Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

за 1 семестр

По дисциплине: «Языки программирования»

Тема: «Перезагрузка операций и принципов обработки исключений»

Выполнил:

Студент 2 курса

Группы ПО-4 (2)

Синяк Д.А

Проверил:

Хацкевич М. В.

2020

**Лабораторная работа №3**

Перезагрузка операций и принципов обработки исключени

**Цель работы:** *получить практические навыки создания иерархии классов и использования статических компонентов класса.*

***Вариант 2***

**Задание:**

Написать программу, в которой описана иерархия классов: средство передвижения

(велосипед, автомобиль, грузовик). Описать класс для хранения коллекции средств

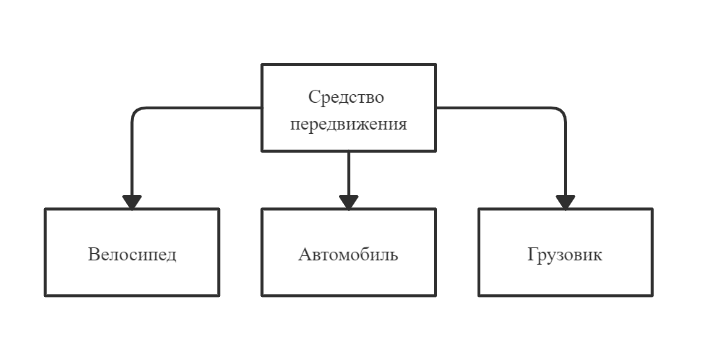
передвижений (массива указателей на базовый класс), в котором перегрузить операцию «[]».

Для базового класса и его потомков перегрузить операции «==», «!=», «=».

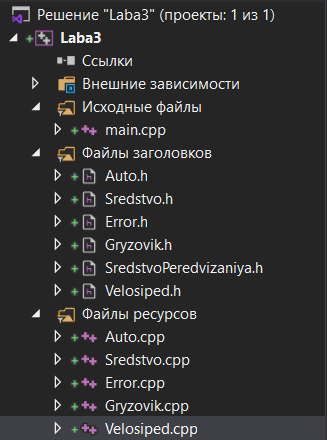
Продемонстрировать работу операторов.

***Ход работы***

1. **Иерархия классов в виде графов:**



1. **Код программы:**



**Main.cpp**

#include "Auto.h"

#include "Error.h"

#include "Gryzovik.h"

#include "SredstvoPeredvizaniya.h"

#include "Velosiped.h"

#include "Sredstvo.h"

void menu(Sredstvo arr);

int main() {

setlocale(LC\_ALL, "ru");

system("color f0");

cout << "Задание. Вариант 2." << endl

<< "Написать программу, в которой описана иерархия классов: геометрические передвижения (велосипед, автомобиль, грузовик)."

<< endl << "писать класс для хранения коллекции фигур (массива указателей на базовый класс), в котором пергрузить операцию [ ],"

<< endl << "операции ==, !=, =. Продеманстрировать работу операторов." << endl;

Sredstvo arr(0);

menu(arr);

return 0;

}

void menu(Sredstvo arr) {

bool exit = false;

while (!exit) {

cout << "1 - добавить Автомобиль" << endl

<< "2 - добавить Велосипед" << endl

<< "3 - добавить Грузовик" << endl

<< "4 - сравнить Грузовики" << endl

<< "5 - показать средство передвижения по индексу" << endl

<< "6 - показать все средства передвижения" << endl

<< "7 - удалить с конца" << endl

<< "8 - удалить по индексу" << endl

<< "0 - Exit" << endl;

try

{

int num;

cin >> num;

cout << endl;

switch (num)

{

case 1:

{

Auto\* obj1 = new Auto;

obj1->Read();

Auto\* obj2 = new Auto;

obj2 = obj1;

bool temp = false;

cout << "Добавить в конец(0) или по индексу(1)?";

cin >> temp;

if (temp) {

int index;

cout << "Введите индекс: ";

cin >> index;

arr.add(index, obj2);

}

else {

arr.addToTheEnd(obj2);

}

break;

}

case 2:

{

Velosiped\* obj = new Velosiped;

obj->Read();

bool temp = false;

cout << "Добавить в конец(0) или по индексу(1)?";

cin >> temp;

if (temp) {

int index;

cout << "Введите индекс: ";

cin >> index;

arr.add(index, obj);

}

else {

arr.addToTheEnd(obj);

}

break;

}

case 3:

{

Gryzovik\* obj = new Gryzovik;

obj->Read();

bool temp = false;

cout << "Добавить в конец(0) или по индексу(1)?";

cin >> temp;

if (temp) {

int index;

cout << "Введите индекс: ";

cin >> index;

arr.add(index, obj);

}

else {

arr.addToTheEnd(obj);

}

break;

}

case 4:

{

Gryzovik obj1;

obj1.Read();

Gryzovik obj2;

obj2.Read();

if (obj1 == obj2) {

cout << "Грузовики одинаковые" << endl;

}

else if (obj1 != obj2) {

cout << "Грузовики не одинаковые" << endl;

}

break;

}

case 5:

{

int index;

cout << "Введите индекс: ";

cin >> index;

if (arr[index] != 0) {

arr[index]->Print();

}

break;

}

case 6:

{

cout << "Количество средств передвижения= " << arr.get\_count() << endl;

for (int i = 0; i < arr.get\_count(); i++) {

arr[i]->Print();

}

break;

}

case 7:

{

arr.deleteFromTheEnd();

break;

}

case 8:

{

int index;

cout << "Введите индекс: ";

cin >> index;

arr.del(index);

break;

}

case 0:

exit = true;

}

cout << endl;

}

catch (IndexError& e)

{

cout << "Ошибка индекса: " << e.get\_message() << endl;

}

catch (...)

{

cout << "Неизвестная ошибка." << endl;

}

}

}

**Auto.h**

#pragma once

#include "SredstvoPeredvizaniya.h"

class Auto : public SredstvoPeredvizaniya

{

private:

int Year; /// год выпуска.

public:

Auto();

Auto(int Year);

Auto(const Auto& other);

~Auto();

bool operator == (const Auto& right);

bool operator != (const Auto& right);

Auto& operator = (const Auto& right);

void Print();

void Read();

};

**Auto.cpp**

#include "Auto.h"

Auto::Auto() {}

Auto::Auto(int Year) {

this->Year = Year;

}

Auto::Auto(const Auto& other) {

Year = other.Year;

}

Auto::~Auto() {}

bool Auto::operator == (const Auto& right) {

return (Year == right.Year);

}

bool Auto::operator != (const Auto& right) {

return !(\*this == right);

}

Auto& Auto::operator = (const Auto& right) {

Year = right.Year;

return \*this;

}

void Auto::Print() {

cout << endl << "Машина" << ", Год выпуска" << Year << endl;

}

void Auto::Read() {

cout << "Введите год выпуска: ";

cin >> Year;

}

**Gryzovik.h**

#pragma once

#include "SredstvoPeredvizaniya.h"

class Gryzovik :public SredstvoPeredvizaniya {

private:

int GryzTonn; /// грузоподъёмность.

public:

Gryzovik();

Gryzovik(int GryzTonn);

Gryzovik(const Gryzovik& other);

~Gryzovik();

bool operator == (const Gryzovik& right);

bool operator != (const Gryzovik& right);

Gryzovik& operator = (const Gryzovik& right);

void Print();

void Read();

};

**Gryzovik.cpp**

#include "Gryzovik.h"

Gryzovik::Gryzovik() {}

Gryzovik::Gryzovik(int GryzTonn) {

this->GryzTonn = GryzTonn;

}

Gryzovik::Gryzovik(const Gryzovik& other) {

GryzTonn = other.GryzTonn;

}

Gryzovik::~Gryzovik() {}

bool Gryzovik::operator == (const Gryzovik& right) {

return (GryzTonn == right.GryzTonn);

}

bool Gryzovik::operator != (const Gryzovik& right) {

return !(\*this == right);

}

Gryzovik& Gryzovik::operator = (const Gryzovik& right) {

GryzTonn = right.GryzTonn;

return \*this;

}

void Gryzovik::Print() {

cout << "Максимальная грузоподъёмность грузовика" << GryzTonn << endl;

}

void Gryzovik::Read() {

cout << "Введите грузоподъёмность грузовика: ";

cin >> GryzTonn;

}

**Velosiped.h**

#include "SredstvoPeredvizaniya.h"

class Velosiped :public SredstvoPeredvizaniya {

private:

int MaxSpeed;

public:

Velosiped();

Velosiped(int MaxSpeed);

Velosiped(const Velosiped& other);

~Velosiped();

bool operator == (const Velosiped& right);

bool operator != (const Velosiped& right);

Velosiped& operator = (const Velosiped& right);

void Print();

void Read();

};

**Velosiped.cpp**

#include "SredstvoPeredvizaniya.h"

class Velosiped :public SredstvoPeredvizaniya {

private:

int MaxSpeed;

public:

Velosiped();

Velosiped(int MaxSpeed);

Velosiped(const Velosiped& other);

~Velosiped();

bool operator == (const Velosiped& right);

bool operator != (const Velosiped& right);

Velosiped& operator = (const Velosiped& right);

void Print();

void Read();

};

**Sredstvo.h**

#pragma once

#include "SredstvoPeredvizaniya.h"

#include "Error.h"

class Sredstvo {

private:

SredstvoPeredvizaniya\*\* arr;

int count;

public:

Sredstvo(int count);

~Sredstvo();

SredstvoPeredvizaniya\* operator[] (int n) const;

SredstvoPeredvizaniya\*& operator[] (int n);

int get\_count();

void addToTheEnd(SredstvoPeredvizaniya\* vehicle);

void add(int index, SredstvoPeredvizaniya\* vehiclee);

void deleteFromTheEnd();

void del(int index); };

**Sredstvo.cpp**

#include "Sredstvo.h"

Sredstvo::Sredstvo(int count) {

arr = new SredstvoPeredvizaniya \* [count];

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

arr[i] = nullptr;

}

this->count = count;

}

Sredstvo::~Sredstvo() {

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

if (arr[i] != nullptr) {

delete arr[i];

}

delete[]arr;

}

}

SredstvoPeredvizaniya\* Sredstvo::operator[] (int n) const {

if (n < 0 || n >= count) {

throw IndexError("Вы вышли за границы массива.");

}

return arr[n];

}

SredstvoPeredvizaniya\*& Sredstvo::operator[] (int n) {

if (n < 0 || n >= count) {

throw IndexError("Вы вышли за границы массива.");

}

return arr[n];

}

int Sredstvo::get\_count() {

return count;

}

void Sredstvo::addToTheEnd(SredstvoPeredvizaniya\* figure) {

SredstvoPeredvizaniya\*\* temp = new SredstvoPeredvizaniya \* [count + 1];

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

temp[i] = arr[i];

}

arr = temp;

temp[count] = figure;

count++;

}

void Sredstvo::add(int index, SredstvoPeredvizaniya\* figure) {

if (index < 0 || index > count) {

addToTheEnd(figure);

throw IndexError("Вы вышли за границы массива. Фигура будет добавлена в конец массива");

}

SredstvoPeredvizaniya\*\* temp = new SredstvoPeredvizaniya \* [count + 1];

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

temp[i] = arr[i];

}

temp[index] = figure;

for (int i = index; i < count; i++) {

temp[i + 1] = arr[i];

}

arr = temp;

count++;

}

void Sredstvo::deleteFromTheEnd() {

SredstvoPeredvizaniya\*\* temp = new SredstvoPeredvizaniya \* [count - 1];

for (int i = 0; i < count - 1; i++) {

temp[i] = arr[i];

}

arr = temp;

count--;

}

void Sredstvo::del(int index) {

SredstvoPeredvizaniya\*\* temp = new SredstvoPeredvizaniya \* [count - 1];

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

temp[i] = arr[i];

}

for (int i = index + 1; i < count; i++) {

temp[i - 1] = arr[i];

}

arr = temp;

count--;

}

**Error.h**

#pragma once

#include <iostream>

class IndexError {

protected:

char\* message;

public:

IndexError(const char\* message);

~IndexError();

char\* get\_message();

};

**Error.cpp**

#include "error.h"

IndexError::IndexError(const char\* message) {

this->message = \_strdup(message);

}

IndexError::~IndexError() {

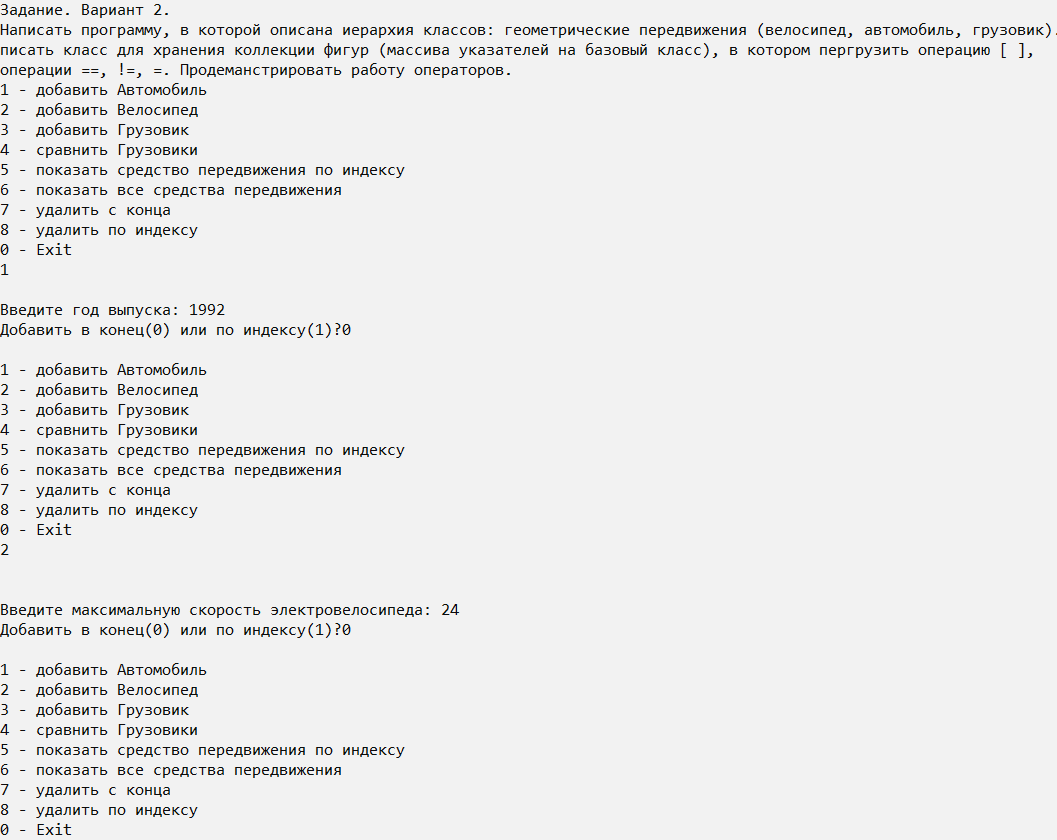
delete this->message;

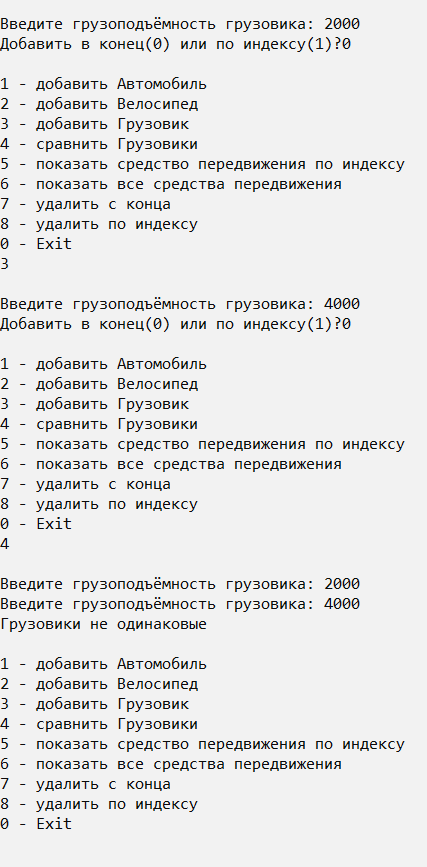
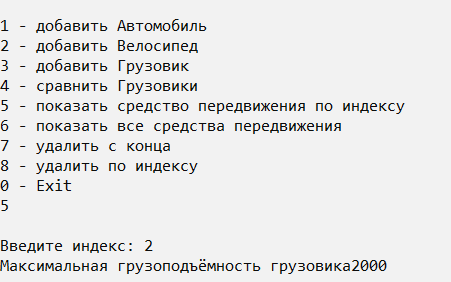
}

char\* IndexError::get\_message() {

return message; }

**Результат программы:**

****

****