Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.8**

Дисциплина: «Информатика»

Тема: “ Объектно-ориентированное программирование.

Программа, управляемая событиями.”

Вариант 15

Выполнил:

Студент группы ИВТ-20-2Б Чувашев Максим

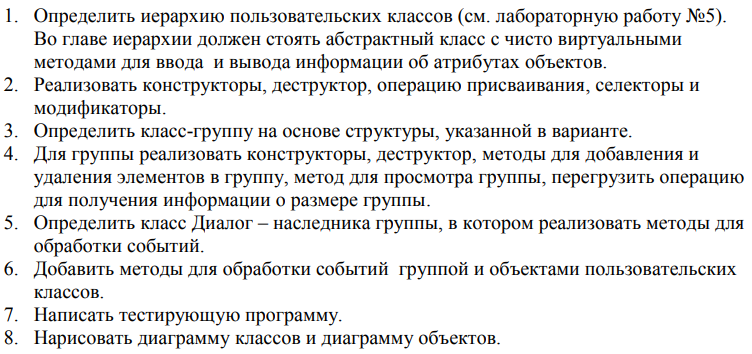
Проверила:

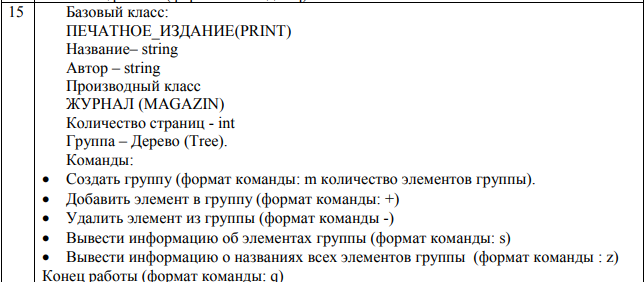
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**





**Анализ задачи**

1. Для решения задачи необходимо:
   1. Организовать абстрактный класс Object с чисто виртуальными функциями для вывода данных на консоль и ввода данных. (необходимо при наследовании)
   2. Организовать класс Vector – контейнер для объектов других классов.
   3. Организовать класс Eventы для работы с событиями, в котором организовать enum EVENTS и COMMANDS для работы с событиям
   4. Организовать класс Diaglog для работы с событиями.
   5. Организовать класс PRINT с полями name и author типа string для ввода наименования и автора.
   6. Организовать класс Magazine с полями name типа string и author типа string для ввода названия издательства и имени автора соответственно.
2. В ходе работы были использованы следующие типы данных:
   1. enum EVENTS и COMMANDS для работы с событиями.

enum EVENTS

{

evNothing = 0,

evMessege = 100

};

enum COMMANDS

{

cmdMake = 1,

cmdAdd,

cmdDel,

cmdShow,

cmdName,

cmdQuit,

cmdHelp

};

* 1. В функции GetEvent() класса Dialog используется switch() для переключения команды для метода HandleEvent()

void Dialog::HandleEvent(TEvent& event)

{

if (event.what == evMessege)

{

switch (event.command)

{

case cmdMake: // кейс создания группы

{

int size = event.parameter;

this->begin = new Object \* [size];

this->size = size;

this->current = 0;

ClearEvent(event);

break;

}

case cmdAdd: // кейс добавления объекта в группу

{

add();

ClearEvent(event);

break;

}

case cmdDel: // кейс удалениия объекта из группы

{

del();

ClearEvent(event);

break;

}

case cmdShow: // кейс вывода всех элементов в консоль

{

show();

ClearEvent(event);

break;

}

case cmdQuit: // кейс выхода из программы

{

EndState = 1;

ClearEvent(event);

break;

}

case cmdHelp:

{

cout <<endl << "Список команд:" << endl;

cout << "m<size>: Создание группы размера size, по умолчанию size = 0" << endl;

cout << "<+>: Добавление элемента в группу" << endl;

cout << "<->: Удаление элемента группы" << endl;

cout << "<s>: Вывод полной информации о всех элементах" << endl;

cout << "<z>: Вывод названий всех элементов" << endl;

cout << "<q>: Выход из программы" << endl;

cout << "<h>: Список команд" << endl;

ClearEvent(event);

break;

}

default:

Vector::HendleEvent(event);

}

}

}

 void Dialog::GetEvent(TEvent& event)

{

string OpInt = "m+-szqh"; // строка содержащая коды операций

string s = "";

cout << "Ваше действие: ";

getline(cin, s);

if (OpInt.find(s[0]) != -1) // является ли символ кодом операции

{

event.what = evMessege;

switch (s[0])

{

case 'm': event.command = cmdMake; break; // создать группу

case '+': event.command = cmdAdd; break; // добавить объект в группу

case '-': event.command = cmdDel; break; // удалить объект из группы

case 's': event.command = cmdShow; break; // вывод в консоль

case 'z': event.command = cmdName; break; // вывод имени объектов группы в консоль

case 'q': event.command = cmdQuit; break; // выход из программы

case 'h': event.command = cmdHelp; break; // вывод списка команд

}

if (s.length() > 1)

{

string param = s.substr(1, s.length() - 1); //строка без первого символа

event.parameter = stoi(param); // достаем числа из строки

}

else

{

event.parameter = 0;

}

}

else

{

event.what = evNothing;

}

}

1. Для решения задачи данные были представлены в следующем виде:
   1. Работа с событиями происходит через структуру TEvent с полем what, отвечающим за тип события. Вторая часть задаёт информацию, передаваемую с событием.

struct TEvent

{

int what;

union

{

int command;

struct

{

int messege;

int parameter;

};

};

};

1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Для выбора события вводится ввод осуществляется через функцию getline() в переменную s. Пользователю предлагается выбор из операций создания группы, удаления элемента из группы, добавления элемента в группу, вывода элементов группы и выхода из программы.

void Dialog::GetEvent(TEvent& event)

{

string OpInt = "m+-szqh"; // строка содержащая коды операций

string s = "";

cout << "Ваше действие: ";

getline(cin, s);

if (OpInt.find(s[0]) != -1) // является ли символ кодом операции

{

event.what = evMessege;

switch (s[0])

{

case 'm': event.command = cmdMake; break; // создать группу

case '+': event.command = cmdAdd; break; // добавить объект в группу

case '-': event.command = cmdDel; break; // удалить объект из группы

case 's': event.command = cmdShow; break; // вывод в консоль

case 'z': event.command = cmdName; break; // вывод имени объектов группы в консоль

case 'q': event.command = cmdQuit; break; // выход из программы

case 'h': event.command = cmdHelp; break; // вывод списка команд

}

if (s.length() > 1)

{

string param = s.substr(1, s.length() - 1); //строка без первого символа

event.parameter = stoi(param); // достаем числа из строки

}

else

{

event.parameter = 0;

}

}

else

{

event.what = evNothing;

}

}

* 1. Для вывода на консоль введённых объектов необходимо при выборе операции ввести “s”, тогда запустится метод Show() для класса Vector.

1. Поставленные задачи будут решены следующими действиями:
   1. В функции main() вызывается метод Execute() для объекта D. В данном методе создаётся объект event структуры TEvent – событие, с которым будут выполняться действия. В цикле do-while() вызываются методы GetEvent(), HandleEvenet() и ClearEvent(). Работа цикла осуществляется, пока параметр EndState не будет равен 1.

void Dialog::Execute()

{

TEvent event;

do

{

GetEvent(event);

HandleEvent(event);

if (event.what != evNothing)

{

ClearEvent(event);

}

} while (EndState == 0);

}

* 1. Метод GetEvent() получает событие: пользователь может создать группу удалить элемент из группы, добавить элемент в группу, вывести элементы группы, выйти из программы. Выбор осуществляется через switch(). Однако, чтобы попасть в ветвление switch() происходить проверка введённых данных в if(): если из заданной программистом строчки из операций находится символ введённый пользователем через метод find(), то происходит проход в ветвление, иначе полю what присваивается значение пустого события.
  2. В случае ввода создания группы, нужно ввести дополнительный параметр через разделитель – размер группы (количество элементов в контейнере) через метод класса string – substr(), далее эта подстрока преобразуется в число через функцию stoi и данное значение присваивается параметру a класса Events.

void Dialog::GetEvent(TEvent& event)

{

string OpInt = "m+-szqh"; // строка содержащая коды операций

string s = "";

cout << "Ваше действие: ";

getline(cin, s);

if (OpInt.find(s[0]) != -1) // является ли символ кодом операции

{

event.what = evMessege;

switch (s[0])

{

case 'm': event.command = cmdMake; break; // создать группу

case '+': event.command = cmdAdd; break; // добавить объект в группу

case '-': event.command = cmdDel; break; // удалить объект из группы

case 's': event.command = cmdShow; break; // вывод в консоль

case 'z': event.command = cmdName; break; // вывод имени объектов группы в консоль

case 'q': event.command = cmdQuit; break; // выход из программы

case 'h': event.command = cmdHelp; break; // вывод списка команд

}

if (s.length() > 1)

{

string param = s.substr(1, s.length() - 1); //строка без первого символа

event.parameter = stoi(param); // достаем числа из строки

}

else

{

event.parameter = 0;

}

}

else

{

event.what = evNothing;

}

}

* 1. В функции HandleEvent() происходит обработка полученного события из метода GetEvent(). Происходит провека: если в поле what записано не пустое событие, то происходит заход в по ветке true к ветвлению switch()

void Dialog::HandleEvent(TEvent& event)

{

if (event.what == evMessege)

{

switch (event.command)

{

case cmdMake: // кейс создания группы

{

int size = event.parameter;

this->begin = new Object \* [size];

this->size = size;

this->current = 0;

ClearEvent(event);

break;

}

case cmdAdd: // кейс добавления объекта в группу

{

add();

ClearEvent(event);

break;

}

case cmdDel: // кейс удалениия объекта из группы

{

del();

ClearEvent(event);

break;

}

case cmdShow: // кейс вывода всех элементов в консоль

{

show();

ClearEvent(event);

break;

}

case cmdQuit: // кейс выхода из программы

{

EndState = 1;

ClearEvent(event);

break;

}

case cmdHelp:

{

cout <<endl << "Список команд:" << endl;

cout << "m<size>: Создание группы размера size, по умолчанию size = 0" << endl;

cout << "<+>: Добавление элемента в группу" << endl;

cout << "<->: Удаление элемента группы" << endl;

cout << "<s>: Вывод полной информации о всех элементах" << endl;

cout << "<z>: Вывод названий всех элементов" << endl;

cout << "<q>: Выход из программы" << endl;

cout << "<h>: Список команд" << endl;

ClearEvent(event);

break;

}

default:

Vector::HendleEvent(event);

}

}

}

* 1. Метод ClearEvent() присваивает полю what константу о пустом событие – evNothing. Метод ClearEvent() присваивает полю what константу о пустом событие – evNothing.

void Dialog::ClearEvent(TEvent& event)

{

event.what = evNothing;

}

**Блок-схема**

































**Код**

#include <iostream>

#include "Dialog.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

cout << "Список команд:" << endl;

cout << "m<size>: Создание группы размера size, по умолчанию size = 0" << endl;

cout << "<+>: Добавление элемента в группу" << endl;

cout << "<->: Удаление элемента группы" << endl;

cout << "<s>: Вывод полной информации о всех элементах" << endl;

cout << "<z>: Вывод названий всех элементов" << endl;

cout << "<q>: Выход из программы" << endl;

cout << "<h>: Список команд" << endl;

Dialog D;

D.Execute();

}

#include "Dialog.h"

#include "Events.h"

#include <string>

#include <iostream>

using namespace std;

Dialog::Dialog() : Vector()

{

EndState = 0;

}

Dialog::~Dialog() {}

void Dialog::GetEvent(TEvent& event)

{

string OpInt = "m+-szqh"; // строка содержащая коды операций

string s = "";

cout << "Ваше действие: ";

getline(cin, s);

if (OpInt.find(s[0]) != -1) // является ли символ кодом операции

{

event.what = evMessege;

switch (s[0])

{

case 'm': event.command = cmdMake; break; // создать группу

case '+': event.command = cmdAdd; break; // добавить объект в группу

case '-': event.command = cmdDel; break; // удалить объект из группы

case 's': event.command = cmdShow; break; // вывод в консоль

case 'z': event.command = cmdName; break; // вывод имени объектов группы в консоль

case 'q': event.command = cmdQuit; break; // выход из программы

case 'h': event.command = cmdHelp; break; // вывод списка команд

}

if (s.length() > 1)

{

string param = s.substr(1, s.length() - 1); //строка без первого символа

event.parameter = stoi(param); // достаем числа из строки

}

else

{

event.parameter = 0;

}

}

else

{

event.what = evNothing;

}

}

void Dialog::HandleEvent(TEvent& event)

{

if (event.what == evMessege)

{

switch (event.command)

{

case cmdMake: // кейс создания группы

{

int size = event.parameter;

this->begin = new Object \* [size];

this->size = size;

this->current = 0;

ClearEvent(event);

break;

}

case cmdAdd: // кейс добавления объекта в группу

{

add();

ClearEvent(event);

break;

}

case cmdDel: // кейс удалениия объекта из группы

{

del();

ClearEvent(event);

break;

}

case cmdShow: // кейс вывода всех элементов в консоль

{

show();

ClearEvent(event);

break;

}

case cmdQuit: // кейс выхода из программы

{

EndState = 1;

ClearEvent(event);

break;

}

case cmdHelp:

{

cout <<endl << "Список команд:" << endl;

cout << "m<size>: Создание группы размера size, по умолчанию size = 0" << endl;

cout << "<+>: Добавление элемента в группу" << endl;

cout << "<->: Удаление элемента группы" << endl;

cout << "<s>: Вывод полной информации о всех элементах" << endl;

cout << "<z>: Вывод названий всех элементов" << endl;

cout << "<q>: Выход из программы" << endl;

cout << "<h>: Список команд" << endl;

ClearEvent(event);

break;

}

default:

Vector::HendleEvent(event);

}

}

}

void Dialog::Execute()

{

TEvent event;

do

{

GetEvent(event);

HandleEvent(event);

if (event.what != evNothing)

{

ClearEvent(event);

}

} while (EndState == 0);

}

void Dialog::ClearEvent(TEvent& event)

{

event.what = evNothing;

}

#include "PRINT.h"

PRINT::PRINT()

{

this->name = "";

this->author = "";

}

PRINT::PRINT(string name, string author)

{

this->name = name;

this->author = author;

}

PRINT::PRINT(const PRINT& other)

{

this->name = other.name;

this->author = other.author;

}

PRINT::~PRINT()

{

}

string PRINT::getName()

{

return name;

}

string PRINT::getAuthor()

{

return author;

}

void PRINT::setName(string name)

{

this->name = name;

}

void PRINT::setAuthor(string author)

{

this->author = author;

}

PRINT& PRINT::operator=(const PRINT& other)

{

if (&other == this) // проверка на самоприсваивание

{

return \*this;

}

else

{

this->name = other.name;

this->author = other.author;

return \*this;

}

}

void PRINT::Show()

{

setlocale(LC\_ALL, "Ru");

cout << "Название печатного издания: " << this->name << endl;

cout << "Автор: " << this->author << endl;

}

void PRINT::Input()

{

cout << "Введите название печатного издания: ";

cin >> this->name;

cout << "Введите автора: ";

cin >> this->author;

}

void PRINT::HandleEvent(TEvent& event)

{

setlocale(LC\_ALL, "Ru");

if (event.what == evMessege)

{

switch (event.command)

{

case cmdName: cout << "Название издания: " << getName() << endl; break;

}

}

}

#include "Vector.h"

#include "PRINT.h"

#include "Magazine.h"

Vector::Vector()

{

begin = nullptr;

current = 0;

size = 0;

}

Vector::Vector(int size)

{

this->begin = new Object \* [size];

this->current = 0;

this->size = size;

}

Vector::~Vector()

{

if (begin != nullptr)

{

delete[] this;

}

begin = nullptr;

}

void Vector::add()

{

setlocale(LC\_ALL, "Ru");

if (this->size == 0)

{

cout << "Вектор не создан!" << endl;

return;

}

if (this->size == this->current)

{

cout << "В векторе не осталось места!" << endl;

return;

}

int choise = 0;

Object\* ptr = new Magazine;

bool f = false;

while (f == false)

{

switch (choise)

{

case 1: {PRINT\* tmp = new PRINT; tmp->Input(); ptr = tmp; f = true; break; }

case 2: {Magazine\* tmp = new Magazine; tmp->Input(); ptr = tmp; f = true; break; }

default: cout << endl << "Объект какого класса вы хотите добавть в группу?" << endl;

cout << "1: Печатное издание" << endl;

cout << "2: Журнал" << endl;

cout << "Выбор: ";

cin >> choise;

break;

}

}

if (this->current < size) // проверка на корректность размера

{

this->begin[this->current] = ptr; //добавляем элемент в вектор

this->current++; // сдвигаем указатель на текущий элемент

}

}

void Vector::show()

{

if (this->current == 0)

{

cout << "Вектор пуст!\n";

}

else

{

Object\*\* ptr = this->begin; // создаем копию

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

{

(\*ptr)->Show();

ptr++;

cout << "\n";

}

}

}

void Vector::del()

{

if (this->size == 0)

{

cout << "Вектор не существует!\n";

}

else

{

this->current--;

}

}

int Vector:: operator()()

{

return this->current;

}

void Vector::HendleEvent(TEvent& event)

{

if (event.what == evMessege)

{

Object\*\* ptr = this->begin;

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

{

(\*ptr)->HandleEvent(event);

ptr++;

}

}

}

#pragma once

#include "Vector.h"

#include "Events.h"

class Dialog : public Vector

{

protected:

int EndState;

public:

Dialog();

~Dialog();

void GetEvent(TEvent& event);

void Execute();

void HandleEvent(TEvent& event);

void ClearEvent(TEvent& event);

};

#pragma once

enum EVENTS

{

evNothing = 0,

evMessege = 100

};

enum COMMANDS

{

cmdMake = 1,

cmdAdd,

cmdDel,

cmdShow,

cmdName,

cmdQuit,

cmdHelp

};

struct TEvent

{

int what;

union

{

int command;

struct

{

int messege;

int parameter;

};

};

};

#pragma once

#include "PRINT.h"

class Magazine :

public PRINT

{

protected:

int pages;

public:

Magazine() :PRINT()

{

this->pages = 0;

}

Magazine(string name, string author, int pages) : PRINT(name, author)

{

this->pages = pages;

}

~Magazine(){}

void Show()

{

setlocale(LC\_ALL, "Ru");

cout << "Название журнала: " << this->name << endl;

cout << "Автор: " << this->author << endl;

cout << "Количество страниц: " << this->pages << endl;

}

void Input()

{

setlocale(LC\_ALL, "Ru");

cout << "Введите название журнала: ";

cin >> this->name;

cout << "Автор: ";

cin >> this->author;

cout << "Введите количество страниц в журнале: ";

cin >> this->pages;

}

};

#pragma once

#include "Events.h"

class Object

{

public:

Object(void) {}

virtual ~Object(void){}

virtual void Show() = 0;

virtual void Input() = 0;

virtual void HandleEvent(TEvent& event) = 0;

};

#pragma once

#include <iostream>

#include <string>

#include "Events.h"

#include "Object.h"

using namespace std;

class PRINT:public Object

{

protected:

string name; // поле названия печатного издания

string author; // поле автора печатного издания

public:

PRINT(); // конструктор без параметров

PRINT(string name, string author); // конструктор с параметрами

PRINT(const PRINT& other); // конструктор копирования

~PRINT(); // деструктор

string getName(); // геттер названия

string getAuthor(); // геттер автора

void setName(string name); // сеттер названия

void setAuthor(string author);// сеттер автора

PRINT& operator = (const PRINT& other); // перегрузка оператора =

void Show(); // функция печати

void Input(); // функция ввода

void HandleEvent(TEvent& event); // функция обратотки события

};

#pragma once

#include "Object.h"

class Vector

{

protected:

Object\*\* begin;

int size;

int current;

public:

Vector();

Vector(int size);

~Vector();

void add();

void del();

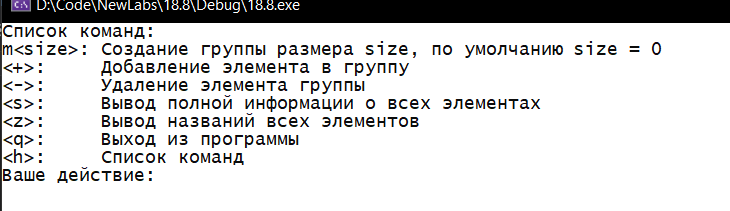
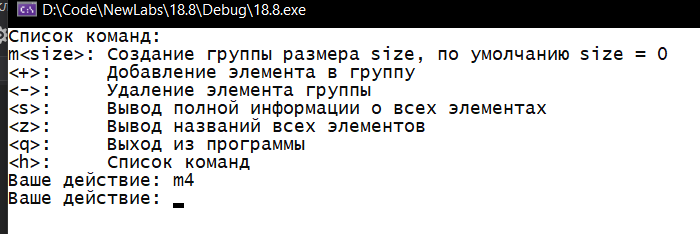
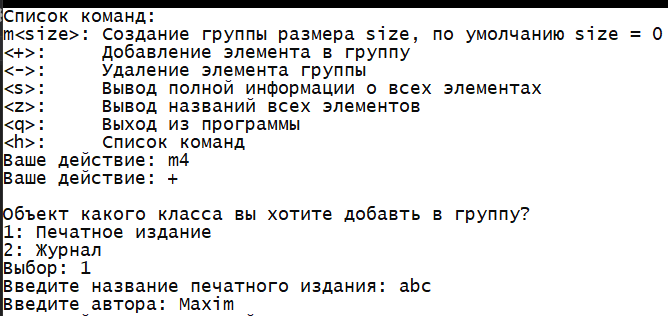
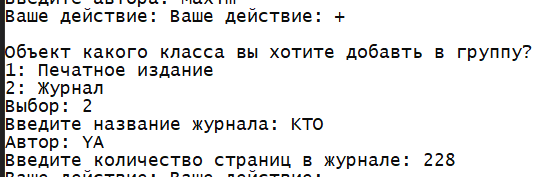
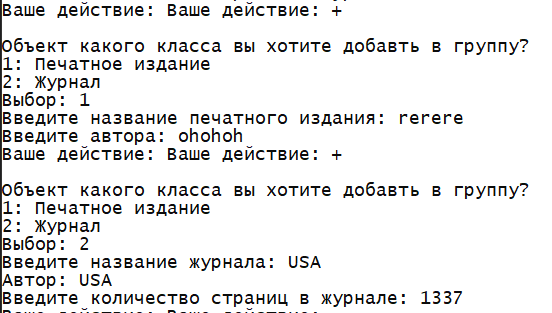
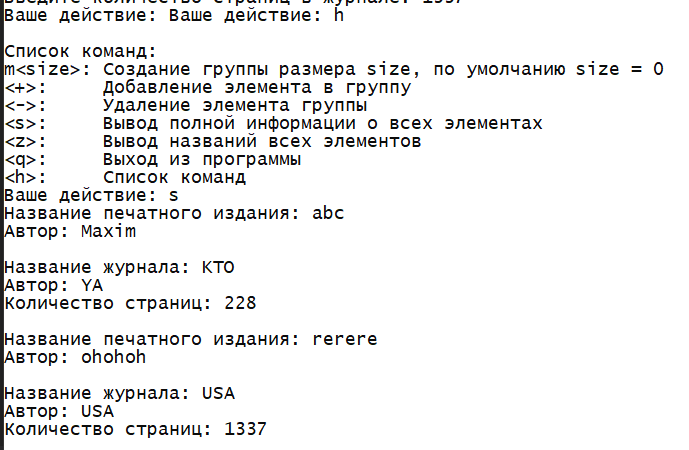
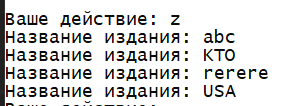
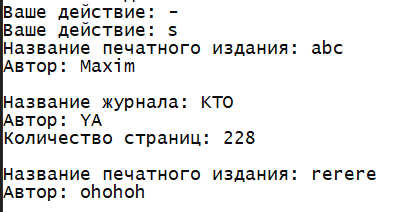
void show();

void HendleEvent(TEvent& event);

int operator()();

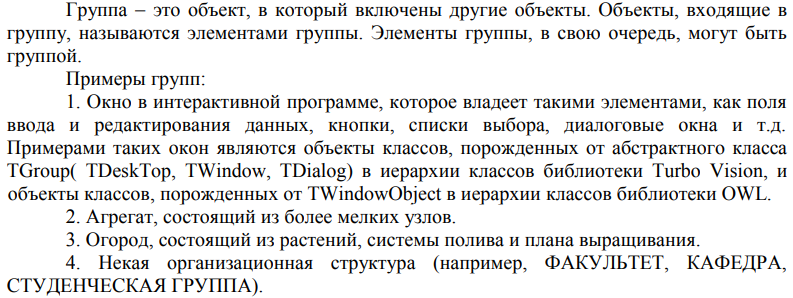
};

**Скриншоты**

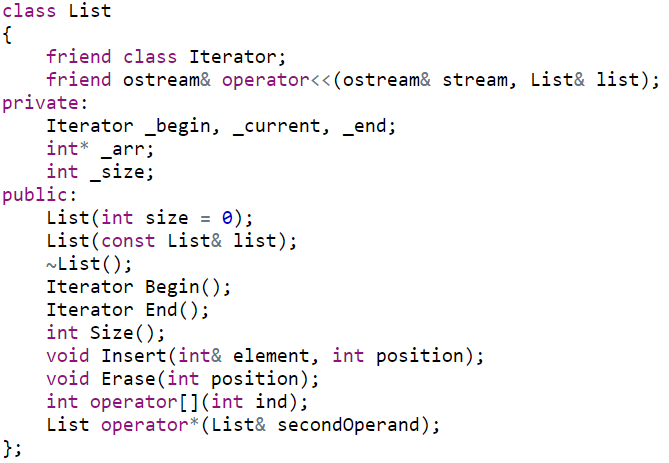
1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 

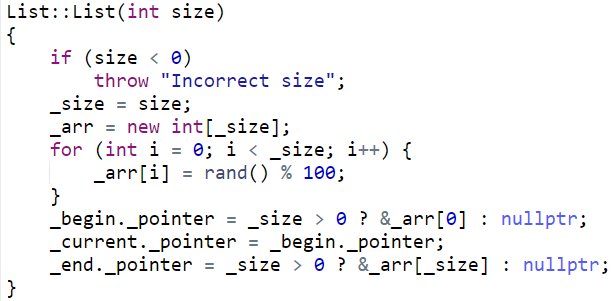
**Контрольные вопросы**

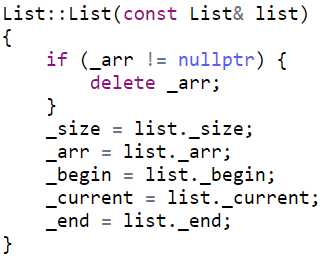


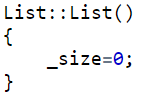
1. 



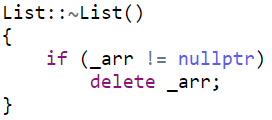
2. 

3. 

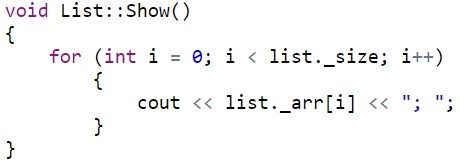




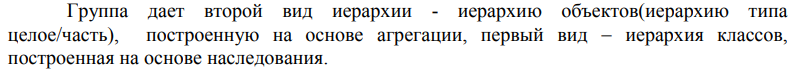


4. 



5. 



6. 

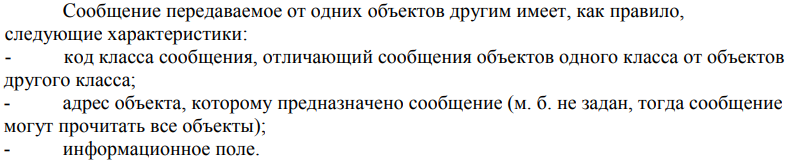


7. Для того, чтобы хранить ссылку на дочерний класс.

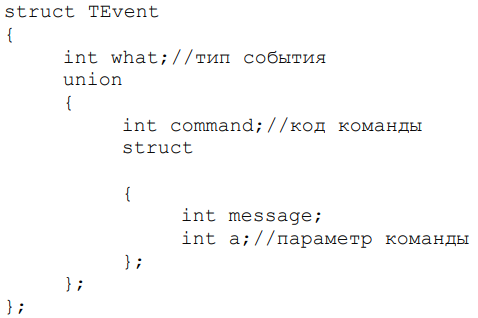


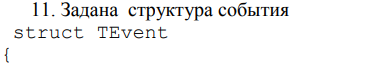
8. Событие – это пакеты информации, которыми обмениваются объекты и которые создаются объектно-ориентированной средой в ответ на те или иные действия пользователя.

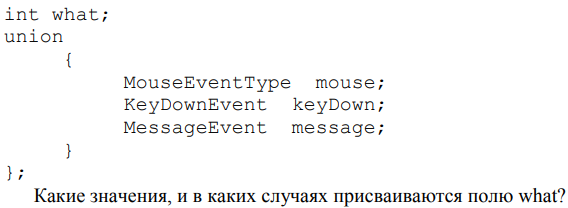


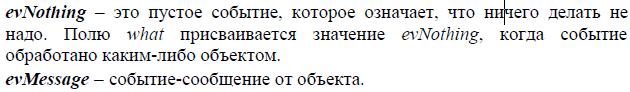
9. 

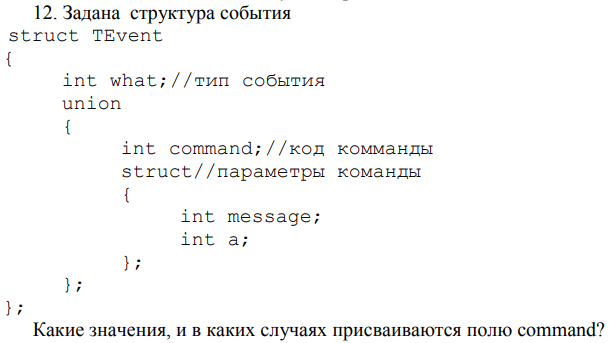


10. 

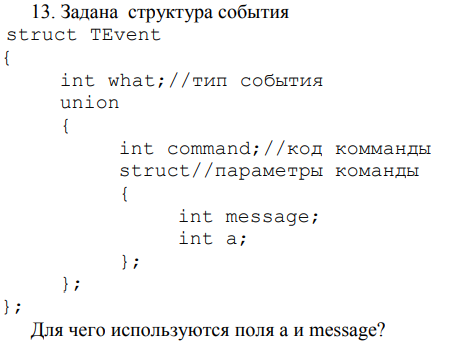




11. 

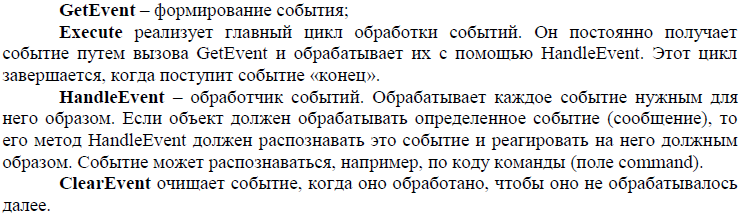


12. Код команды, которую необходимо выполнить при появлении некого события (через ветвление switch).

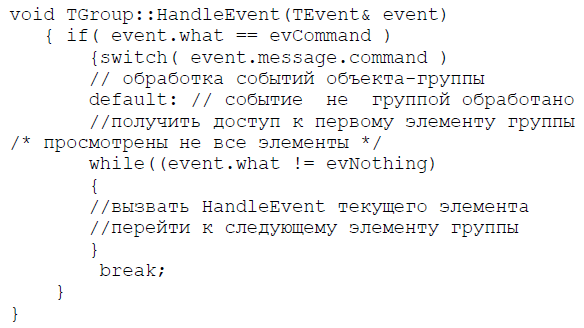


13. В поле a передаётся необходимый в дальнейшем параметр, а поле message содержит константу из переменной command.



14. 

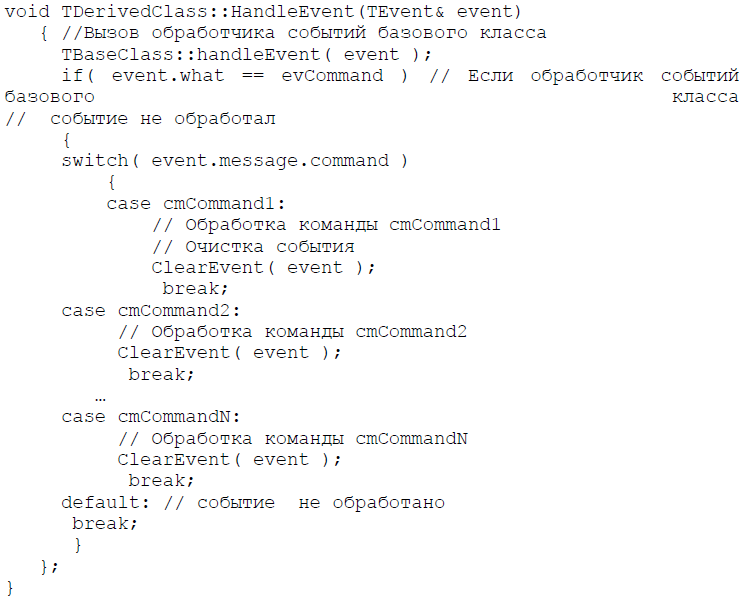


15. 



16. 



17. 



18. Формирование события.



19. Для регулирования работы программы событий: если равна 0, то программа продолжает работу, если равна 1, то завершает работу.



20. Valid отвечает за состояние параметра EndState: если параметр равен 0, цикл в методе Execute() продолжается, а если параметр равен 1, то происходит выход из цикла и остановка программы.