   // нашли новый
                            // запомнить A[i]
    iMax = i;               // запомнить i
  }
```



Как упростить?

По номеру элемента **iMax** всегда можно найти его значение **A[iMax]**. Поэтому везде меняем **max** на **A[iMax]** и убираем переменную **max**.

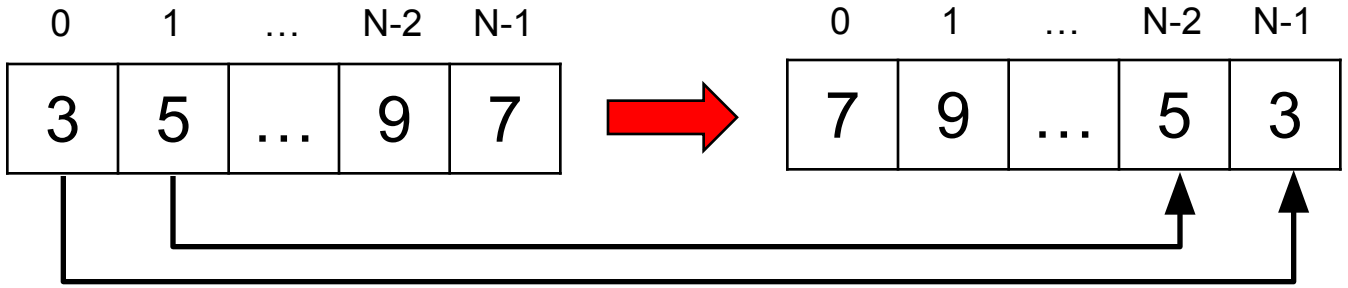
Программа

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  const int N = 5;
  int A[N], i, iMax;
  // заполнить случайными числами [100,150]
  // найти максимальный элемент и его номер
  printf("\nМаксимальный элемент A[%d] = %d",
         iMax, A[iMax]);
}
```

на предыдущих слайдах

Реверс массива

Задача: переставить элементы массива в обратном порядке (выполнить инверсию).



Алгоритм:

сумма индексов **N-1**

поменять местами **A[0]** и **A[N-1]**, **A[1]** и **A[N-2]**, **…**

Псевдокод:

```
for ( i = 0; i < N / 2 ; i++ )
  // поменять местами A[i] и A[N-1-i]
```

Что неверно?

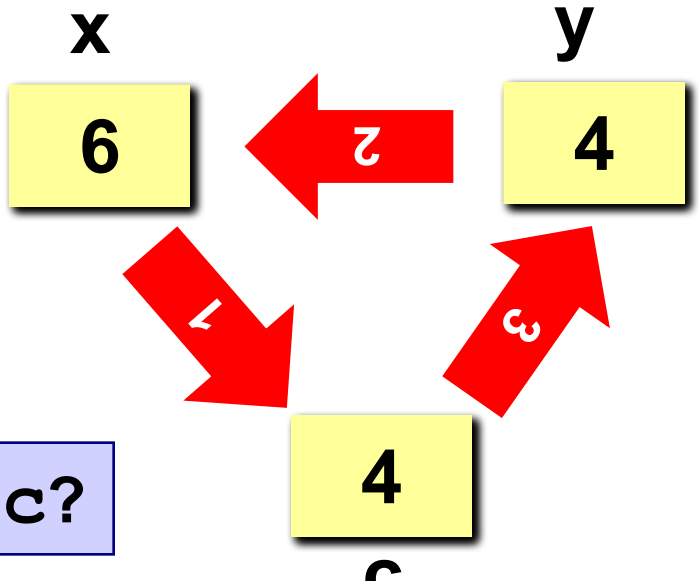
Как переставить элементы?

Задача: поменять местами содержимое двух чашек.

Задача: поменять местами содержимое двух ячеек памяти.

```
x = y;
y = x;
```

```
c = x;
x = y;
y = c;
```



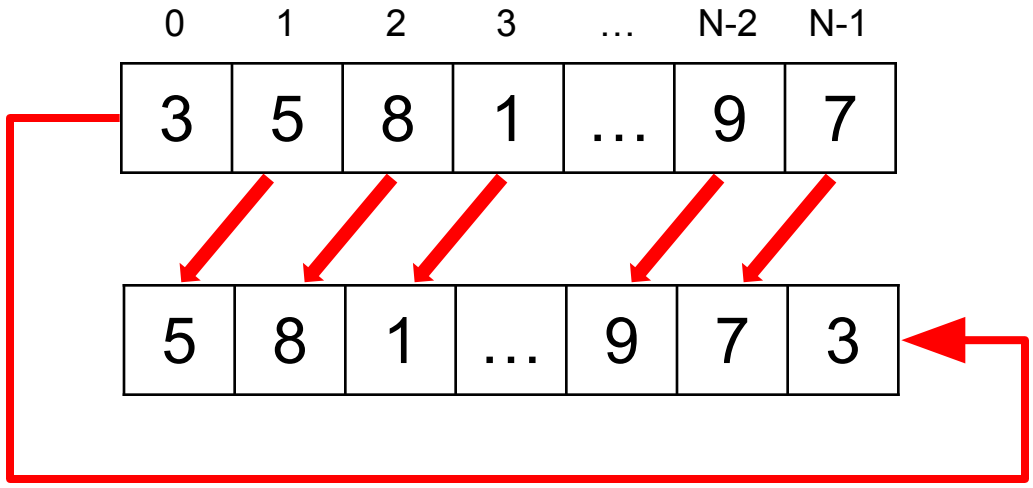
Можно ли обойтись без c?

Программа

```
main()
{
  const int N=10;
  int A[N], i, c;
  // заполнить массив
  // вывести исходный массив
  for (i=0; i<N/2; i++) {
    c=A[i];
    A[i]=A[N-1-i];
    A[N-1-i]=c;
    }
  // вывести полученный массив
}
```

Циклический сдвиг

Задача: сдвинуть элементы массива влево на 1 ячейку, первый элемент становится на место последнего.



Алгоритм:

```
A[0]=A[1]; A[1]=A[2];… A[N-2]=A[N-1];
```

Цикл:

почему не **N**?

```
for (i=0; i<N-1; i++)
  A[i]=A[i+1];
```

Что неверно?

Программа

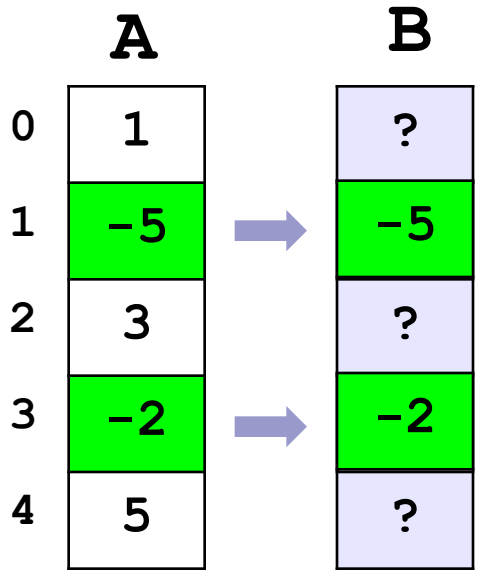
```
main()
{
  const int N = 10;
  int A[N], i, c;
  // заполнить массив
  // вывести исходный массив
  c=A[0];
  for(i=0; i<N-1; i++)
    A[i]=A[i+1];
  A[N-1]=c;
  // вывести полученный массив
}
```

Формирование массива по условию

Задача – найти в массиве элементы, удовлетворяющие некоторому условию (например, отрицательные), и скопировать их в другой массив.

Примитивное решение:

```
const int N = 5;
int A[N], B[N];
// здесь заполнить массив A
for( i=0; i< N; i++)
  if( A[i] < 0 ) B[i] = A[i];
```

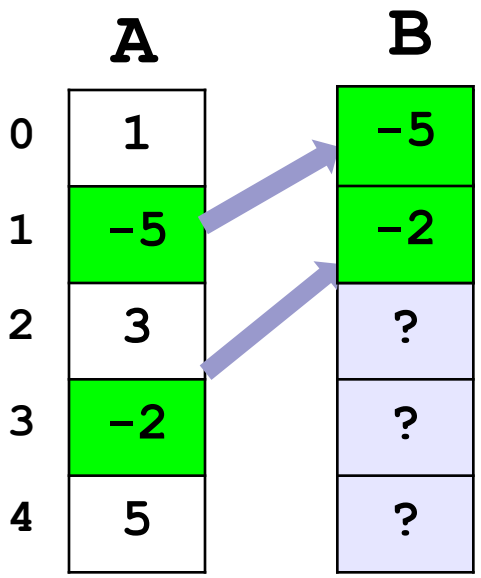


- выбранные элементы не рядом, не в начале массива
- непонятно, как с ними работать

Формирование массива по условию

Решение: ввести счетчик найденных элементов **count**, очередной элемент ставится на место **B[count]**.

```
int A[N], B[N], count = 0;
// здесь заполнить массив A
for( i = 0; i < N; i ++)
  if( A[i] < 0 ) {
    B[count] = A[i];
    count ++;
  }
// вывод массива B
for( i = 0; i < count; i ++)
  printf("%d\n", B[i]);
```



Поиск в массиве

Задача – найти в массиве элемент, равный **X**, или установить, что его нет.

Решение: для произвольного массива: **линейный поиск** (перебор)

недостаток: **низкая скорость**

Как ускорить? – заранее подготовить массив для поиска

- как именно подготовить?
- как использовать «подготовленный» массив?

Линейный поиск

nX – номер нужного элемента в массиве

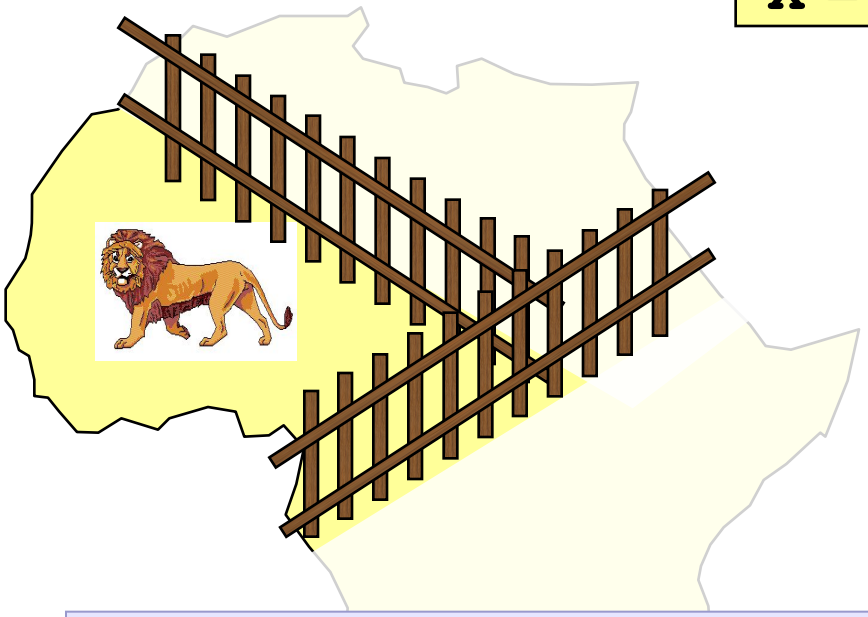
```
nX = -1; // пока не нашли ...
for ( i = 0; i < N; i ++) // цикл по всем элементам
  if ( A[i] == X )        // если нашли, то ...
     nX = i;              // ... запомнили номер
if (nX < 0) printf("Не нашли...")
else        printf("A[%d]=%d", nX, X);
```

Что можно улучшить?

Улучшение: после того, как нашли **X**, выходим из цикла.

```
nX = -1;
for ( i = 0; i < N; i ++)
 if ( A[i] == X ) {
    nX = i;
    break ;  //выход из цикла
 }
```

Двоичный поиск



X = 7

X < 8

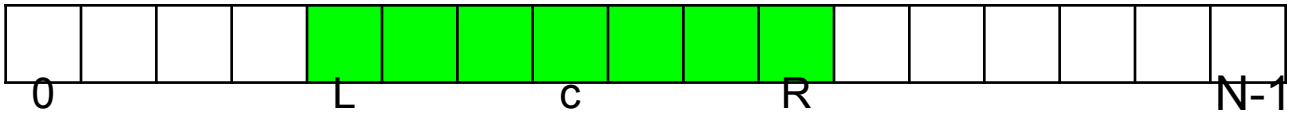
X > 4

X > 6

1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12
13	13	13
14	14	14
15	15	15
16	16	16

1. Выбрать средний элемент `A[c]` и сравнить с X.
2. Если `X = A[c]`, нашли (выход).
3. Если `X < A[c]`, искать дальше в первой половине.
4. Если `X > A[c]`, искать дальше во второй половине.

Двоичный поиск




```
nX = -1;
L = 0; R = N-1; // границы: ищем от A[0] до A[N-1]
while ( R >= L ){
  c = (R + L) / 2;
  if (X == A[c]) {
    nX = c;
    break;
    }
  if (X < A[c]) R = c - 1;
  if (X > A[c]) L = c + 1;
  }
if (nX < 0) printf("Не нашли...");
else        printf("A[%d]=%d", nX, X);
```

номер среднего элемента

если нашли …

выйти из цикла

сдвигаем границы

Почему нельзя `while ( R > L ) { … }` ?

Сравнение методов поиска

	Линейный	Двоичный
подготовка	нет	**отсортировать**
	число шагов	
N = 2	**2**	**2**
N = 16	**16**	**5**
N = 1024	**1024**	**11**
N= 1048576	**1048576**	**21**
N	**≤ N**	**≤ $\log_2 N+1$**

Упражнения

1. Написать программу, которая сортирует массив **ПО УБЫВАНИЮ** и ищет в нем элемент, равный X (это число вводится с клавиатуры). Использовать **двоичный поиск**.

2. Написать программу, которая считает **среднее число шагов в двоичном поиске** для массива из 32 элементов в интервале [0,100]. Для поиска использовать 1000 случайных чисел в этом же интервале.

Массивы как параметры функций

- Массивы в функциях
- Массивы как параметры

Массивы в функциях

Задача: составить процедуру, которая переставляет элементы массива в обратном порядке.

параметр-массив

размер массива

```
void Reverse ( int A[] , int N )
{
  int i, c;
  for ( i = 0; i < N/2; i ++ ) {
    c = A[i];
    A[i] = A[N-1-i];
    A[N-1-i] = c;
  }
}
```

Массивы как параметры функций

Особенности:

- при описании параметра-массива в заголовке функции его размер не указывается (функция работает с массивами **любого размера**)

Почему здесь размер не обязателен?

- размер массива надо передавать как отдельный параметр
- в процедура передается **адрес** исходного массива: все **изменения**, сделанные в процедуре **влияют** на массив в основной программе

Массивы в функциях

```
void Reverse ( int A[], int N )
{
...
}
main()
{
  int A[10];
    // здесь надо заполнить массив
  Reverse ( A, 10 );  // весь массив
  // Reverse ( A, 5 );   // первая половина
  // Reverse ( A+5, 5 ); // вторая половина
}
```

это адрес начала массива в памяти

A или **&A[0]**

A+5 или **&A[5]**

Упражнения

1. Написать функцию, которая сортирует массив по возрастанию, и показать пример ее использования.

2. Написать функцию, которая ставит в начало массива все четные элементы, а в конец – все нечетные.

Массивы в функциях

Задача: составить функцию, которая находит сумму элементов массива.

результат – целое число

параметр-массив

размер массива

```
int Sum ( int A[], int N )
{
int i, sum = 0;
for ( i = 0; i < N; i ++ )
   sum += A[i];
return sum;
}
```

Массивы в функциях

```
int Sum ( int A[], int N )
{
...
}
main()
{
  int A[10], sum, sum1, sum2;
    // заполнить массив
  sum  = Sum ( A, 10 );  // весь массив
  sum1 = Sum ( A, 5 );   // первая половина
  sum2 = Sum ( A+5, 5 ); // вторая половина
  ...
}
```

Упражнения

1. Написать функцию, которая находит максимальный элемент в массиве.

2. Написать логическую функцию, которая определяет, верно ли, что среди элементов массива есть два одинаковых. Если ответ «да», функция возвращает 1; если ответ «нет», то 0.

   Подсказка: для отладки удобно использовать массив из 5 элементов, задаваемых вручную:

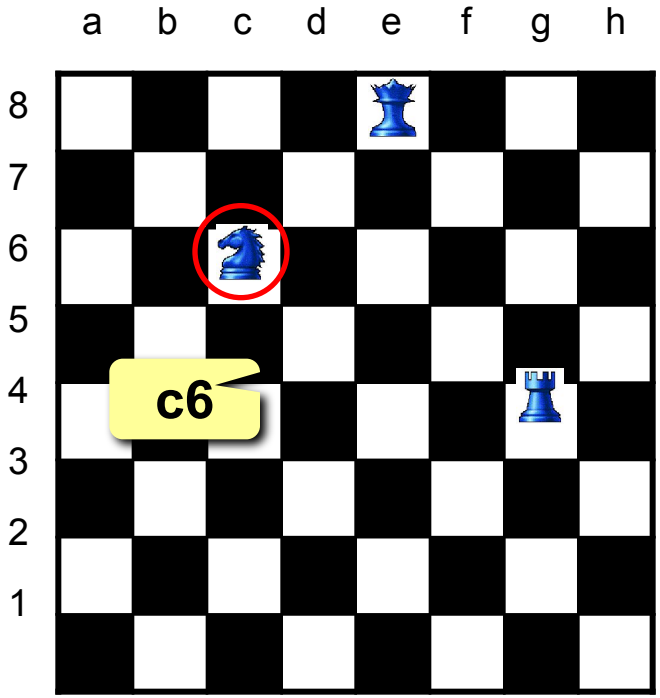
```
const int N = 5;
int A[N] = { 1, 2, 3, 3, 4
};
```

Двумерные массивы

- Двумерные массивы и матрицы
- Объявление двумерных массивов
- Ввод и вывод двумерных массивов
- Обработка двумерных массивов

Двумерные массивы (матрицы)

Задача: запомнить положение фигур на шахматной доске.



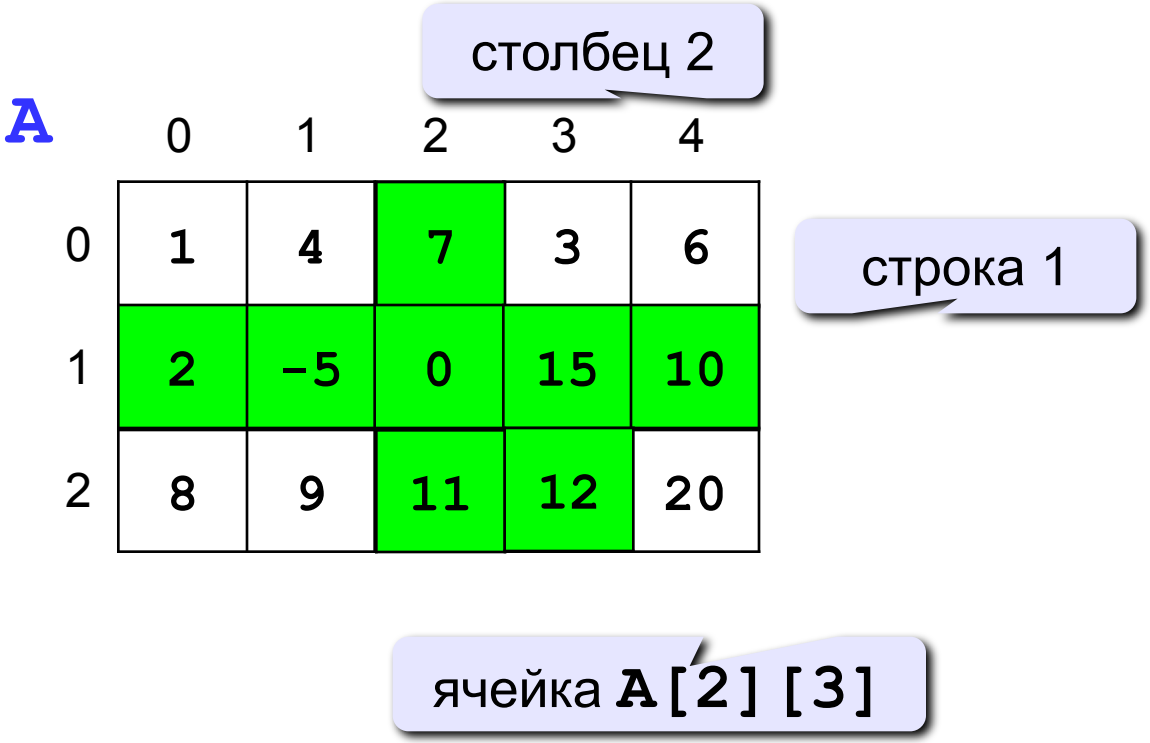
	0	1	2	3	4	5	6	7
7	0	0	0	0	**2**	0	0	0
6	0	0	0	0	0	0	0	0
5	0	0	**3**	0	0	0	0	0
4	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	**4**	0
2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

A[5][2]

Двумерные массивы (матрицы)

Матрица – это прямоугольная таблица однотипных элементов.

Матрица – это массив, в котором каждый элемент имеет два индекса (номер строки и номер столбца).



Двумерные массивы (матрицы)

Объявление:

```
const int N = 3, M = 4;
int A[N][M];
float a[2][2] = {{3.2, 4.3}, {1.1, 2.2}};
char sym[2][2] = { 'a', 'b', 'c', 'd' };
```

Ввод с клавиатуры:

```
for ( j = 0; j < M; j ++ )
  for ( i = 0; i < N; i ++ ) {
    printf ( "A[%d][%d]=", i, j);
    scanf ( "%d", &A[i][j] );
    }
```

i	j	
A[0][0	2	
A[0][1	5	
A[0][2	1	
]=	4	
A[2][3	5	
]=	4	

Если переставить циклы?

Двумерные массивы (матрицы)

Заполнение случайными числами

```
for (i=0; i<N; i++)
  for (j=0; j<M; j++)
    A[i][j] = random(25)-10;
```

цикл по строкам

цикл по столбцам

интервал?

Вывод на экран

```
for (i=0; i<N; i++) {
  for (j=0; j<M; j++)
    printf("%5d", A[i][j]);
  printf("\n");
  }
```

вывод строки

в той же строке

перейти на новую строку

```
   12    25     1    13
  156     1    12   447
    1   456   222    23
```

? Если переставить циклы?

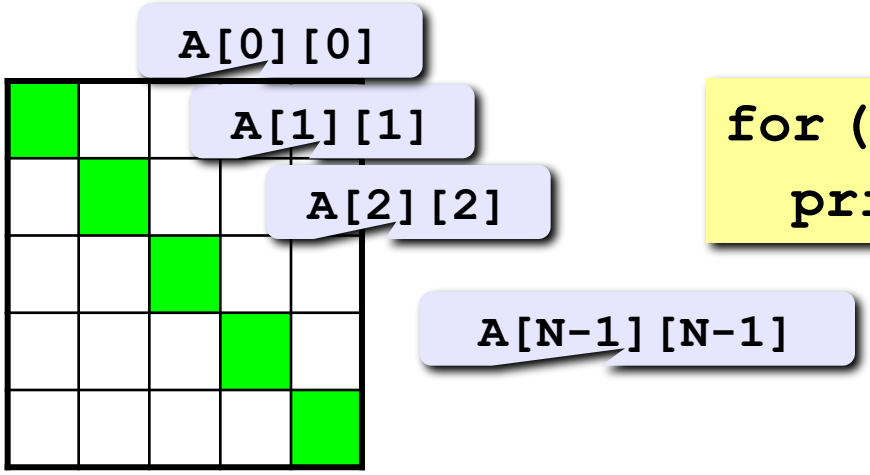
Обработка всех элементов матрицы

Задача: заполнить матрицу из 3 строк и 4 столбцов случайными числами и вывести ее на экран. Найти сумму элементов матрицы.

```
main()
{
  const int N=3, M=4;
  int A[N][M], i, j, S=0;
  ... // заполнение матрицы и вывод на экран
  for(i=0; i<N; i++)
    for(j=0; j<M; j++)
      S+=A[i][j];
  printf("Сумма элементов матрицы S=%d", S);
}
```

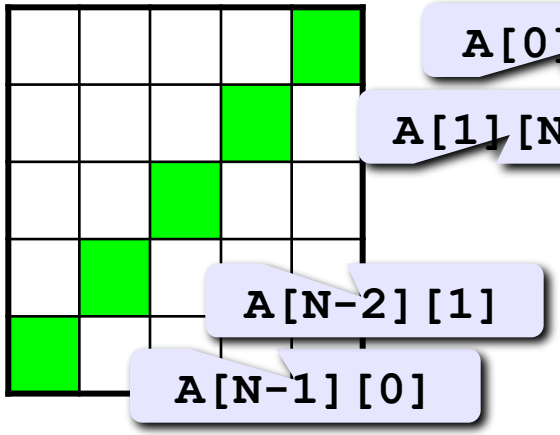
Операции с матрицами

Задача 1. Вывести на экран главную диагональ квадратной матрицы из N строк и N столбцов.




```
for ( i = 0; i < N; i ++ )
  printf ( "%5d", A[i][i] );
```

Задача 2. Вывести на экран вторую диагональ.

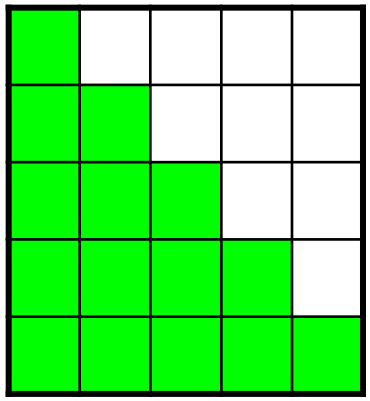


сумма номеров строки и столбца N-1

```
for ( i = 0; i < N; i ++)
  printf ( "%5d", A[i][ N-1-i ]);
```

Операции с матрицами

Задача 3. Найти сумму элементов, стоящих на главной диагонали и ниже ее.



Одиночный цикл или вложенный?

строка 0: `A[0][0]`

строка 1: `A[1][0]+A[1][1]`

`...`

строка i: `A[i][0]+A[i][2]+...+A[i][i]`

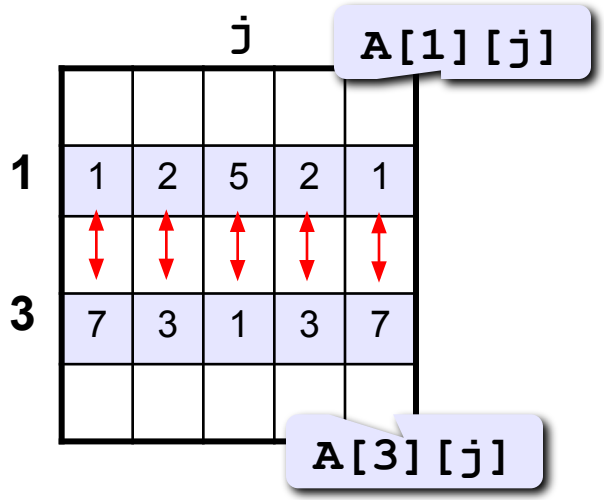
цикл по всем строкам

```
S = 0;
for ( i = 0; i < N; i ++ )
  for ( j = 0; j <= i; j ++
  )
    S += A[i][j];
```

складываем нужные элементы строки **i**

Операции с матрицами

Задача 4. Перестановка строк или столбцов. В матрице из N строк и M столбцов переставить 1-ую и 3-ю строки.




```
for ( j = 0; j <= M; j ++ ) {
  c = A[1][j];
  A[1][j] = A[3][j];
  A[3][j] = c;
  }
```

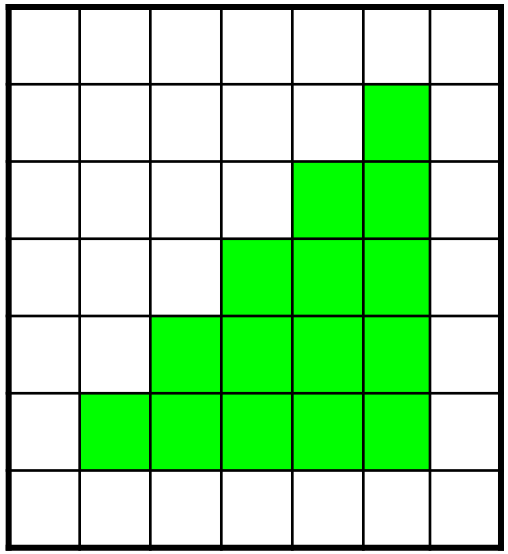
Задача 5. К третьему столбцу добавить шестой.

```
for ( i = 0; i < N; i ++ )
  A[i][3] += A[i][6];
```

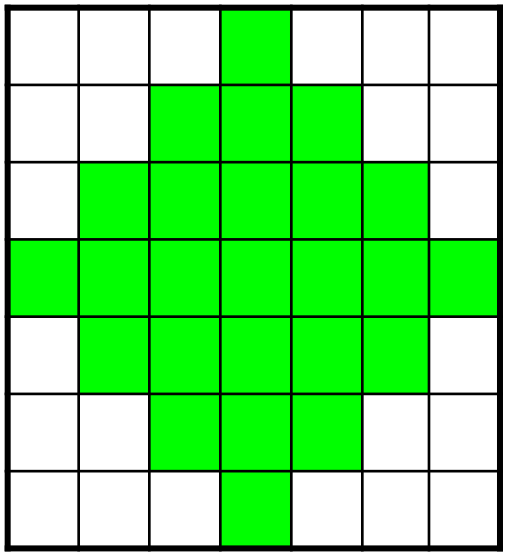
Задания

Заполнить матрицу из 7 строк и 7 столбцов случайными числами в интервале [-10,10] и вывести ее на экран. Обнулить элементы, отмеченные зеленым фоном, и вывести полученную матрицу на экран.

1.



2.



Вопросы?

- Массивы
  - Основные понятия
  - Объявление массивов
  - Ввод и вывод массивов
  - Заполнение массива случайными числами
  - Поэлементная обработка массивов
- Массивы: типичные задачи
  - Поиск максимального элемента
  - Перестановка элементов
  - Отбор элементов массива
  - Линейный и двоичный поиск в массиве
- Массивы как параметры функций
  - Массивы в функциях
  - Массивы как параметры
- Двумерные массивы
  - Двумерные массивы и матрицы
  - Объявление двумерных массивов
  - Ввод и вывод двумерных массивов
  - Обработка двумерных массивов

Дубовая роща. Незнакомка с расческой