Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Программа, управляемая событиями.

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Токарев Павел Аркадьевич

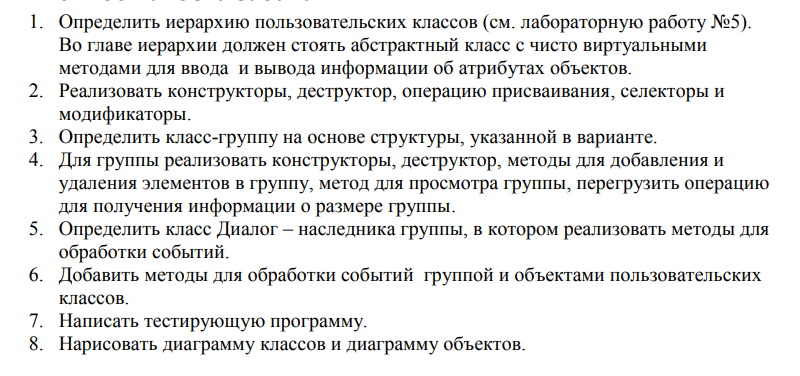
Проверил

Доцент кафедры ИТАС

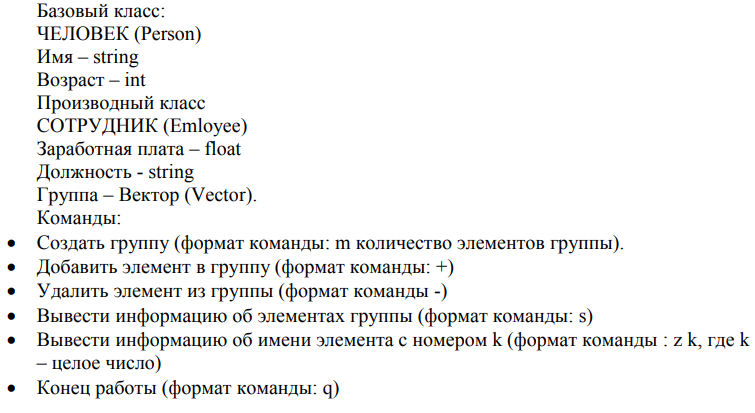
Полякова Ольга Андреевна

Г. Пермь-2023

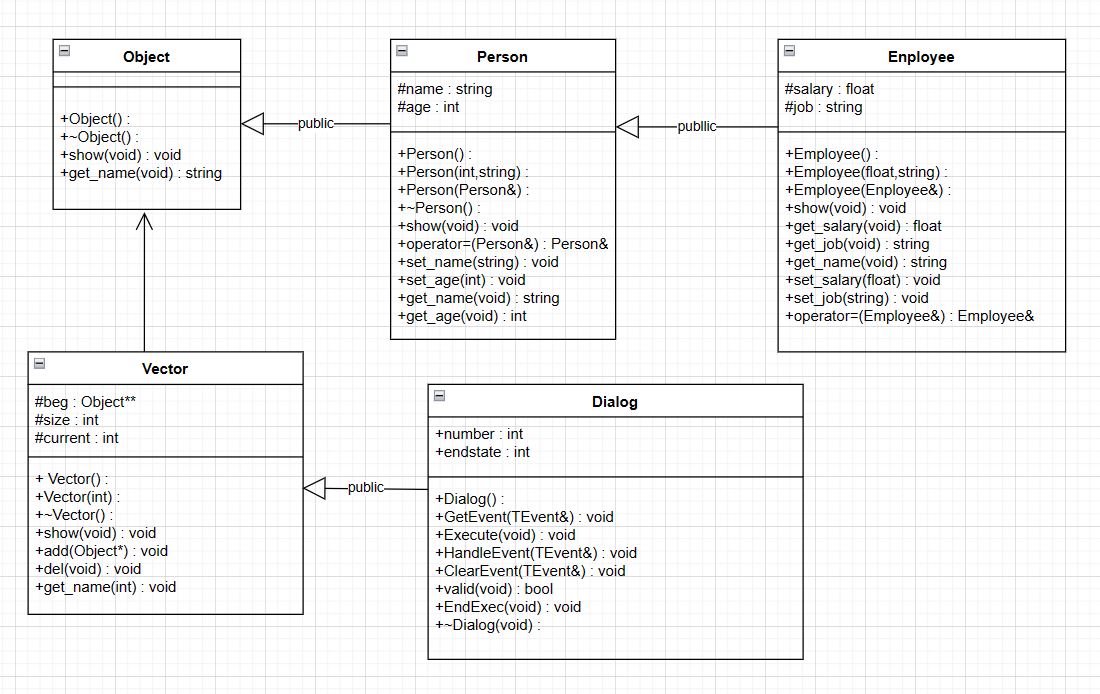
**Постановка задачи**



Вариант 2:

****

**Диаграмма UML**

****

**Код программы**

main.cpp

#include <iostream>

#include"Dialog.h"

#include"Employee.h"

#include"Event.h"

#include"Object.h"

#include"Person.h"

#include"Vector.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

cout << "m: Создать группу" << endl;

cout << "+: Добавить элемент" << endl;

cout << "-: Удалить элемент" << endl;

cout << "s: Показать элементы группы" << endl;

cout << "z: Вывести информацию о элементе" << endl;

cout << "q: Завершить работу" << endl;

Dialog D;

D.Execute();

return 0;

}

Object.h

#pragma once

#include<iostream>

using namespace std;

class Object

{

public:

Object();

virtual ~Object(void);

virtual void show() = 0;

virtual string get\_name() = 0;

};

Object.cpp

#include"Object.h"

Object::Object() {

}

Object::~Object() {

}

Person.h

#pragma once

#include <iostream>

#include"Object.h"

using namespace std;

class Person:public Object

{

protected:

string name;

int age;

public:

Person(void);

Person(int, string);

Person(const Person&);

virtual ~Person(void);

void show();

Person& operator=(const Person&);

void set\_name(string a);

void set\_age(int a);

int get\_age();

string get\_name();

friend istream& operator>>(istream& in, Person& a);

friend ostream& operator<<(ostream& out, const Person& a);

};

Person.cpp

include "Person.h"

Person::Person() {

name = "";

age = 0;

}

Person::Person(int a, string b) {

name = b;

age = a;

}

Person::Person(const Person& a) {

name = a.name;

age = a.age;

}

Person::~Person() {

}

void Person::show() {

cout << "Имя: " << name << endl;

cout << "Возрас: " << age << endl;

}

Person& Person::operator=(const Person& a) {

name = a.name;

age = a.age;

return \*this;

}

void Person::set\_name(string a) {

name = a;

}

void Person::set\_age(int a) {

age = a;

}

int Person::get\_age() {

return age;

}

string Person::get\_name() {

return name;

}

istream& operator>>(istream& in, Person& a) {

cout << "Имя: ";

in >> a.name;

cout << "Возраст: ";

in >> a.age;

return in;

}

ostream& operator<<(ostream& out, const Person& a) {

out << "Имя: " << a.name << endl;

out << "Возраст: " << a.age << endl;

return out;

}

Enployee.h

#pragma once

#include "Person.h"

using namespace std;

class Employee:

public Person

{

protected:

float salary;

string job;

public:

Employee(void);

~Employee(void);

void show();

Employee(int, string, float, string);

Employee(const Employee&);

float get\_salary();

string get\_job();

string get\_name();

void set\_salary(float a);

void set\_job(string a);

Employee& operator=(const Employee&);

friend istream& operator>>(istream& in, Employee& a);

friend ostream& operator<<(ostream& out, const Employee& a);

};

Employee.cpp

#include "Employee.h"

Employee::Employee():Person() {

salary = 0.0;

job = "";

}

Employee::Employee(int a, string b, float c, string d) :Person(a, b) {

salary = c;

job = d;

}

Employee::Employee(const Employee& a) {

age = a.age;

name = a.name;

salary = a.salary;

job = a.job;

}

Employee::~Employee() {

}

void Employee::show() {

cout << "Имя: " << name << endl;

cout << "Возраст: " << age << endl;

cout << "Зарплата: " << salary << endl;

cout << "Должность: " << job << endl;

}

float Employee::get\_salary() {

return salary;

}

string Employee::get\_job() {

return job;

}

void Employee::set\_job(string a) {

job = a;

}

void Employee::set\_salary(float a) {

salary = a;

}

Employee& Employee::operator=(const Employee& a) {

age = a.age;

name = a.name;

job = a.job;

salary = a.salary;

return \*this;

}

istream& operator>>(istream & in, Employee & a) {

cout << "Имя: ";

in >> a.name;

cout << "Возраст: ";

in >> a.age;

cout << "Зарплата: ";

in >> a.salary;

cout << "Должность: ";

in >> a.job;

return in;

}

ostream& operator<<(ostream& out, const Employee& a) {

out << "Имя: " << a.name << endl;

out << "Возраст: " << a.age << endl;

out << "Зарплата: " << a.salary << endl;

out << "Должность: " << a.job << endl;

return out;

}

string Employee::get\_name() {

return name;

}

Vector.h

#pragma once

#include<iostream>

#include"Object.h"

using namespace std;

class Vector

{

public:

Vector(void);

Vector(int);

~Vector();

void show();

void add(Object\*);

void del(void);

void getname(int);

friend ostream& operator<<(ostream& out, const Vector& t);

protected:

Object\*\* beg;

int size;

int current;

};

Vector.cpp

#include "Vector.h"

#include<iostream>

using namespace std;

Vector::Vector() {

beg = 0;

size = 0;

current = 0;

}

Vector::~Vector() {

if (beg != 0) {

delete[] beg;

}

beg = 0;

}

Vector::Vector(int n) {

beg = new Object \* [n];

current = 0;

size = n;

}

void Vector::add(Object\* p) {

if (current < size) {

beg[current] = p;

current++;

}

}

void Vector::del() {

beg[current] = 0;

current--;

}

ostream& operator<<(ostream& out, const Vector& t) {

if (t.size == 0) {

cout << "Вектор пуст" << endl;

}

Object\*\* p = t.beg;

for (int i = 0;i < t.size;i++) {

(\*p)->show();

p++;

}

return out;

}

void Vector::show() {

Object\*\* pt = beg;

cout << " Группа" << endl << endl;;

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

(\*pt)->show();

pt++;

cout << endl;

}

}

void Vector::getname(int k) {

Object\*\* pt = beg;

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

pt++;

}

cout << (\*pt)->get\_name()<<endl;

}

Dialog.h

#pragma once

#include"Vector.h"

#include"Event.h"

class Dialog:public Vector

{

public:

Dialog();

void GetEvent(TEvent& event);

void Execute();

void HandleEvent(TEvent& event);

void ClearEvent(TEvent& event);

bool valid();

void EndExec();

~Dialog();

protected:

int endstate;

int number;

};

Dialog.cpp

#include "Dialog.h"

#include"Person.h"

#include"Employee.h"

#include <iostream>

using namespace std;

void Dialog::Execute() {

TEvent event;

while (valid()) {

endstate = 0;

GetEvent(event);

HandleEvent(event);

}

}

bool Dialog::valid(){

return (endstate == 0);

}

void Dialog::HandleEvent(TEvent& event) {

if (event.what == cmsg) {

int a;

switch (event.command)

{

case cmake:

cout << "Введите размер группы: " << endl;

cin >> a;

beg = new Object \* [a];

current = 0;

size = a;

ClearEvent(event);

break;

case cadd:

Object\* ptm;

cout << "1.Person" << endl;

cout << "2.Employee" << endl;

cin >> a;

if (a == 1) {

Person \*sw=new Person;

cin >> \*sw;

ptm = sw;

add(ptm);

}

else {

Employee \*sw=new Employee;

cin >> \*sw;

ptm = sw;

add(ptm);

}

ClearEvent(event);

break;

case cdel:

del();

ClearEvent(event);

break;

case cshow:

show();

ClearEvent(event);

break;

case cinf:

cout << "Введите номер элемента" << endl;

int k;

cin >> k;

if (k < current+1) {

getname(k-1);

}

ClearEvent(event);

break;

case cquit:

EndExec();

break;

}

}

}

void Dialog::GetEvent(TEvent& event) {

string var = "m+-szq";

string s;

cin >> s;

char num = s[0];

if (var.find(num)>=0) {

event.what = cmsg;

switch (num) {

case 'm':

event.command = cmake;

break;

case '+':

event.command = cadd;

break;

case '-':

event.command = cdel;

break;

case 's':

event.command = cshow;

break;

case 'z':

event.command = cinf;

break;

case 'q':

event.command = cquit;

break;

}

}

else {

event.what = cnothing;

}

}

void Dialog::ClearEvent(TEvent& event) {

event.what = cnothing;

}

void Dialog::EndExec() {

endstate = 1;

}

Dialog::Dialog() {

endstate = 0;

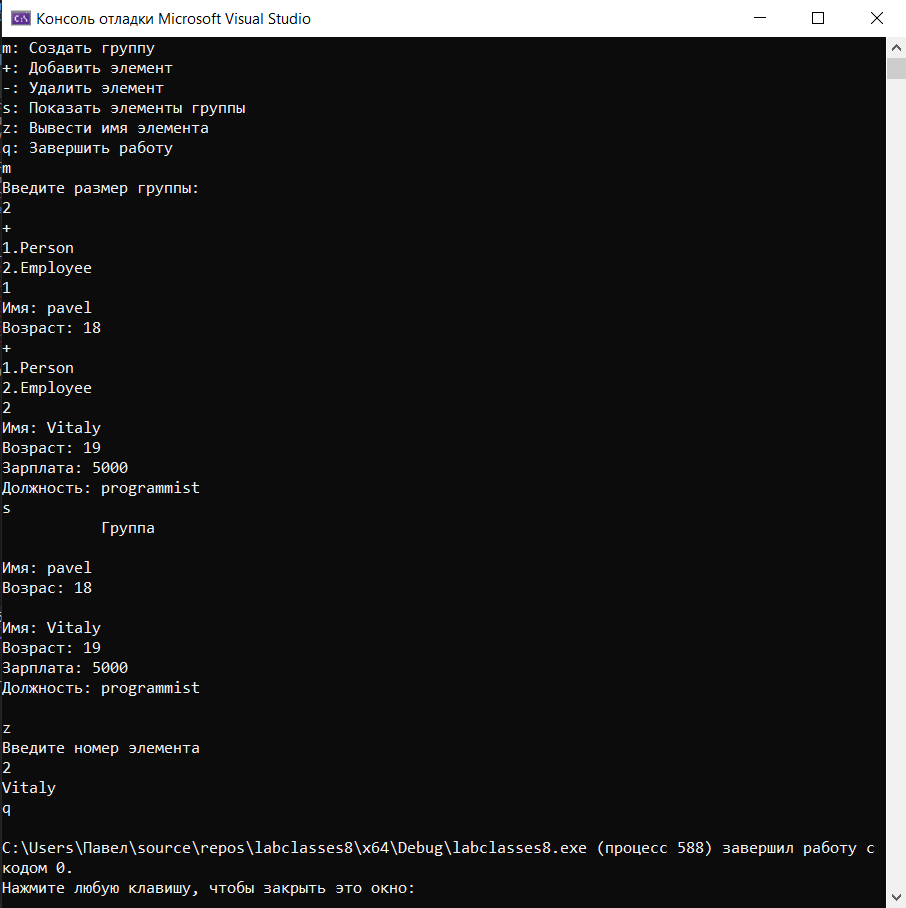
number = 0;

}

Dialog::~Dialog() {

}

**Результат работы программы**

****

**Анализ результатов**

Удалось успешно реализовать программу, управляемую событиями. Программа позволяет создавать группу из объектов разных классов.

**Контрольные вопросы**

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

Группа − это объект, в который включены другие объекты. Объекты, входящие в группу, называются элементами группы. Элементы группы, в свою очередь, могут быть группой. Окно в интерактивной программе, которое владеет такими элементами, как поля ввода и редактирования данных, кнопки, списки выбора, диалоговые окна и т.д.. Пример: студенческая группа, состоящая из студентов.

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

class List

{

public:

List(int);//конструктор с параметрами

List();

public:

~List(void);//деструктор

void Add();//добавление элемента в вектор

void Del();

void Show();

void Task();

int operator()();//размер вектора

protected:

Object\*\*beg;//указатель на первый элемент вектора

int size;//размер

int cur;//текущая позиция

};

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

List(int);//конструктор с параметрами

List();

List (const List&);//Копирования

4.Привести пример деструктора для класса-группы Список.

~List(void);//деструктор

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

void Vector::Show()

{

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

Object\*\* p = beg;//указатель на указатель типа Object

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

{

(\*p)->Show();//вызов метода Show() (позднее связывание)

p++;

}

}

6.Какой вид иерархии дает группа?

Второй.

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

От него все наследуются.

8.Что такое событие? Для чего используются события?

События лучше всего представить себе как пакеты информации, которыми обмениваются объекты и которые создаются объектно-ориентированной средой в ответ на те или иные действия пользователя.

9.Какие характеристики должно иметь событие-сообщение?

- код класса сообщения, отличающий сообщения объектов одного класса от объектов другого класса;

- адрес объекта, которому предназначено сообщение (м. б. не задан, тогда сообщение могут прочитать все объекты);

- информационное поле.

10.Привести пример структуры, описывающей событие.

struct TEvent

{

int what; //тип события

union

{

int command; //код команды

struct

{

int message;

int a; //параметр команды

};

};

};

11. Задана структура события

struct TEvent

{ int what;

union

{ MouseEventType mouse; KeyDownEvent keyDown; MessageEvent message;

} };

Какие значения, и в каких случаях присваиваются полю what?

У структуры будет размер равный максимальному типу в union + твой int what.

12. Задана структура события struct TEvent { int what;//тип события union { int command;//код комманды struct//параметры команды { int message; int a; }; }; }; Какие значения, и в каких случаях присваиваются полю command?

Код команды, которую необходимо выполнить при появлении данного события;

Когда присваивают значение command.

13. Задана структура события struct TEvent { int what;//тип события union { int command;//код комманды struct//параметры команды { int message; int a; }; }; }; Для чего используются поля a и message?

A параметр команды, message хранит в себе сообщение.

14.Какие методы необходимы для организации обработки сообщений?

evNothing – это пустое событие, которое означает, что ничего делать не надо. Полю what присваивается значение evNothing, когда событие обработано каким-либо объектом. evMessage − событие-сообщение от объекта

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

Execute

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

ClearEvent очищает событие, когда оно обработано, чтобы оно не обрабатывалось далее.

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

HandleEvent – обработчик событий. Обрабатывает каждое событие нужным для него образом. Если объект должен обрабатывать определенное событие (сообщение), то его метод HandleEvent должен распознавать это событие и реагировать на него должным образом. Событие может распознаваться, например, по коду команды (поле command).

18.Какую функцию выполняет метод GetEvent ()?

GetEvent – формирование события;

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

Конец выполнения программы, класс Dialog

20.Для чего используется функция Valid()?

Проверка атрибута EndState.