

# Лекция 4: Задача сортировка и функции в языке Си (продолжение)

Д. А. Караваев

Санкт-Петербургский государственный университет телекоммуникаций  
им. проф. М. А. Бонч-Бруевича

Факультет РТС, Кафедра РОС

Факультатив «Программирование в ЦОС»

Осень 2019

11.11.2019 Санкт-Петербург

Существует два способа передачи аргументов в функцию:

- 1 **По значению:** в качестве аргументов функции выступают соответствующие *копии* значений выражений, переданных в функцию в момент её вызова. Подобный механизм неявно реализован в Си (поведение по-умполчанию).
- 2 **По ссылке:** в качестве аргументов функции выступают *синонимы* переменных, переданных функцию. Подобный механизм используется в Java. В Си его можно реализовать явно при помощи *указателей*!

**Задача:** Что будет если мы захотим изменить значение той переменной, которая была передана в функцию в качестве параметра?

# Классический пример: функция swap

```
/* Реализуем функцию, которая меняет значения двух переменных. */  
/* Не правильно (оперируем с копиями, а не с самими переменными): */  
void swap_error(int x, int y)  
{  
    int temp = x;  
    x = y;  
    y = temp;  
}  
  
/* Правильно: используем указатели (передаём адреса, а не значения): */  
void swap_correct(int* x, int* y)  
{  
    int temp = *x;  
    *x = *y; /* (*x) - синоним переменной x! */  
    *y = temp;  
}
```

# Задача сортировки

## Формулировка

**Вход:** Массив  $a$  длины  $N$ ;

**Выход:** Массив  $a$  длины  $N$ :  $a[0] \leq a[1] \leq \dots \leq a[N-1]$  (по убыванию).

**Fuction** InsertionSort ( $a$  : массив,  $N$  : длина  $a$ ):

```
  for  $n \leftarrow 0$  to  $N - 1$  do
    |  $j \leftarrow \text{MinIndex}(a, n, N)$ 
    | Swap ( $a[n]$ ,  $a[j]$ )
  end
```

**Fuction** MinIndex ( $a$  : массив,  $begin$  : начало,  $end$  : конец):

```
  | // Ваш (псевдо)код тут
```

**Замечание:** Простейшим примером сортировки, является сортировка вставками (сложность  $O(N^2)$ ), основанная на поиске минимума.

**Посылка:** Представим, что правая и левая половины  $a$  отсортированы в нужном порядке:

$$mid = \left\lfloor \frac{N}{2} \right\rfloor,$$

$$a[0 \dots mid - 1] : a[0] \leq \dots \leq a[mid - 1],$$

$$a[mid \dots N - 1] : a[mid] \leq \dots \leq a[N - 1].$$

**Задача:** Как получить (слить) отсортированный массив  $a$  на основе двух его отсортированных половин?

**Fuction** Merge( $a, begin, mid, end$ ):

$n_1 \leftarrow mid - begin, n_2 \leftarrow end - mid, L[n_1 + 1], R[n_2 + 1]$

**for**  $i \leftarrow 0$  to  $n_1 - 1$  **do**

$L[i] \leftarrow a[begin + i]$

**end**

**for**  $j \leftarrow 0$  to  $n_2 - 1$  **do**

$R[j] \leftarrow a[mid + j]$

**end**

$L[n_1] \leftarrow \infty, R[n_2] \leftarrow \infty, i \leftarrow 0, j \leftarrow 0$

**for**  $k \leftarrow begin$  to  $end - 1$  **do**

**if**  $L[i] \leq R[j]$  **then**

$a[k] \leftarrow L[i], i \leftarrow i + 1$

**else**

$a[k] \leftarrow R[j], j \leftarrow j + 1$

**end**

**end**

# Сортировка слиянием

**Рекурсивный переход:** Когда  $a$  состоит из двух элементов, то обе его части отсортированы (начальное условие).

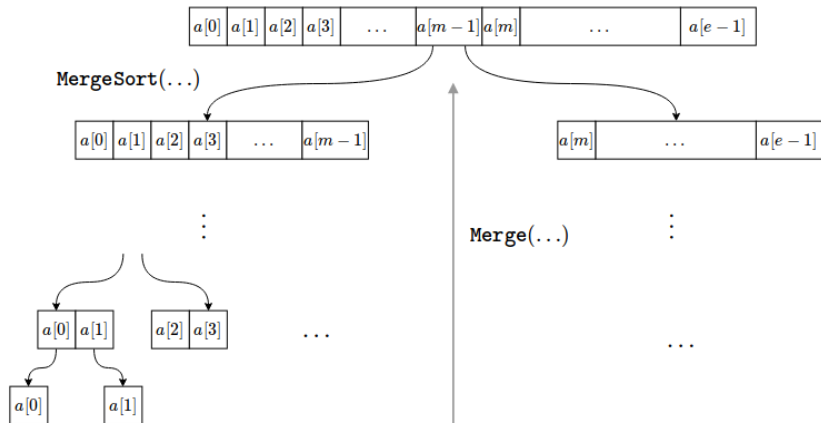
*Следовательно, рекурсивно разбивая массив на части до единичных элементов, можно отсортировать массив, применяя процедуру слияния для каждого этапа разбиения.*

**Fuction** MergeSort ( $a$  : массив,  $begin$  : начало,  $end$  : конец):

```
if  $(end - begin) \neq 1$  then
     $mid = \left\lfloor \frac{begin + end}{2} \right\rfloor$ 
    MergeSort ( $a, begin, mid$ )
    MergeSort ( $a, mid, end$ )
end
```

**Сложность:** Временная  $\Theta(N \log_2 N)$ , пространственная:  $\Theta(N)$ .

# Иллюстрация сортировки слиянием





- ❶ **Быстрая сортировка\***: Временная сложность в среднем  $O(N \log_2 N)$ , память - in-place;
- ❷ **Пирамидальная сортировка**: Временная сложность  $O(N \log_2 N)$ , память - in-place (основана на структуре данных - пирамида);
- ❸ **Подсчётом**: Временная сложность  $\Theta(k)$ , где  $k$  - максимально возможное значение элемента, память -  $\Theta(k)$ ;
- ❹ **Сортировка пузырьком**: Временная сложность  $O(N^2)$ , память - in-place.

В файле проекта `Sort/source/main.c`

- 1 Реализовать сортировку вставками (3 процедуры);
- 2 Реализовать сортировку слиянием (2 процедуры).

**Замечание:** Для сборки используйте те же команды, что и для прокетов `Convolution` и `Search`. Для проверки результатов воспользуйтесь функцией `printf(...)`.

Спасибо за внимание!