



ВЫСШАЯ ШКОЛА ЭКОНОМИКИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Департамент программной инженерии
Алгоритмы и структуры данных

Семинар №3. 2021-2022 учебный год

Нестеров Роман Александрович, ДПИ ФКН и НУЛ ПОИС

Бессмертный Александр Игоревич, ДПИ ФКН

План



- Объявление и инициализация массива
- Массивы в памяти и указатели
- Передача массива в функцию
- Задачи, в которых используются массивы
- Многомерные и динамические массивы
- Реализация класса динамического массива


Объявление и инициализация массива



Структуры vs. массивы

```
struct Employee {  
    int age;  
    double salary;  
    string name  
}
```

```
struct TestRes {  
    int resSt1;  
    int resSt2;  
    int resSt3;  
    ...  
    int resSt265;  
}
```



Использование структуры для хранения большого количества однотипных данных **крайне неудобно**

Структуры vs. массивы



Массив – совокупный тип данных, который позволяет получать доступ ко всем переменным одного типа через один и тот же идентификатор.

```
int resStudents[265];
```

Элементы массива индексируются с **0**.

Размер фиксированного массива



```
int array[265];  
  
const int size = 265;  
  
int size1;  
cin >> size1;  
int array1[size1];  
  
int x = 265;  
const int size2 = x;  
int array2[size2];
```

Размер фиксированного массива

```
int array[265];  
  
const int size = 265;
```

```
int size1;  
cin >> size1;  
int array1[size1];
```

```
int x = 265;  
const int size2 = x;  
int array2[size2];
```

Ошибка компиляции!

Ошибка компиляции!

Инициализация массива

```
int array1[19] = {-45, 36, -989};
```

```
int array2[64] = { };
```

```
int array3[] = {1, 1, 0, 1, 1};
```


Инициализация массива

```
int array1[19] = {-45, 36, -989};  
  
int array2[64] = { };  
  
int array3[] = {1, 1, 0, 1, 1};
```

```
const n = 50;  
int array4[n] = { };  
  
for (size_t i = 0; i < n; i++) {  
    array[i] = ...;  
}
```

Использование перечислений для индексации

```
enum Students {  
    IVAN,  
    ELLIJAH,  
    ANTON,  
    IGOR,  
    MAX_STUDENTS  
}
```

```
int resStudents[MAX_STUDENTS] = { };  
  
resStudents[ANTON] = 95;  
resStudents[IGOR] = 97;  
...
```

Классы перечислений для индексации использовать не получится.

Массивы в памяти и указатели



Выделение памяти для массива

```
const int n = 150;  
int array[n] = { };  
  
cout << sizeof(array);  
600 (150 * sizeof(int))
```

Выделение памяти для массива

```
const int n = 150;  
int array[n] = { };  
  
cout << sizeof(array);  
600 (150 * sizeof(int))
```

```
int array[n] = {1, 1, 0, 1, 1, 0, 1};  
cout << sizeof(array) / sizeof(array[0]);  
7
```

Расположение массива в памяти

<ИМЯ_массива> хранит адрес первого элемента массива.

```
const int n = 150;
int array[n] = {-1453, 225, 54, -3698, -7 };

cout << array << ' ' << &array[0] << '\n';
cout << *array;
```

00BAF750 00BAF750
-1453

Расположение массива в памяти

<ИМЯ_массива> хранит адрес первого элемента массива.

```
const int n = 150;
int array[n] = {-1453, 225, 54, -3698, -7 };

cout << array << ' ' << &array[0] << '\n';
cout << *array;
```

00BAF750 00BAF750
-1453

<ИМЯ_массива> **не является указателем!**

Расположение массива в памяти

Элементы массива хранятся в смежных ячейках памяти.

```
const int n = 3;  
int array[n] = {-1453, 225, 54};  
  
for (int i = 0; i < n; i++) {  
    cout << &array[i] << ' ';  
}
```

00B3F714 00B3F718 00B3F71C

Память		

array[0]	00B3F714	-1453
array[1]	00B3F718	225
array[2]	00B3F71C	54

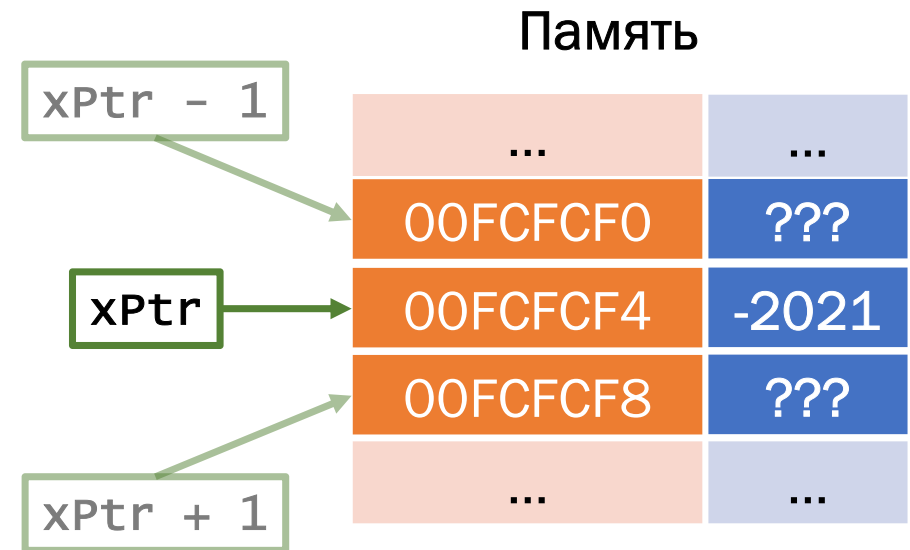
Расположение массива в памяти

Указатели **допускают** выполнение арифметических операций над собой.

```
int x = -2021;  
int *xPtr = &x;
```

```
cout << xPtr << '\n';  
cout << xPtr + 1 << '\n';  
cout << xPtr - 1 << '\n';
```

```
00FCFCF4 00FCFCF8 00FCFCF0
```



Расположение массива в памяти

Итерация по массиву с помощью указателя

```
int numUnits = 0;
int array[] = { 1, 1, 0, 1, 1, 0, 1 };

int *last = array1 + sizeof(array) / sizeof(array[0]);

for (int *ptr = array1; ptr < last ; ++ptr)
{
    if (*ptr == 1) {
        numUnits = numUnits + 1;
    }
}
```

Передача массивов в функции



От массива остается только указатель...

```
#include <iostream>

void func(int array[]) {
    std::cout << sizeof(array);
}

int main() {
    int array[] = {1, 1, 0, 1, 1};
    func(array);
    std::cout << ' ' << sizeof(array);
}
```

4 20

От массива остается только указатель...

Передача размера массива в функцию

```
#include <iostream>

void func(int array[], int n) {
    for (int i = 0; i < n; i++) {
        array[i] = array[i] * 3;
    }
}

int main() {
    int array[] = {1, 1, 0, 1, 1};
    func(array, sizeof(array) / sizeof(array[0]));
    std::cout << array[3];
}

3
```

От массива остается только указатель...

Передача размера массива в функцию

```
#include <iostream>

void func(int *array, int n) {
    for (int i = 0; i < n; i++) {
        array[i] = array[i] * 3;
    }
}

int main() {
    int array[] = {1, 1, 0, 1, 1};
    func(array, sizeof(array) / sizeof(array[0]));
    std::cout << array[3];
}

3
```

Задачи, в которых используются массивы

- Сортировка элементов
- Вычисление порядковых статистик
- Поиск элемента(-ов) в отсортированном/неотсортированном массиве
- ...
- Интервальные запросы
- ...

Задачи, в которых используются массивы

- Сортировка элементов
- Вычисление порядковых статистик
- Поиск элемента(-ов) в отсортированном/неотсортированном массиве
- ...
- Интервальные запросы
- ...

Интервальные запросы к массиву



Дан целочисленный массив размера N .

Поступает большое число запросов вида:

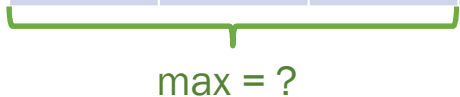
- Сумма элементов на некотором интервале $[L, R]$
- Поиск минимального/максимального элемента на интервале $[L, R]$
- Изменение всех элементов на интервале $[L, R]$
- ...

Интервальные запросы к массиву

Дан целочисленный массив размера N .
Поступает большое число запросов вида:

- Сумма элементов на некотором интервале $[L, R]$
- Поиск минимального/максимального элемента на интервале $[L, R]$
- Изменение всех элементов на интервале $[L, R]$
- ...

0	1	2	3	4	5	6	...	N-1	N
567	-33	0	-1	-9	7	555	...	0	-2

max = ?

Интервальные запросы к массиву

Дан целочисленный массив размера N .
Поступает большое число запросов вида:

- Сумма элементов на некотором интервале $[L, R]$
- Поиск минимального/максимального элемента на интервале $[L, R]$
- Изменение всех элементов на интервале $[L, R]$
- ...

0	1	2	3	4	5	6	...	N-1	N
567	-33	0	-1	-9	7	555	...	0	-2

max = ?

min = ?

Интервальные запросы к массиву

Дан целочисленный массив размера N .
Поступает большое число запросов вида:

- Сумма элементов на некотором интервале $[L, R]$
- Поиск минимального/максимального элемента на интервале $[L, R]$
- Изменение всех элементов на интервале $[L, R]$
- ...

0	1	2	3	4	5	6	...	N-1	N
567	-33	0	-1	-9	7	555	...	0	-2

max = ? *2 sum = ?

Корневая декомпозиция. Суммы на интервалах

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8	7	7	3
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Корневая декомпозиция. Суммы на интервалах

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

блоки																
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8	7	7	3
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Корневая декомпозиция. Суммы на интервалах

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

Предварительно вычисляем сумму для блоков.

блоки	7				5				17				9			
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8	7	7	3
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Корневая декомпозиция. Суммы на интервалах

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

Предварительно вычисляем сумму для блоков.

$$\text{sum}(4, 11) = ?$$

блоки	7				5				17				9			
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8	7	7	3
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Корневая декомпозиция. Суммы на интервалах

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

Предварительно вычисляем сумму для блоков.

$$\text{sum}(4, 11) = 5 + 17 = 22$$

блоки	7				5				17				9			
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8	7	7	3
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Корневая декомпозиция. Суммы на интервалах

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

Предварительно вычисляем сумму для блоков.

$$\text{sum}(4, 11) = 5 + 17 = 21$$

$$\text{sum}(6, 13) = ?$$

блоки	7				5				17				9			
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8	7	7	3
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Корневая декомпозиция. Суммы на интервалах

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

Предварительно вычисляем сумму для блоков.

$$\text{sum}(4, 11) = 5 + 17 = 22$$

$$\text{sum}(6, 13) = 1 + 0 + 17 - 8 + 7 = 17$$

блоки	7				5				17				9			
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8	7	7	3
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Корневая декомпозиция. Суммы на интервалах

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

Предварительно вычисляем сумму для блоков.

$$\text{sum}(4, 11) = 5 + 17 = 22$$

$$\text{sum}(6, 13) = 1 + 0 + 17 - 8 + 7 = 17$$

...

блоки	7				4				17				9			
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8	7	7	3
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Корневая декомпозиция. Интервальное изменение

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

В него будем записывать изменения значений.

блоки													
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12

Корневая декомпозиция. Интервальное изменение

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

В него будем записывать изменения значений.

$\text{ch}(0, 3, -6)$

блоки	0				0				0				0
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12

Корневая декомпозиция. Интервальное изменение

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

В него будем записывать изменения значений.

ch(0, 3, -6)

ch(4, 7, 2)

блоки	-6				0				0				0
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12

Корневая декомпозиция. Интервальное изменение

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

В него будем записывать изменения значений.

ch(0, 3, -6)

ch(4, 7, 2)

блоки	-6				2				0				0
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12

Корневая декомпозиция. Интервальное изменение

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

В него будем записывать изменения значений.

ch(0, 3, -6)

ch(4, 7, 2)

ch(4, 10, -3)

блоки	-6				2				0				0
массив	5	-5	4	3	2	2	1	0	0	-12	14	15	-8
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12

Корневая декомпозиция. Интервальное изменение

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

В него будем записывать изменения значений.

$\text{ch}(0, 3, -6)$

$\text{ch}(4, 7, 2)$

$\text{ch}(4, 10, -3)$

блоки	-6				-1				0				0
массив	5	-5	4	3	2	2	1	0	-3	-15	11	15	-8
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12

Корневая декомпозиция. Интервальное изменение

Исходный массив имеет размер N .

Создаем дополнительный массив размера $\text{ceil}(\sqrt{N})$.

В него будем записывать изменения значений.

$\text{ch}(0, 3, -6)$

$$\text{arr}[0] = 5 - 6 = -1$$

$\text{ch}(4, 7, 2)$

$$\text{arr}[6] = 1 - 1 = 0$$

$\text{ch}(4, 10, -3)$

$$\text{arr}[8] = -3 + 0 = 0$$

блоки	-6				-1				0				0
массив	5	-5	4	3	2	2	1	0	-3	-15	11	15	-8
индексы	0	1	2	3	4	5	6	7	8	9	10	11	12

Цикл for по диапазону

Итерация по фиксированному массиву



```
int testResults[] = {100, 95, 43, 67, 77, 69};  
for (int mark : testResults) {  
    cout << mark << ' ';  
}
```

Итерация по фиксированному массиву

```
int testResults[] = {100, 95, 43, 67, 77, 69};  
  
for (int mark : testResults) {  
    cout << mark << ' ';  
}
```

- Происходит копирование значения `testResults[i]` в `mark`

Итерация по фиксированному массиву

```
int testResults[] = {100, 95, 43, 67, 77, 69};  
  
for (auto mark : testResults) {  
    cout << mark << ' ';  
}
```

- Происходит копирование значения `testResults[i]` в `mark`
- Удобно использовать `auto`

Итерация по фиксированному массиву



```
int testResults[] = {100, 95, 43, 67, 77, 69};  
  
for (int &mark : testResults) {  
    mark += 5;  
}
```

- Использование ссылки для изменения значений элементов

Итерация по фиксированному массиву

```
int testResults[] = {100, 95, 43, 67, 77, 69};  
  
for (const int &mark : testResults) {  
    cout << mark << ' ';  
}
```

- Использование ссылки для изменения значений элементов
- Использование константной ссылки для доступа “read-only”

Итерация по фиксированному массиву



Цикл `for` по диапазону

- Не допускаются ошибки индексирования (вне диапазона)
- Нет возможности получить индекс элемента
- Нельзя использовать с указателями (=динамическим массивами и при передаче в функцию)

Динамические и многомерные массивы



Динамический массив. new[] и delete[]

```
int size;  
cin >> size;  
  
int *array = new int[size] {-5, 6, 7, 9};  
  
int *arrayD = new int[] {1, 0, 1, 1, -1};
```

- `array == &array[0]`
- `array + x == &array[0 + x]`

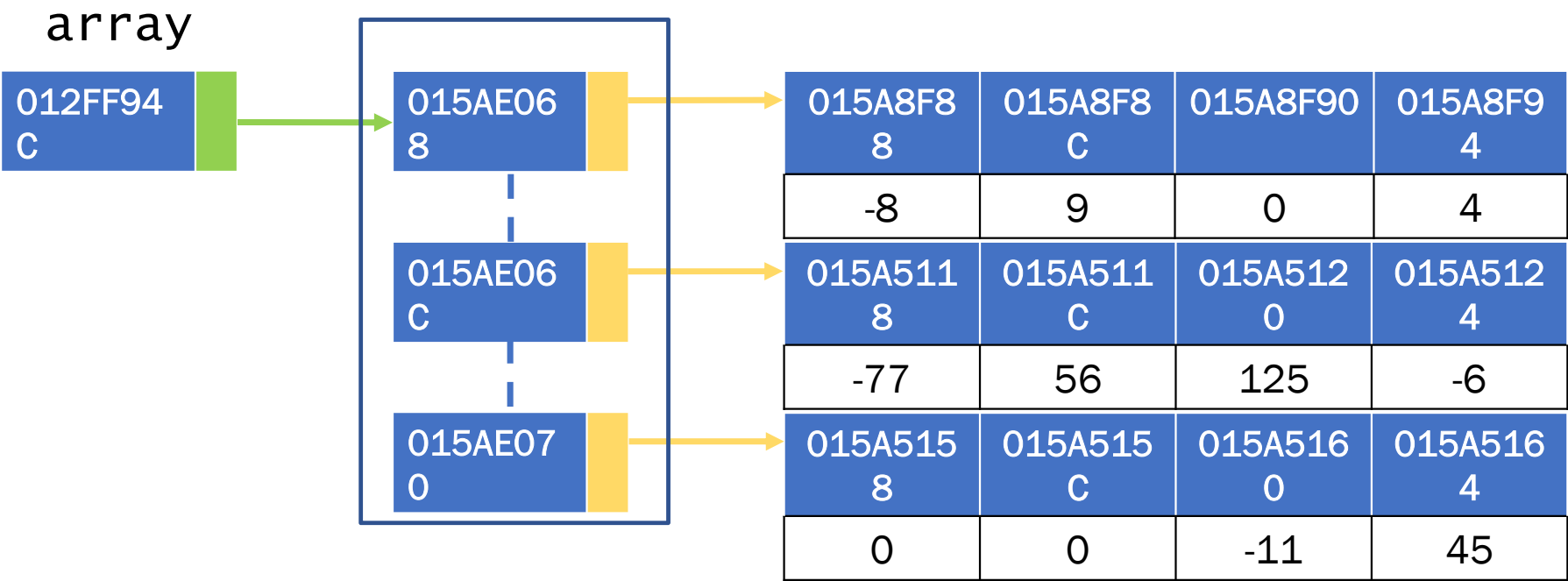
Динамический массив. new[] и delete[]

```
int size;  
cin >> size;  
  
int *array = new int[size] {-5, 6, 7, 9};  
  
int *arrayD = new int[] {1, 0, 1, 1, -1};  
  
...;  
  
delete[] array;  
array = nullptr;
```

Сколько операций нужно выполнить, чтобы изменить длину динамического массива?

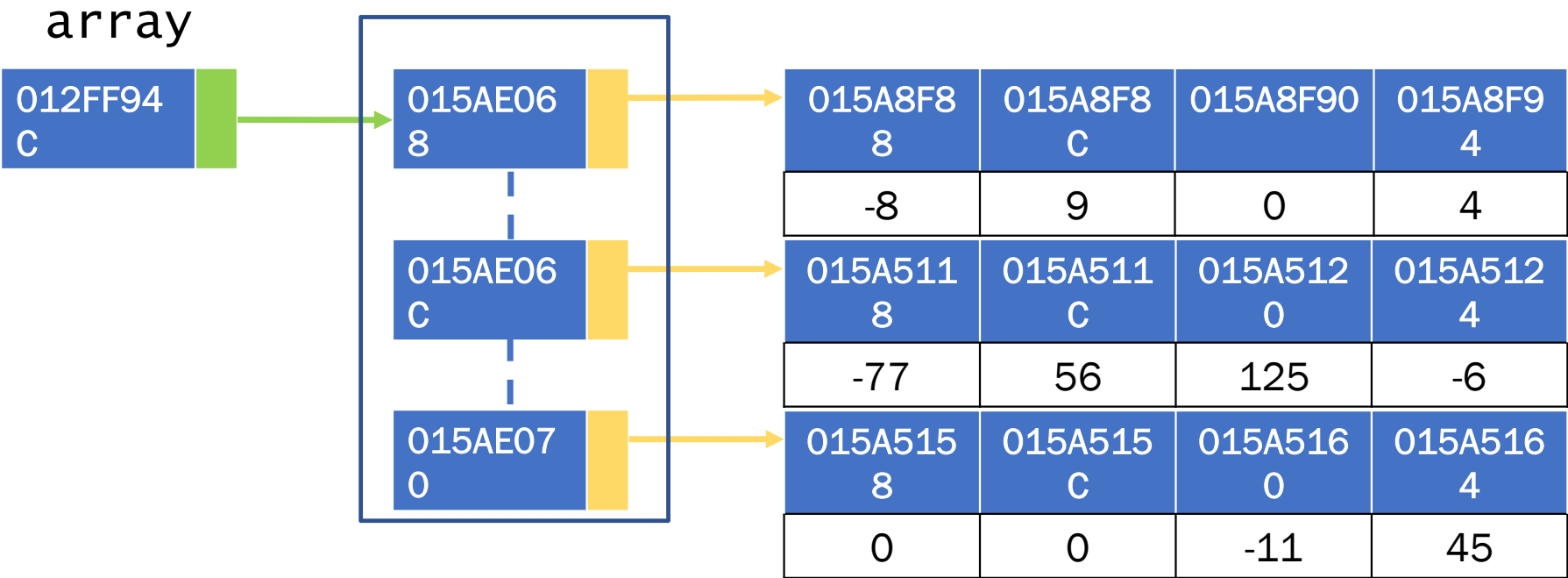
Динамический многомерный массив

Массив массивов



Динамический многомерный массив

Массив массивов `int **array;`



Динамический многомерный массив. Создание

Массив массивов `int **array;`

```
int rows, cols;
cin >> rows >> cols;

int **array;
array = new int *[rows];

for (int i = 0; i < rows; i++) {
    array[i] = new int[cols];
}

for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        array[i][j] = i+j;
    }
}
```


Динамический многомерный массив. Удаление

Массив массивов `int **array;`

```
int rows, cols;
cin >> rows >> cols;

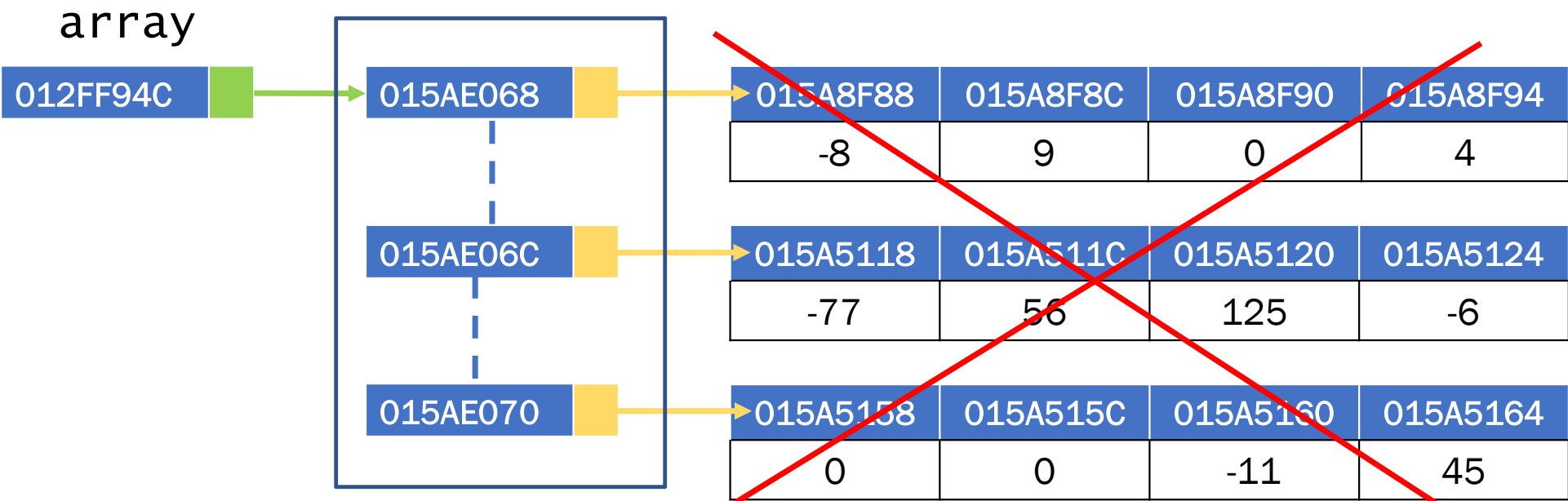
int **array;
array = new int *[rows];
...;

for (int i = 0; i < rows; i++) {
    delete [] array[i];
}

delete [] array;
array = nullptr;
```

Динамический многомерный массив. Удаление

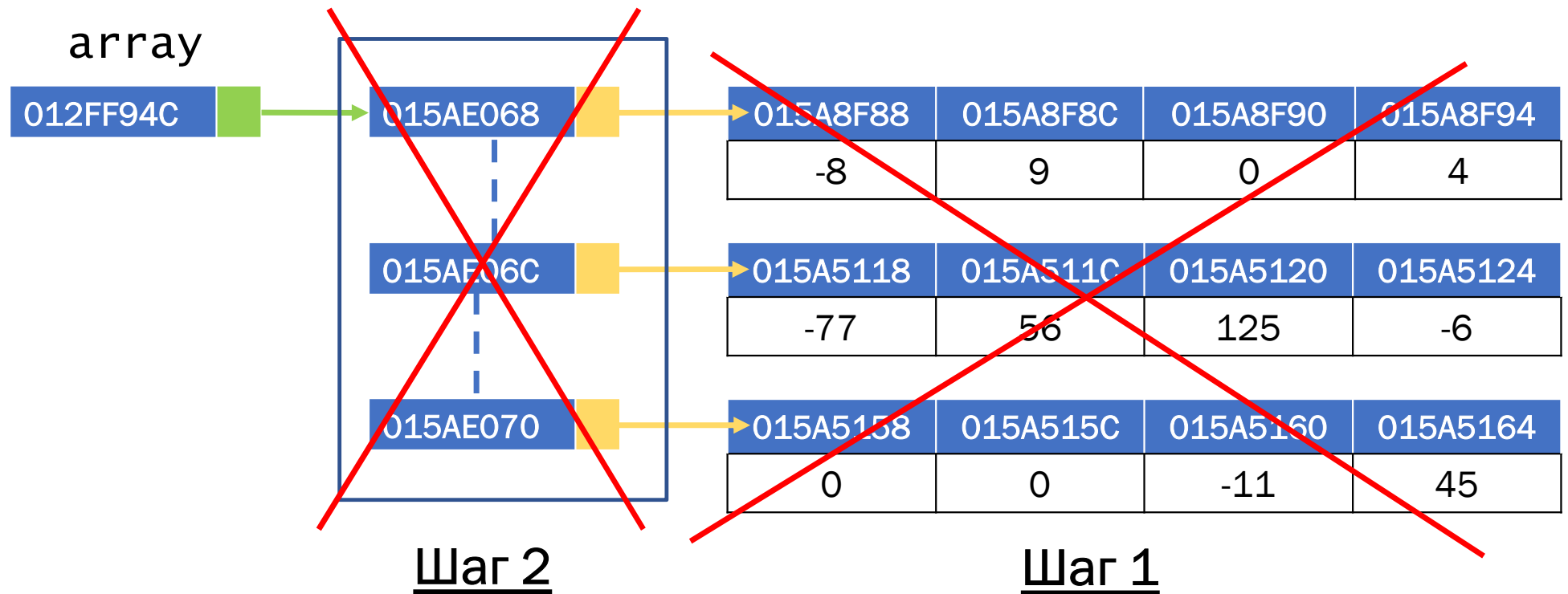
Массив массивов `int **array;`



Шаг 1

Динамический многомерный массив. Удаление

Массив массивов `int **array;`



Реализуем динамический массив

