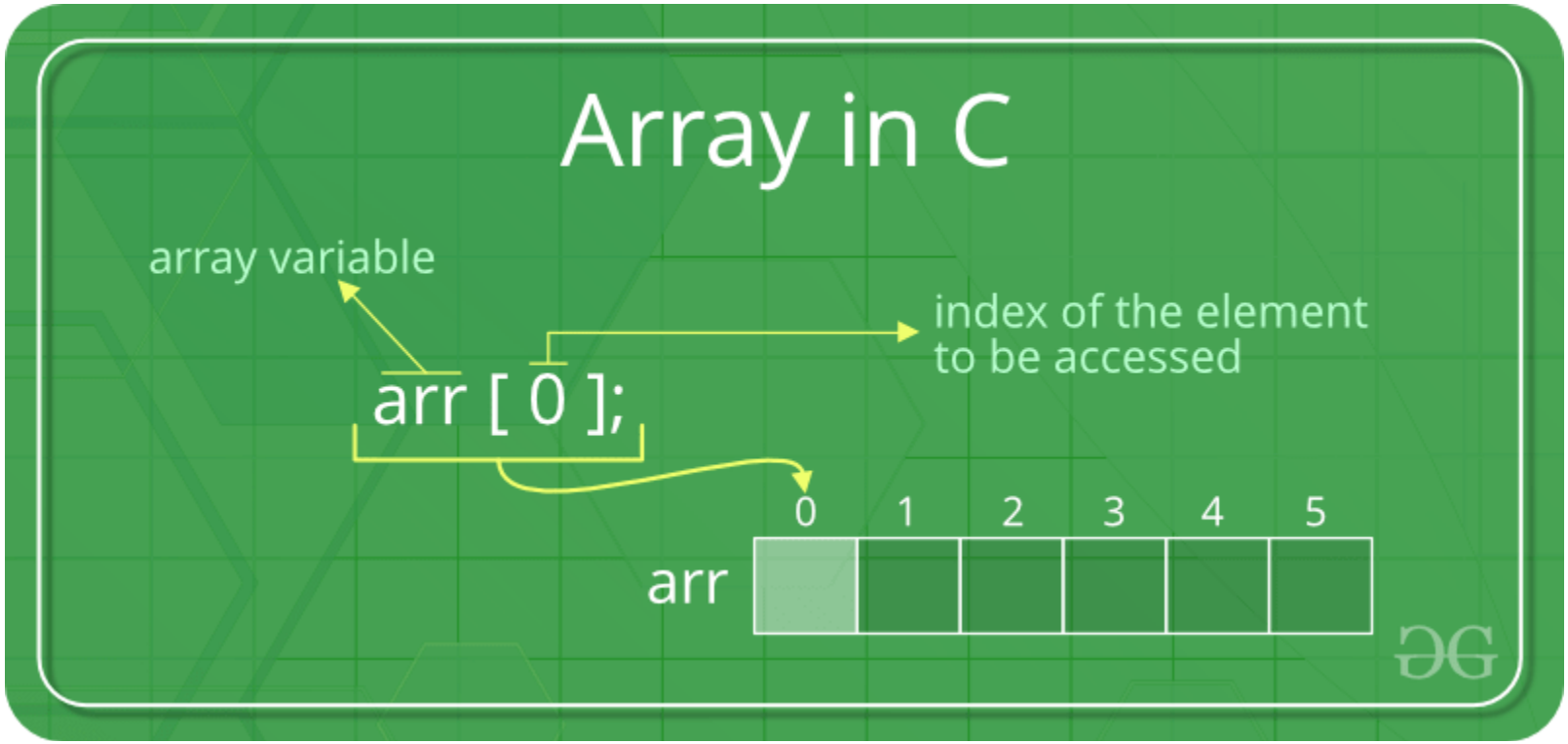
Едномерни масиви

доц. д-р Нора Ангелова

Едномерни масиви

- Редица от елементи от един и същ тип
- Клетките са разположени на последователни адреси в паметта



Едномерни масиви

- Дефиниция

<тип> <идентификатор>[<размер>$_{опц}$] = {<ст/ст>, <ст/ст>, …}$_{опц}$;
<размер> ::= константна стойност или израз

```
// Декларация на масив от цели числа (заделя памет за 32 цели числа)
// Елементите остават неинициализирани
int intArr[32];
// Декларация на масив от символи (заделя памет за 16 символа)
char strArr[16];
```

Едномерни масиви

- Дефиниция

<тип> <идентификатор>[<размер>$_{опц}$] = {<ст/ст>, <ст/ст>, …}$_{опц}$;
<размер> ::= константна стойност или израз

```
// Чрез {} може да се задават ст-ти на част от елементите в масива

// Създава масив и инициализира елементите му
// Създава масив от 6 елемента и ги инициализира със ст-тите 2,5,4,…
int arr2[6] = {2, 5, 4, 3, 2, 1};
// Създава масив от 6 елемента, инициализира първите два
// със ст-ти 2 и 5, а останалите с 0
int arr3[6] = {2, 5};
// Създава масив от 2 елемента със стойности 2 и 5
int arr4[] = {2, 5};
// Създава масив от 6 елемента със стойности 0
int arr5[6] = {0};
int arr6[6] = {};
```

Едномерни масиви

- Индекси

От 0 до <размер - 1>

Пример:

```
int intArr[32]; // Индексите са 0, 1, …, 31
char strArr[16]; // Индексите са 0, 1, …, 15

// Елементът 2 е на индекс 0, 5 е на индекс 1 и 1 е на индекс 5
int arr2[6] = {2, 5, 4, 3, 2, 1};

// 2 - индекс 0, 5 – индекс 1, 0 – 2-5
int arr3[6] = {2, 5};

// елементът 2 е индекс 0, а елементът 5 е на индекс 1
int arr4[] = {2, 5};
```

Едномерни масиви

- Достъп до елемент

```
<идентификатор>[<индекс>]
```

Пример:

```
int intArr[32];

// Достъп до първия елемент
intArr[0];

// Достъп до втория елемент
intArr[1];

// Извеждане на елементи на масива
std::cout << arr2[1] << "  " << arr2[0] << std::endl;
```

Едномерни масиви

- Достъп до елемент – позволява и промяна

```
<идентификатор>[<индекс>]
```

Пример:

```
int intArr[32];

// Инициализира първия елемент със стойност 5
intArr[0] = 5;

// Инициализира втория елемент със стойността на intValue
int intValue = 6;
intArr[1] = intValue;
```

Едномерни масиви

- Обхождане

```
int intArr[32] = {0};

for (int index = 0; index < 32; ++index) {
   std::cout << intArr[index] << " ";
}

-> 0, 1, 2, …, 30, 31
```

Адрес на променлива

- Адрес – число в шеснайсетична бройна система
- Оператор & - връща адресът на променливата

Пример:

```
int intValue = 5;
std::cout << &intValue; // Адресът на променливата intValue
```

Едномерни масиви

- Адрес – името на масива съвпада с адреса на първия елемент

```
intArr == &intArr[0]
```

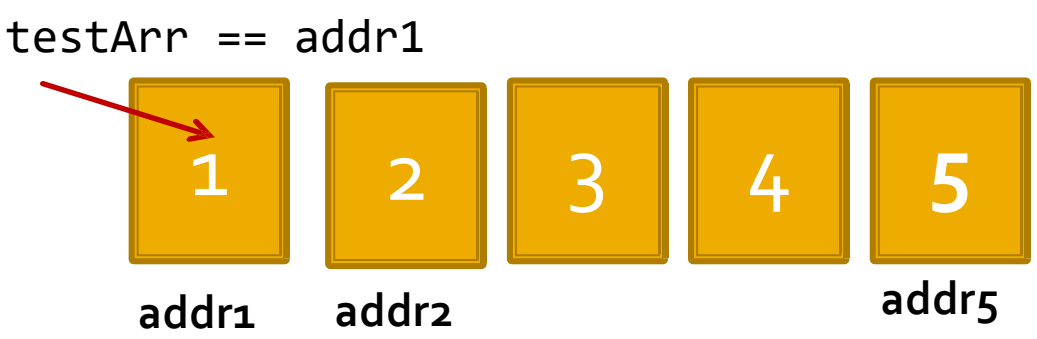
Едномерни масиви

- Адрес – името на масива съвпада с адреса на първия елемент

```
intArr == &intArr[0]
```

Пример:

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



Едномерни масиви

- Въвеждане и извеждане на елементи

```
for (int index = 0; index < 32; ++index) {
  std::cin >> intArr[index];
}

for (int index = 0; index < 32; ++index) {
  std::cout << intArr[index] << " ";
}
```

Едномерни масиви

- Въвеждане и извеждане на елементи

```
// Грешка
std::cin >> intArr;

// Извежда адрес
std::cout << intArr;
```

Едномерни масиви

- Присвояване

```
// Не се допуска директно присвояване
arr2 = arr
```

Едномерни масиви

- Сравнение (>=, <=, <, >, ==, !=)

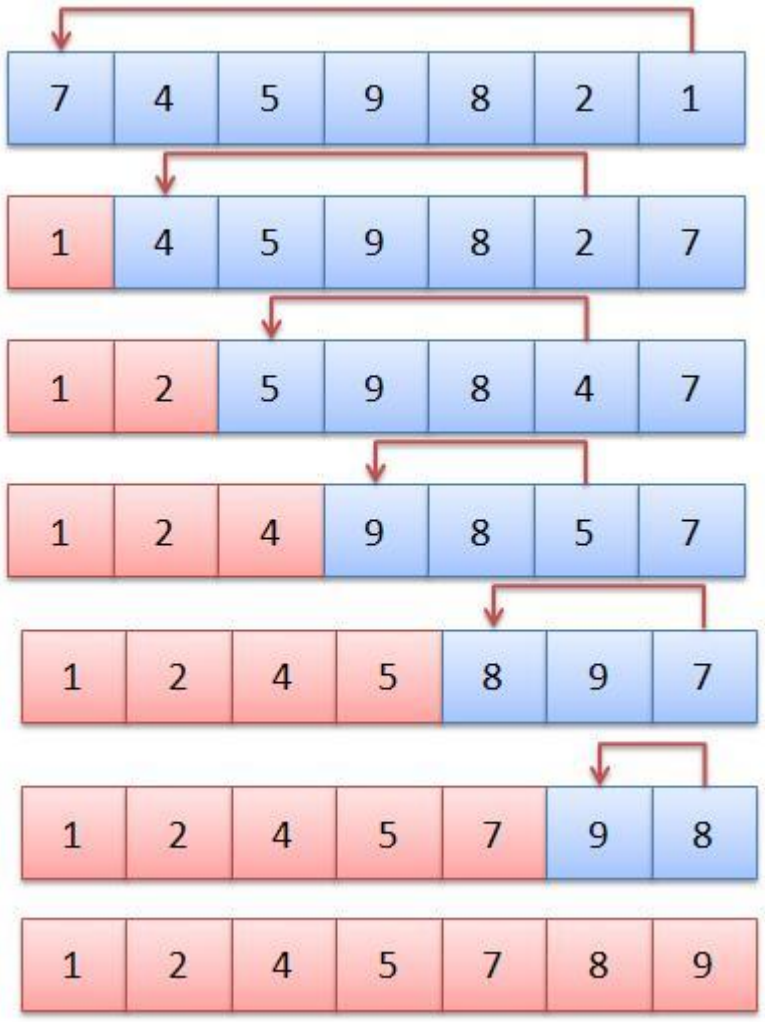
```
// Сравнява адреси
arr2 == arr
```

Пряка селекция

$a_0, a_1, \ldots, a_{n-1}$

За `i = 0, 1, 2, …, n-2`

- Намира се к, така че
  $a_k = \min\{a_i, a_{i+1}, \ldots, a_{n-1}\}$
- Разменят се стойностите на a_k и a_i.



Метод на мехурчето

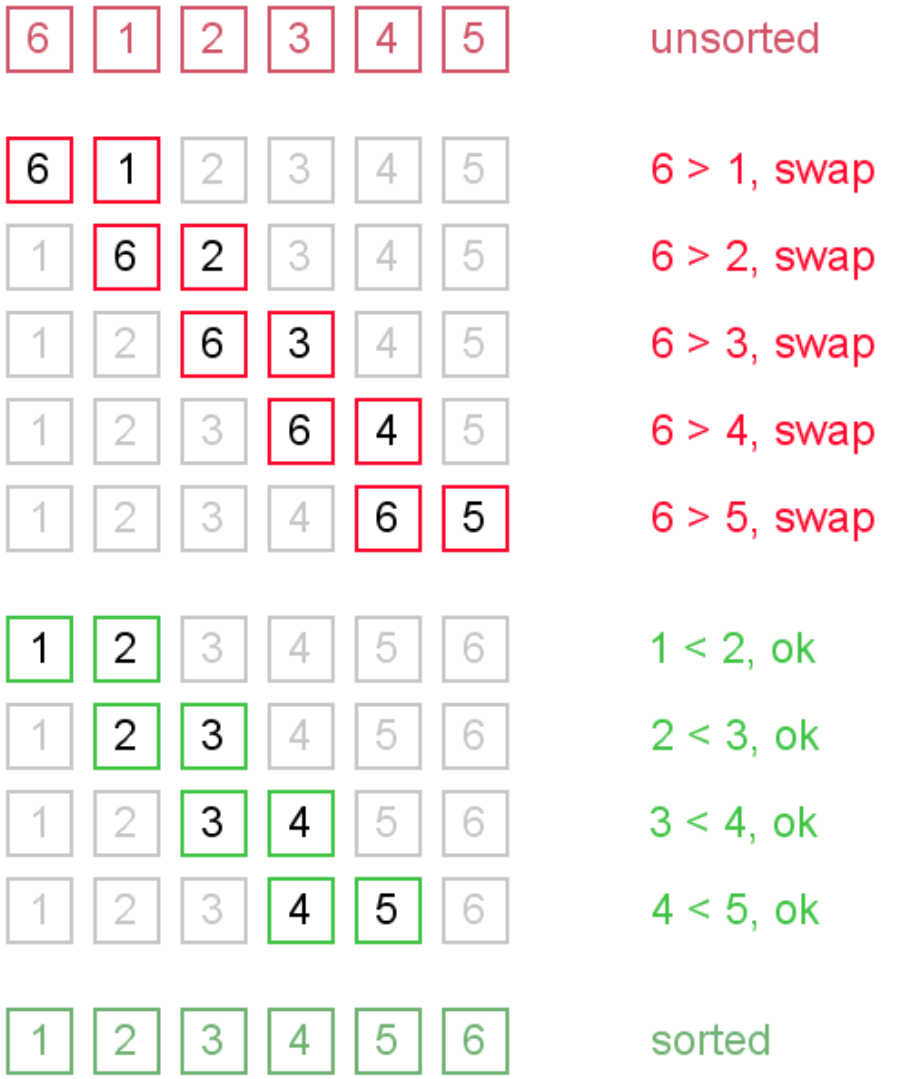
$a_0, a_1, \ldots, a_n$

Нека `right = n-1`

- Разглежда се редицата $a_0, a_1, \ldots, a_{right}$.
- Сравняват се всеки два последователни елемента a_i и a_{i+1}.
- Ако $a_i > a_{i+1}$, елементите се разменят и позицията на размяната се запомня в k.
- Ако $a_i <= a_{i+1}$, елементите не се разменят.

След направените размени:

- `right = k.`
- Ако `right > 0`, действията се повтарят.
- Ако `right == 0`, редицата е сортирана.



Търсене

- Дали елемент се среща в масив
- Дали елемент се среща в сортиран масив

Примери

Да се напише програмен фрагмент, който обръща елементите на масив с фиксиран размер SIZE.

Примери

```
const int SIZE = 9;
int arr[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9};


int temp, i;
for (i = 0; i < SIZE/2; ++i) {
     temp = arr[SIZE-i-1];
     arr[SIZE-i-1] = arr[i];
     arr[i] = temp;
}

for (i = 0; i < SIZE; ++i) {
     std::cout << arr[i] << " ";
}
```

Край