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 19

Выполнил:

Студент группы РИС-20-2Б

Пономарёв Артём Викторович

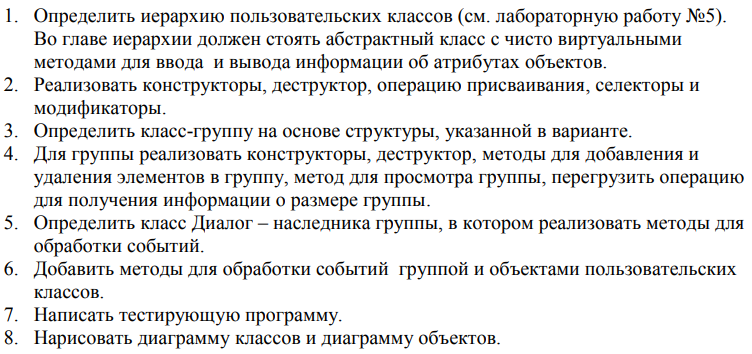
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**



**Анализ задачи**

**1.** Для решения задач необходимо:

**1.1.** Организовать абстрактный класс Object с чисто виртуальными функциями для вывода данных на консоль и ввода данных.

**1.2.** Организовать класс Vector – контейнер для объектов других классов.

**1.3.** Организовать класс TEvent для работы с событиями, в котором организовать enum EVENTS и COMMANDS для работы с событиями.

**1.4.** Организовать класс Diaglog для работы с событиями.

**1.5.** Организовать класс Print с полями name и author типа string для ввода наименования и автора.

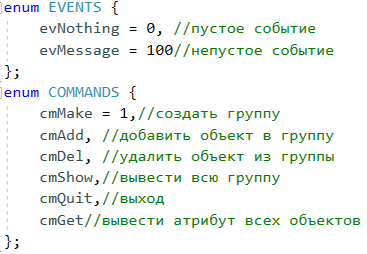
**1.6.** Организовать класс Book с полями office типа string и pages типа int для ввода названия издательства и количества страниц соответственно.

**2.** В ходе работы были использованы следующие типы данных:

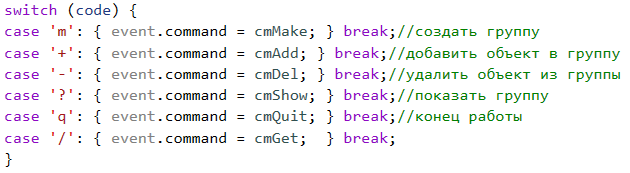
**2.1.** Экземпляр класса Dialog – D, над которым будут производиться действия.

Dialog D;

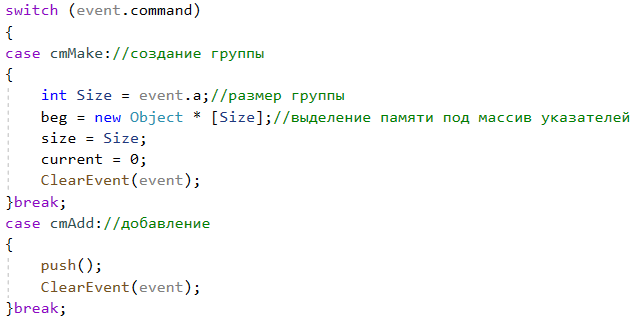
**2.2.** enum EVENTS и COMMANDS для работы с событиями.

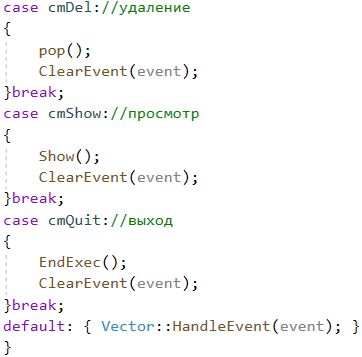


**2.3.** В функции GetEvent() класса Dialog используется switch() для переключения команды для метода HandleEvent()



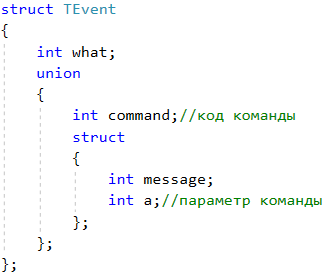
**2.3.** В функции HandleEvent() класса Dialog используется switch() для обработки полученного события





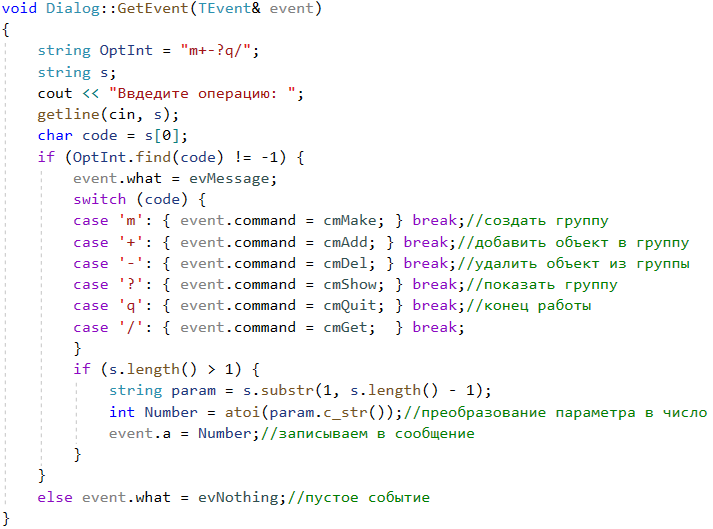
**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Работа с событиями происходит через структуру TEvent с полем what, отвечающим за тип события. Вторая часть задаёт информацию, передаваемую с событием.

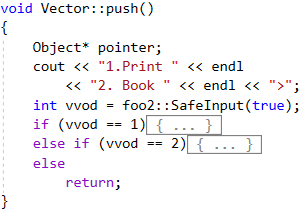


**4.** Для операций ввода и вывода использовались следующие операторы и функции:

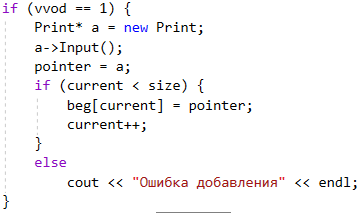
**4.1.** Для выбора события вводится ввод осуществляется через функцию getline() в переменную s. Пользователю предлагается выбор из операций создания группы, удаления элемента из группы, добавления элемента в группу, вывода элементов группы и выхода из программы.

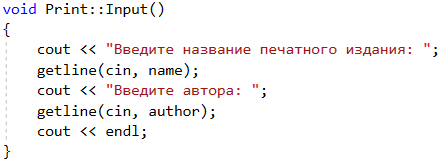


**4.2.** Для ввода экземпляров классов Print и Book используется метод push() класса Vector. Функция SafeInput() необходима для безопасного ввода (отсекаются дробные числа, отрицательные числа, символы).

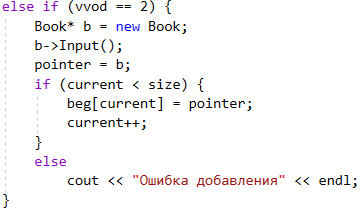


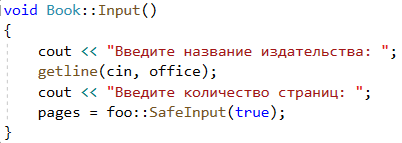
В случае выбора 1, то добавляется объект класса Print через метод Input().



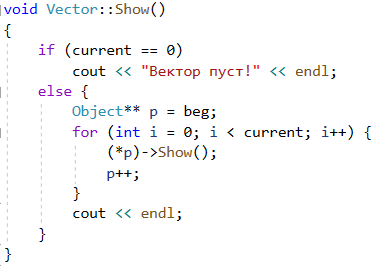


В случае выбора 2, добавляется объект класса Book.

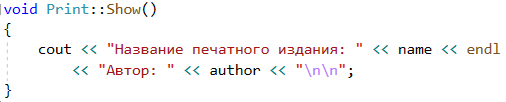


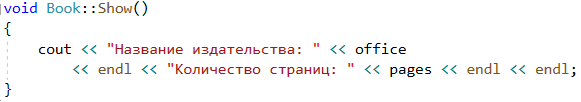


**4.3.** Для вывода на консоль введённых объектов необходимо при выборе операции ввести “?”, тогда запустится метод Show() для класса Vector.



В цикле for указатель p ссылается на область памяти, в которой записаны объекты классов Print или Book, через метод Show() вызывается метод Show() для соответствующих классов.

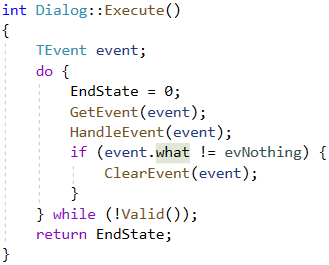


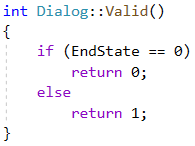


**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** В функции main() вызывается метод Execute() для объекта D.

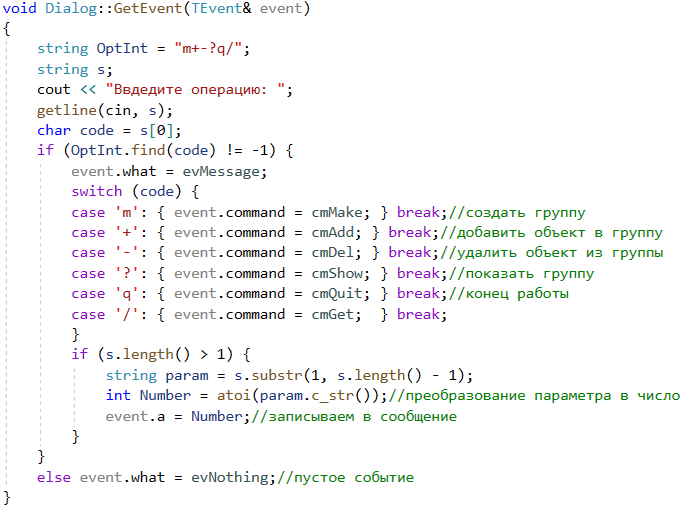
В данном методе создаётся объект event структуры TEvent – событие, с которым будут выполняться действия. В цикле do-while() вызываются методы GetEvent(), HandleEvenet() и ClearEvent(). Работа цикла осуществляется, пока параметр EndState не будет равен 1. Метод Valid необходим для возвращения значения 1/0.



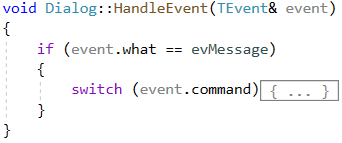


**5.2.** Метод GetEvent() получает событие: пользователь может создать группу удалить элемент из группы, добавить элемент в группу, вывести элементы группы, выйти из программы. Выбор осуществляется через switch(). Однако, чтобы попасть в ветвление switch() происходить проверка введённых данных в if(): 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.

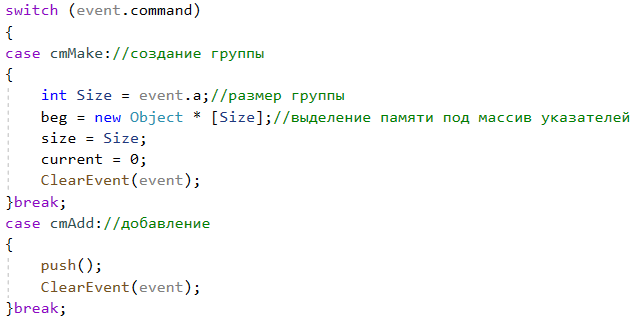
В случае ввода создания группы, нужно ввести дополнительный параметр через разделитель – размер группы (количество элементов в контейнере) через метод класса string – substr(), далее эта подстрока преобразуется в число через функцию atoi и данное значение присваивается параметру a класса TEvent.

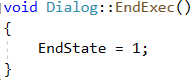


**5.3.** В функции HandleEvent() происходит обработка полученного события из метода GetEvent(). Происходит провека: если в поле what записано не пустое событие, то происходит заход в по ветке true к ветвлению switch()

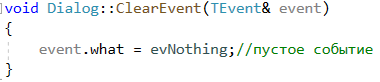


В ветвлении switch() в зависимости от выбора пользователя вызываются методы для обработки событий: если пользователь ввёл “m” – создание группы, создаётся динамический массив класса Object, наследниками которого являются классы Print, Book; если пользователь ввёл “+”, то вызывается метод push() для добавления элемента; если пользователь ввёл “-“, происходит вызов метода pop(), который удаляет элемент; если пользователь ввёл “q”, вызывается метод EndExec(), в котором параметру EndState присваивается 1 и происходит выход из цикла do-while() метода Execute().

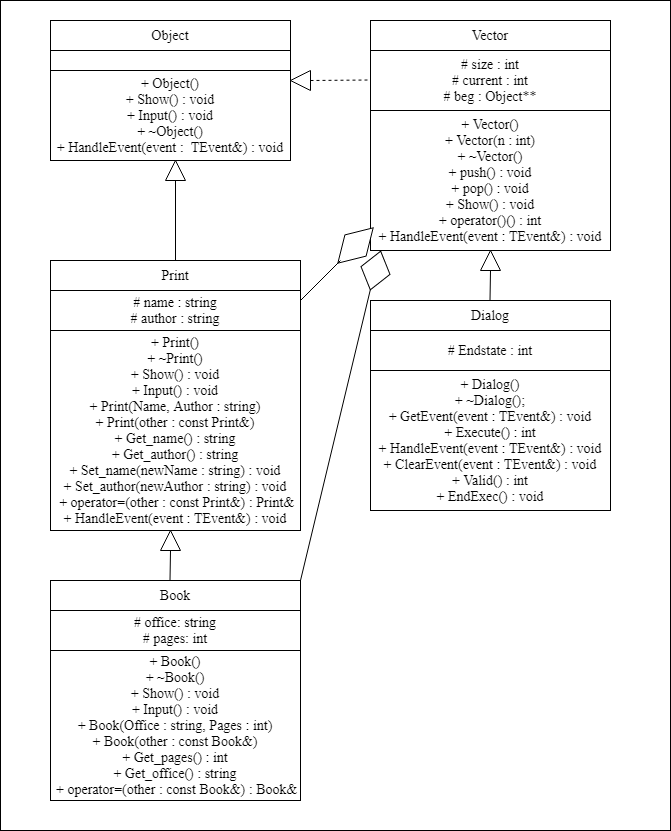




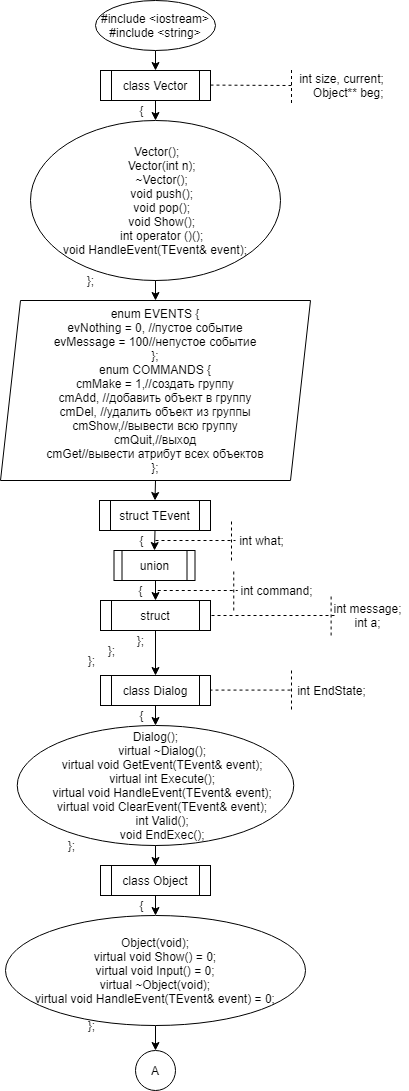
**5.4.** Метод ClearEvent() присваивает полю what константу о пустом событие – evNothing.

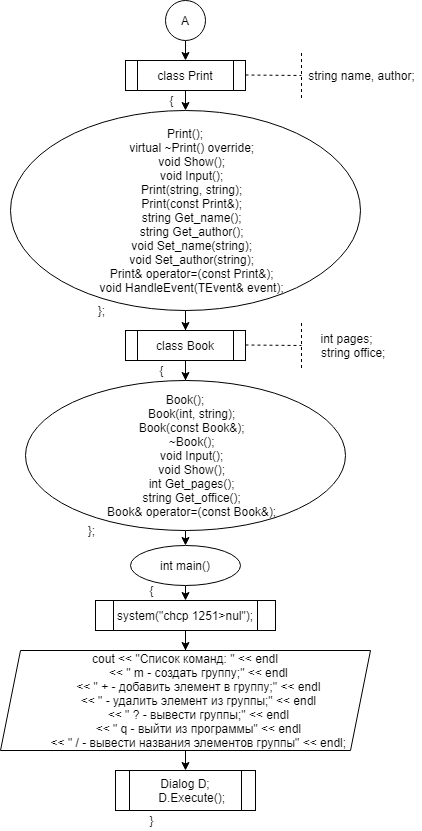


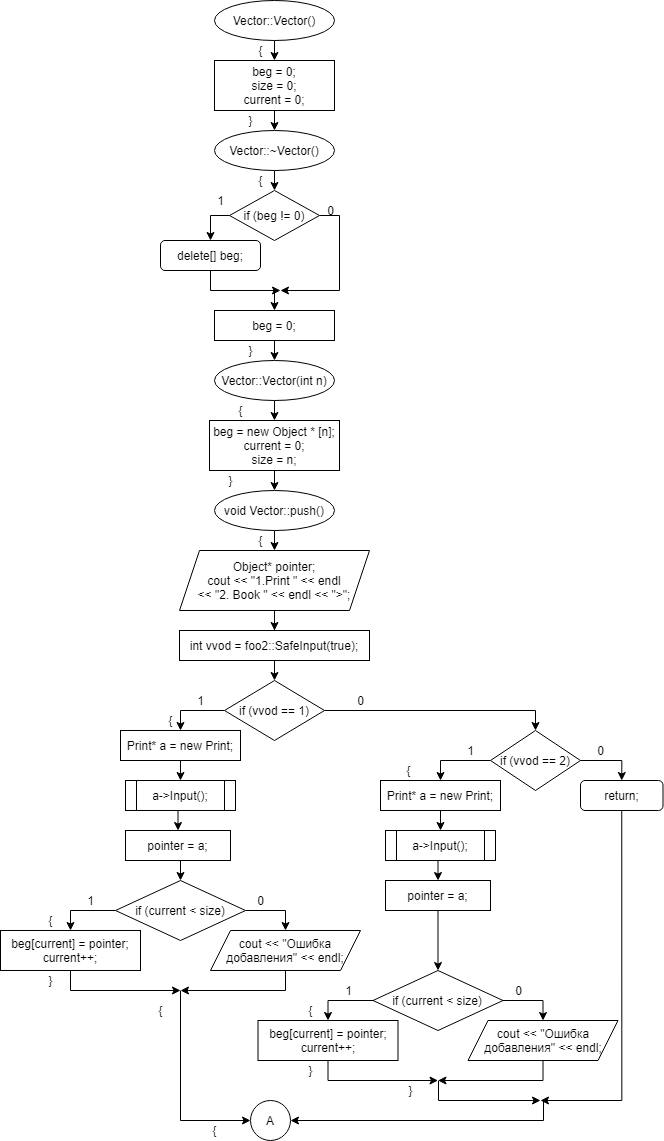
**Диаграммы классов**

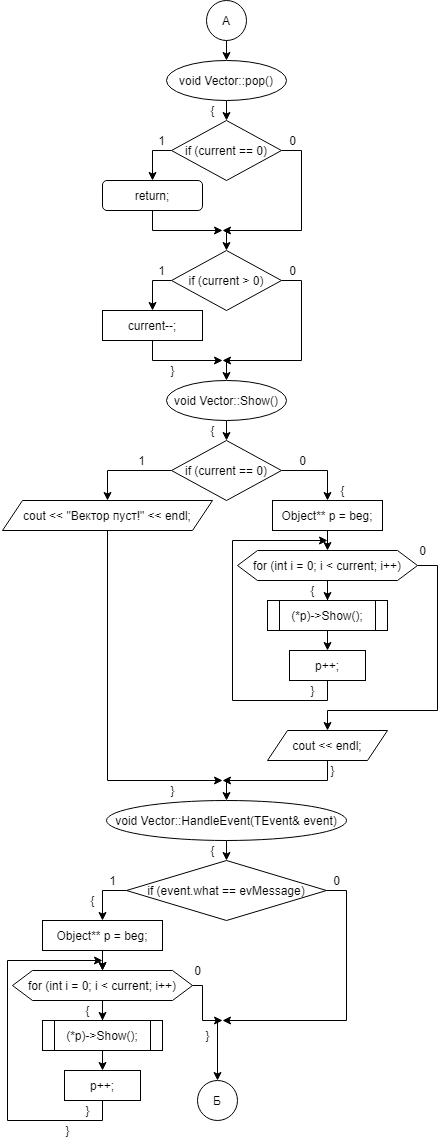


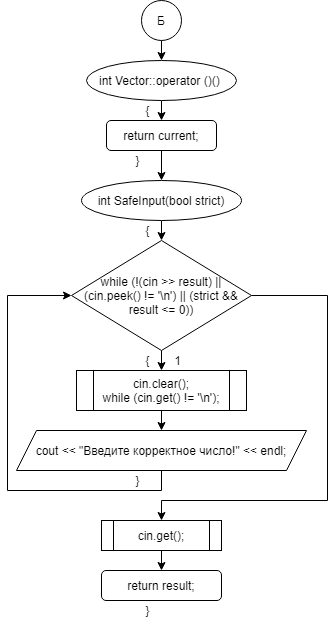
**Блок-схема**

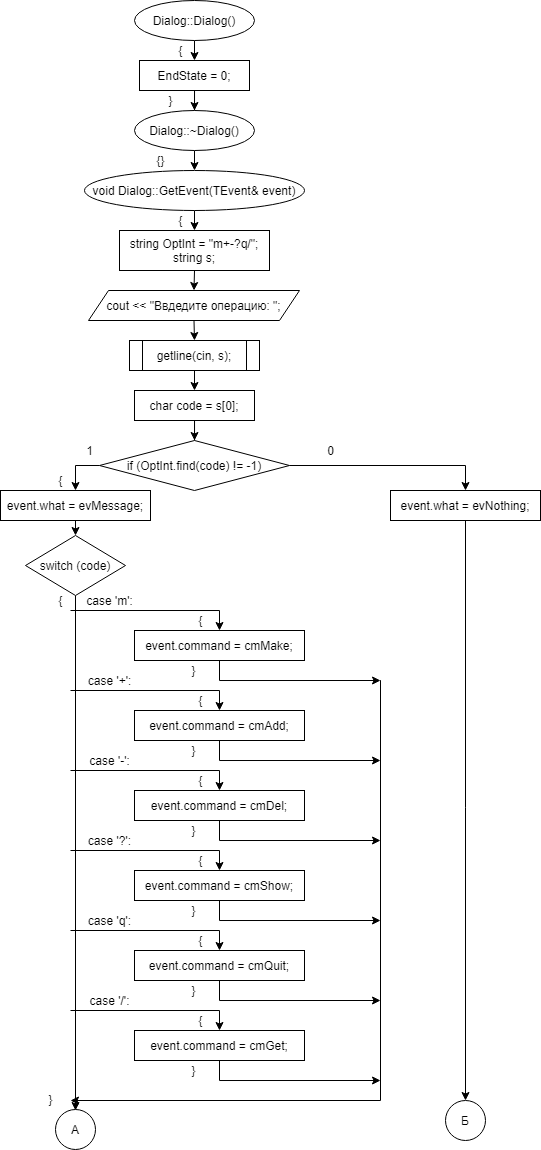


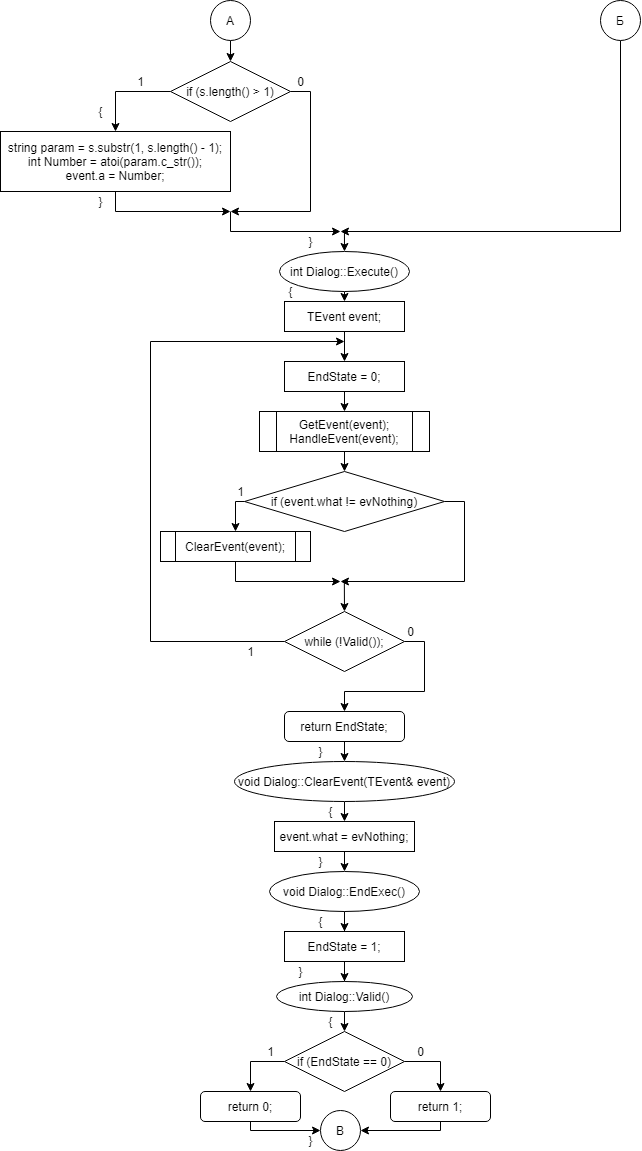


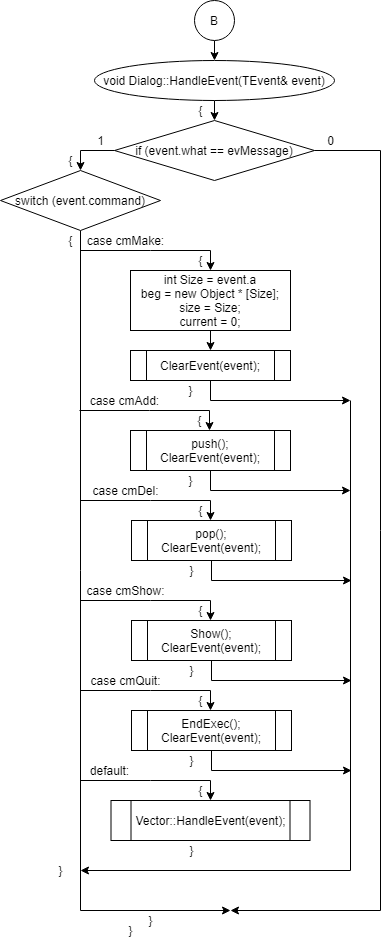


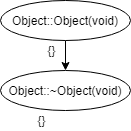


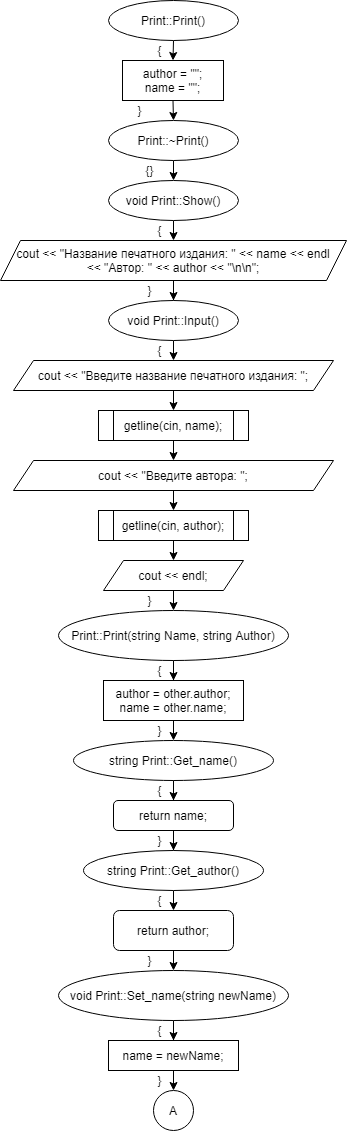


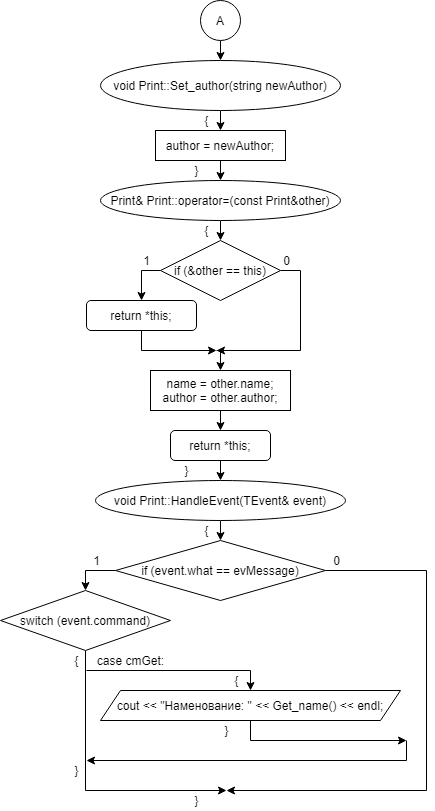


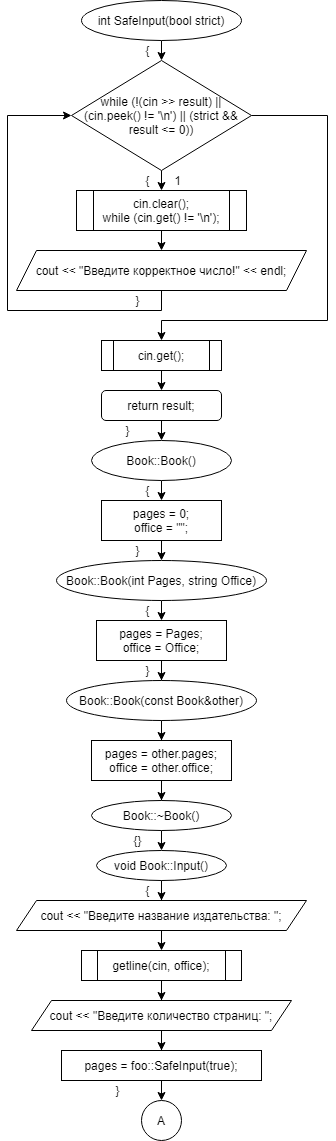


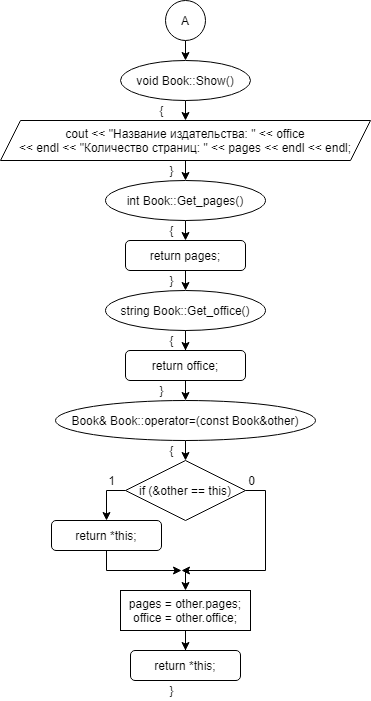




****

****

****

****

**Код**

#include "Dialog.h"

int main()

{

system("chcp 1251>nul");

/\* Print\* a = new Print;

a->Input();

a->Show();

Book\* b = new Book;

b->Input();

b->Show();\*/

/\*Vector vec(10);

vec.push();

vec.Show();

vec.pop();\*/

cout << "Список команд: " << endl

<< " m - создать группу;" << endl

<< " + - добавить элемент в группу;" << endl

<< " - удалить элемент из группы;" << endl

<< " ? - вывести группы;" << endl

<< " q - выйти из программы" << endl

<< " / - вывести названия элементов группы" << endl;

Dialog D;

D.Execute();

}

#pragma once

enum EVENTS {

evNothing = 0, //пустое событие

evMessage = 100//непустое событие

};

enum COMMANDS {

cmMake = 1,//создать группу

cmAdd, //добавить объект в группу

cmDel, //удалить объект из группы

cmShow,//вывести всю группу

cmQuit,//выход

cmGet//вывести атрибут всех объектов

};

struct TEvent

{

int what;

union

{

int command;//код команды

struct

{

int message;

int a;//параметр команды

};

};

};

#pragma once

#include <iostream>

#include <string>

#include "Event.h"

using namespace std;

class Object

{

public:

Object(void);

virtual void Show() = 0;

virtual void Input() = 0;

virtual ~Object(void);

virtual void HandleEvent(TEvent& event) = 0;

};

#include "Object.h"

Object::Object(void)

{

}

Object::~Object(void)

{

}

#pragma once

#include "Object.h"

class Vector {

protected:

int size, current;

Object\*\* beg;

public:

Vector();

Vector(int n);

~Vector();

void push();

void pop();

void Show();

int operator ()();

//friend ostream& operator<<(ostream& out, const Vector& t);

void HandleEvent(TEvent& event);

};

#include "Vector.h"

#include "Object.h"

#include "Print.h"

#include "Book.h"

namespace foo2

{

int SafeInput(bool strict)

{

int result;

while (!(cin >> result) || (cin.peek() != '\n') || (strict && result <= 0)) {

cin.clear();

while (cin.get() != '\n');

cout << "Введите корректное число!" << endl;

}

cin.get();

return result;

}

}

Vector::Vector()

{

beg = 0;

size = 0;

current = 0;

}

Vector::~Vector()

{

if (beg != 0)

delete[] beg;

beg = 0;

}

Vector::Vector(int n)

{

beg = new Object \* [n];

current = 0;

size = n;

}

void Vector::push()

{

Object\* pointer;

cout << "1.Print " << endl

<< "2. Book " << endl << ">";

int vvod = foo2::SafeInput(true);

if (vvod == 1) {

Print\* a = new Print;

a->Input();

pointer = a;

if (current < size) {

beg[current] = pointer;

current++;

}

else

cout << "Ошибка добавления" << endl;

}

else if (vvod == 2) {

Book\* b = new Book;

b->Input();

pointer = b;

if (current < size) {

beg[current] = pointer;

current++;

}

else

cout << "Ошибка добавления" << endl;

}

else

return;

}

void Vector::pop()

{

if (current == 0)

return;//пустой

if (current > 0)

current--;

}

void Vector::Show()

{

if (current == 0)

cout << "Вектор пуст!" << endl;

else {

Object\*\* p = beg;

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

(\*p)->Show();

p++;

}

cout << endl;

}

}

void Vector::HandleEvent(TEvent& event)

{

if (event.what == evMessage) {

Object\*\* p = beg;

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

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

p++;

}

}

}

int Vector::operator ()()

{

return current;

}

#pragma once

#include "Object.h"

class Print :

public Object

{

protected:

string name, author;

public:

Print();

virtual ~Print();

void Show();

void Input();

Print(string, string);

Print(const Print&);

string Get\_name();

string Get\_author();

void Set\_name(string);

void Set\_author(string);

Print& operator=(const Print&);

void HandleEvent(TEvent& event);

};

#include "Print.h"

Print::Print()

{

author = "";

name = "";

}

Print::~Print()

{

}

void Print::Show()

{

cout << "Название печатного издания: " << name << endl

<< "Автор: " << author << "\n\n";

}

void Print::Input()

{

cout << "Введите название печатного издания: ";

getline(cin, name);

cout << "Введите автора: ";

getline(cin, author);

cout << endl;

}

Print::Print(string Name, string Author)

{

name = Name;

author = Author;

}

Print::Print(const Print&other)

{

author = other.author;

name = other.name;

}

string Print::Get\_name()

{

return name;

}

string Print::Get\_author()

{

return author;

}

void Print::Set\_name(string newName)

{

name = newName;

}

void Print::Set\_author(string newAuthor)

{

author = newAuthor;

}

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

{

if (&other == this)

return \*this;

name = other.name;

author = other.author;

return \*this;

}

void Print::HandleEvent(TEvent& event)

{

if (event.what == evMessage) {

switch (event.command) {

case cmGet: { cout << "Наменование: " << Get\_name() << endl; }break;

}

}

}

#pragma once

#include "Print.h"

class Book :

public Print

{

protected:

int pages;

string office;

public:

Book();

Book(int, string);

Book(const Book&);

~Book();

void Input();

void Show();

int Get\_pages();

string Get\_office();

Book& operator=(const Book&);

};

#include "Book.h"

namespace foo

{

int SafeInput(bool strict)

{

int result;

while (!(cin >> result) || (cin.peek() != '\n') || (strict && result <= 0)) {

cin.clear();

while (cin.get() != '\n');

cout << "Введите корректное число!" << endl;

}

cin.get();

return result;

}

}

Book::Book()

{

pages = 0;

office = "";

}

Book::Book(int Pages, string Office)

{

pages = Pages;

office = Office;

}

Book::Book(const Book&other)

{

pages = other.pages;

office = other.office;

}

Book::~Book()

{

}

void Book::Input()

{

cout << "Введите название издательства: ";

getline(cin, office);

cout << "Введите количество страниц: ";

pages = foo::SafeInput(true);

}

void Book::Show()

{

cout << "Название издательства: " << office

<< endl << "Количество страниц: " << pages << endl << endl;

}

int Book::Get\_pages()

{

return pages;

}

string Book::Get\_office()

{

return office;

}

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

{

if (&other == this)

return \*this;

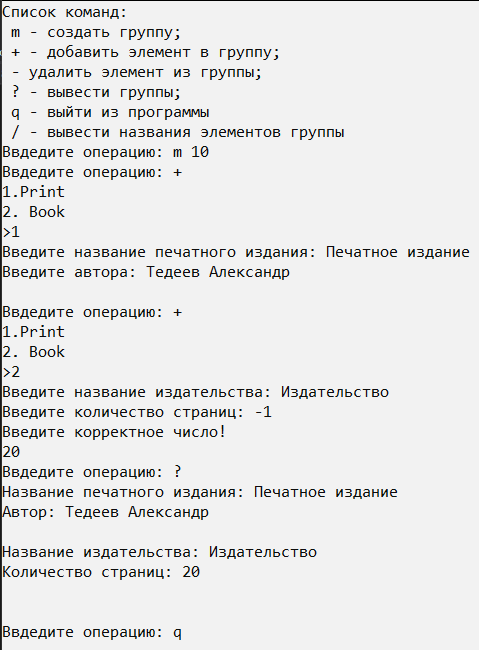
pages = other.pages;

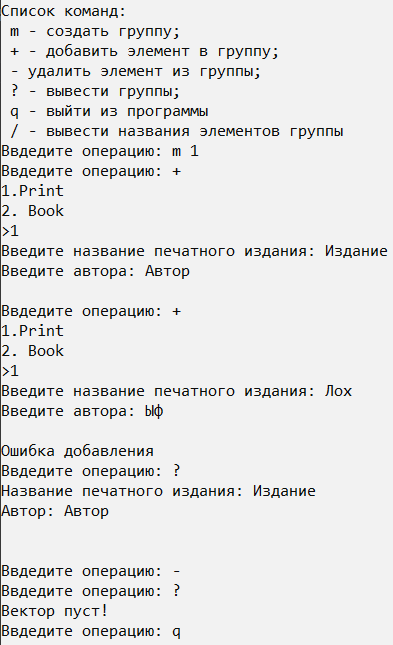
office = other.office;

return \*this;

}

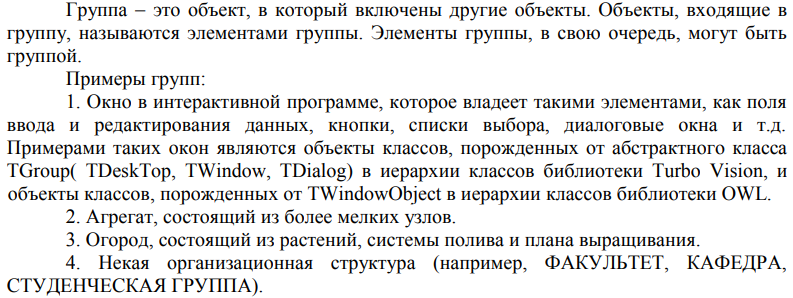
**Скриншоты**



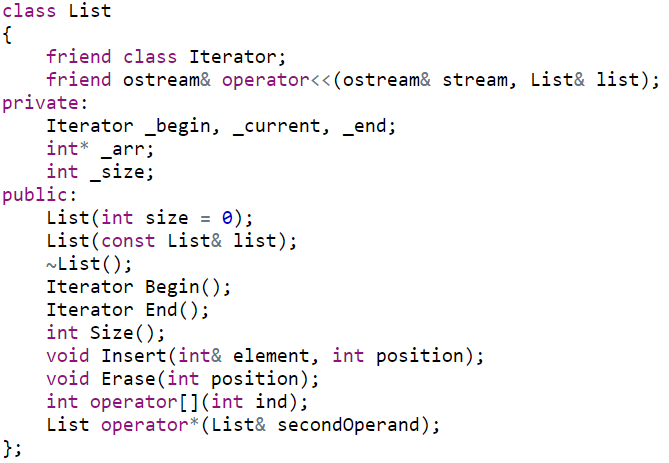


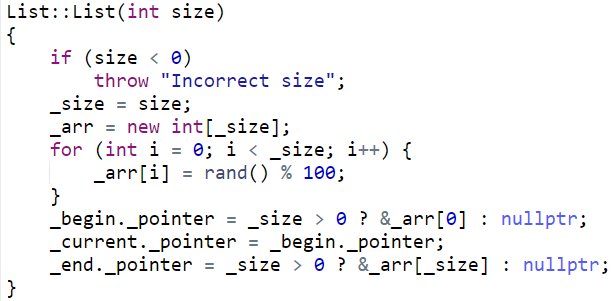
**Контрольные вопросы**

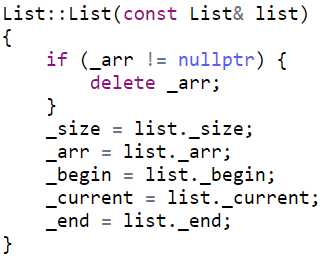


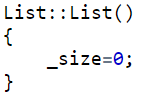
1. 



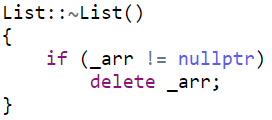
2. 

3. 

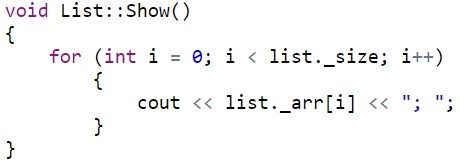




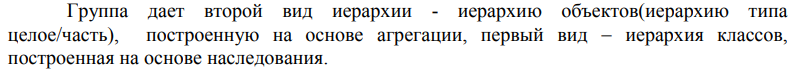


4. 



5. 



6. 

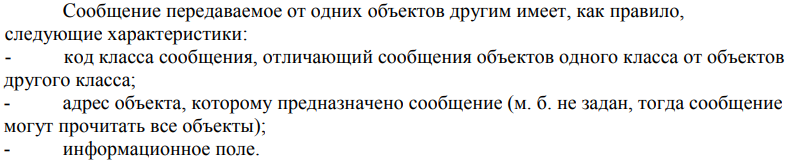


7. Для того, чтобы хранить ссылку на дочерний класс.

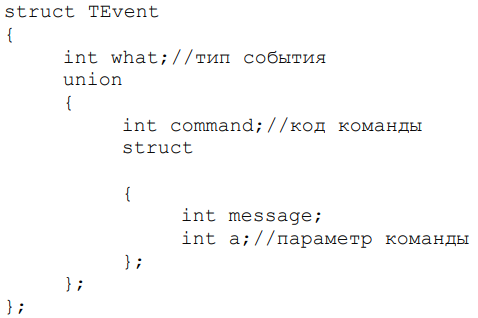


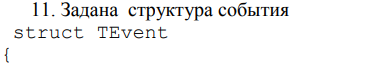
8. Событие – это пакеты информации, которыми обмениваются объекты и которые создаются объектно-ориентированной средой в ответ на те или иные действия пользователя.

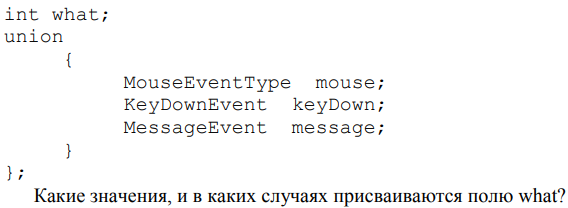


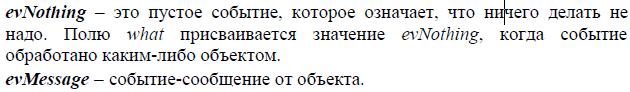
9. 

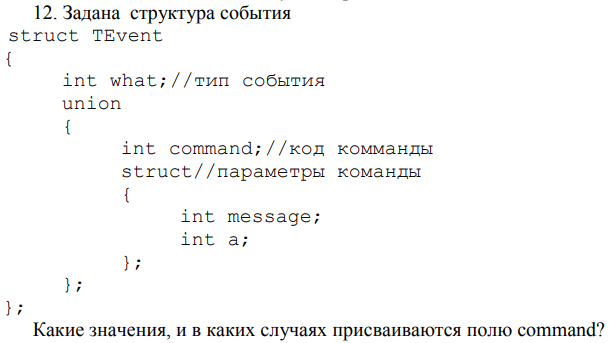


10. 

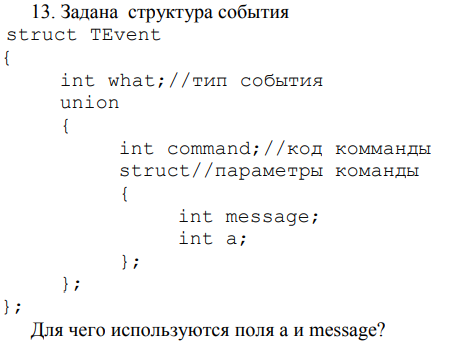




11. 

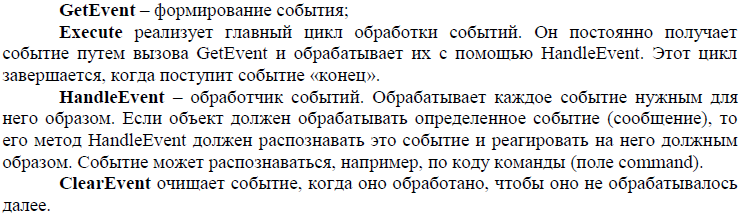


12. Код команды, которую необходимо выполнить при появлении некого события (через ветвление switch).

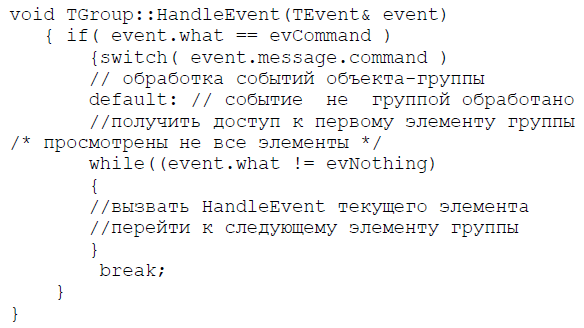


13. В поле a передаётся необходимый в дальнейшем параметр, а поле message содержит константу из переменной command.



14. 

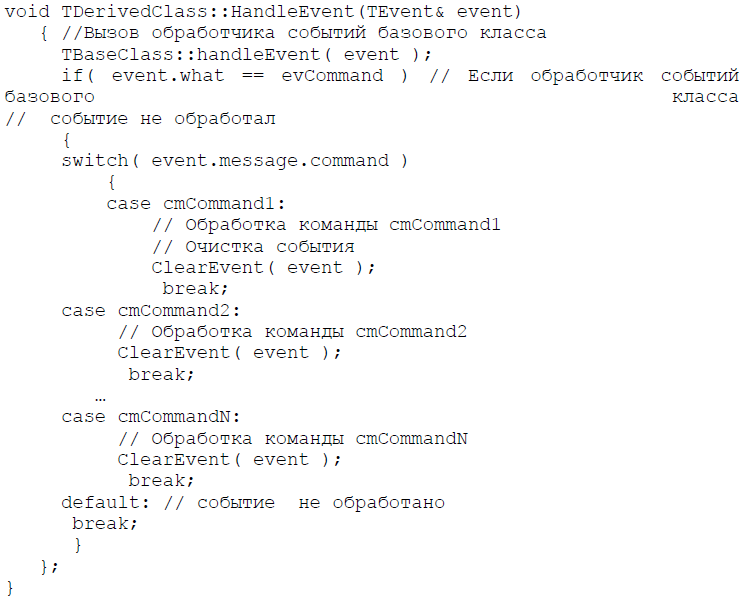


15. 



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.