Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №7
по дисциплине: **ООТПиСТ**
Тема: ШАБЛОНЫ ФУНКЦИЙ И КЛАССОВ

Выполнил
студент 2 курса
Корнасевич И. Д.

Проверил
Миндер А. В.

Брест 2021

Задание Реализовать шаблонный класс-обёртку над массивом. Необходимые методы:

- `add` — добавить элемент в конец коллекции
- `insert` — вставить элемент по индексу
- `erase` — удалить элемент по индексу
- `[]` — получение элемента внутреннего массива по индексу.
- `<<` — перегрузка вывода в поток
- `==` и `!=` — реализация неглубокого сравнения

main.cpp

```cpp
#include <iostream>
#include "ArrayList.h"

int main() {
    ArrayList<int> list(10);
    list.add(1);
    list.add(2);
    list.add(3);
    list.add(4);
    list.add(5);
    list.add(6);
    list.add(7);
    list.add(8);
    std::cout << list << std::endl;
    list.insert(3, 100);
    std::cout << list << std::endl;
    list.erase(4);
    std::cout << list << std::endl;
}
```

ArrayList.h

```cpp
#ifndef SRC_ARRAYLIST_H
#define SRC_ARRAYLIST_H

#include <cstring>
#include <ostream>

template<typename T>
class ArrayList {
public:
    explicit ArrayList(size_t capacity) {
        this->capacity = capacity;
        _size = 0;
        array = new T[capacity];
    }

    ArrayList() : ArrayList(16) {};

```

```
18      ~ArrayList() {
19          delete[] array;
20      }
21
22      void add(const T &item) {
23          _size++;
24          expandIfFull();
25          array[_size - 1] = item;
26      }
27
28      void insert(size_t index, const T &item) {
29          _size++;
30          if (index > _size) {
31              _size = index + 1;
32          }
33          expandIfFull();
34          memmove(&array[index + 1], &array[index], (_size - index) *
    sizeof(T));
35          array[index] = item;
36      }
37
38      void erase(size_t index) {
39          memmove(&array[index], &array[index + 1], (_size - index) *
    sizeof(T));
40          _size--;
41      }
42
43      size_t size() {
44          return this->_size;
45      }
46
47      friend std::ostream &operator<<(std::ostream &os, const ArrayList
     &list) {
48          os << "ArrayList[";
49          for (int i = 0; i < list._size - 1; i++) {
50              os << list[i] << ", ";
51          }
52          os << list[list._size - 1] << "]";
53          return os;
54      }
55
56      T operator[](size_t index) const {
57          return this->array[index];
58      }
59
60      bool operator==(const ArrayList &rhs) const {
61          return _size == rhs._size &&
62                 capacity == rhs.capacity &&
63                 array == rhs.array;
64      }
65
66      bool operator!=(const ArrayList &rhs) const {
67          return rhs != *this;
68      }
69
70  private:
```

```
71      void expand() {
72          capacity <<= 1;
73          auto newArray = new T[capacity];
74          memcpy(newArray, array, (capacity >> 1) * sizeof(T));
75          delete[] array;
76          array = newArray;
77      }
78
79      void expandIfFull() {
80          while (_size >= capacity) {
81              expand();
82          }
83      }
84
85      size_t _size{};
86      size_t capacity{};
87      T *array;
88  };
89
90
91  #endif //SRC_ARRAYLIST_H
```

results.txt

```
1  ArrayList[1, 2, 3, 4, 5, 6, 7, 8]
2  ArrayList[1, 2, 3, 100, 4, 5, 6, 7, 8]
3  ArrayList[1, 2, 3, 100, 5, 6, 7, 8]
```

Вывод: Я разработал класс generic ArrayList<T>, который является обёрткой над массивом. Этот класс может работать с любым типом данных. Также я использовал функции memcpy и memmove для наиболее быстрого перемещения элементов при копировании.