ФГОБУ ВПО "СибГУТИ"
Кафедра вычислительных систем

ПРОГРАММИРОВАНИЕ
ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Массивы данных

Преподаватель:
Доцент Кафедры ВС, к.т.н.
Поляков Артем Юрьевич

Задача сортировки (sorting problem)

Дано: последовательность из n чисел $\langle a_1, a_2, a_3, \ldots, a_n \rangle$

Необходимо: переставить элементы последовательности так, чтобы для любых элементов новой последовательности $\langle a'_1, a'_2, a'_3, \ldots, a'_n \rangle$ выполнялось соотношение:

$$a'_1 \leq a'_2 \leq a'_3 \leq \ldots \leq a'_n$$

Пример:

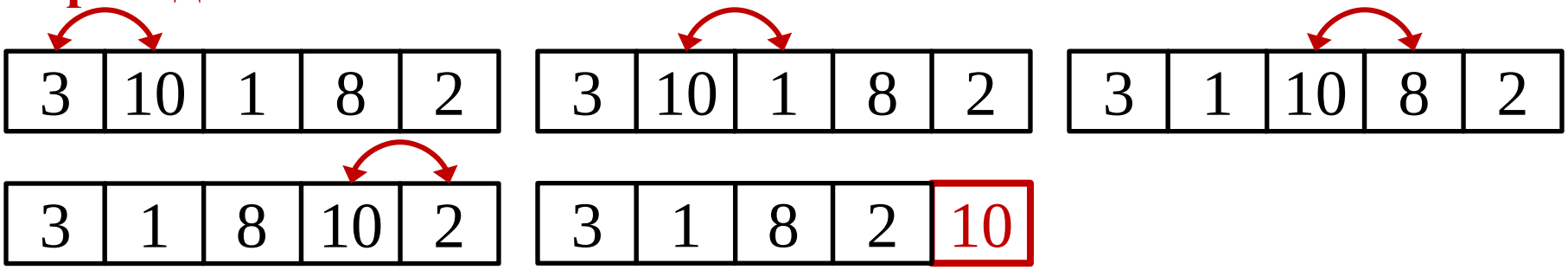
Входная последовательность: ‹5, 3, 8, 9, 4›

Выходная последовательность: ‹3, 4, 5, 8, 9›

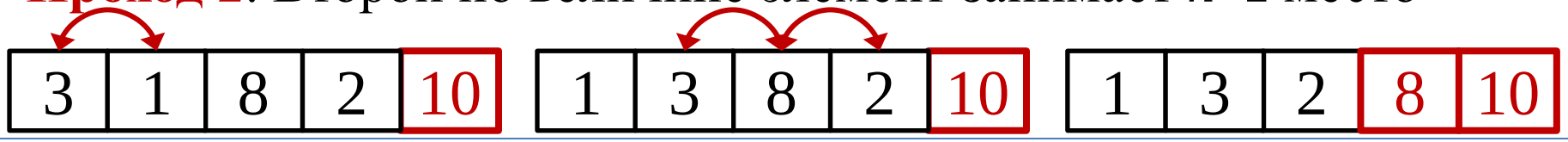
Входные данные (последовательность), удовлетворяющие всем заданным ограничениям задачи, называется **экземпляром задачи**.

Сортировка методом пузырька (пример)

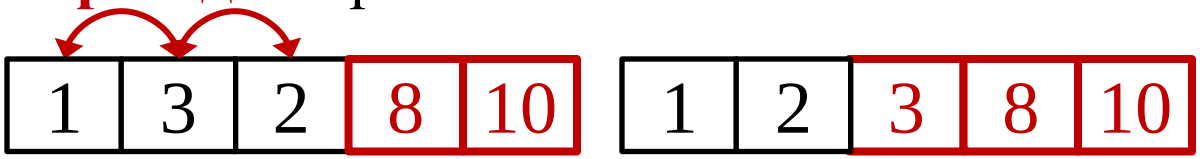
Проход 1. Наибольший элемент занимает *n*-е место



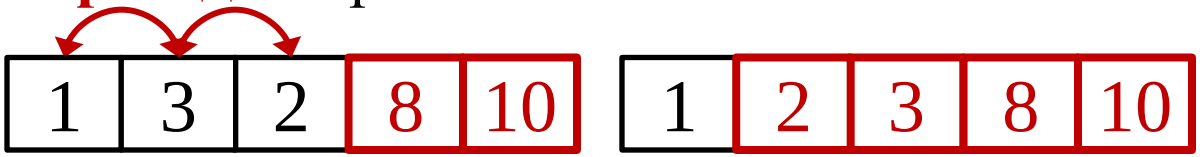
Проход 2. Второй по величине элемент занимает *n*–1 место



Проход 3. Третий по величине элемент занимает *n*–2 место



Проход 4. Третий по величине элемент занимает *n*–3 место



Сортировка методом пузырька (псевдокод)

Входные данные: последовательность a из n элементов

```
ввод a
i ← 1
while i ≤ n do
    j ← 1
    while j ≤ (n–i) do // На i-й итерации i правых эл-тов отсортированы
        if a_j > a_{j+1} then
            t ← a_j
            a_j ← a_{j+1}
            a_{j+1} ← t
        j ← j + 1
    i ← i + 1
```

Правила оформления псевдокода

1. Циклические конструкции обозначаются английскими словами (аналогично языку программирования СИ): **while**, **do-while**, **for.**

2. Конструкция ветвления обозначается ключевыми словами **if-then-else**.

3. Структура блоков указывается с помощью отступов.

4. Для описания комментариев используется '//'

5. Присваивание обозначается как '←': $x \leftarrow y$ для отличи от сравнения (обозначается как "=").

Программная реализация алгоритма сортировки методом пузырька

- Алгоритм поиска минимального значения в последовательности: **достаточно однократной обработки элемента** (сравнение с текущим минимумом-рекордом)
- Алгоритм сортировки методом пузырька: **многократное обращение к элементам в процессе работы**.

Для реализации алгоритма сортировки методом пузырька требуется хранение всех элементов сортируемой последовательности в памяти программы.

Массивы

- Массивы предназначены для хранения **наборов однотипных данных** в памяти программы.
- Доступ к конкретному элементу производится с использованием его **целочисленного индекса**.
- Индексы задают порядок следования элементов в массиве, поэтому его можно рассматривать как упорядоченное множество.
- Технически это последовательность однотипных переменных.
- В памяти элементы массива располагаются друг за другом непрерывно. Для каждого элемента справедливо, что **между элементами** с индексами i и i+1 **не может находиться** никаких **ячеек** данных.

Объявление массивов

При объявлении массива требуется следующая информация:

1. Тип данного составляющих его элементов.
2. Имя массива.
3. Количество элементов в массиве.
4. [Необязательно] начальное содержимое массива (инициализация).

Например: массив с именем m из N элементов типа int объявляется так:

```
int m[N];
```

Объявление массивов (инициализация)

При объявлении массива требуется следующая информация:

1. Тип данного составляющих его элементов.
2. Имя массива.
3. Количество элементов в массиве.
4. [Необязательно] начальное содержимое массива (инициализация).

Например: массив с именем *mas* из 5 элементов типа float, со значениями 1.1, 2.0, 3.5, 6.7, 8.1 объявляется так:

```
int mas[5] = {1.1, 2.0, 3.5, 6.7, 8.1};
```

или

```
int mas[] = {1.1, 2.0, 3.5, 6.7, 8.1};
```

Операция индексации

- В языке СИ не предусмотрено операций над всем массивом.
- Основная часть операций производится поэлементно.
- Доступ к конкретному элементу производится с использованием операции индексации:

<имя-массива>[<индекс>].

Например, для массива:

	0	1	2	3	4	
float mas[] = {	1.1	2.0	3.5	6.7	8.1	}

доступ к элементу со значением 3.5, расположенному 3-ем в массиве осуществляется следующим образом:

```
x = mas[2];
```

Индексация элементов в языке СИ начинается с **НУЛЯ**!

Индексы

- В качестве индекса может использоваться **любое выражение, целого типа**: char, short, int, long.
- Индексы элементов массива языка СИ **начинаются с 0.**
- Индексы могут быть:
  - положительными, тогда обращение производится к ячейке, располагающейся **после первой**.
  - отрицательными, тогда обращение производится к ячейке, располагающейся **перед первой**.

...	N[-2]	N[-1]	N[0]	N[1]	N[2]	. . .

Ввод массива

- В функции scanf не предусмотрено спецификатора для ввода массива как единого целого. Это сделано из соображений универсальности и сохранения относительной простоты использования.
- Для ввода массива необходимо выполнить чтение каждого из его элементов.
- При решении данной задачи удобно использовать циклы.

```
int mas[10], i;
for(i=0;i<10;i++){
    scanf("%d",&mas[i]);
}
```

Обработка массива

- К массиву как единому целому может быть применена только операция индексации [ ] и оператор sizeof.
- Любые другие действия требуют **поэлементной обработки**!
- Например, не предусмотрено операции присваивания массивов. Для достижения желаемого эффекта необходимо поэлементно присвоить каждому элементу изменяемого массива значение соответствующего элемента исходного:

```
int mas1[10] = {...}, mas2[10], i;
for(i=0;i<10;i++){
    mas2[i] = mas1[i];
}
```

Обработка массива (2)

- К массиву как единому целому может быть применена только операция индексации.
- Любые другие действия требуют **поэлементной обработки**!
- Например, программа вычисления суммы элементов массива выглядит следующим образом:

```
int mas[10], i, sum = 0;

// ввод массива с клавиатуры (слайд 12)
for(i=0;i<10;i++){
    sum += mas[i];
}
```

Перебор индексов массива

В задачах, рассмотренных выше, обработка массива сводилась к последовательному перебору его индексов и обработке соответствующего элемента согласно условиям задачи.

В связи с тем, что индексация массивов начинается с 0, в условии продолжения цикла используют **строгое** неравенство:

```
...
for(i = 0; i < 10; i++) { ... }
...
```

i: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ~~10~~

Особенности инициализации массива

- Статические массивы можно объявлять с инициализацией, перечисляя значения их элементов в {} через запятую. **Если задано меньше элементов, чем длина массива остальные элементы считаются нулями**:

```
int a[10] = { 1, 2, 3, 4 }; /* и 6 нулей */
```

- Если при описании массива с инициализацией не указать его размер, он будет подсчитан компилятором:

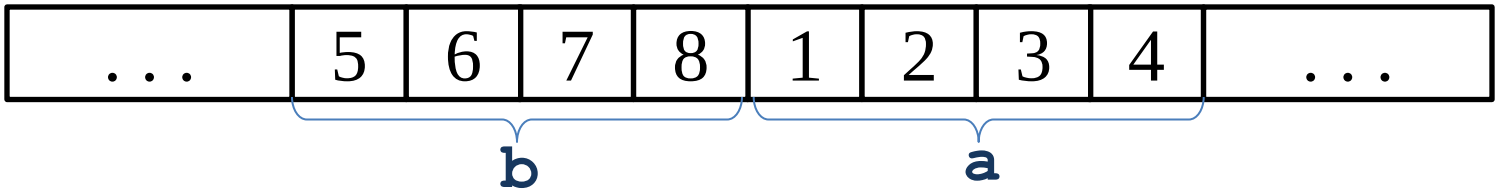
```
int b[] = { 1, 2, 3 };
```

Особенности реализации массивов в языке СИ

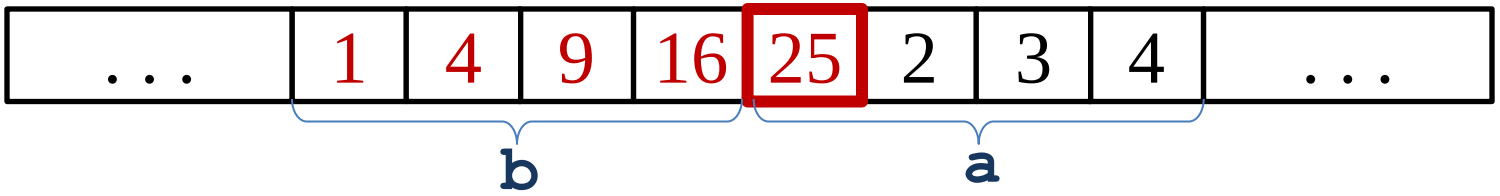
- «Си — инструмент, острый, как бритва: с его помощью можно создать и элегантную программу, и кровавое месиво», **Б. Керниган**.
- В связи со сравнительно низким уровнем языка многие случаи неправильного использования опасных элементов не обнаруживаются и не могут быть обнаружены ни при компиляции, ни во время исполнения.
- В Си **не предусмотрено средств проверки индексов** массивов (проверки выхода за границы массива).
- Например, возможна запись в шестой элемент массива из пяти элементов, что, естественно, приведёт к непредсказуемым результатам.

Выход за границы массива

```
int i, a[4] = {1,2,3,4}, b[4] = {5,6,7,8};
```




```
for(i=0; i<=4; i++){
    b[i] = (i+1)*(i+1);
}
```



Демонстрация выхода за границы массива

```
#include <stdio.h>
int main()
{
  int i, a[4] = {1,2,3,4}, b[] = {5,6,7,8};
  for(i=0;i<=4;i++)
    b[i] = (i+1)*(i+1);
  printf("a: ");
  for(i=0;i<4;i++)
    printf("%d ",a[i]);
  printf(" b: ");
  for(i=0;i<4;i++)
    printf("%d ",b[i]);
  printf("\n");
}
```

```
$ ./array_borders
a: 25 2 3 4  b: 1 4 9 16
```

Применение GDB

```
(gdb) b main
(gdb) r
. . .
(gdb) watch a[0]
Hardware watchpoint 5: a[0]
(gdb) c
Continuing.
Hardware watchpoint 5: a[0]
Old value = -1073745688
New value = 25
main () at array_borders.c:6
5   for(i=0;i<=4;i++)
(gdb) inspect &b[i]
$5 = (int *) 0xbffff0cc
(gdb) inspect &a[0]
$6 = (int *) 0xbffff0cc
```

```
5   for(i=0;i<=4;i++)
6     b[i] = (i+1)*(i+1);
7   printf("a: ");
```

Многомерные массивы

- в языке СИ определены **только одномерные массивы;**
- элементом массива в свою очередь может быть массив.

Правила 1 и 2 позволяют определять многомерные массивы следующим образом:

```
int m1[2][5], m2[4][10][10];
float m3[9][9][9][9];
```

Для обращения к элементу массива необходимо зафиксировать индекс **каждого измерения**, например:

```
m1[1][3] = 8; m1[1][5] = 10; m[2][4] = 1;
m3[5][4][2][6] = 1.6;
```

Логическое и физическое размещение многомерных массивов

Логическое расположение элементов массива в памяти

Строки \ Столбцы	0	1	2
0	N[0][0]	N[0][1]	N[0][2]
1	N[1][0]	N[1][1]	N[1][2]
2	N[2][0]	N[2][1]	N[2][2]

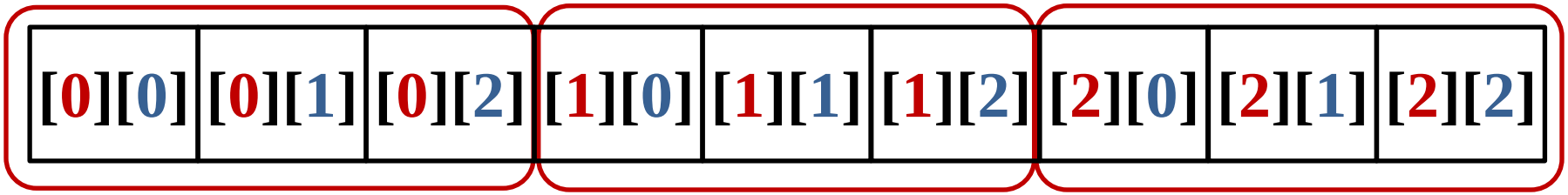
Физическое расположение элементов массива в памяти

[0][0]	[0][1]	[0][2]	[1][0]	[1][1]	[1][2]	[2][0]	[2][1]	[2][2]

Адресация двумерных массивов

Физическое расположение элементов массива в памяти

```
int A[3][3];
```



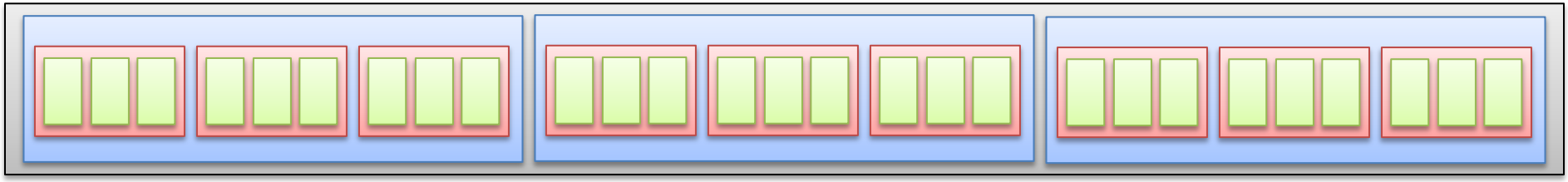
Задача определения смещения элемента в физическом представлении:
Дано: индексы (i, j) элемента двумерного массива $A[N][M]$.
Найти: функцию смещения $t(i, j)$ элемента $A[i][j]$ в физическом представлении A относительно первого элемента $A[0][0]$.
Решение:

$$t(i, j) = i \cdot M + j$$

Адресация трехмерных массивов

Физическое расположение элементов массива в памяти

```
int A[3][3][3];
```



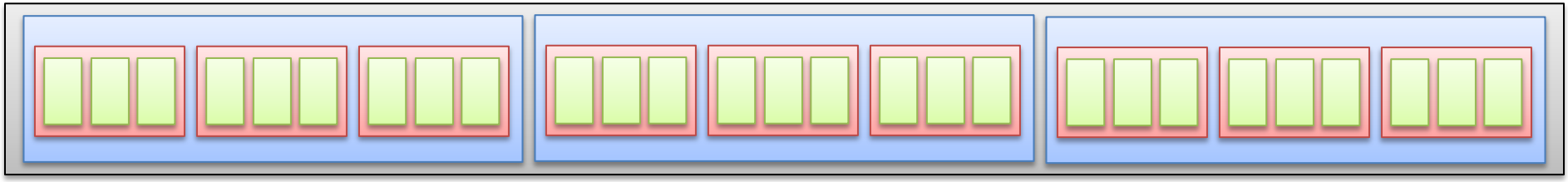
Задача определения смещения элемента в физическом представлении:
Дано: индексы (i, j, k) элемента массива $A[N][M][K]$.
Найти: функцию смещения $t(i, j, k)$ элемента $A[i][j][k]$ в физическом представлении A относительно первого элемента $A[0][0][0]$.
Решение:

? ? ? ?

Адресация трехмерных массивов

Физическое расположение элементов массива в памяти

```
int A[3][3][3];
```



Задача определения смещения элемента в физическом представлении:
Дано: индексы (i, j, k) элемента массива $A[N][M][K]$.
Найти: функцию смещения $t(i, j, k)$ элемента $A[i][j][k]$ в физическом представлении A относительно первого элемента $A[0][0][0]$.
Решение:

$$t(i, j, k) = i \cdot M \cdot K + j \cdot K + k$$

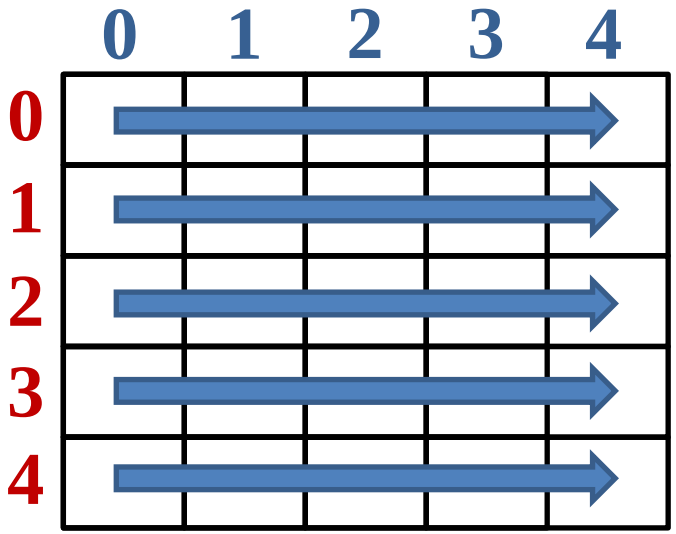
Обработка двумерного массива

Строки \ Столбцы	0	1	2
0	N[0][0]	N[0][1]	N[0][2]
1	N[1][0]	N[1][1]	N[1][2]
2	N[2][0]	N[2][1]	N[2][2]

- Элемент двумерного массива описывается **двумя индексами**.
- Обработка массива предусматривает просмотр/изменение **каждого элемента**.
- **Последовательность** обработки элементов **определяется задачей**.
- Для перебора индексов массива обычно используют **вложенные циклы**, каждый цикл отвечает за изменение своего индекса.

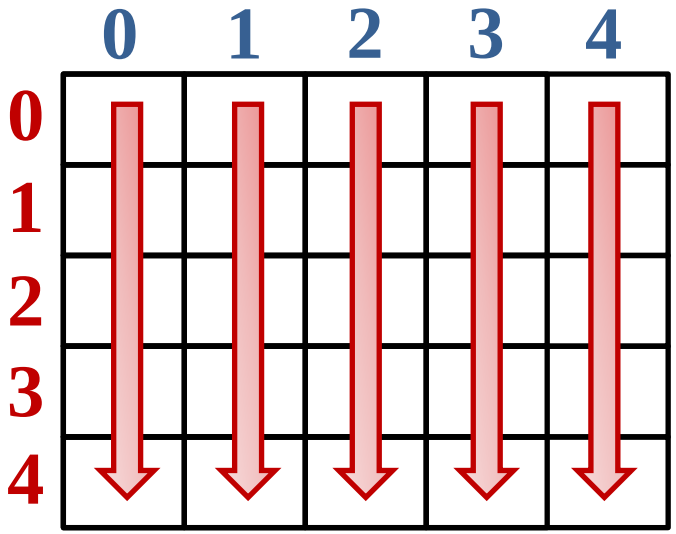
Построчный просмотр элементов

```
int a[5][5] = {{1},{4},{2},{3},{8}};
int i, j;
for(i = 0; i < 5; i++){
  for(j = 0; j < 5; j++)
        printf("%d ", a[i][j]);
}
```



Просмотр элементов по столбцам

```
int a[5][5] = {{1},{4},{2},{3},{8}};
int i, j;
for(j = 0; j < 5; j++){
  for(i = 0; i < 5; i++)
        printf("%d ", a[i][j]);
}
```



Произведение матриц

$$A \times B = C$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$c_{11} = a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31}$$

$$\dots$$

$$c_{23} = a_{21} b_{13} + a_{22} b_{23} + a_{23} b_{33}$$

$$\dots$$

Порядок обработки элементов

Программа вычисления c_{ij}

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$k \leftarrow 0$

$c_{ij} \leftarrow 0$

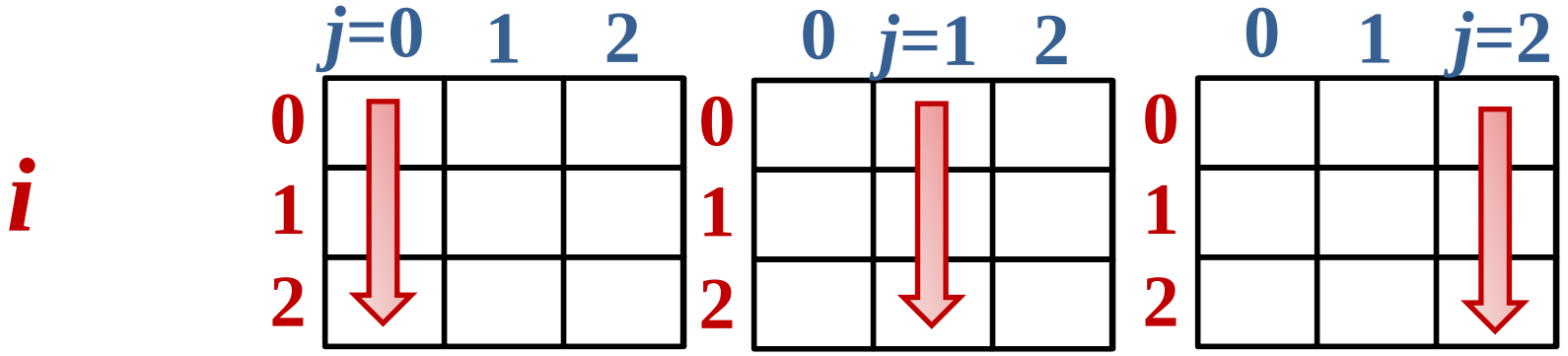
while $k < n$ **do**

$c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$

$k \leftarrow k + 1$

Программа вычисления строки

$$c_i = (c_{i1}, c_{i2}, c_{i3}, \ldots c_{in})$$



$j \leftarrow 0$

while $j < n$ **do**

$k \leftarrow 0$

$c_{ij} \leftarrow 0$

while $k < n$ **do**

$c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$

$k \leftarrow k + 1$

$j \leftarrow j + 1$

Программа вычисления всей матрицы C

$i \leftarrow 0$

while $i < n$ **do**

$j \leftarrow 0$

while $j < n$ **do**

$k \leftarrow 0$

$c_{ij} \leftarrow 0$

while $k < n$ **do**

$c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$

$k \leftarrow k + 1$

$j \leftarrow j + 1$

$i \leftarrow i + 1$

Инициализация двумерных массивов

- Для инициализации статических многомерных массивов необходимо инициализировать *каждый вложенный массив*, являющийся элементом внешнего:

```
int a[2][3] = { {1, 2, 3}, {4, 5, 6} };
```

Строки \ Столбцы	0	1	2
0	1	2	3
1	4	5	6

Инициализация двумерных массивов (2)

- Если задано меньше элементов, чем длина массива остальные элементы **заполняются нулями**:

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

	0	1	2
0	1	0	0
1	2	0	0

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

	0	1	2
0	1	0	0
1	2	0	0
2	0	0	0

	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0

Сечение многомерных массивов

$$b[n_1][n_2][n_3]...[n_k]$$

Фиксация (обязательно слева направо) l первых индексов массива b приведет к формированию сечения массива.

Сечение массива представляет собой массив b' меньшей (по сравнению с b) размерности, элементы которого образуются по следующему правилу:

$$b'[i_1][i_2]\ldots[i_{(k-l)}] = b[c_1][c_2]\ldots[c_l]\,[i_1][i_2]\ldots[i_{(k-l)}]\,,$$

где $c_1, \ldots c_l$ – зафиксированные индексы, а

$i_1, i_2, \ldots i_{(k-l)}$ – свободные индексы.

Обращение к элементу – частный случай сечения

Сечение двумерных массивов

```
int b[3][3]={ {1,2,3},{4,5,6},{7,8,9} }
```

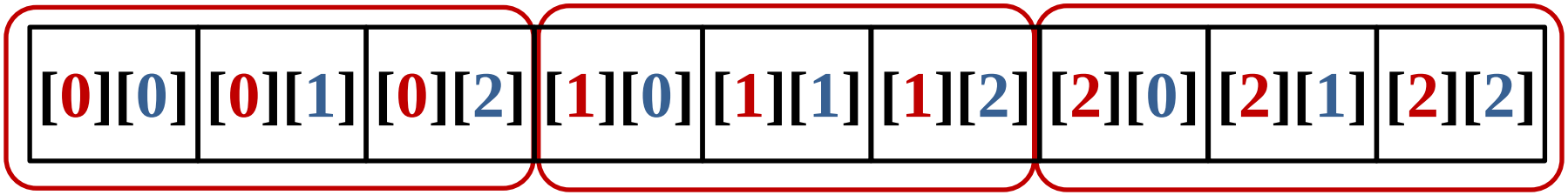
	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

b[0]	1	2	3
b[1]	4	5	6
b[2]	7	8	9

Расположение двумерных массивов в памяти программы

Физическое расположение элементов массива в памяти

```
int A[3][3];
```



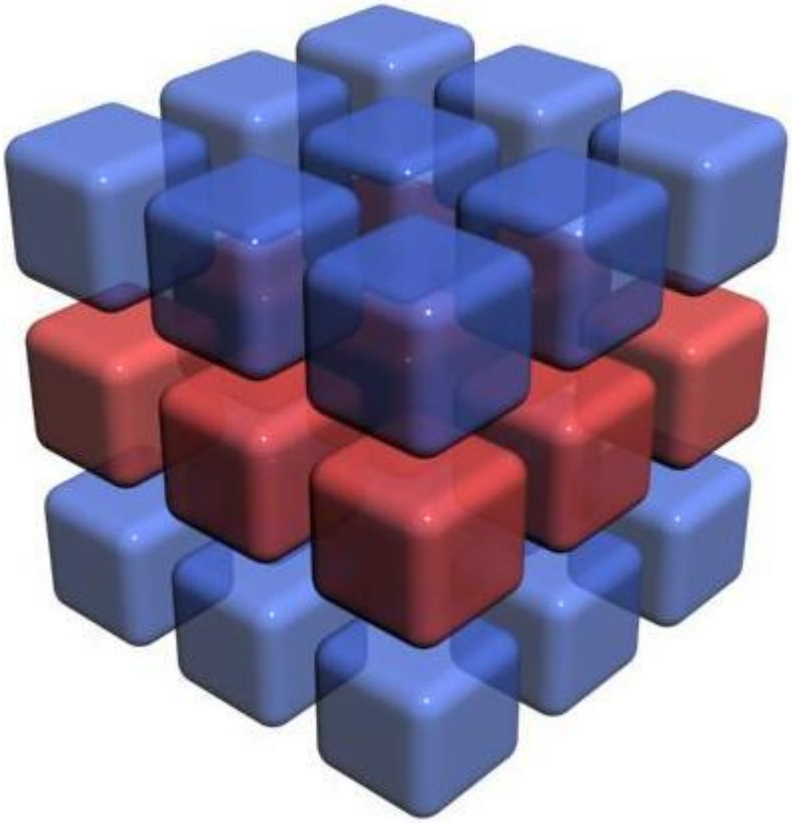
Задача определения смещения элемента в физическом представлении:
Дано: индексы (i, j) элемента двумерного массива $A[N][M]$.
Найти: функцию смещения $t(i, j)$ элемента $A[i][j]$ в физическом представлении A относительно первого элемента $A[0][0]$.
Решение:

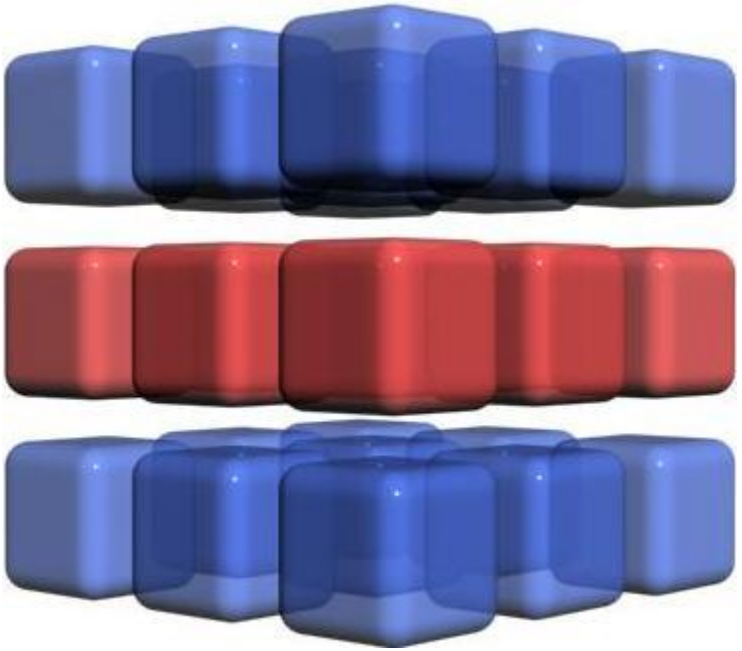
$$t(i, j) = i \cdot M + j$$

Сечение трехмерных массивов

```
int b[3][3][3];
```

b[1]





Оператор **sizeof**

```
//     z  y  x
int a[5][5][5];
int size, xysize, xsize, elemsize;
int cnt, xcnt, ycnt, zcnt;
size = sizeof(a);
xysize = sizeof(a[0]);
xsize = sizeof(a[0][0]);
elemsize = sizeof(a[0][0][0]));
cnt = sizeof(a)/sizeof(a[0][0][0]);
zcnt = sizeof(a)/sizeof(a[0]);
ycnt = sizeof(a[0])/sizeof(a[0][0]);
xcnt = sizeof(a[0][0])/sizeof(a[0][0][0]);
```

СПАСИБО ЗА ВНИМАНИЕ!