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

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

# Лабораторная работа " Программа, управляемая событиями."

Выполнил студент гр. РИС-24-3б

Жиряков Леонид Антонович

Проверил:

Доцент кафедры ИТАС   
Ольга Андреевна Полякова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г. Пермь, 2024

Вариант 8.

Постановка задачи:

Общая:

1. Определить иерархию пользовательских классов (см. лабораторную работу №5). Во главе иерархии должен стоять абстрактный класс с чисто виртуальными методами для ввода и вывода информации об атрибутах объектов.

2. Реализовать конструкторы, деструктор, операцию присваивания, селекторы и модификаторы.

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 для просмотра группы, перегрузить операцию для получения информации о размере группы.

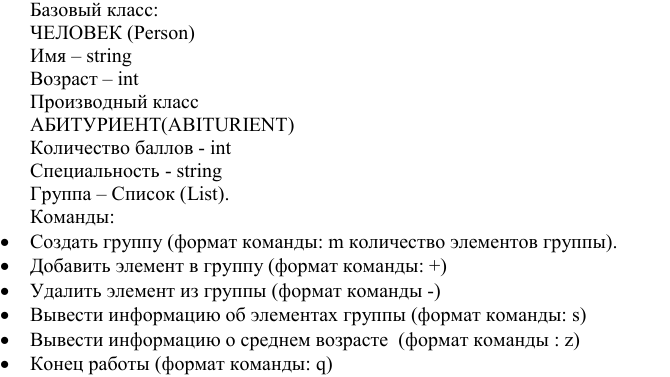
5. Определить класс Диалог - наследника группы, в котором реализовать методы для обработки событий.

6. Добавить методы для обработки событий группой и объектами пользовательских классов.

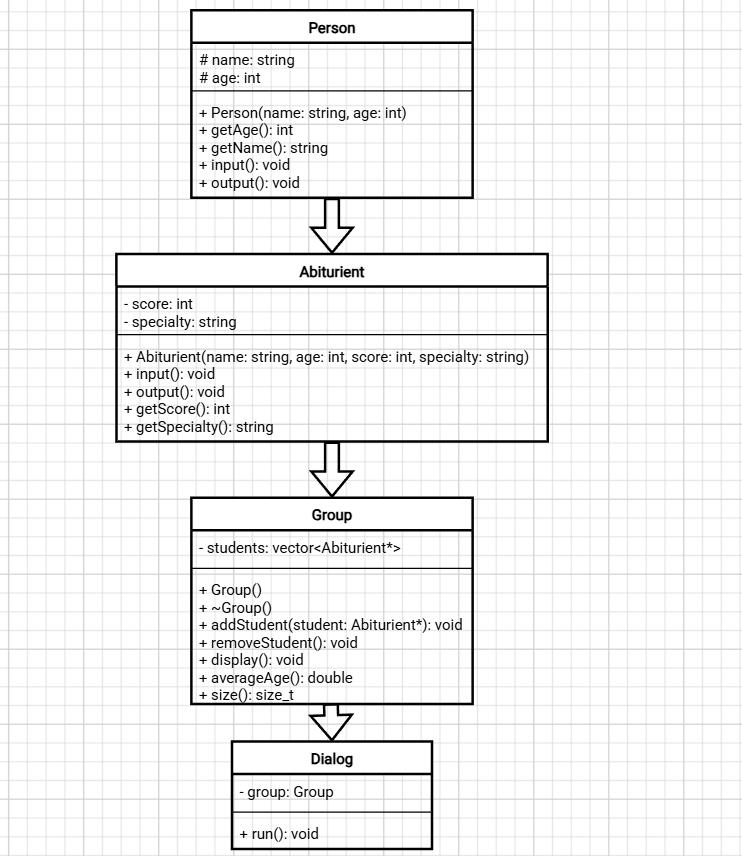
7. Написать тестирующую программу.

8. Нарисовать диаграмму классов и диаграмму объектов.

Персональная:



UML – диаграмма:



Программная реализация

**Заголовочный файл**

#pragma once

#include <string>

#include <iostream>

class Person

{

protected:

std::string name;

int age;

public:

Person(const std::string& name, int age);

virtual ~Person() = default;

virtual void input() = 0;

virtual void output() const = 0;

int getAge() const;

const std::string& getName() const;

};

#pragma once

#include "Person.h"

class Abiturient : public Person

{

private:

int score;

std::string specialty;

public:

Abiturient(const std::string& name, int age, int score, const std::string& specialty);

void input() override;

void output() const override;

int getScore() const;

const std::string& getSpecialty() const;

};

#pragma once

#include <vector>

#include "Abiturient.h"

class Group

{

private:

std::vector<Abiturient\*> students;

public:

Group();

~Group();

void addStudent(Abiturient\* student);

void removeStudent();

void display() const;

double averageAge() const;

size\_t size() const;

};

#pragma once

#include "Group.h"

class Dialog

{

private:

Group group;

public:

void run();

};

**Реализация методов**

#include "Person.h"

Person::Person(const std::string& name, int age) : name(name), age(age) {}

int Person::getAge() const

{

return age;

}

const std::string& Person::getName() const

{

return name;

}

#include "Abiturient.h"

#include <iostream>

Abiturient::Abiturient(const std::string& name, int age, int score, const std::string& specialty)

: Person(name, age), score(score), specialty(specialty) {

}

void Abiturient::input()

{

setlocale(LC\_ALL, "Russian");

std::cout << "Введите имя: ";

std::cin >> name;

std::cout << "Введите возраст: ";

std::cin >> age;

std::cout << "Введите количество баллов: ";

std::cin >> score;

std::cout << "Введите специальность: ";

std::cin >> specialty;

}

void Abiturient::output() const

{

std::cout << "Имя: " << name << ", Возраст: " << age

<< ", Баллы: " << score << ", Специальность: " << specialty << std::endl;

}

int Abiturient::getScore() const

{

return score;

}

const std::string& Abiturient::getSpecialty() const

{

return specialty;

}

#include "Group.h"

#include <iostream>

Group::Group() {}

Group::~Group()

{

for (auto student : students)

{

delete student;

}

}

void Group::addStudent(Abiturient\* student)

{

students.push\_back(student);

}

void Group::removeStudent()

{

if (!students.empty())

{

delete students.back();

students.pop\_back();

}

}

void Group::display() const

{

for (const auto& student : students)

{

student->output();

}

}

double Group::averageAge() const

{

if (students.empty()) return 0.0;

int totalAge = 0;

for (const auto& student : students)

{

totalAge += student->getAge();

}

return static\_cast<double>(totalAge) / students.size();

}

size\_t Group::size() const

{

return students.size();

}

#include "Dialog.h"

#include <iostream>

void Dialog::run()

{

std::string command;

while (true)

{

std::cout << "Введите команду (m, +, -, s, z, q): ";

std::cin >> command;

if (command == "m")

{

int count;

std::cout << "Введите количество элементов группы: ";

std::cin >> count;

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

{

Abiturient\* student = new Abiturient("", 0, 0, "");

student->input();

group.addStudent(student);

}

}

else if (command == "+")

{

Abiturient\* student = new Abiturient("", 0, 0, "");

student->input();

group.addStudent(student);

}

else if (command == "-")

{

group.removeStudent();

std::cout << "Последний элемент удалён из группы." << std::endl;

}

else if (command == "s")

{

std::cout << "Информация об элементах группы:" << std::endl;

group.display();

}

else if (command == "z")

{

double avgAge = group.averageAge();

std::cout << "Средний возраст: " << avgAge << std::endl;

}

else if (command == "q")

{

std::cout << "Завершение работы." << std::endl;

break;

}

else

{

std::cout << "Неизвестная команда. Пожалуйста, попробуйте снова." << std::endl;

}

}

}

**Функция main**

#include "Dialog.h"

int main()

{

setlocale(LC\_ALL, "Russian");

system("chcp 1251");

system("cls");

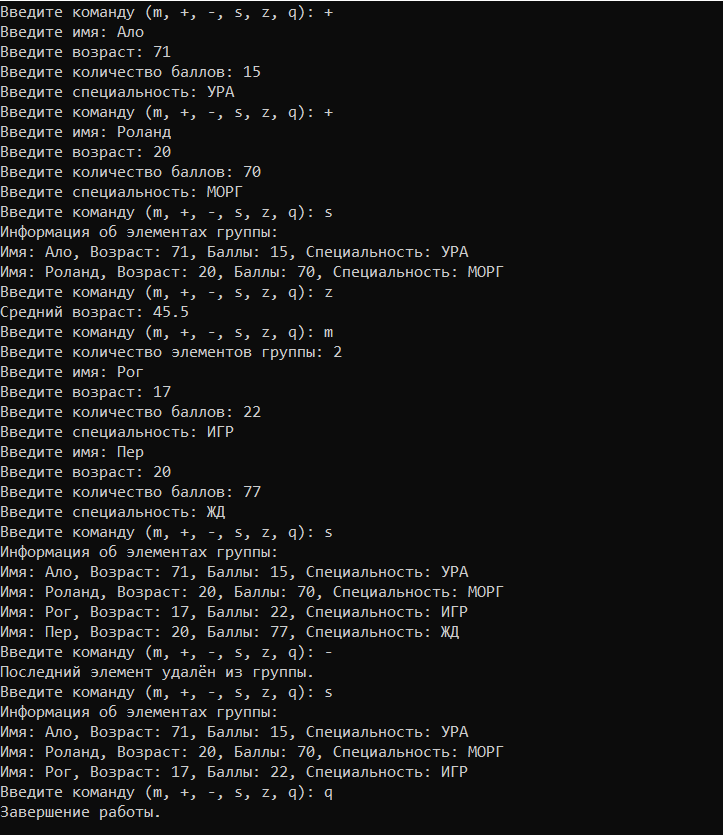
Dialog dialog;

dialog.run();

return 0;

}

Результат работы:



Контрольные вопросы:

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

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.

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

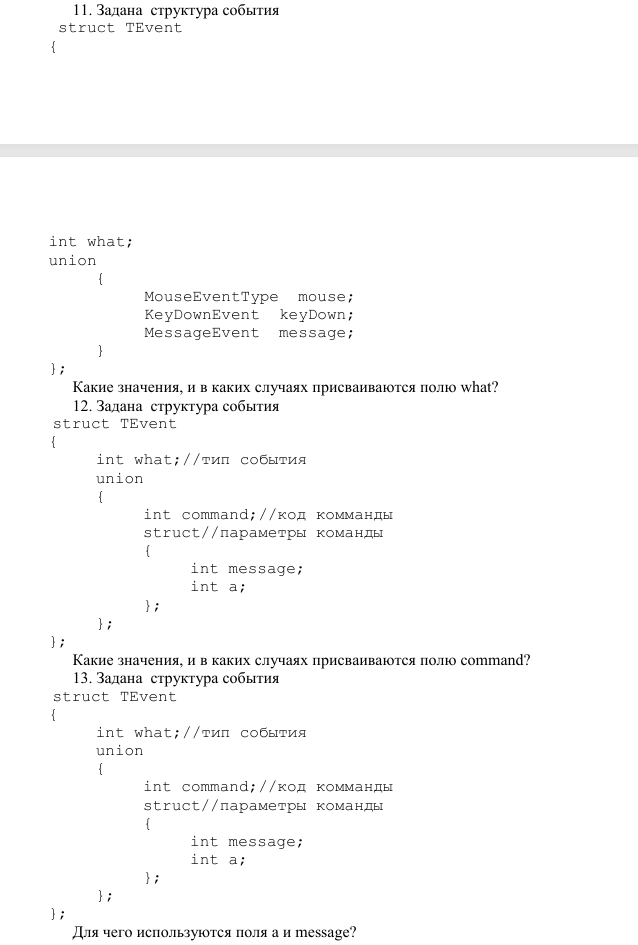
6. Какой вид иерархии дает группа?

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

8. Что такое событие? Для чего используются события?

9. Какие характеристики должно иметь событие-сообщение?

10. Привести пример структуры, описывающей событие.



14. Какие методы необходимы для организации обработки сообщений?

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

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

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

18. Какую функцию выполняет метод GetEvent ()?

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

это поле?

20. Для чего используется функция Valid()?

Ответы:

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

* **Класс-группа** – это объект, который включает в себя другие объекты, называемые элементами группы.
* **Примеры классов-групп**:
  + Окно в интерактивной программе (например, TWindow, TDialog).
  + Агрегат узлов (например, дерево).
  + Огород с растениями.
  + Организационная структура (например, факультет, кафедра).

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

class List

{

private:

Node\* head; // Указатель на первый элемент списка

int size; // Количество элементов в списке

public:

List(); // Конструктор без параметров

List(const List& l); // Конструктор копирования

~List(); // Деструктор

void add(int value); // Метод для добавления элемента

void remove(int value); // Метод для удаления элемента

void display(); // Метод для отображения элементов

};

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

List::List() : head(nullptr), size(0) {} // Конструктор без параметров

List::List(const List& l)

{

// Логика копирования элементов из l в текущий объект

this->size = l.size;

// Дополнительная логика...

}

List::List(int initialValue) : head(new Node(initialValue)), size(1) {} // Конструктор с параметром

### 4. Привести пример деструктора для класса-группы Список.

List::~List()

{

Node\* current = head;

while (current != nullptr)

{

Node\* next = current->next;

delete current; // Удаление текущего узла

current = next; // Переход к следующему узлу

}

}

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

void List::display()

{

Node\* current = head;

while (current != nullptr)

{

std::cout << current->data << " "; // Вывод значения узла

current = current->next; // Переход к следующему узлу

}

std::cout << std::endl;

}

### 6. Какой вид иерархии дает группа?

* Группа дает иерархию объектов (иерархию типа целое/часть), построенную на основе **агрегации**.

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

* Абстрактный класс определяет общее поведение и интерфейс для всех классов-наследников, что обеспечивает гибкость и расширяемость системы.

### 8. Что такое событие? Для чего используются события?

* **Событие** – это пакет информации, который создается объектно-ориентированной средой в ответ на действия пользователя (например, нажатие клавиш или движения мыши).
* События используются для организации взаимодействия между объектами в программе.

### 9. Какие характеристики должно иметь событие-сообщение?

* Код класса сообщения.
* Адрес объекта, которому предназначено сообщение.
* 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. Какие значения, и в каких случаях присваиваются полю what?

* **evNothing** – присваивается, когда событие обработано каким-либо объектом.
* **evMessage** – присваивается, если событие связано с сообщением от объекта.

### 12. Какие значения, и в каких случаях присваиваются полю command?

* Поле **command** присваивается значением кода команды, когда событие является событием-сообщением (evMessage), содержащим информацию о том, что нужно выполнить.

### 13. Для чего используются поля a и message?

* Поля **a** и **message** используются для передачи дополнительной информации, связанной с командой, в обработчиках событий, позволяя более точно управлять действиями объектов.

**14. Необходимые методы для организации обработки сообщений:**

* **GetEvent** — для формирования события.
* **Execute** — реализует главный цикл обработки событий.
* **HandleEvent** — обработчик событий, который обрабатывает каждое событие.
* **ClearEvent** — очищает событие после его обработки.

**15. Главный цикл обработки событий-сообщений:**

* Находится в методе **Execute**.
* Состоит из постоянного получения событий с помощью **GetEvent** и их обработки через **HandleEvent**.

**16. Функция метода ClearEvent():**

* Очищает событие, присваивая полю **what** значение **evNothing**.
* Это предотвращает повторную обработку события.

**17. Функция метода HandleEvent():**

* Обрабатывает каждое событие нужным образом.
* Вызов метода базового класса для дальнейшей обработки, проверка типа события и его обработки по заданным параметрам.

**18. Функция метода GetEvent():**

* Формирует и возвращает событие, создаваемое в ответ на действия пользователя.

**19. Использование поля EndState:**

* Хранит состояние завершения работы программы.
* Содержится в объекте главной группы, которая представляет "прикладную программу".

**20. Использование функции Valid():**

* Проверяет, завершили ли работу все элементы группы.
* Возвращает true, если все подпорядоченные элементы завершили свою работу, позволяя программе корректно завершиться.

Ссылка:

**https://github.com/LeonidZhir/-**