

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных**  **технологий** | **Кафедра**  **информационных систем** |

**Основная образовательная программа 09.03.02  
«Информационные системы и технологии»**

**Отчет по дисциплине «Объектно-ориентированное программирование»**

**по лабораторной работе № 4**

**по теме: «Множественное наследование в C++»**

|  |  |
| --- | --- |
| Студент группы ИДБ-23-08 | Прохоров А.А. |
|  |  |
| Старший преподаватель | Варварюк А.В. |

# Задание 1(ВАРИАНТ 2)

**Описание**

Написать программу, в которой описана иерархия классов: средство передвижения (велосипед, автомобиль, грузовик). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.

**Использованные переменные**



**Программа**

#include <iostream>

#include <vector>

#include <memory>

#include <string>

#include <cstdlib>

using namespace std;

class Movement

{

public:

virtual ~Movement() = default;

virtual void gg() {

cout << "Средства передвижения\n";

};

};

class car : public Movement {

void gg() override {

cout << "Передвижние на машине\n";

}

};

class bicycle : public Movement {

void gg() override {

cout << "Передвижение на велосипеде\n";

}

};

class truck : public Movement {

void gg() override {

cout << "Передвижение на грузовике\n";

}

};

class OverflowError : public Movement {

public:

void gg() override {

cout << "Ошибка переполнения\n";

}

};

template <typename T>

class PointerArray {

private:

vector<T\*> items;

public:

void add(T\* item) {

items.push\_back(item);

}

T\* operator[](std::size\_t index) {

if (index >= items.size()) {

throw out\_of\_range("Индекс выходит за пределы допустимого диапазона\n");

}

return items[index];

}

~PointerArray() {

for (T\* item : items) {

delete item;

}

}

};

int main() {

setlocale(LC\_ALL, "ru");

try {

PointerArray<Movement> errors;

errors.add(new car());

errors.add(new bicycle());

errors.add(new truck());

for (int i = 0; i < 3; i++) {

errors[i]->gg();

}

errors[10]->gg();

}

catch (const out\_of\_range& e) {

cerr << "Исключения из диапазона" << e.what() << endl;

}

catch (const exception& e) {

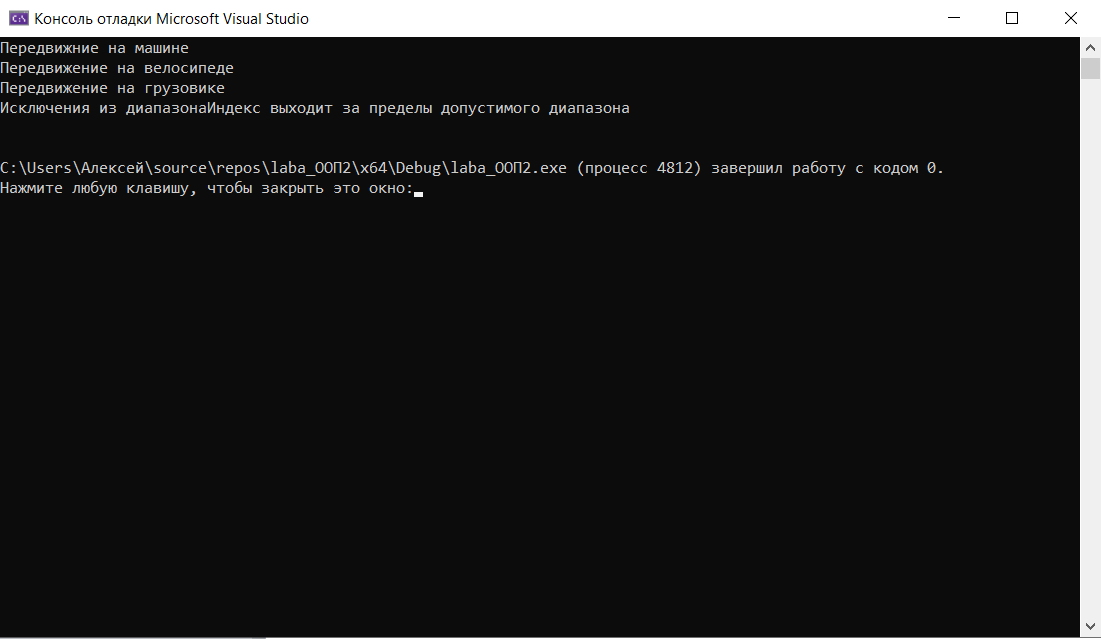
cerr << "Исключение" << e.what() << endl;

}

return 0;

}

**Примеры работы программы**



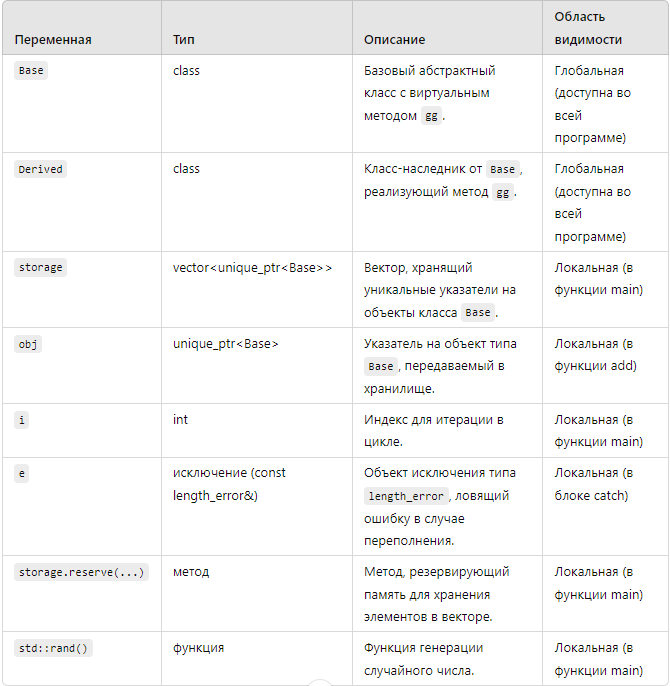
# Задание 2

**Описание**

В программе необходимо дополнительно реализовать обработку одного из исключений (вариант выбрать по номеру списка группы):

8. length\_error: исключение, которое возникает при попытке создать объект большего размера, чем допустим для данного типа.

**Использованные переменные**



**Программа**

#include <iostream>

#include <vector>

#include <memory>

#include <cstdlib>

#include <ctime>

#include <stdexcept>

using namespace std;

class Base {

public:

virtual ~Base() = default;

virtual void gg() const = 0;

};

class Derived : public Base {

public:

void gg() const override {

cout << "This is Derived class.\n";

}

};

void add(vector<unique\_ptr<Base>>& storage, unique\_ptr<Base> obj) {

storage.push\_back(move(obj));

}

int main() {

try {

vector<unique\_ptr<Base>> storage;

storage.reserve(static\_cast<size\_t>(-1));

for (int i = 0; i < 10; ++i) {

if (std::rand() % 2 == 0) {

add(storage, make\_unique<Derived>());

}

else {

add(storage, make\_unique<Derived>());

}

}

for (const auto& obj : storage) {

obj->gg();

}

}

catch (const length\_error& e) {

cerr << "Caught length\_error: " << e.what() << "\n";

}

catch (const std::exception& e) {

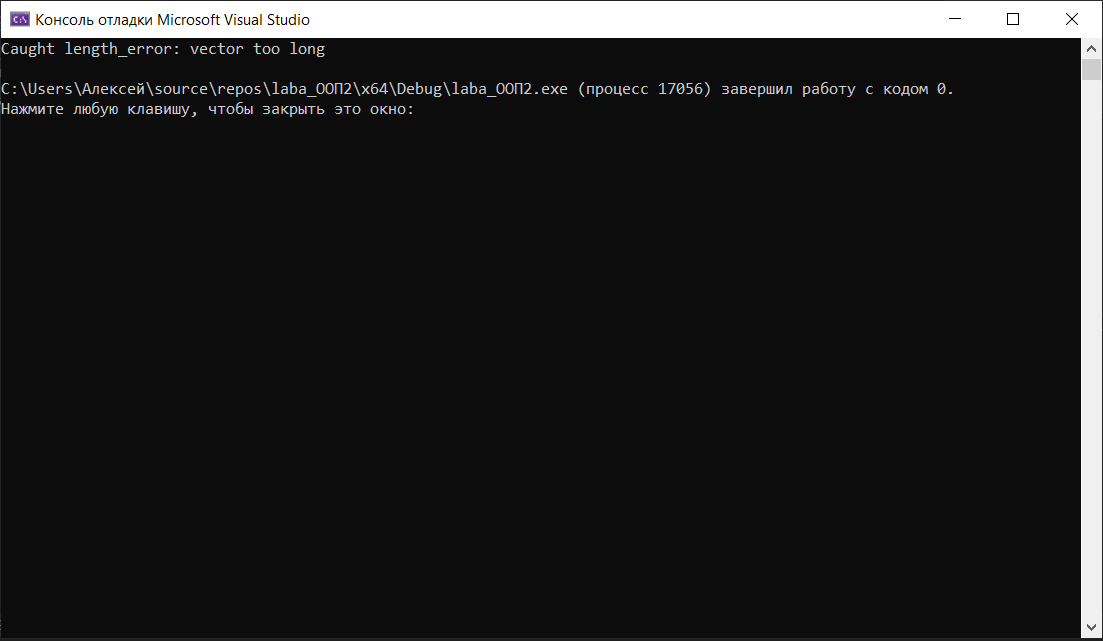
cerr << "Caught exception: " << e.what() << "\n";

}

return 0;

}

**Примеры работы программы**



**Вывод**

В ходе выполнения лабораторной работы была разработана и реализована иерархия классов для представления ошибок и их обработки. Шаблонный класс PointerArray продемонстрировал возможность работы с динамическими массивами указателей и корректную обработку исключений. Программа не только обрабатывает ошибки, но и освобождает всю выделенную память, предотвращая утечки. Таким образом, полученный результат отвечает всем заданным требованиям.