**Пермский национальный исследовательский политехнический университет**

Кафедра “Информационные технологии и автоматизированные системы”

**Лабораторная работа №25**

По дисциплине «Основы алгоритмизации и программирования»

**Тема:**

Программа, управляемая событиями.

Выполнила:

Студентка группы ИВТ-19-1б

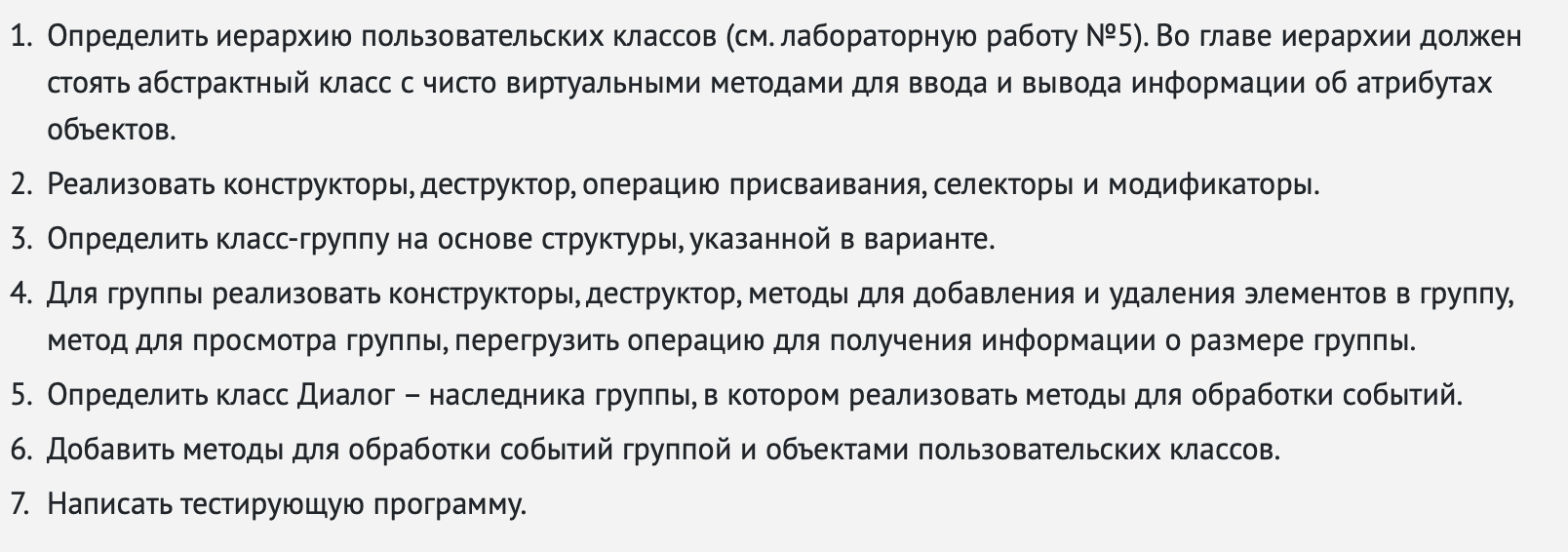
Еремеева Полина Алексеевна

Проверила:

доцент кафедры “ИТАС”

Полякова О.А.

Пермь, 2020

**Постановка задачи**

**Код**

**Main.cpp**

#include <iostream>

#include "Tree.hpp"

#include "Print.hpp"

#include "Dialog.hpp"

#include "Magazin.hpp"

**using** **namespace** std;

**int** main()

{

setlocale(LC\_ALL, "rus");

cout << "m: Создать группу\n+: Добавить элемент\n-: Удалить элемент\ns: Информация о членах группы\n";

cout << "z: Информация о названиях элементов группы\nq: Конец работы\n";

Dialog D;

D.Execute();

}

**Magazin.cpp**

#include "Magazin.hpp"

Magazin::Magazin(**void**) :Print()

{

pages = 0;

}

Magazin::~Magazin(**void**)

{

}

Magazin::Magazin(string n, string a, **int** page) :Print(n, a)

{

pages = page;

}

Magazin::Magazin(**const** Magazin& m)

{

name = m.name;

author = m.author;

pages = m.pages;

}

**void** Magazin::Set\_pages(**int** page)

{

pages = pages;

}

Magazin& Magazin::**operator**=(**const** Magazin& m)

{

**if** (&m == **this**) **return** \***this**;

name = m.name;

author = m.author;

pages = m.pages;

**return** \***this**;

}

**void** Magazin::Show()

{

cout << "\nName: " << name;

cout << "\nAuthor: " << author;

cout << "\nPages: " << pages;

cout << endl;

}

**void** Magazin::Input()

{

cout << "\nEnter name: "; cin >> name;

cout << "\nEnter author: "; cin >> author;

cout << "\nEnter pages: "; cin >> pages;

cout << endl;

}

**Magazin.hpp**

#pragma once

#include "Print.hpp"

**using** **namespace** std;

**class** Magazin :

**public** Print

{

**public**:

Magazin(**void**);

**public**:

~Magazin(**void**);

**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;

};

**Event.hpp**

#pragma once

**const** **int** evNothung = 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;

**union**

{

**int** command;

**struct**

{

**int** message;

**int** a;

};

};

};

**Dialod.cpp**

#include "Dialog.hpp"

Dialog::Dialog(**void**) :Tree()

{

EndState = 0;

}

Dialog::~Dialog(**void**)

{

}

**void** Dialog::GetEvent(TEvent& event)

{

string OpInt = "+-?/qam";

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** 'z': event.command = cmGet; **break**;

**case** 'q': event.command = cmQuit; **break**;

}

**if** (s.length() > 1)

{

param = s.substr(1, s.length() - 1);

**int** A = atoi(param.c\_str());

event.a = A;

}

}

**else** event.what = evNothung;

}

**int** Dialog::Execute()

{

TEvent event;

**do** {

EndState = 0;

GetEvent(event);

HandleEvent(event);

} **while** (!Valid());

**return** EndState;

}

**int** Dialog::Valid()

{

**if** (EndState == 0)**return** 0;

**else** **return** 1;

}

**void** Dialog::ClearEvent(TEvent& event)

{

event.what = evNothung;

}

**void** Dialog::EndExec()

{

EndState = 1;

}

**void** Dialog::HandleEvent(TEvent& event)

{

**if** (event.what == evMessage)

{

**switch** (event.command)

{

**case** cmMake:

cout << "Enter size: ";

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**:Tree::HandleEvent(event);

};

};

}

**Dialod.hpp**

#pragma once

#include <iostream>

#include "Event.hpp"

#include "Tree.hpp"

**using** **namespace** std;

**class** Dialog :

**public** Tree

{

**public**:

Dialog(**void**);

**public**:

**virtual** ~Dialog(**void**);

**virtual** **void** GetEvent(TEvent& event);

**virtual** **int** Execute();

**virtual** **void** HandleEvent(TEvent& event);

**virtual** **void** ClearEvent(TEvent& event);

**int** Valid();

**void** EndExec();

**protected**:

**int** EndState;

};

**Object.cpp**

#include "Object.hpp"

Object::Object(**void**)

{

}

Object::~Object(**void**)

{

}

**Object.hpp**

#pragma once

#include "Event.hpp"

**class** Object

{

**public**:

Object(**void**);

**virtual** **void** Show() = 0;

**virtual** **void** Input() = 0;

**virtual** ~Object(**void**);

**virtual** **void** HandEvent(**const** TEvent& e) = 0;

};

**Tree.cpp**

#include "Tree.hpp"

Tree::Tree()

{

beg = 0;

cur = 0;

size = 0;

}

Tree::~Tree(**void**)

{

**if** (beg != 0)**delete**[] beg;

beg = 0;

}

Tree::Tree(**int** n)

{

beg = **new** Object \* [n];

cur = 0;

size = n;

}

**void** Tree::Add()

{

Object\* p;

cout << "1. Print" << endl;

cout << "2. Magazin" << endl;

**int** x;

cin >> x;

**if** (x == 1)

{

Print\* a = **new** (Print);

a->Input();

p = a;

**if** (cur < size)

{

beg[cur] = p;

cur++;

}

}

**else**

**if** (x == 2)

{

Magazin\* b = **new** Magazin;

b->Input();

p = b;

**if** (cur < size)

{

beg[cur] = p;

cur++;

}

}

**else** **return**;

}

**void** Tree::Add(Object\* o)

{

Object\* p = o;

beg[cur] = p;

cur++;

**return**;

}

**void** Tree::Show()

{

**if** (cur == 0) cout << "Empty" << endl;

Object\*\* p = beg;

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

{

(\*p)->Show();

p++;

}

}

**int** Tree::**operator**()()

{

**return** cur;

}

**void** Tree::HandleEvent(**const** TEvent& e)

{

**if** (e.what == evMessage)

{

Object\*\* p = beg;

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

{

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

p++;

}

}

}

**void** Tree::Del()

{

**if** (cur == 0) **return**;

cur--;

}

**Tree.hpp**

#pragma once

#include <iostream>

#include "Object.hpp"

#include "Magazin.hpp"

**using** **namespace** std;

**class** Tree

{

**public**:

Tree();

Tree(**int**);

**public**:

~Tree(**void**);

**void** Add();

**void** Add(Object\*);

**void** Del();

**void** Show();

**int** **operator**()();

**void** HandleEvent(**const** TEvent& e);

**protected**:

Object\*\* beg;

**int** size;

**int** cur;

};

**Print.cpp**

#include "Print.hpp"

#include <iostream>

**using** **namespace** std;

Print::Print(**void**)

{

name = "";

author = "";

}

Print::~Print(**void**)

{

}

Print::Print(string n, string a)

{

name = n;

author = a;

}

Print::Print(**const** Print& p)

{

name = p.name;

author = p.author;

}

**void** Print::Set\_name(string n)

{

name = n;

}

**void** Print::Set\_author(string a)

{

author = a;

}

Print& Print::**operator**=(**const** Print& p)

{

**if** (&p == **this**) **return** \***this**;

name = p.name;

author = p.author;

**return** \***this**;

}

**void** Print::Show()

{

cout << "\nName: " << name;

cout << "\nAuthor: " << author;

cout << endl;

}

**void** Print::Input()

{

cout << "\nEnter name: "; cin >> name;

cout << "\nEnter author: "; cin >> author;

cout << endl;

}

**void** Print::HandEvent(**const** TEvent& e)

{

**if** (e.what == evMessage)

{

**switch** (e.command)

{

**case** cmGet:cout << "Name: " << Get\_name() << endl;

**break**;

}

}

}

**Print.hpp**

#pragma once

#include <iostream>

#include "Object.hpp"

#include "Event.hpp"

**using** **namespace** std;

**class** Print :

**public** Object

{

**public**:

Print(**void**);

**public**:

**virtual** ~Print(**void**);

**void** Show();

**void** Input();

Print(string, string);

Print(**const** Print&);

string Get\_name() { **return** name; }

string Get\_author() { **return** author; }

**void** Set\_name(string);

**void** Set\_author(string);

Print& **operator**=(**const** Print&);

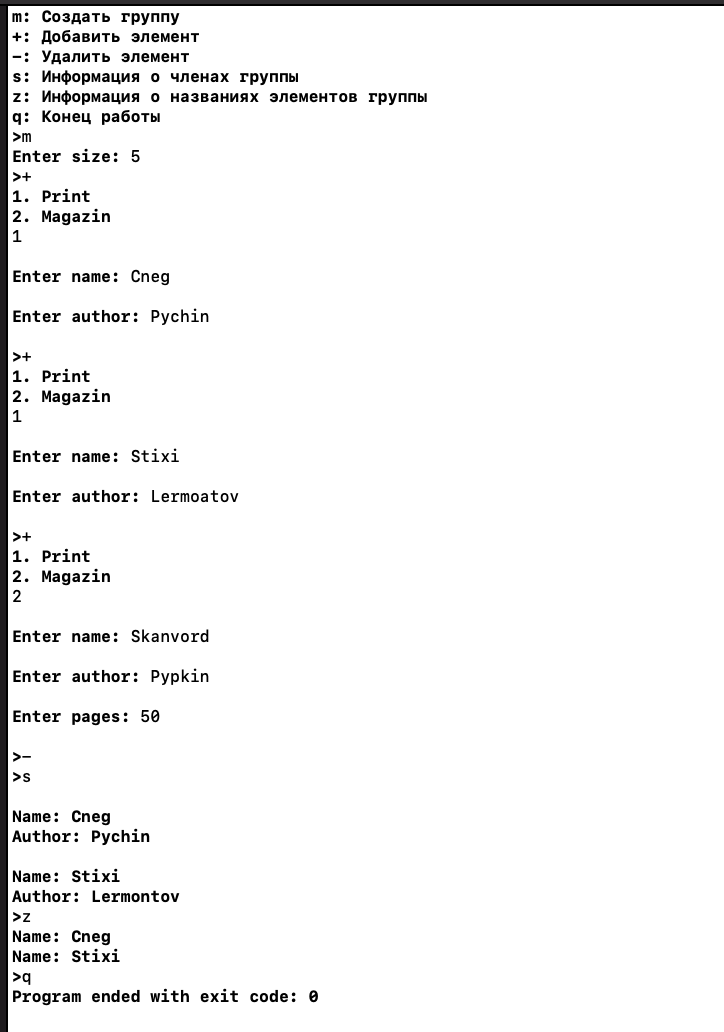
**void** HandEvent(**const** TEvent& e);

**protected**:

string name;

string author;

};

**Тестирование**