

Алгоритмы обработки массивов. Решение задач по обработке массивов исходных данных: подсчет количества элементов, обладающих заданным свойством, поиск минимальных и максимальных значений, поиск подпоследовательностей.

Заполнение массива

Ввод данных в массив может осуществляться множеством способов. **Все зависит от задачи.**

1. Ввод элементов вручную с клавиатуры
2. **Заполнение массива случайными числами**
3. Заполнение массива в процессе вычислений
4. **Чтение элементов массива из файла**
5. Элементы массива могут поступать через порт с внешнего устройства (???)

Заполнение массива случайными числами

Функция стандартной библиотеки языка C: `rand()`

`int rand(void)`; - генерирует псевдослучайное на интервале 0
RAND_MAX (= 32767, но зависит от реализации языка) .

Числа `0...9`

`rand()%10`

Числа `1...9`

`rand()%10 + 1`

Последовательность
псевдослучайная, но всегда
одинаковая !!!

Исправим ситуацию:

`void srand (unsigned int seed);`

`time_t time (time_t* timer);`

`srand(time(NULL));`

Заполнение массива случайными числами

```
//от 1 до 10
```

```
srand((unsigned)time(NULL));  
for (int i = 0; i < n; i++ ) A[i] = rand()%10+1;
```

```
//от -20 до 20
```

```
srand((unsigned)time(NULL));  
for (int i = 0; i < n; i++ ) A[i] = rand() % 41-20;
```

```
//на отрезке от a до b
```

```
srand(time(NULL)*1000);  
for (int i=0; i<n; i++)  
{  
A[i]=(rand()*1.0/(RAND_MAX)*(b-a)+a);  
}
```

(См. Example3, Project FillingArray)

Заполнение массива числами из файла

Библиотека `fstream` (`#include <fstream>`)

`ifstream` - для чтения данных из потока

`eof()` - end of file

```
//создать поток для чтения данных из файла input.txt
ifstream in("input.txt");
//проверка существования файла
if (!in)
{
    cout << "File not found\n";
    return 1;
}

//чтение чисел из файла и их запись в массив
for(int i=0; !in.eof() && i < n; i++) in>>A[i];
```

(См. Example3, Project FillingArray)

Операции с большими числами (длинная арифметика)

Задача. Найти произведение длинного целого числа (451095723598) на цифру

1. Будем хранить каждую цифру числа в массиве

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 5 | 1 | 0 | 9 | 5 | 7 | 2 | 3 | 5 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|

2. Умножим каждую цифру числа на цифру (например на 3)

| | | | | | | | | | | | |
|----|----|---|---|----|----|----|---|---|----|----|----|
| 12 | 15 | 3 | 0 | 27 | 15 | 21 | 6 | 9 | 15 | 27 | 24 |
|----|----|---|---|----|----|----|---|---|----|----|----|

3. Организуем перенос: в каждой ячейке оставляем младшую цифру хранящегося там числа, а старшую суммируем с числом в левой ячейке

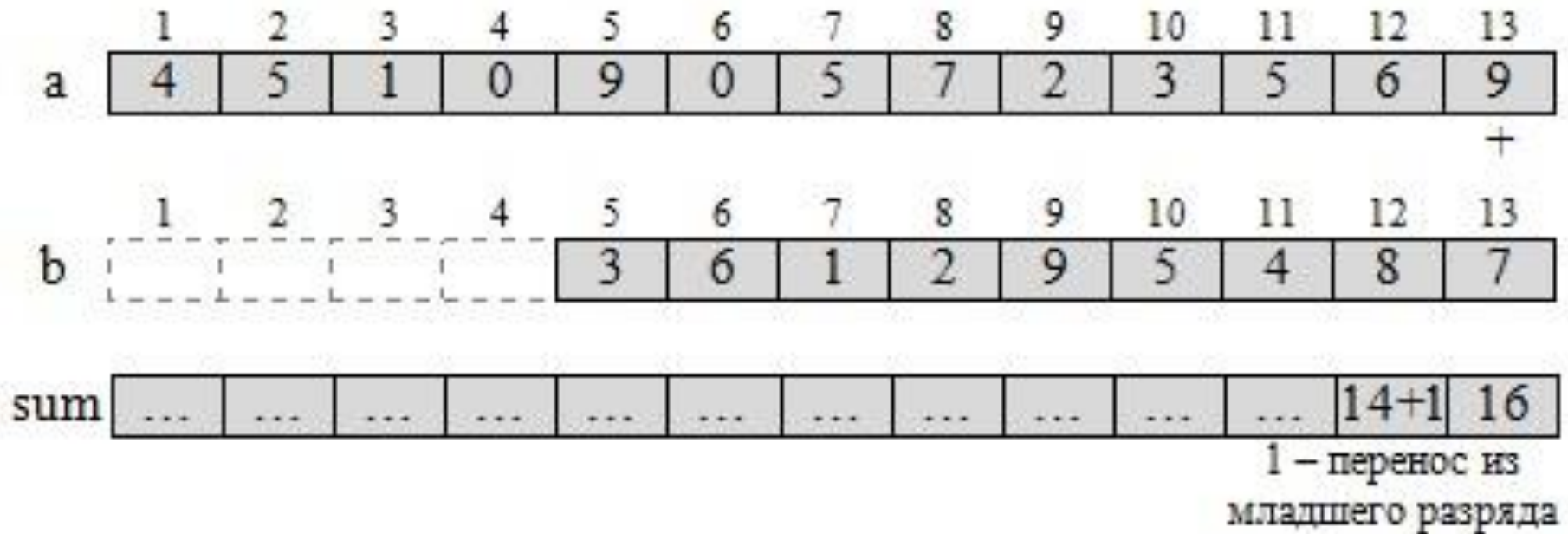
| | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---------|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | 27+2=29 | 4 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---------|---|



(см. Example3, Project BigInt)

Операции с большими числами (длинная арифметика)

Задача. Найти сумму двух длинных целых чисел



(см. Example3, Project BigInt)

Двумерные массивы

| | Column 0 | Column 1 | Column 2 | Column 3 |
|-------|-------------|-------------|-------------|-------------|
| Row 0 | a[0][0] | a[0][1] | a[0][2] | a[0][3] |
| Row 1 | a[1][0] | a[1][1] | a[1][2] | a[1][3] |
| Row 2 | a[2][0] | a[2][1] | a[2][2] | a[2][3] |

```
int array2D[3][3]={10,20,30,  
40,50,60,  
70,80,90};
```

| | | | |
|-----|-----|-----|-----|
| [0] | 10 | 20 | 30 |
| [1] | 40 | 50 | 60 |
| [2] | 70 | 80 | 90 |
| | [0] | [1] | [2] |

Column subscript
Row subscript
Array name

(см. Example3, Project TwoDimArray)

Двумерные массивы. Обработка.

| i/j | 0 | 1 | 2 | 3 |
|-----|-------|-------|-------|-------|
| 0 | (0,0) | (0,1) | (0,2) | (0,3) |
| 1 | (1,0) | (1,1) | (1,2) | (1,3) |
| 2 | (2,0) | (2,1) | (2,2) | (2,3) |
| 3 | (3,0) | (3,1) | (3,2) | (3,3) |
| 4 | (4,0) | (4,1) | (4,2) | (4,3) |

$i=1..n$

$j=1..m$

//1. Заполнение (построчно, слева направо)

```
for (int i = 0; i < n; i++)  
    for (int j = 0; j < m; j++)  
        a[i][j] = something;
```

Двумерные массивы. Обработка.

| i/j | 0 | 1 | 2 | 3 |
|-----|-------|-------|-------|-------|
| 0 | (0,0) | (0,1) | (0,2) | (0,3) |
| 1 | (1,0) | (1,1) | (1,2) | (1,3) |
| 2 | (2,0) | (2,1) | (2,2) | (2,3) |
| 3 | (3,0) | (3,1) | (3,2) | (3,3) |
| 4 | (4,0) | (4,1) | (4,2) | (4,3) |

$i=1..n$

$j=1..m$

//2. Сумма всех элементов

```
int sum=0;
for (int i = 0; i<n; i++)
    for(int j = 0; j<m; j++)
        sum += a[i][j];
```

Двумерные массивы. Обработка.

| i/j | 0 | 1 | 2 | 3 |
|-----|-------|-------|-------|-------|
| 0 | (0,0) | (0,1) | (0,2) | (0,3) |
| 1 | (1,0) | (1,1) | (1,2) | (1,3) |
| 2 | (2,0) | (2,1) | (2,2) | (2,3) |
| 3 | (3,0) | (3,1) | (3,2) | (3,3) |
| 4 | (4,0) | (4,1) | (4,2) | (4,3) |

$i=1..n$

$j=1..m$

//3. Сумма элементов строки

```
int sum=0;
for(int j = 0; j<m; j++)
    sum += a[3][j];
```

Двумерные массивы. Обработка.

| i/j | 0 | 1 | 2 | 3 |
|-----|-------|-------|-------|-------|
| 0 | (0,0) | (0,1) | (0,2) | (0,3) |
| 1 | (1,0) | (1,1) | (1,2) | (1,3) |
| 2 | (2,0) | (2,1) | (2,2) | (2,3) |
| 3 | (3,0) | (3,1) | (3,2) | (3,3) |
| 4 | (4,0) | (4,1) | (4,2) | (4,3) |

$i=1..n$

$j=1..m$

//4. Сумма элементов столбца

```
int sum=0;
for(int i = 0; i<n; i++)
    sum += a[i][3];
```

Двумерные массивы. Обработка.

| i/j | 0 | 1 | 2 | 3 |
|-----|-------|-------|-------|-------|
| 0 | (0,0) | (0,1) | (0,2) | (0,3) |
| 1 | (1,0) | (1,1) | (1,2) | (1,3) |
| 2 | (2,0) | (2,1) | (2,2) | (2,3) |
| 3 | (3,0) | (3,1) | (3,2) | (3,3) |
| 4 | (4,0) | (4,1) | (4,2) | (4,3) |

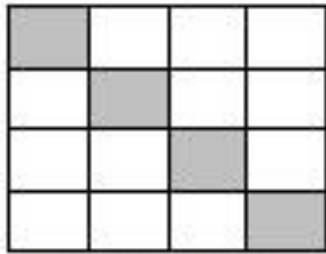
$i=1..n$

$j=1..m$

//4. Максимальный в каждом столбце

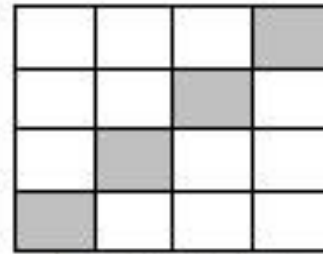
```
int max[4];
for(int j = 0; j<m; j++)
{
    max[j]=max[j,0];
    for(int i=1;i<n;i++)
        if (a[i][j] > max[j]) max[j]=a[i][j];
}
```

Обработка относительно диагоналей



Главная
диагональ:

$$i=j$$



Побочная
диагональ:

$$i=n-j+1$$

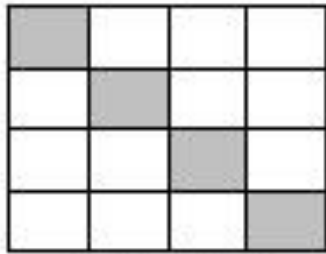
//Сумма элементов главной диагонали

```
for(int i = 0; i<n; i++) sum+=a[i][i];
```

//Сумма элементов выше главной диагонали

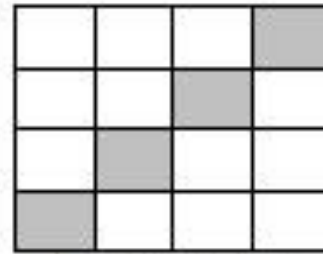
```
for(int i = 0; i<n; i++)  
    for(int j = 0; i<n; j++)  
        if (i < j) sum+=a[i][j];
```

Обработка относительно диагоналей



Главная
диагональ:

$$i=j$$



Побочная
диагональ:

$$i=n-j+1$$

//Сумма элементов побочной диагонали

```
for(int i = 0; i<n; i++) sum+=a[i][n-i-1];
```

//Сумма элементов выше главной диагонали

```
for(int i = 0; i<n; i++)  
    for(int j = 0; i<n; j++)  
        if (i < j) sum+=a[i][j];
```

| 1. Ниже и на главной диагонали | 2. Выше и на главной диагонали | 3. Выше и на побочной диагонали | 4. Ниже и на побочной диагонали |
|--|--|--|--|
| 100000 110000 111000 111100 111110 111111 | 111111 011111 001111 000111 000011 000001 | 111111 111110 111100 111000 110000 100000 | 000001 000011 000111 001111 011111 111111 |

```
1. for (int i=0;i<n;i++)
    for (int j=0; j<=i; j++) a[i][j]=1;
```

```
2. for (int i=0;i<n;i++)
    for (int j=i; j<n; j++) a[i][j]=1;
```

```
3. for (int i=0;i<n;i++)
    for (int j=0; j < n-i; j++) a[i][j]=1;
```

```
4. for (int i=0;i<n;i++)
    for (int j=(n-i-1); j < n; j++) a[i][j]=1;
```


informatics.msk.ru

| № гр | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---------------------------|---|---|---|---|---|---|---|---|---|
| Одномерные массивы | G | O | E | A | B | C | D | I | F |
| Двухмерные массивы | A | B | C | D | E | F | G | H | K |