Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

ОТЧЁТ

по лабораторной работе №3
по дисциплине “Проектирование программного обеспечения в информационных системах”

Выполнил:
Р. В. Липский, гр. 121701
Проверил:
С. А. Никифоров

Постановка задачи

Тема: Обобщенное программирование. Стандартная библиотека шаблонов (STL).
Цель: Получить навыки обобщенного программирования с использованием шаблонов.

Задание: В соответствии с вариантом нужно реализовать шаблонную функцию(функции) для сортировки. Необходимо, чтобы разработанная функция(функции) позволяла сортировать массивы и векторы(std::vector<>) любых объектов (как встроенных типов, так и пользовательских), продемонстрировать это путём создания собственного класса, массив объектов которого необходимо отсортировать. Для этого можно изучить реализацию функций сортировки, из стандартной библиотеки, позволяющих сортировать массивы и векторы.

Вариант: 14. Quicksort. Sorting networks

Реализация

Quicksort

Исходный код:

```
template<typename T, typename C>
   size_t partition(T array[], size_t low, size_t high, C compare) {
       T pivot = array[high];
       size_t i = (low - 1);
       for (size_t j = low; j < high; j++) {
           if (compare(array[j], pivot)) {
               i++;
               T temp = array[j];
               array[j] = array[i];
               array[i] = temp;
           }
       }
       T temp = array[i + 1];
       array[i + 1] = array[high];
       array[high] = temp;
       return (i + 1);
   }

   template<typename T, typename C>
   void qsort(T* array, size_t low, size_t high, C compare) {
       if (low < high) {
```

```
            size_t pi = partition(array, low, high, compare);
            qsort(array, low, pi - 1, compare);
            qsort(array, pi + 1, high, compare);
        }
    }
```

Функция неявно принимает два шаблонных параметра:

- T - тип элементов массива
- C - тип функтора (любой тип, реализующий метод operator(), включая указатели на функции, std::function, lamda-выражения и т.д.)

Функция принимает четыре параметра:

- T array[] - указатель на массив элементов типа T
- size_t low, high - границы массива, в рамках которого производится сортировка
- C compare - функтор сравнения

std::vector гарантирует, что все элементы расположены в памяти последовательно, потому мы может использовать указатель на первый его элемент, как указатель на массив аналогичного типа.

Пример использования с массивом:

```
int arr[] = {1, 6, 3, 0, -4, 126, 31, -35, 35, 7, 8, -3, 4, -7};
nstd::qsort(arr, 0, 13, [](int a, int b) { return a < b; });
```

Пример использования с вектором:

```
std::vector<int> vect = {1, 6, 3, 0, -4, 126, 31, -35, 35, 7, 8, -3, 4,
-7};
nstd::qsort(&vect[0], 0, 13, [](int a, int b) { return a < b; });
```

Sorting network

Исходный код:

```
class sorting_network : public std::vector<std::pair<size_t, size_t>> {
    public:
        template<typename T, typename C>
        void apply(T* array, C compare) {
            for (const auto& pair : *this) {
                if (compare(array[pair.first], array[pair.second])) {}
                T temp = array[pair.first];
                array[pair.first] = array[pair.second];
                array[pair.second] = temp;
            }
        }

    };
```

nstd::sorting_network - шаблон класса, наследующий std::vector<std::pair<size_t, size_t>>, где элементы вектора означают связи между элементами в сети сортировки.

Шаблон класса реализует метод apply(), который принимает два шаблонных параметра:

- T - тип элементов массива
- C - тип функтора (любой тип, реализующий метод operator(), включая указатели на функции, std::function, lamda-выражения и т.д.)

Функция принимает два параметра:

- T array[] - указатель на массив элементов типа T
- C compare - функтор сравнения

std::vector гарантирует, что все элементы расположены в памяти последовательно, потому мы может использовать указатель на первый его элемент, как указатель на массив аналогичного типа.

Пример использования с массивом:

```
nstd::sorting_network sn;
sn.emplace_back(0, 13);
sn.emplace_back(1, 12);

int arr[] = {-35, -7, -4, -3, 0, 1, 3, 4, 6, 7, 8, 31, 35, 126};
sn.apply(arr, [](int a, int b) { return a > b; });
```

Пример использования с вектором:

```
nstd::sorting_network sn;
sn.emplace_back(0, 13);
sn.emplace_back(1, 12);

std::vector<int> vect = {-35, -7, -4, -3, 0, 1, 3, 4, 6, 7, 8, 31, 35,
126};
sn.apply(&vect[0], [](int a, int b) { return a > b; });
```