Тема 1.10 Реализация методов внутренней сортировки с помощью рекурсии

Рассмотрим примение рекурсии на примере быстрой сортировки. Определим следующий код:

```
#include <iostream>

void sort(int[], size_t, size_t);
void swap(int[], size_t, size_t);

int main()
{
   int nums[] {3, 0, 6, -2, -6, 11, 3};
   sort(nums, 0, std::size(nums)-1);
   for(auto num: nums)
   {
      std::cout << num << "\t";
   }
   std::cout << std::endl;
}

void sort(int numbers[], size_t start, size_t end)
{
    // начальный индекс должен быть меньше конечного индекса для массива из 2 и более элементов
   if (start >= end)
      return;
   // проверяем все элементы относительно элемента с индексом start
   size_t current {start};
   for (size_t i {start + 1}; i <= end; i++)
   {
      // если i-ый элемент меньше начального
      if (numbers[i] < numbers[start])
      {
         swap(numbers, ++current, i); // меняем его с левым
      }
   }
   swap(numbers, start, current); // Меняем выбранный (start) и последний обмененный элементы
   if (current > start)
   {
      sort(numbers, start, current - 1); // Сортируем элементы слева
   }
   if (end > current + 1)
   {
      sort(numbers, current + 1, end); // Сортируем элементы справа
   }
}
void swap(int numbers[], size_t first, size_t second)
{
```

```
    auto temp{numbers[first]};
    numbers[first] = numbers[second];
    numbers[second] = temp;
}
```

Для сортировки массива здесь определена функция sort, которая принимает три параметра: сортируемый массив, начальный и конечный индексы сортируемой части массива:

```
void sort(int numbers[], size_t start, size_t end)
```

При первом вызове функции предполагается, что начальный индес, start, будет равен 0, а конечный индекс, end, будет представлять индекс последнего элемента массива.

Сначала проверяем размер сортируемой части:

```
if (start >= end)
    return;
```

Если начальный При каждом выполнении функции sort() в конце текущая сортируемая последовательность разбивается на две меньшие подпоследовательности, для каждой из которых также рекурсивно вызывается функция sort(). Поэтому в конечном итоге мы должны получить подпоследовательность, которая содержит только один элемент.

Далее устанавливаем индекс текущего элемента

```
size_t current {start};
```

Далее в цикле сравниваем этот элемент с остальными элементами, которые идут после индекса start:

```
for (size_t i {start + 1}; i <= end; i++)
{
    // если i-ый элемент меньше начального
    if (numbers[i] < numbers[start])
    {
        swap(numbers, ++current, i); // меняем его с тем, который слева
    }
}
```

Если i-ый элемент меньше выбранного начального элемента, то меняем i-ый элемент с элементом, который следует за start. В результате все элементы, которые меньше выбранного элемента, помещаются перед всеми элементами, которые больше или равны ему.

Когда цикл заканчивается, переменная current содержит индекс последнего найденного элемента, которое меньше выбранного начального элемента с индексом start. И эти элементы с индексами current и start меняем местами:

```
swap(numbers, start, current);
```

Таким образом, элемент, относительного которого производилась проверка других элементов, тепень имеет индекс current и помещается после элементов, которые меньше него.

В конце сортируем две подпоследовательности по обе стороны от текущего элемента с индексом current, вызывая sort() для каждого подмножества. Индексы элементов, меньших выбранного слова, идут от начала до индекса current-1, а индексы тех, которые больше, идут от индекса current+1 до конца.

```
if (current > start)
{
    sort(numbers, start, current - 1); // Сортируем элементы слева
}
if (end > current + 1)
{
    sort(numbers, current + 1, end); // Сортируем элементы справа
}
```

Например, при первом выполнении исходные данные будут иметь следующие значения:

```
start = 0
end = 6
current = 0
```

В итоге цикл произведет последовательно 7 итераций, при которых элементы будут меняться следующим образом

после 1-итерации:	3	0	6	-2	-6	11	3
после 2-итерации:	3	0	6	-2	-6	11	3
после 3-итерации:	3	0	-2	6	-6	11	3
после 4-итерации:	3	0	-2	-6	6	11	3
после 5-итерации:	3	0	-2	-6	6	11	3
после 6-итерации:	3	0	-2	-6	6	11	3
после 7-итерации:	**3**	0	-2	**-6**	6	11	3

А переменная current будет равна 3, то есть было три элемента, которые меньше выбранного элемента с индексом start. И далее меняем местами current и start меняем местами. В итоге получаем:

-6	0	-2	**3**	6	11	3

Затем разбиваем на две подпоследовательности: слева от current

-6	0	-2

и справа от current

6	11	3