Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.8**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Программа, управляемая событиями

Вариант 8

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

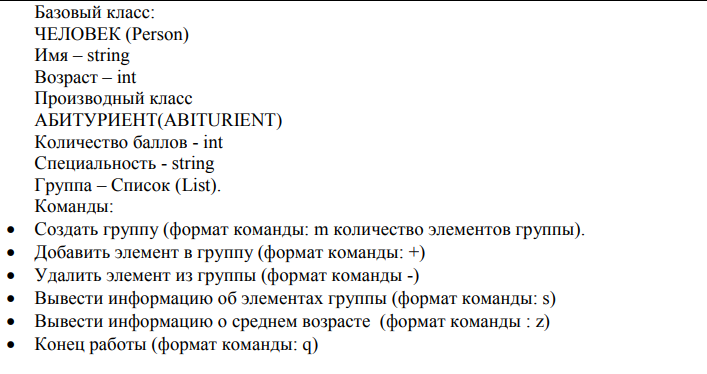
# **Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Разработка программы, управляемой событиями.

# **Постановка задачи**

1. Определитель иерархию пользовательских классов. Во главе иерархии должен стоять абстрактный класс с чисто виртуальными методами ввода и вывода информации об атрибутах объектов.
2. Реализовать конструкторы, деструкторы, селекторы и модификаторы, операцию присваивания.
3. Определить класс группу на основе структуры указанной в варианте.
4. Для группы реализовать конструкторы, деструктор, методы для добавления и удаления элементов в группу, метод просмотра группы, перегрузить операцию для получения информации о размере группы.
5. Определить класс Диалог – наследника группы, в котором реализовать методы для обработки событий.
6. Добавить методы для обработки событий группой и объектами пользовательских классов.
7. Перегрузить операции необходимые для выполнения операций контейнерного класса.
8. Написать демонстрационную программу, иллюстрирующую выполнение задачи.

(8 вариант)



# **Анализ задачи**

1. Определить какие действия предстоит выполнить:

* Разработать структуры событий Event.

const int evNothing = 0; //Пустое сообщение

const int evMessage = 100; //Непустое сообщение

const int cmAdd = 1; //Добавить объект в группу

const int cmDel = 2; //Удалить объект из группы

const int cmGet = 3; //Вывести средний возраст всех объектов

const int cmShow = 4; //Вывести всю группу

const int cmMake = 6; //Создать группу

const int cmQuit = 101; //Выход

struct TEvent {

int what;

union {

int command;

struct {

int message;

int a;

};

};

};

* Разработать абстрактный класс Object.

class Object {

public:

Object() {};

virtual ~Object() {};

virtual void Show() = 0;

virtual void Input() = 0;

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

};

* Разработать класс Person c полями name, age.
* Разработать класс Abiturient c полями points, special.
* Организовать методы для ввода данных, селекторы и модификатор, конструкторы и деструкторы, перегрузки операторов классам Person, Abiturient.

class Person : public Object {

protected:

string name;

int age;

public:

Person();

Person(string, int);

Person(const Person&);

virtual ~Person();

void Show();

void Input();

string get\_name();

int get\_age();

void set\_name(string);

void set\_age(int);

Person& operator = (const Person& );

int HandleEvent(const TEvent&);

};

class Abiturient : public Person {

protected:

int points; //Баллы

string special; //Специализация

public:

Abiturient();

Abiturient(string, int, int, string);

Abiturient(const Abiturient&);

~Abiturient();

void Show();

void Input();

int get\_points();

string get\_special();

void set\_points(int);

void set\_special(string);

Abiturient& operator = (const Abiturient&);

};

* Разработать контейнер группу List с полями head, tail, size.
* Организовать методы для ввода данных, конструкторы и деструкторы, перегрузки операторов контейнер группы List.

class List {

public:

List();

~List();

void Show();

void Push\_Front();

void Push\_Back();

void Pop\_Front();

void Pop\_Back();

virtual void HandleEvent(const TEvent&);

protected:

Node\* head;

Node\* tail;

int size;

};

* Разработать класс обработчик событий Dialog c полем EndState.
* Организовать методы обработчика событий.

class Dialog : public List {

protected:

int EndState;

public:

Dialog();

virtual ~Dialog();

virtual void GetEvent(TEvent&);

virtual int Execute();

virtual void HandleEvent(TEvent&);

virtual void ClearEvent(TEvent&);

int Valid();

void EndExec();

};

* Протестировать работу шаблон контейнера в функции main.

void main() {

system("chcp 1251");

Dialog D;

D.Execute();

system("pause");

}

1. C какими типами данных предстояло работать:

* Для хранения имени персоны реализована переменная name типа string.

string name;

* Для хранения возраста персоны реализована переменная age типа int.

int age;

* Для хранения кол-ва баллов абитуриента реализована переменная points типа int.

int points;

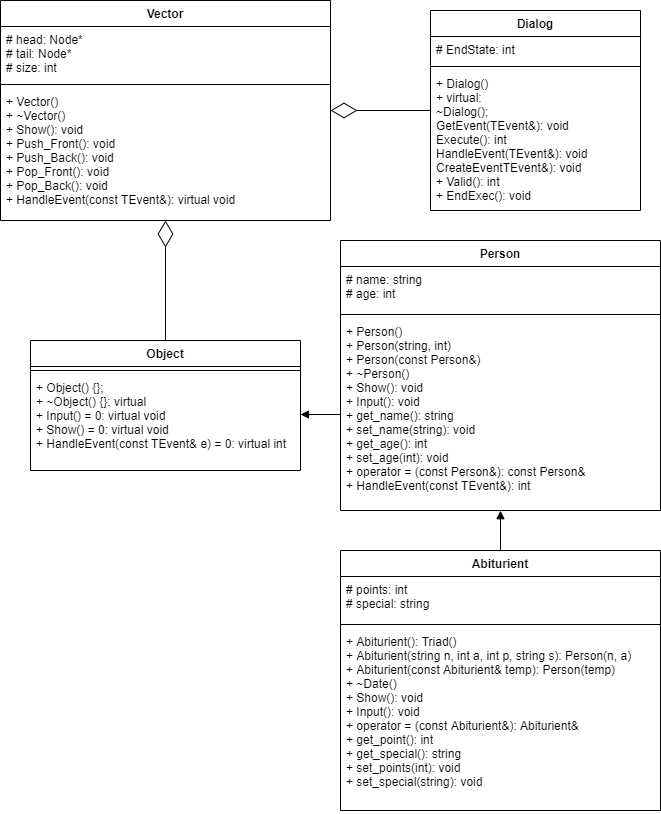
* Для хранения специализации абитуриента реализована переменная special типа string.

string special;

* Для хранения состояния обработчика событий реализована переменная EndState типа int.

int EndState;

# **UML диаграмма**



# **Код программы на языке C++**

Заголовочный файл Event.h

#pragma once

const int evNothing = 0; //Пустое сообщение

const int evMessage = 100; //Непустое сообщение

const int cmAdd = 1; //Добавить объект в группу

const int cmDel = 2; //Удалить объект из группы

const int cmGet = 3; //Вывести средний возраст всех объектов

const int cmShow = 4; //Вывести всю группу

const int cmMake = 6; //Создать группу

const int cmQuit = 101; //Выход

struct TEvent {

int what;

union {

int command;

struct {

int message;

int a;

};

};

};

Заголовочный файл Dialog.h

#pragma once

#include <iostream>

#include "List.h"

#include "Event.h"

using namespace std;

class Dialog : public List {

protected:

int EndState;

public:

Dialog();

virtual ~Dialog();

virtual void GetEvent(TEvent&);

virtual int Execute();

virtual void HandleEvent(TEvent&);

virtual void ClearEvent(TEvent&);

int Valid();

void EndExec();

};

Обьявление класса в Dialog.cpp

#include "Dialog.h"

Dialog::Dialog() : List() {

EndState = 0;

}

Dialog::~Dialog() { }

void Dialog::GetEvent(TEvent& event) {

string OpInt = "+-mszq";

string s, param;

char code;

cout << "\nm - Создать список\ns - Показать список\n+ - Добавить объект в список\n- - Удалить объект из списка\nz - Узнать средний возраст группы\nq - Выход\n\n> ";

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 = evNothing;

}

}

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 = evNothing;

}

void Dialog::EndExec() {

EndState = 1;

}

void Dialog::HandleEvent(TEvent& event) {

if (event.what == evMessage) {

switch (event.command)

{

case cmMake:

int otvet;

head = nullptr;

tail = nullptr;

for (int i = 0; i < event.a; i++) {

Push\_Back();

}

ClearEvent(event);

break;

case cmAdd:

if (size == 0) {

cout << "\nСписок не создан!\n";

}

else {

cout << "\nДобавить объект в:\n1 - Конец списка\n0 - Начало списка\n> ";

cin >> otvet;

if (otvet == 1) {

Push\_Back();

}

if (otvet == 0) {

Push\_Front();

}

}

ClearEvent(event);

break;

case cmDel:

if (size == 0) {

cout << "\nСписок пуст!\n";

}

else {

cout << "\nУдалить объект из:\n1 - Конца списка\n0 - Начала списка\n> ";

cin >> otvet;

if (otvet == 1) {

Pop\_Back();

}

if (otvet == 0) {

Pop\_Front();

}

}

ClearEvent(event);

break;

case cmShow:

if (size == 0) {

cout << "\nСписок пуст!\n";

}

else {

Show();

}

ClearEvent(event);

break;

case cmQuit:

EndExec();

ClearEvent(event);

break;

default:

if (size == 0) {

cout << "\nСписок пуст!\n";

}

else {

List::HandleEvent(event);

}

}

}

}

Заголовочный файл List.h

#pragma once

#include <iostream>

#include "Object.h"

#include "Abiturient.h"

using namespace std;

struct Node {

Object\* data;

Node\* next;

Node\* prev;

};

class List {

public:

List();

~List();

void Show();

void Push\_Front();

void Push\_Back();

void Pop\_Front();

void Pop\_Back();

virtual void HandleEvent(const TEvent&);

protected:

Node\* head;

Node\* tail;

int size;

};

Обьявление класса в List.cpp

#include "List.h"

List::List() {

head = nullptr;

tail = nullptr;

size = 0;

}

List::~List() {

while (head != nullptr) {

Pop\_Front();

}

}

void List::Show() {

Node\* temp = head;

while (temp != nullptr) {

temp->data->Show();

temp = temp->next;

}

}

void List::Push\_Front() {

cout << "\nКакой объект добавить\n1 - Person\n0 - Abiturient\n> ";

int otvet;

cin >> otvet;

if (otvet == 1) {

Person\* a = new Person;

a->Input();

Node\* temp = new Node;

temp->data = a;

if (size == 0) {

temp->next = tail;

temp->prev = head;

tail = temp;

head = temp;

}

else {

head->prev = temp;

temp->next = head;

head = temp;

head->prev = nullptr;

}

size++;

}

if (otvet == 0) {

Abiturient\* a = new Abiturient;

a->Input();

Node\* temp = new Node;

temp->data = a;

if (size == 0) {

temp->next = tail;

temp->prev = head;

tail = temp;

head = temp;

}

else {

head->prev = temp;

temp->next = head;

head = temp;

head->prev = nullptr;

}

size++;

}

}

void List::Push\_Back() {

cout << "\nКакой объект добавить\n1 - Person\n0 - Abiturient\n> ";

int otvet;

cin >> otvet;

if (otvet == 1) {

Person\* a = new Person;

a->Input();

Node\* temp = new Node;

temp->data = a;

if (size == 0) {

temp->next = tail;

temp->prev = head;

tail = temp;

head = temp;

}

else {

tail->next = temp;

temp->prev = tail;

tail = temp;

tail->next = nullptr;

}

size++;

}

if (otvet == 0) {

Abiturient\* a = new Abiturient;

a->Input();

Node\* temp = new Node;

temp->data = a;

if (size == 0) {

temp->next = tail;

temp->prev = head;

tail = temp;

head = temp;

}

else {

tail->next = temp;

temp->prev = tail;

tail = temp;

tail->next = nullptr;

}

size++;

}

}

void List::Pop\_Front() {

Node\* temp = head;

head = head->next;

delete temp;

size--;

}

void List::Pop\_Back() {

Node\* temp = tail;

tail = tail->prev;

delete temp;

tail->next = nullptr;

size--;

}

void List::HandleEvent(const TEvent& event) {

if (event.what == evMessage) {

Object\* object;

int sum = 0;

Node\* temp = head;

while (temp!=nullptr) {

object = temp->data;

sum += object->HandleEvent(event);

temp = temp->next;

}

sum = sum / size;

cout << "\nСредний возвраст = " << sum << "\n";

}

}

Заголовочный файл Object.h

#pragma once

#include "Event.h"

class Object {

public:

Object() {};

virtual ~Object() {};

virtual void Show() = 0;

virtual void Input() = 0;

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

};

Заголовочный файл Person.h

#pragma once

#include <iostream>

#include "Object.h"

using namespace std;

class Person : public Object {

protected:

string name;

int age;

public:

Person();

Person(string, int);

Person(const Person&);

virtual ~Person();

void Show();

void Input();

string get\_name();

int get\_age();

void set\_name(string);

void set\_age(int);

Person& operator = (const Person& );

int HandleEvent(const TEvent&);

};

Обьявление класса в Person.cpp

#include "Person.h"

Person::Person() {

name = "";

age = 0;

}

Person::Person(string temp\_name, int temp\_age) {

name = temp\_name;

age = temp\_age;

}

Person::Person(const Person& temp\_Person) {

name = temp\_Person.name;

age = temp\_Person.age;

}

Person::~Person() { }

void Person::Show() {

cout << "\nИмя - " << name;

cout << "\nВозраст - " << age;

cout << "\n";

}

void Person::Input() {

cout << "\nИмя:\n> "; cin >> name;

cout << "\nВозраст:\n> "; cin >> age;

}

string Person::get\_name() {

return name;

}

int Person::get\_age() {

return age;

}

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

name = temp\_name;

}

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

age = temp\_age;

}

Person& Person::operator=(const Person& temp\_Person) {

if (&temp\_Person == this) {

return \*this;

}

name = temp\_Person.name;

age = temp\_Person.age;

return \*this;

}

int Person::HandleEvent(const TEvent& event) {

if (event.what == evMessage) {

switch (event.command) {

case cmGet:

return get\_age();

}

}

}

Заголовочный файл Abiturient.h

#pragma once

#include <iostream>

#include "Person.h"

using namespace std;

class Abiturient : public Person {

protected:

int points; //Баллы

string special; //Специализация

public:

Abiturient();

Abiturient(string, int, int, string);

Abiturient(const Abiturient&);

~Abiturient();

void Show();

void Input();

int get\_points();

string get\_special();

void set\_points(int);

void set\_special(string);

Abiturient& operator = (const Abiturient&);

};

Обьявление класса в Abiturient.cpp

#include "Abiturient.h"

Abiturient::Abiturient() :Person() {

points = 0;

special = "";

}

Abiturient::Abiturient(string temp\_name, int temp\_age, int temp\_points, string temp\_special) :Person(temp\_name, temp\_age) {

points = temp\_points;

special = temp\_special;

}

Abiturient::Abiturient(const Abiturient& temp\_Abiturient) {

name = temp\_Abiturient.name;

age = temp\_Abiturient.age;

points = temp\_Abiturient.points;

special = temp\_Abiturient.special;

}

Abiturient::~Abiturient() { }

void Abiturient::Show() {

cout << "\nИмя - " << name;

cout << "\nВозраст - " << age;

cout << "\nКол-во баллов - " << points;

cout << "\nСпециализация - " << special;

cout << "\n";

}

void Abiturient::Input() {

cout << "\nИмя:\n> "; cin >> name;

cout << "\nВозраст:\n> "; cin >> age;

cout << "\nКол-во баллов:\n> "; cin >> points;

cout << "\nСпециализация:\n> "; cin >> special;

}

int Abiturient::get\_points() {

return points;

}

string Abiturient::get\_special() {

return special;

}

void Abiturient::set\_points(int temp\_points) {

points = temp\_points;

}

void Abiturient::set\_special(string temp\_special) {

special = temp\_special;

}

Abiturient& Abiturient::operator=(const Abiturient& temp\_Abiturient) {

if (&temp\_Abiturient == this) {

return \*this;

}

name = temp\_Abiturient.name;

age = temp\_Abiturient.age;

points = temp\_Abiturient.points;

special = temp\_Abiturient.special;

return \*this;

}

Main файл

#include <iostream>

#include "Abiturient.h"

#include "List.h"

#include "Dialog.h"

using namespace std;

void main() {

system("chcp 1251");

Dialog D;

D.Execute();

system("pause");

}

# **Скриншоты тестов**

