

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

старший преподаватель		Е.О. Шумова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6

Стандартная библиотека C++. Последовательные и ассоциативные
контейнеры. Обобщенные алгоритмы

по дисциплине: Объектно-ориентированное программирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №	Z1431		М.Д. Быстров
	номер группы	подпись, дата	инициалы, фамилия

Студенческий билет №	2021/3572
----------------------	-----------

Санкт-Петербург 2023

Условие

Цель работы: изучить принципы построения консольных приложений, применив на практике знания базовых синтаксических конструкций языка C++ и объектно-ориентированного программирования.

Общая часть задания

Реализовать класс, содержащий:

- массив данных (вектор), заполненный случайными числами в диапазоне $m1 - m2$;
- методы, обеспечивающие выполнение действий (согласно варианту) с использованием обобщенных алгоритмов, объектов-функций и предикатов;
- обеспечить вывод результатов после выполнения каждого действия.

Обобщенные алгоритмы использовать обязательно.

Индивидуальное задание (вариант 2):

Вариант 2.

- $m1=-10, m2=10$
- найти максимальный элемент массива
- прибавить к каждому элементу массива найденный максимальный

элемент

- отсортировать массив по абсолютному значению

Полный текст (листинг) программы

1. Файл «main.cpp»

```
1. #include <iostream>
2. #include "MyVector.h"
3.
4. using namespace std;
5.
6. int main()
7. {
8.     cout << "LR#6 VAR2" << endl << endl;
9.
10.    MyVector* vector = new MyVector(20);
11.
12.    cout << "Vector: " << vector->getVectorStr() << endl << endl;
13.
14.    cout << "Max: " << vector->max() << endl;
15.
16.    vector->addMax();
17.
18.    cout << "Max added to each element: " << vector->getVectorStr() <<
    endl;
19.
20.    vector->sortByAbs();
21.
22.    cout << "Vector sorted by absolute value: " << vector->getVectorStr()
    << endl;
23.
24.    return 0;
25. }
```

2. Файл «MyVector.h»

```
1. #include <vector>
2. #include <string>
3.
4. #pragma once
5. class MyVector
6. {
7. private:
8.
9.     std::vector<int>* vector;
10.
11. public:
12.
```

```

13.     MyVector(int size);
14.     ~MyVector();
15.
16.     std::string getVectorStr();
17.
18.     int max();
19.     void addMax();
20.     void sortByAbs();
21. };
22.
23.

```

3. Файл «MyVector.cpp»

```

1. #include <vector>
2. #include <limits>
3. #include <cstdlib>
4. #include <string>
5. #include <sstream>
6. #include <ctime>
7. #include "MyVector.h"
8. #include "AddFunctor.h"
9.
10.
11. #define m1 -10
12. #define m2 10
13.
14. MyVector::MyVector(int size)
15. {
16.     this->vector = new std::vector<int>(size);
17.
18.     srand((unsigned)time(NULL));
19.
20.     for (int i = 0; i < vector->size(); i++)
21.     {
22.         vector->at(i) = (rand() % (abs(m1 - m2) + 1)) + m1;
23.     }
24. }
25.
26. MyVector::~MyVector()
27. {
28.     delete this->vector;
29. }
30.
31. std::string MyVector::getVectorStr()
32. {

```

```

33.     std::stringstream ss;
34.
35.     std::vector<int>::iterator iter = this->vector->begin();
36.
37.     for (iter; iter < this->vector->end(); iter++)
38.     {
39.         ss << *iter << " ";
40.     }
41.
42.     ss << std::endl;
43.
44.     return ss.str();
45. }
46.
47. template <class T> T findMax(std::vector<T>* vector, T minValue)
48. {
49.     T maxValue = minValue;
50.
51.     for (int i = 0; i < vector->size(); i++)
52.     {
53.         if (vector->at(i) > maxValue)
54.         {
55.             maxValue = vector->at(i);
56.         }
57.     }
58.
59.     return maxValue;
60. }
61.
62. int MyVector::max()
63. {
64.     int max = findMax<int>(vector, std::numeric_limits<int>::min());
65.
66.     return max;
67. }
68.
69. void MyVector::addMax()
70. {
71.     AddFunctor add{};
72.
73.     int max = this->max();
74.
75.     for (int i = 0; i < vector->size(); i++)
76.     {
77.         add(&(vector->at(i)), max);
78.     }
79. }
80.
81. bool intSortCriterion(const int int1, const int int2)
82. {

```

```

83.     return abs(int1) <= abs(int2);
84. }
85.
86. void mergeArray(std::vector<int>* vector, int left, int middle, int right,
    bool(*op)(int, int))
87. {
88.     int leftArrayLength = middle - left + 1;
89.     int rightArrayLength = right - middle;
90.     std::vector<int>* leftTempArray = new
        std::vector<int>(leftArrayLength);
91.     std::vector<int>* rightTempArray = new
        std::vector<int>(rightArrayLength);
92.
93.     int i, j;
94.
95.     for (i = 0; i < leftArrayLength; i++)
96.     {
97.         leftTempArray->at(i) = vector->at(left + i);
98.     }
99.     for (j = 0; j < rightArrayLength; j++)
100.    {
101.        rightTempArray->at(j) = vector->at(middle + 1 + j);
102.    }
103.
104.    i = 0;
105.    j = 0;
106.    int k = left;
107.
108.    while (i < leftArrayLength
109.        && j < rightArrayLength)
110.    {
111.        if (op(leftTempArray->at(i), rightTempArray->at(j)))
112.        {
113.            vector->at(k++) = leftTempArray->at(i++);
114.        }
115.        else
116.        {
117.            vector->at(k++) = rightTempArray->at(j++);
118.        }
119.    }
120.
121.    while (i < leftArrayLength)
122.    {
123.        vector->at(k++) = leftTempArray->at(i++);
124.    }
125.
126.    while (j < rightArrayLength)
127.    {
128.        vector->at(k++) = rightTempArray->at(j++);
129.    }

```

```

130.
131.         delete leftTempArray;
132.         delete rightTempArray;
133.     }
134.
135.     std::vector<int>* sort(std::vector<int>* vector, int left, int right,
        bool(*op)(int, int))
136.     {
137.         if (left < right)
138.         {
139.             int middle = left + (right - left) / 2;
140.             sort(vector, left, middle, op);
141.             sort(vector, middle + 1, right, op);
142.             mergeArray(vector, left, middle, right, op);
143.         }
144.
145.         return vector;
146.     }
147.
148.     std::vector<int>* mergeSort(std::vector<int>* vector, bool(*op)(int,
        int))
149.     {
150.         if (vector->size() == 0
151.             || vector->size() == 1)
152.         {
153.             return new std::vector<int>(*(vector));
154.         }
155.
156.         std::vector<int>* ret = new std::vector<int>(*(vector));
157.
158.         ret = sort(ret, 0, ret->size() - 1, op);
159.
160.         return ret;
161.     }
162.
163.     void MyVector::sortByAbs()
164.     {
165.         std::vector<int>* prevVector = this->vector;
166.
167.         this->vector = mergeSort(this->vector, intSortCriterion);
168.
169.         delete prevVector;
170.     }
171.
172.

```

4. Файл «AddFunctor.h»

```
1. #pragma once
2. class AddFunctor
3. {
4. public:
5.
6.     void operator()(int* a, int b);
7. };
8.
9.
```

5. Файл «AddFunctor.cpp»

```
1. #include "AddFunctor.h"
2.
3. void AddFunctor::operator()(int* a, int b)
4. {
5.     *(a) += b;
6. }
7.
```


Работа программы



Консоль отладки Microsoft Visual Studio

```
Vector: 2 4 -9 0 -2 -10 2 8 -10 4 -8 -9 9 4 -2 0 8 8 -6 -8  
Max: 9  
Max added to each element: 11 13 0 9 7 -1 11 17 -1 13 1 0 18 13 7 9 17 17 3 1  
Vector sorted by absolute value: 0 0 -1 -1 1 1 3 7 7 9 9 11 11 13 13 13 17 17 17 18  
C:\Users\user\source\repos\OOP_lab6\x64\Debug\OOP_lab6.exe (процесс 19224) завершил р  
аботу с кодом 0.  
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 1 Работа программы

Выводы

В ходе выполнения лабораторной работы №6 изучены принципы построения консольных приложений, применены на практике знания базовых синтаксических конструкций языка C++ и объектно-ориентированного программирования.

Получен опыт в работе со структурами данных STL, использованы объекты-функции, предикаты.

Создана и использована шаблонизированная функция, позволяющая обрабатывать различные типы данных.