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

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

Электротехнический факультет

Кафедра Информационные технологии и автоматизированные системы

Лабораторные работы по

«Основам алгоритмизации и программирования»

вариант № 15

за 2 семестр

Выполнил:

студент группы РИС-21-1бз

Тимолянов Григорий Константинович

Проверила:

Доцент кафедры ИТАС,

к.т.н. Полякова О.А.

2024

###### Лабораторная работа №8

**Шаблоны классов**

**Цель:** 1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.

2) Разработка программы, управляемой событиями.

**Постановка задачи**

1. Определить иерархию пользовательских классов (см. лабораторную работу №5). Во главе иерархии должен стоять абстрактный класс с чисто виртуальнымиметодами 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.

Базовый класс:

ЧЕЛОВЕК (Person)

Имя – string

Возраст – int

Производный класс

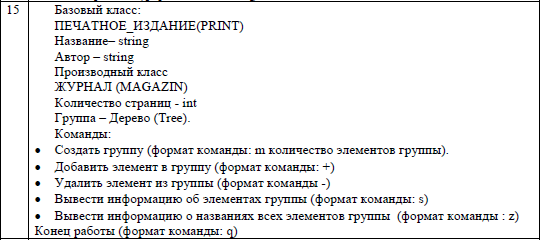
СОТРУДНИК (Employee)

Должность - string

Группа – Список (List).

Команды:

* Создать группу (формат команды: m количество элементов группы).
* Добавить элемент в группу (формат команды: +)
* Удалить элемент из группы (формат команды -)
* Вывести информацию об элементах группы (формат команды: s)
* Вывести информацию об элементе группы (формат команды :z k, где к – целое число)
* Конец работы (формат команды: q)



**Решение:**

Файл Программа с событиями.cpp

#include <iostream>

#include "Windows.h"

#include "Print.h"

#include "Magazin.h"

#include "Vector.h"

#include "Dialog.h"

#include "Event.h"

#include "Object.h"

// Имеется класс Object, от которого наследуется класс Print. От Print наследуется Magazine.

// Класс-группа Vector содержит указатель на динамический массив указателей типа Object. От Vector наследуется Dialog.

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Введите данные печатного издания (класс Print):";

Print\* a = new Print;

a->Input();

a->Show();

cout << "\nВведите данные печатного издания (класс Magazine):";

Magazine\* b = new Magazine;

a->Input();

a->Show();

Print\* c = new Print;

c->Input();

Vector v(10);

Object\* p = a;

v.Add();

p = b;

v.Add();

v.Show();

v.Del();

cout << "\nДлина вектора = " << v();

Dialog D;

D.Execute();

return 0;

}

Файл Object.h

// pragma once - подключение файла при компиляции только один раз

#pragma once

#include <string>

#include "Event.h"

using namespace std;

struct Object

{

Object(void) {};

virtual ~Object(void) {};

virtual void Show() = 0;

virtual void Input() = 0;

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

protected:

string name;

string author;

};

Файл Dialog.cpp

#include "Dialog.h"

#include "Object.h"

#include "Vector.h"

#include <iostream>

using namespace std;

Dialog::Dialog(void): Vector() { EndState = 0; }

Dialog::~Dialog(void) {}

void Dialog::GetEvent(TEvent& event) {

// Строка содержит коды операций

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 '?': event.command = cmShow;

break;

case 'q': event.command = cmQuit;

break;

case '/': event.command = cmGet;

break;

}

if (s.length() > 1) {

// Возвращаем подстроку строки s начиная с символа с индексом 1

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

// Конвертируем строку в число

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() {

int result;

result = (EndState == 0) ? 0 : 1;

return result;

}

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:

size = event.a;

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: Vector::HandleEvent(event);

}

}

}

Файл Dialog.h

#pragma once

#include "Vector.h"

#include "Event.h"

class Dialog : public Vector {

int EndState;

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();

};

Файл Magazin.cpp

#include "Magazin.h"

Magazine::Magazine(void) : Print () { page = 0; }

Magazine::~Magazine(void) {}

Magazine::Magazine(string N, string A, int P) : Print (N, A) { page = P; }

Magazine::Magazine(const Magazine &M) { name = M.name; author = M.author; page = M.page; }

void Magazine::Set\_Page(int P) { page = P; }

Magazine& Magazine::operator = (const Magazine& m) {

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

name = m.name;

author = m.author;

return \*this;

}

void Magazine::Show() {

cout << "\nНазвание : " << name;

cout << "\nАвтор : " << author;

cout << "\nСтраница : " << page;

cout << "\n";

}

void Magazine::Input() {

cout << "\nНазвание : ";

cin >> name;

cout << "Автор : ";

cin >> author;

cout << "Страница : ";

cin >> page;

}

Файл Magazin.h

#pragma once

#include "Print.h"

#include <iostream>

using namespace std;

class Magazine : public Print {

int page;

public:

Magazine(void);

~Magazine(void);

void Show();

void Input();

Magazine(string, string, int);

Magazine(const Magazine&);

int Get\_page() { return page; }

void Set\_Page(int);

Magazine& operator = (const Magazine&);

};

Файл Print.cpp

#include "Print.h"

#include "Event.h"

Print::Print(void) { name = ""; author = ""; }

Print::~Print(void) {}

Print::Print(string N, string A) { name = N; author = A; }

Print::Print(const Print& print) { name = print.name; author = print.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 << "\nНазвание: " << name;

cout << "\nАвтор: " << author;

cout << "\n";

}

void Print::Input() {

cout << "\nНазвание: ";

cin >> name;

cout << "Автор: ";

cin >> author;

}

void Print::HandleEvent(const TEvent& e) {

if (e.what == evMessage) {

switch (e.command) {

case cmGet:

cout << "Название = " << Get\_name() << endl;

break;

}

}

}

Файл Print.h

#pragma once

#include "Object.h"

#include <iostream>

using namespace std;

struct Print : public Object {

Print(void);

virtual ~Print(void);

void Show();

void Input();

void HandleEvent(const TEvent&);

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&);

protected:

string name;

string author;

};

Файл Vector.cpp

#include "Vector.h"

#include "Object.h"

#include "Print.h"

#include "Magazin.h"

#include <iostream>

using namespace std;

Vector::~Vector(void) {

if (beg != 0) { delete[] beg; }

beg = 0;

}

Vector::Vector(int n) {

beg = new Object \* [n];

cur = 0;

size = n;

}

Vector::Vector() {

beg = 0;

cur = 0;

size = 0;

}

void Vector::Add() {

Object\* p;

cout << "1. Журнал\n";

cout << "2. Страница\n";

int y;

cin >> y;

if (y == 1) {

Print\* a = new (Print);

a->Input();

p = a;

if (cur < size) { beg[cur] = p; cur++; }

else

if (y == 2) {

Magazine\* b = new Magazine;

b->Input();

p = b;

if (cur < size)

{ beg[cur] = p; cur++; }

}

else { return; }

}

}

void Vector::Show() {

if (cur == 0) { cout << "Пусто!\n"; }

Object\*\* p = beg;

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

(\*p)->Show(); p++;

}

}

int Vector::operator() () { return cur; }

void Vector::Del() {

if (cur == 0) return;

cur--;

}

void Vector::HandleEvent(const TEvent& e) {

if (e.what == evMessage) {

Object\*\* p = beg;

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

{ (\*p)->HandleEvent(e); p++; }

}

}

Файл Vector.h

#pragma once

#include "Object.h"

struct Vector {

Vector(int);

~Vector(void);

Vector();

void Add();

void Del();

void Show();

int operator() ();

protected:

int size;

int cur;

Object\*\* beg;

void HandleEvent(const TEvent& e);

};

###### **Результат работы программы:**

# 

###### **Ответы на 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.

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

Шаблон класса контейнеров последовательностей, которые поддерживают их элементы в линейном расположении и разрешают эффективные вставки и удаления в любом расположении в последовательности. Последовательность хранится в виде двунаправленного связанного списка элементов

1. Привести пример конструктора (с параметром, без параметров, копирования) для

класса-группы Список.

template <class Type, class Allocator= allocator<Type>>

class list

1. Привести пример деструктора для класса-группы Список.
2. Привести пример метода для просмотра элементов для класса-группы Список.
3. Какой вид иерархии дает группа?

Иерархию объектов по типу целое/часть. Человек/студент. Животное/мамонт.

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

Делается для удобства. В нём определяется интерфейс взаимодействия с наследуемыми классами, какие-то общие методы, которые потом переопределяются.

1. Что такое событие? Для чего используются события?

Событие — это способ вызвать методы других объектов, когда событие срабатывает.

1. 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.

1. Привести пример структуры, описывающей событие.

struct TEvent

{

int what;//тип события

union

{

int command;//код команды

struct

{

int message;

int a;//параметр команды

};

};

};

1. Задана структура события

Какие значения, и в каких случаях присваиваются полю what?

После what несёт в себе информацию, пустое или же не пустое событие.

1. Задана структура события

Какие значения, и в каких случаях присваиваются полю command?

Код команды, которую необходимо выполнить при появлении данного событи.

1. 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 – дополнительное поле, в котором в данном случае передаётся размер массива для хранения объектов, а так же передаётся номер элемента для вывода на экран.

Message – дополнительное поле, в котором хранится максимальный размер массива + 1.

1. Какие методы необходимы для организации обработки сообщений?

GetEvent – формирование события.

HandleEvent – обработчик события.

ClearEvent – очистка события.

Execute – запускает весь цикл обработки события.

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

int Dialog::Execute()

{

TEvent event{};

do

{

EndState = 0;

GetEvent(event);

HandleEvent(event);

} while (!Valid());

return EndState;

}

Пока переменная Valid не равна нулю, выполняется цикл по формированию события и обработке события.

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

Очищает событие после его обработки, присваивая ему значение 0, означающее в данном случае пустое событие.

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

Обрабатывает событие. В работе обрабатываются заданные события и вызываются необходимые методы для этого события.

1. Какую функцию выполняет метод GetEvent ()?

Метод берёт на себя функционал обработки пользовательского ввода, в результате которого переменной command присваивается код события. Далее этот код передаётся в обработчик событий HandleEvent.

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

Класс, содержащий поле EndState называется диалог, то есть класс для работы с событиями. Самое поле необходимо для корректного выхода из цикла. Значение поля обрабатывается функцией Valid, и если оно отличается от единицы, то цикл в методе Execute заканчивает свою работу.

1. Для чего используется функция Valid()?

Для выхода из цикла.