Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**Тема:** Лабораторная работа №8 по ООП

Семестр: 2

Выполнил студент ИВТ-23-2б:

Синицын Владислав Сергеевич

(дата, подпись)

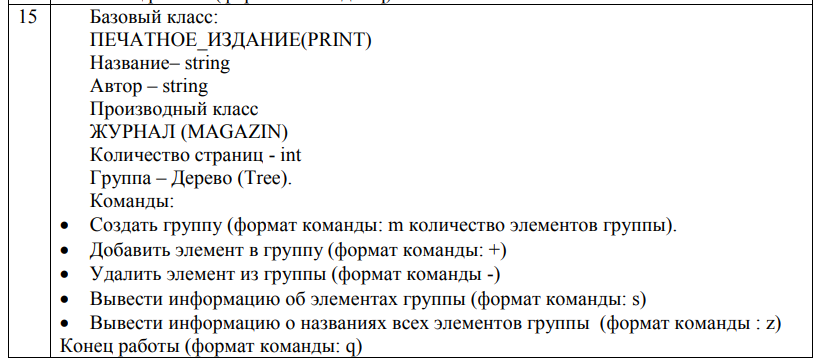
Проверила:

Полякова Ольга Андреевна

(дата, подпись)

Пермь 2024

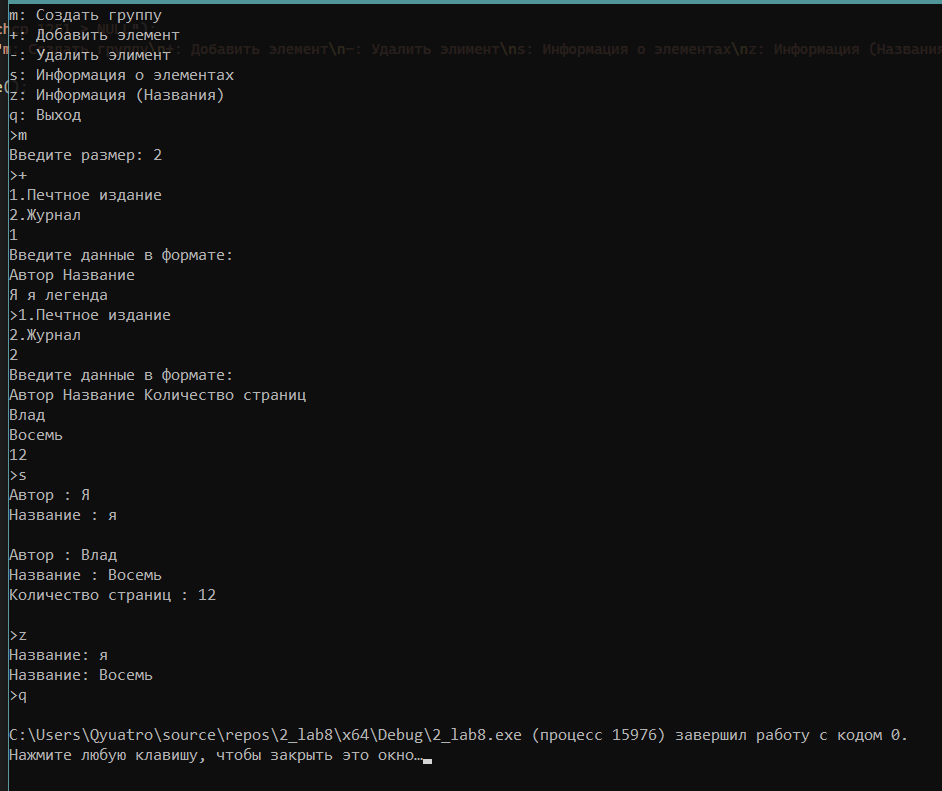
**Задание**

****

**UML-Диаграмма**



**Программное решение**

****

**2\_lab8.cpp**

#include <iostream>

#include "Magazin.h"

#include "Vector.h"

#include "Dialog.h"

using namespace std;

int main() {

system("chcp 1251 > NULL");

cout << "m: Создать группу\n+: Добавить элемент\n-: Удалить элимент\ns: Информация о элементах\nz: Информация (Названия)\nq: Выход\n";

Dialog D;

D.Execute();

return 0;

}

**Dialog.h**

#pragma once

#include "Vector.h"

#include "Event.h"

class Dialog : public Vector {

public:

Dialog();

virtual ~Dialog(void);

virtual void GetEvent(TEvent& Event);

virtual void Execute();

virtual void HandleEvent(TEvent& Event);

virtual void ClearEvent(TEvent& Event);

bool Valid();

void EndExec();

protected:

int EndState;

};

**Dialog.cpp**

#include "Dialog.h"

#include "Event.h"

#include "Vector.h"

#include <iostream>

using namespace std;

Dialog::Dialog() : Vector() {

EndState = 0;

}

Dialog::~Dialog(void) {}

void Dialog::GetEvent(TEvent& Event) {

string OpInt = "+m-szq";

string s;

string param;

char code;

cout << '>';

cin >> s; code = s[0];

if (OpInt.find(code) >= 0) {

Event.what = evMessage;

switch (code) {

case 'm':Event.command = cmMake; break;

case '+': Event.command = cmAdd; break;

case '-': Event.command = cmDel; break;

case 's': Event.command = cmShow; break;

case'q': Event.command = cmQuit; break;

case 'z':Event.command = cmGet; break;

}

}

else Event.what = evNothing;

}

void Dialog::Execute() {

TEvent Event;

do {

EndState = 0;

GetEvent(Event);

HandleEvent(Event);

} while (Valid());

}

bool Dialog::Valid() { return (EndState == 0); }

void Dialog::ClearEvent(TEvent& Event) {

Event.what = evNothing;

}

void Dialog::EndExec() {

EndState = 1;

}

void Dialog::HandleEvent(TEvent& Event) {

if (Event.what == evMessage) {

switch (Event.command) {

case cmMake:

cout << "Введите размер: ";

cin >> size;

beg = new Object \* [size];

cur = 0;

ClearEvent(Event);

break;

case cmAdd:

Add();

ClearEvent(Event);

break;

case cmDel:Del();

ClearEvent(Event);

break;

case cmShow:Show();

ClearEvent(Event);

break;

case cmQuit:EndExec();

ClearEvent(Event);

break;

default:

Vector::HandleEvent(Event);

break;

};

};

}

**Evant.h**

#pragma once

const int evNothing = 0;

const int evMessage = 100;

const int cmAdd = 1;

const int cmDel = 2;

const int cmGet = 3;

const int cmShow = 4;

const int cmMake = 6;

const int cmQuit = 101;

struct TEvent {

int what;

int command;

//union {

// int command;

// struct {

// int message;

// int a;

// };

//};

};

**Magazin.h**

#pragma once

#include "Print.h"

class Magazin : public Print {

public:

Magazin();

~Magazin() {};

void Show();

void Input();

Magazin(string, string, int);

Magazin(const Magazin&);

int Get\_pages() {

return pages;

}

void Set\_pages(int);

Magazin& operator=(const Magazin&);

protected:

int pages;

};

**Magazin.cpp**

#include "Magazin.h"

Magazin::Magazin(void) :Print() {

pages = 0;

}

Magazin::Magazin(string m\_author, string m\_name, int m\_pages) : Print(m\_author, m\_name) {

pages = m\_pages;

}

Magazin::Magazin(const Magazin& L) {

author = L.author;

name = L.name;

pages = L.pages;

}

void Magazin::Set\_pages(int m\_pages) {

pages = m\_pages;

}

Magazin& Magazin::operator=(const Magazin& l) {

if (&l == this) {

author = l.author;

name = l.name;

pages = l.pages;

}

return \*this;

}

void Magazin::Show() {

cout << "Автор : " << author << '\n';

cout << "Название : " << name << '\n';

cout << "Количество страниц : " << pages << "\n\n";

}

void Magazin::Input() {

cout << "Введите данные в формате:\nАвтор Название Количество страниц\n";

cin >> author >> name >> pages;

}

**Object.h**

#pragma once

#include "Event.h"

class Object {

public:

Object(void) {};

virtual void Show() = 0;

virtual void Input() = 0;

virtual ~Object(void) {};

virtual void HandleEvent(const TEvent& e) = 0;

};

**Print.h**

#include "Object.h"

#include <iostream>

using namespace std;

class Print :public Object {

public:

Print();

virtual ~Print() {};

void Show();

void Input();

Print(string, string);

Print(const Print&);

string Get\_author() {

return author;

}

string Get\_name() {

return name;

}

void Set\_author(string);

void Set\_name(string);

Print& operator=(const Print&);

void HandleEvent(const TEvent& e);

protected:

string author, name;

};

**Print.cpp**

#include "Print.h"

#include "Object.h"

#include "Event.h"

#include <iostream>

using namespace std;

void Print::HandleEvent(const TEvent& e) {

if (e.what == evMessage) {

switch (e.command) {

case cmGet:

cout << "Название: " << Get\_name() << '\n';

break;

}

}

}

Print::Print() {

author = "";

name = "";

}

Print::Print(string m\_author, string m\_name) {

author = author;

name = m\_name;

}

Print::Print(const Print& Print) {

author = Print.author;

name = Print.name;

}

void Print::Set\_name(string m\_name) {

name = m\_name;

}

void Print::Set\_author(string m\_name) {

author = m\_name;

}

Print& Print::operator=(const Print& c) {

if (&c != this) {

author = c.author;

name = c.name;

}

return \*this;

}

void Print::Show() {

cout << "Автор : " << author << '\n';

cout << "Название : " << name << "\n\n";

}

void Print::Input() {

cout << "Введите данные в формате:\nАвтор Название\n";

cin >> author >> name;

}

**Vector.h**

#pragma once

#include "Object.h"

#include "event.h"

class Vector {

public:

Vector(int);

Vector();

void HandleEvent(const TEvent& e);

~Vector(void);

void Add();

void Del();

void Show();

int operator()();

protected:

Object\*\* beg;

int size;

int cur;

};

**Vector.cpp**

#include "Vector.h"

#include "Magazin.h"

#include "Event.h"

#include <iostream>

using namespace std;

void Vector::HandleEvent(const TEvent& e) {

if (e.what == evMessage) {

Object\*\* p = beg;

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

(\*p)->HandleEvent(e);

p++;

}

}

}

Vector::~Vector() {

if (beg != 0) {

delete[] beg;

}

beg = 0;

}

Vector::Vector(int n)

{

beg = new Object \* [n];

cur = 0;

size = n;

}

Vector::Vector() {

beg = NULL;

cur = -1;

size = 0;

}

void Vector::Add() {

Object\* p;

cout << "1.Печтное издание\n";

cout << "2.Журнал\n";

int y;

cin >> y;

if (y == 1) {

Print\* a = new (Print);

a->Input();

p = a;

if (cur < size) {

beg[cur] = p;

cur++;

}

}

else if (y == 2) {

Magazin\* b = new Magazin;

b->Input();

p = b;

if (cur < size) {

beg[cur] = p;

cur++;

}

}

else return;

}

void Vector::Show() {

int k;

cout << "Введите индекс элемента: ";

cin >> k;

if (k < cur && k > -1) {

beg[k]->Show();

}

}

int Vector::operator ()() {

return cur;

}

void Vector::Del() {

if (cur == 0) {

return;

}

cur--;

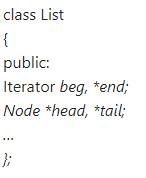
}

**Контрольные вопросы**

1.      Что такое класс-группа? Привести примеры таких классов.

Ответ: класс-группа — объект, в который включены другие объекты.

2.      Привести пример описания класса-группы Список (List).



|  |
| --- |
|  |
|  |  |

3. Привести пример конструктора (с параметром, без параметров, копирования) для класса-группы Список.

|  |
| --- |
|  |
|  |  |

4.      Привести пример деструктора для класса-группы Список.

|  |
| --- |
|  |
|  |  |

~List() {

While (head != NULL)

Pop\_front();

}

5.      Привести пример метода для просмотра элементов для класса-группы Список.

|  |
| --- |
|  |
|  |  |

6.      Какой вид иерархии дает группа?

Иерархию объектов

7.      Почему во главе иерархии классов, содержащихся в группе объектов должен находится абстрактный класс?

Для удобного расширения кода

8.      Что такое событие? Для чего используются события?

Событие — информация, полученная в ходе действий пользователя.

События используют для взаимодействия пользователя с интерфейсом программы.

9.      Какие характеристики должно иметь событие-сообщение?

Событие-сообщение должно иметь код сообщения.

10.   Привести пример структуры, описывающей событие.

|  |
| --- |
|  |
|  |  |

11.  Задана структура события:

*struct TEvent*

*{*

*int what;*

*union*

*{*

*MouseEventType mouse;*

*KeyDownEvent keyDown;*

*MessageEvent message;*

*}*

*};*

Какие значения, и в каких случаях присваиваются полю *what*?

Полю what присваиваются значения пришло ли событие.

12.  В: Задана структура события:

*struct TEvent*

*{*

*int what;*

*union*

*{*

*int command;*

*struct*

*{*

*int message;*

*int a;*

*};*

*}*

*};*

Какие значения, и в каких случаях присваиваются полю *command*?

Полю command присваиваются значения типа события в случае требуемого what

13.  Задана структура события:

*struct TEvent*

*{*

*int what;*

*union*

*{*

*int command;*

*struct*

*{*

*int message;*

*int a;*

*};*

*}*

*};*

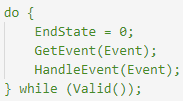
Для чего используются поля a и message?

message – данные

14. Какие методы необходимы для организации обработки сообщений?

Метод получения события и метод обработчик сообщений

15. Какой вид имеет главный цикл обработки событий-сообщений?



16. Какую функцию выполняет метод ClearEvent()? Каким образом?

Устанавливает значения по умолчанию для данного экземпляра

17. Какую функцию выполняет метод HandleEvent()? Каким образом?

Обрабатывает событие

18. Какую функцию выполняет метод GetEvent ()?

Получает новое событие

19. Для чего используется поле EndState? Какой класс (объект) содержит это поле?

EndState – это поле для отображения завершения обработки

20. Для чего используется функция Valid()?

Проверка значения EndState