**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ижевский государственный  
технический университет имени М.Т. Калашникова»

(ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

Кафедра «Вычислительная техника»

Отчет по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

на тему «Библиотека STL, последовательные контейнеры и алгоритмы»

Вариант 14

Выполнил: студент гр. Б06-781-1

Пермитин И.А.

Проверил: к.т.н., доцент кафедры ВТ

Марков Е.М.

Ижевск 2020

**Задание**

1. Разработать программу по заданию, сама обработка выполняется через алгоритмы STL, обязательно использовать обработку исключений.

2. Использовать подходящий для обработки последовательный контейнер с элементами bitset 32 бита (заполняется случайно, все 32 разряда, использовать генератор из класса ), допускается использовать произвольные типы или стандартные по согласованию с преподавателем.

3. В зависимости от задания массив выводиться до и после обработки в двоичном, десятичном и шестнадцатеричном виде.

Задание: Найти максимальный элемент и удалить его из контейнера.

**Описание использованных технологий**

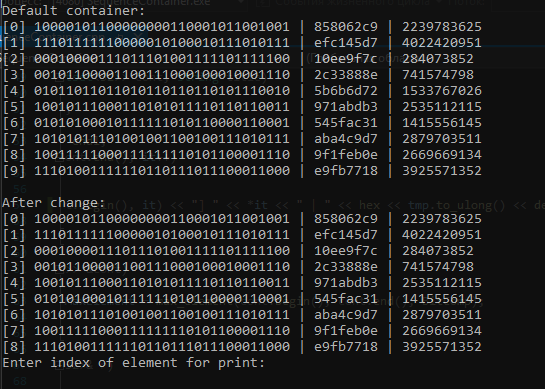
Лабораторная работа выполнялась в IDE «Visual Studio 2019».

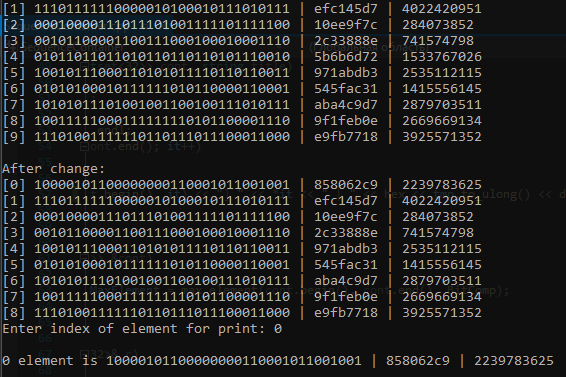
Используемый язык программирования – C++

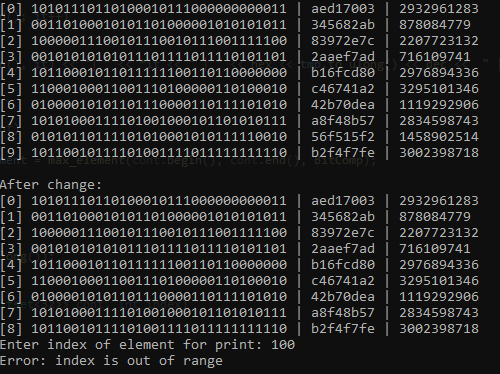
Тип проекта – Консольное приложение.

**Работа приложения**

Создается последовательный контейнер vector. Заполняется сгенерированными значениями. Выводятся в консоль изначальные значения его элементов. Далее находится элемент с максимальным значением в контейнере и удаляется. В конце программа просит ввести индекс элемента для вывода его на экран. Если введенный индекс больше, чем количество элементов контейнера, то срабатывает исключение.







**Листинг программы**

//Найти максимальный элемент и удалить его из контейнера.

#include <iostream>

#include <random>

#include <vector>

#include <bitset>

using namespace std;

void fillBitsetContainer(vector<bitset<32>> &cont);

void printBitsetContainer(vector<bitset<32>> cont, string title);

void printBitsetContainerElement(vector<bitset<32>> cont, int index);

void delMaxElement(vector<bitset<32>> &cont);

bool BitComp(bitset<32>& l, bitset<32>& r);

const int SIZE = 10;

int main()

{

vector<bitset<32>> container;

int index;

fillBitsetContainer(container);

printBitsetContainer(container, "Default container");

delMaxElement(container);

printBitsetContainer(container, "After change");

cout << "Enter index of element for print: ";

cin >> index;

printBitsetContainerElement(container, index);

}

void fillBitsetContainer(vector<bitset<32>> &cont)

{

random\_device rd;

mt19937 gen(rd());

int value = gen();

for (int i = 0; i < SIZE; i++)

{

int value = gen();

cont.push\_back(value);

}

}

void printBitsetContainer(vector<bitset<32>> cont, string title)

{

vector<bitset<32>>::iterator it;

cout << "\n" << title << ":" << endl;

for (it = cont.begin(); it != cont.end(); it++)

{

bitset<32> tmp = \*it;

cout << "[" << distance(cont.begin(), it) << "] " << \*it << " | " << hex << tmp.to\_ulong() << dec << " | " << tmp.to\_ulong() << endl;

}

}

void delMaxElement(vector<bitset<32>> &cont)

{

vector<bitset<32>>::iterator itMaxElement = max\_element(cont.begin(), cont.end(), BitComp);

cont.erase(itMaxElement);

}

bool BitComp(bitset<32>& l, bitset<32>& r)

{

return int(l.to\_ulong()) < int(r.to\_ulong());

}

void printBitsetContainerElement(vector<bitset<32>> cont, int index)

{

try

{

bitset<32> tmp = cont.at(index);

cout << "\n" << index << " element is " << tmp << " | " << hex << tmp.to\_ulong() << dec << " | " << tmp.to\_ulong() << endl;

}

catch (out\_of\_range)

{

cout << "Error: index is out of range" << endl;

}

}