Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №6**

По дисциплине: “ООТПиСП”

Тема: “Механизм обработки исключений C++”

Вариант №6

**Выполнил**:

студент 2 курса

группы ПО-7

Комиссаров А.Е.

**Проверила:**

Хацкевич М.В.

Брест, 2022

Цель работы: изучение методов обработки ошибок в языке С++.

**Вариант №6**

Варианты заданий: 7) Напишите программу на С++, которая показывает, что важна последовательность обработчиков исключений. Выполняется первый соответствующий обработчик. Скомпилируйте и выполните вашу программу с двумя различными последовательностями обработчиков, чтобы показать, что при этом наблюдаются различные результаты.

13) Напишите программу подобную программе из задания 12 для проверки terminate и set\_terminate.

Напишите программу на С++, которая использует функцию set\_unexpected для установки определенной пользователем функции обработки непредусмотренных исключений unexpected для установки определенной пользователем функции обработки непредусмотренных исключений unexpected, затем снова возвращает unexpected обратно

к предыдущей функции.

**Ход работы.**

**Код программы.**

**Задание 1.**

#include <iostream>

using namespace std;

int main() {

try { throw 13;}

catch (int e) {

cout << "Handling exception..." << e << endl;

}catch (...) {

cout << "Unknown exception." << endl;

}return 0;}

**Результат работы программы:**

****

Изменим порядок обработчиков исключений.

**Код программы:**

#include <iostream>

using namespace std;

int main() {

try { throw 13; }

catch (bool) {

cout << "Unknown exception." << endl; // skip this catch section

} catch (int e) {

cout << "Handling exception..." << e << endl;// Handling the exception here

} return 0;}

**Вывод:**



Таким образом, выполняется первый и единственный из обработчиков событий, начиная с первого.

**Задание 2.**

Функция void terminate() вызывается в случае, когда отсутствует процедура для обработки некоторого сформированного исключения.  
По умолчанию terminate() вызывает библиотечную функцию abort(), что влечет завершение программы. Если такая последовательность действий программиста не устраивает, он может написать собственную функцию (terminate\_function) и зарегистрировать ее с помощью функции set\_terminate(). В этом случае terminate() будет вызывать эту новую функцию вместо функции abort().

**Код программы:**

#include <iostream>

using namespace std;

void new\_terminate()

{ cout << "Terminate called." << endl;

abort();}

int main()

{ set\_terminate(new\_terminate);

try {throw 13;}

catch (string) {cout << "String exception." << endl;}

catch (bool) { cout << "Boolean exception." << endl;}

return 0;}

**Результат работы программы:**



Теперь вызовем событие bool.

#include <iostream>

using namespace std;

void new\_terminate(){cout << "Terminate called." << endl;abort();}

int main(){

set\_terminate(new\_terminate);

try {throw true;}

catch (string){cout << "String exception." << endl;}

catch (bool){cout << "Boolean exception." << endl;}

return 0;}

**Результат работы программы:**

****

**Вывод:** изучил некоторые механизмы и методы обработки исключений и ошибок в ЯВУ C++.