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**

Тема: «Программа, управляемая событиями»

Выполнил работу

Студент группы РИС-22-2Б

Вахрушева А.В.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

Постановка задачи.

1. Определить иерархию пользовательских классов (см. лаб. 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

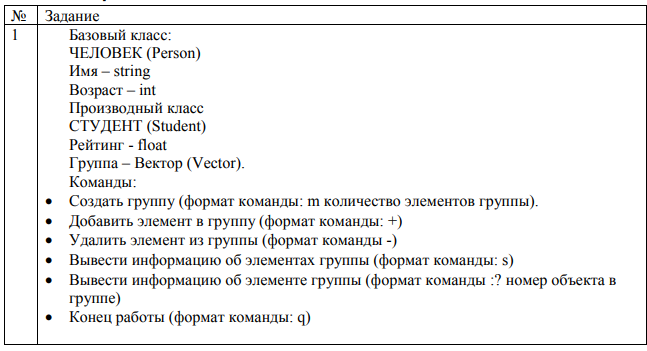
обработки событий.

6. Добавить методы для обработки событий группой и объектами пользовательских

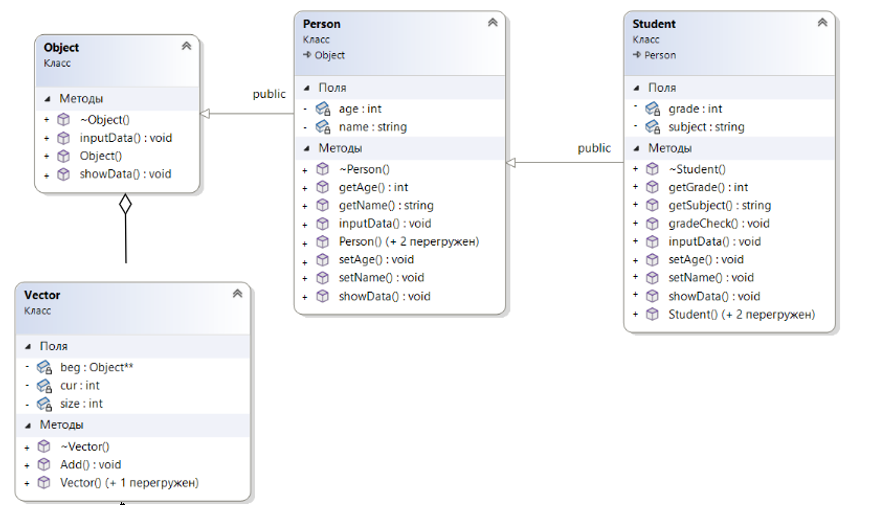
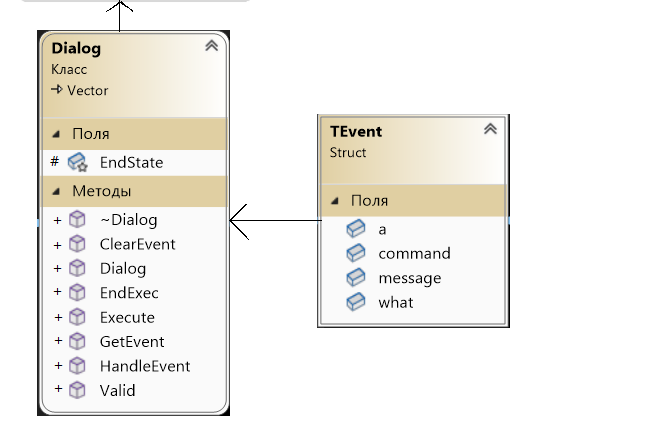
классов.

7. Написать тестирующую программу.

8. Нарисовать диаграмму классов и диаграмму объектов.



UML - диаграмма



Код программы.

Sourse.cpp

#include "Dialog.h"

#include <iostream>

#include "windows.h"

using namespace std;

int main()

{

system("chcp 1251 >>null");

Dialog D;

D.Execute();

return 0;

}

Event.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;

int message;

int a;

};

Dialog.h

#pragma once

#include "Vector.h"

using namespace std;

class Dialog :public Vector

{

public:

Dialog(void);

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;

};

Dialog.cpp

#include "Dialog.h"

Dialog::Dialog(void) :Vector()

{

EndState = 0;

}

Dialog::~Dialog(void) {}

void Dialog::GetEvent(TEvent& event)

{

string OpInt = "+-mszq";

string s;

string param;

char code;

cout << "\n\nYour choise: ";

cin >> s;

code = s[0];

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

{

event.what = evMessage;

switch (code)

{

case 'm':

event.command = cmMake;

break;

case 's':

event.command = cmShow;

break;

case 'z':

event.command = cmGet;

break;

case '+':

event.command = cmAdd;

break;

case '-':

event.command = cmDel;

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()

{

cout << "Нажмите m чтобы создать список.\nНажмите + чтобы добавить.\nНажмите - чтобы удалить последнего.\nНажмите z чтобы вывелся средний возраст людей в списке.\nНажмите s чтобы вывести информацию.\nНажмите q чтобы завершить работу.";

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:

cout << "Enter size: "; cin >> size;

beg = new Object \* [size];

cur = 0;

ClearEvent(event);

break;

case cmAdd:

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

{

Add();

ClearEvent(event);

}

break;

case cmDel:

Del();

ClearEvent(event);

break;

case cmShow:

Show();

ClearEvent(event);

break;

case cmQuit:

EndExec();

ClearEvent(event);

break;

case cmGet:

rAge();

ClearEvent(event);

break;

default:

ClearEvent(event);

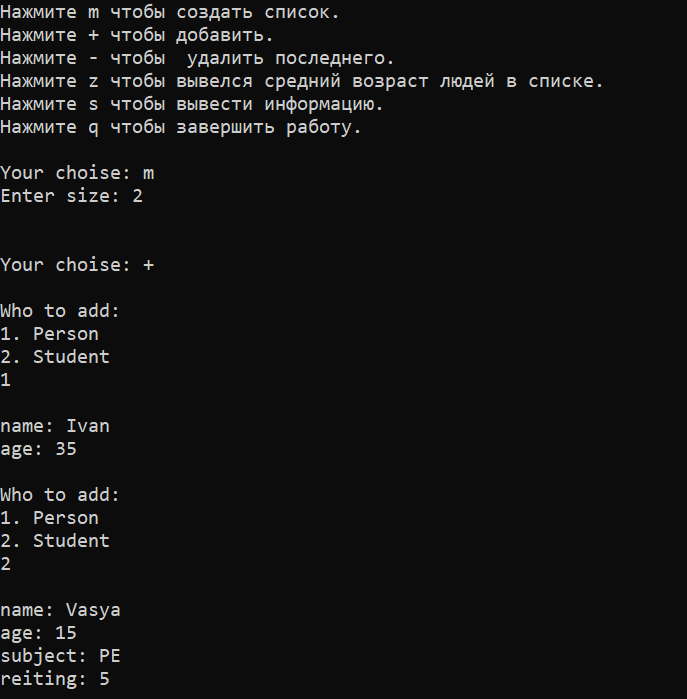
HandleEvent(event);

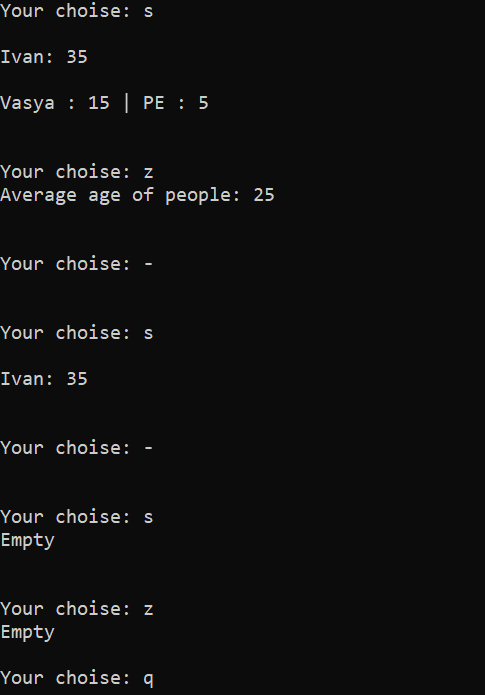
};

};

}

Вывод.





Контрольные вопросы.

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

Класс-группа (или просто группа) - это совокупность объектов, имеющих одинаковые свойства и методы. Например, класс-группа "Фрукты" может включать объекты "Яблоко", "Апельсин", "Банан" и т.д.

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

Класс-группа "Список" может быть описан следующим образом: это набор элементов, упорядоченных по индексам. Методы класса-группы "Список" могут включать добавление элемента в конец списка, удаление элемента по индексу, получение элемента по индексу и т.д.

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

Примеры конструкторов для класса-группы "Список":

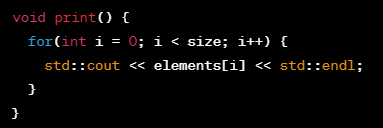
* Конструктор без параметров: создает пустой список.
* Конструктор с параметром "размер списка": создает список заданного размера.
* Конструктор копирования: создает копию уже существующего списка.

4. Привести пример деструктора для класса-группы Список.

Деструктор для класса-группы "Список" должен освобождать память, занимаемую списком.

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

Метод "просмотр элементов" для класса-группы "Список" может быть реализован следующим образом:



6. Какой вид иерархии дает группа?

Группа может создавать иерархию объектов, где каждый последующий объект наследует свойства и методы предыдущего.

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

Абстрактный класс находится во главе иерархии классов, содержащихся в группе объектов, потому что он определяет базовый интерфейс и не может быть использован напрямую. Конкретные классы могут наследоваться от абстрактного класса и реализовывать его методы.

8. Что такое событие? Для чего используются события?

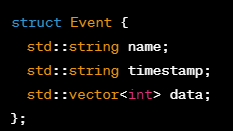
Событие - это уведомление об изменении состояния объекта или системы. События используются для связи между объектами и реагирования на изменения в системе.

9. Какие характеристики должно иметь событие-сообщение?

Событие-сообщение должно содержать информацию о том, что произошло, когда это произошло и какие данные были изменены.

10. Привести пример структуры, описывающей событие.

Структура, описывающая событие, может выглядеть следующим образом:



Здесь "name" - название события, "timestamp" - время, когда событие произошло, "data" - данные, связанные с событием.

11. Задана структура события

struct Kevent int what;

union

Mouseevent mouse;

KeyDownEvent keyDown;

MessageEvent message;

} ;

Какие значения, и в каких случаях присваиваются полю what?

Поле "what" в структуре события определяет тип события, и его значение может быть присвоено в зависимости от того, какое событие произошло. Например, если произошло событие "Mouseevent", то полю "what" будет присвоено значение, связанное с этим событием, а именно значение, которое было определено для события "Mouseevent". Аналогично, если произошло событие "KeyDownEvent" или "MessageEvent", полю "what" будет присвоено соответствующее значение.

12. Задана структура события

struct Kevent

int what;//тип события

union

{

int command; //код команды

struct//параметры команды

int message;

int a;

} ;

};

Какие значения, и в каких случаях присваиваются полю command?

Поле "command" в структуре события может использоваться для хранения кода команды, связанной с событием. Например, если событие связано с выполнением определенной команды, то полю "command" будет присвоено соответствующее значение. Значение, которое будет присвоено полю "command", зависит от конкретной команды, которая выполняется в рамках данного события. Например, если команда выполняет какое-то действие с числом "a", то полю "command" может быть присвоено значение, связанное с этой командой.

13. Задана структура события

struct Kevent

{

int what;//тип события

union

{

int command; //код команды

struct//параметры команды

int message;

int a;

} ;

};

};

Для чего используются поля а и message?

Поля "a" и "message" в структуре события могут использоваться для хранения параметров команды, связанной с событием. Например, если событие связано с выполнением определенной команды, то полям "a" и "message" могут быть присвоены соответствующие значения параметров, связанных с этой командой.

14. Какие методы необходимы для организации обработки сообщений?

Для организации обработки сообщений необходимо реализовать методы, которые будут получать события из очереди событий, обрабатывать их и принимать соответствующие действия в зависимости от типа события. Например, методы для добавления событий в очередь, извлечения событий из очереди, обработки событий и т.д.

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

Главный цикл обработки событий-сообщений имеет вид бесконечного цикла, который получает события из очереди, обрабатывает их и принимает соответствующие действия в зависимости от типа события. Этот цикл продолжается до тех пор, пока не произойдет событие, связанное с завершением работы программы.

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

Метод ClearEvent() очищает содержимое события, устанавливая все его поля в значения по умолчанию или в пустые значения, если они не определены. Это может быть полезно, если необходимо повторно использовать объект события.

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

Метод HandleEvent() выполняет обработку события, полученного из очереди. Он проверяет тип события и вызывает соответствующий обработчик для выполнения действий, связанных с этим событием.

18. Какую функцию выполняет метод GetEvent ()?

Метод GetEvent() используется для получения события из очереди. Он извлекает первое событие из очереди и возвращает его.

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

Поле EndState используется для хранения состояния завершения работы программы. Это поле может быть использовано в главном цикле обработки событий-сообщений для проверки, необходимо ли завершить работу программы. Это поле может содержаться в объекте, управляющем главным циклом программы.

20. Для чего используется функция Valid()?

Функция Valid() используется для проверки, действительно ли событие было успешно создано и содержит корректные данные. Она возвращает логическое значение true, если событие действительно, и false, если событие не действительно. Это может быть полезно при обработке событий, чтобы убедиться, что они корректны и могут быть обработаны без ошибок.