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 10

Выполнил:

Студент группы ИВТ-20-2Б Галинов О.Ю.

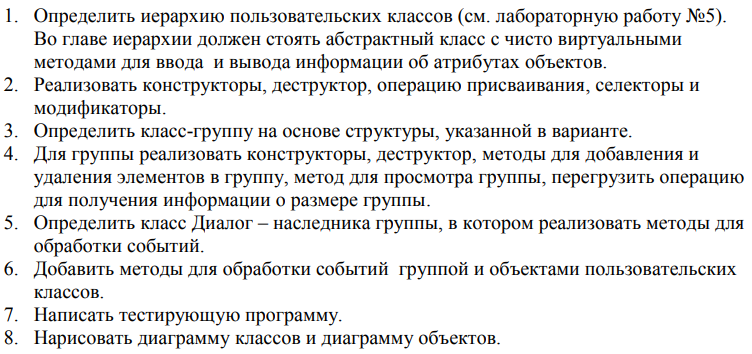
Проверила:

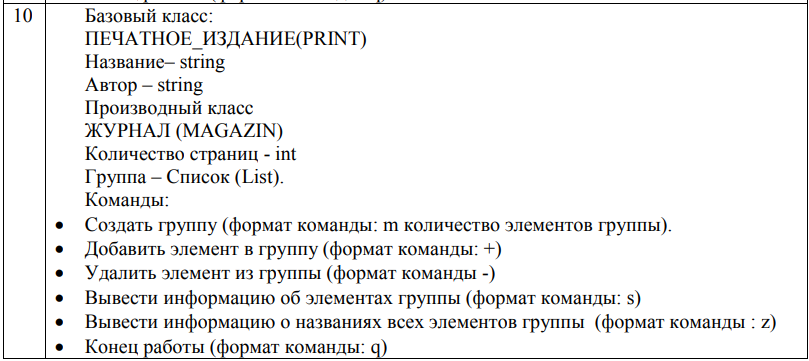
Доцент кафедры ИТАС

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 структуру TEvent для работы с событиями, в котором организовать поля типа int message и command для работы с событиями.

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

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

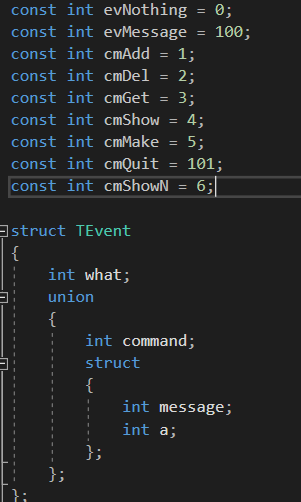
**1.6.** Организовать класс Magazine с полем pages типа int для ввода количества страниц.

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

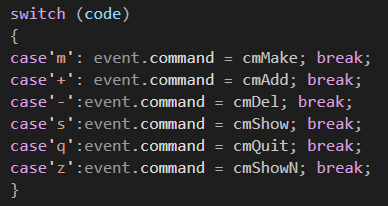
**2.1.** Экземпляр класса Dialog – D, над которым будут производиться действия.

Dialog D;

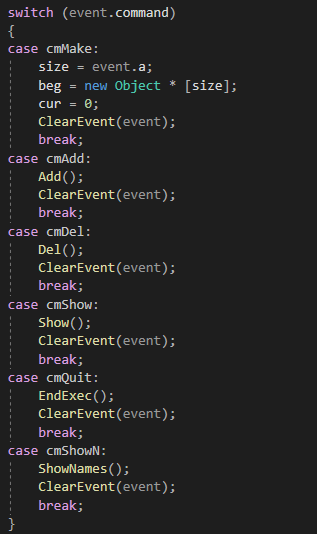
**2.2.** message и command для работы с событиями.



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

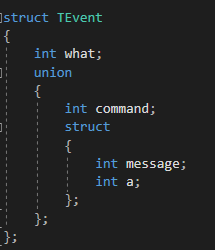


**2.3.** В функции HandleEvent() класса Dialog используется switch() для обработки полученного события



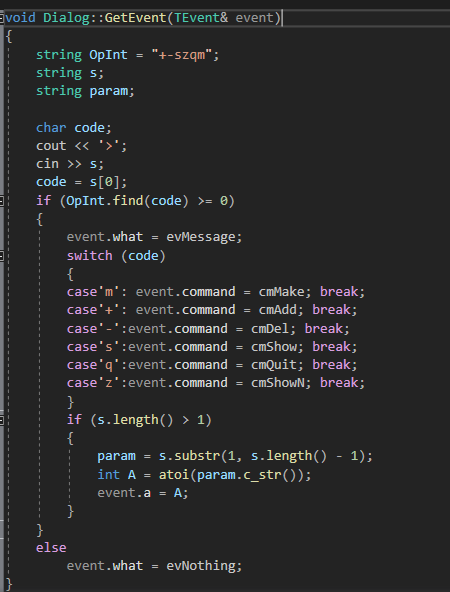
**3.** Для решения задачи данные были представлены в следующем виде:

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

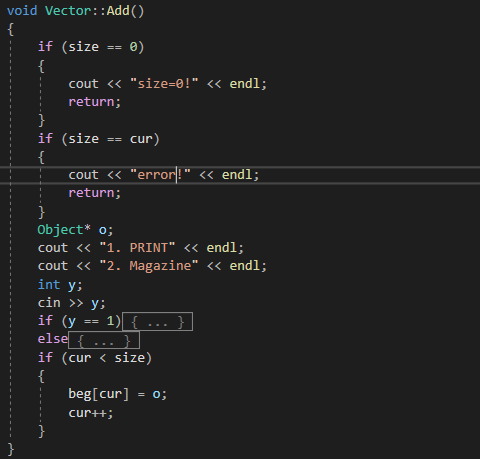


**4.** Для операций ввода и вывода использовались следующие операторы и функции:

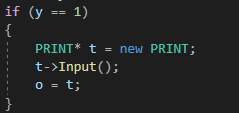
**4.1.** Для выбора события вводится ввод осуществляется через функцию cin в переменную s. Пользователю предлагается выбор из операций создания группы, удаления элемента из группы, добавления элемента в группу, вывода элементов группы и выхода из программы.

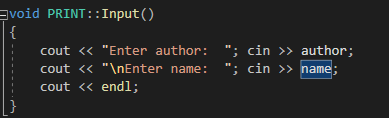


**4.2.** Для ввода экземпляров классов Print и Magazine используется метод Add() класса Vector.

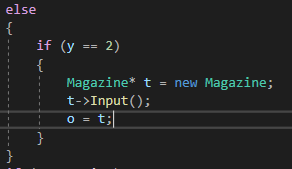


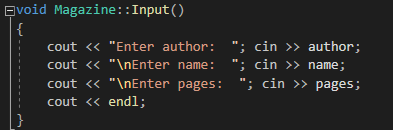
В случае выбора 1, то добавляется объект класса Print через метод Input().



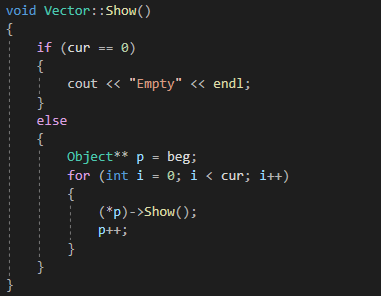


В случае выбора 2, добавляется объект класса Magazine.

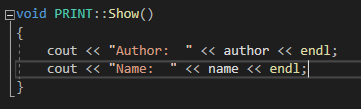


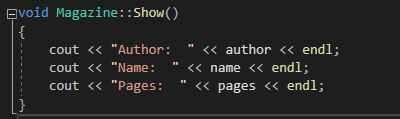


**4.3.** Для вывода на консоль введённых объектов необходимо при выборе операции ввести “s”, тогда запустится метод Show() для класса Vector.



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

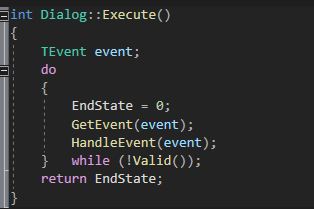


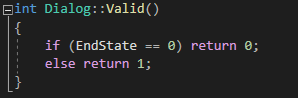


**5.** Поставленные задачи будут решены следующими действиями:

**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() и HandleEvenet().Работа цикла осуществляется, пока параметр EndState не будет равен 1. Метод Valid необходим для возвращения значения 1/0.



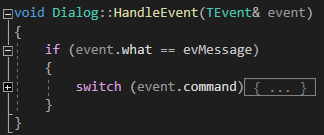


**5.2.** Метод GetEvent() получает событие: пользователь может создать группу удалить элемент из группы, добавить элемент в группу, вывести элементы группы, выйти из программы. Выбор осуществляется через switch(). Однако, чтобы попасть в ветвление switch() происходить проверка введённых данных в if():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

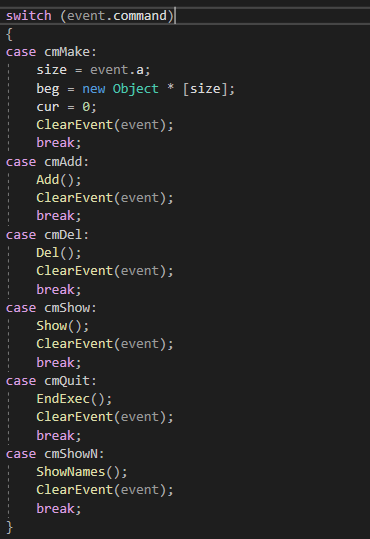
В случае ввода создания группы, нужно ввести дополнительный параметр через разделитель – размер группы (количество элементов в контейнере) через метод класса string – substr(), далее эта подстрока преобразуется в число через функцию atoi и данное значение присваивается параметру a класса TEvent.

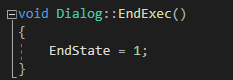


**5.3.** В функции HandleEvent() происходит обработка полученного события из метода GetEvent(). Происходит провека: если в поле what записано не пустое событие, то происходит заход в по ветке true к ветвлению switch()

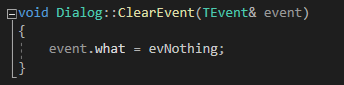


В ветвлении switch() в зависимости от выбора пользователя вызываются методы для обработки событий: если пользователь ввёл “m” – создание группы, создаётся динамический массив класса Object, наследниками которого являются классы Print, Magazine; если пользователь ввёл “+”, то вызывается метод push() для 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() и тд.

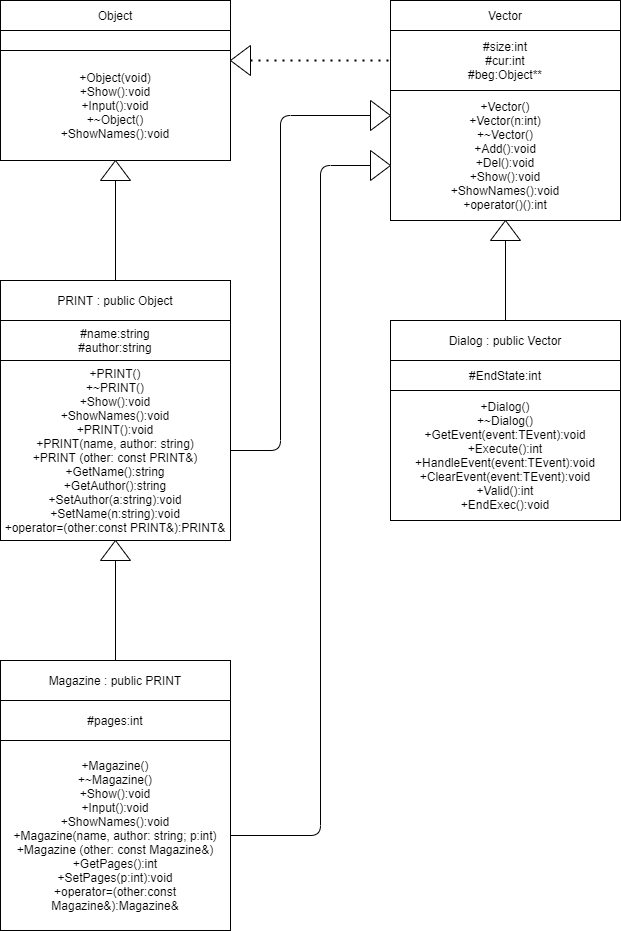




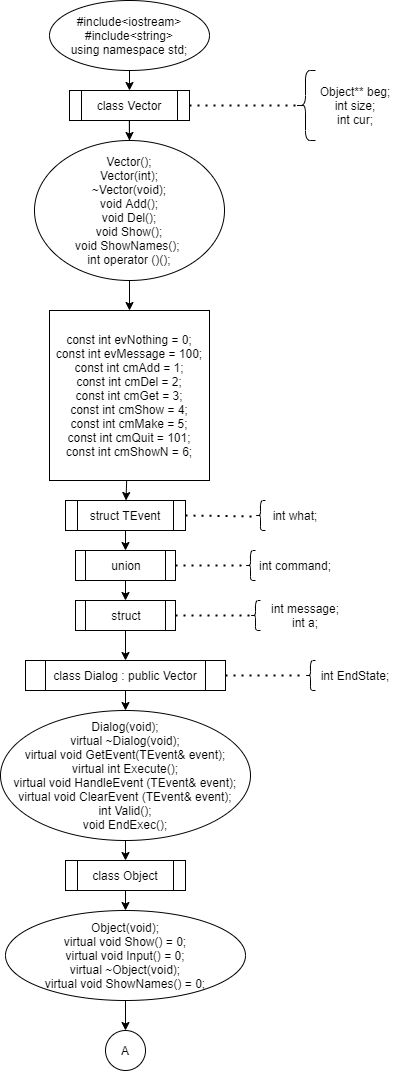
**5.4.** Метод ClearEvent() присваивает полю what константу о пустом событие – evNothing.

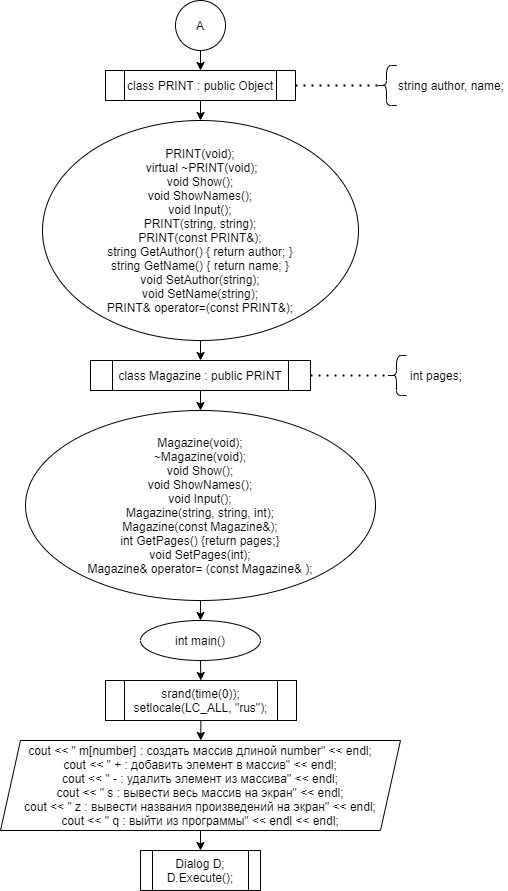


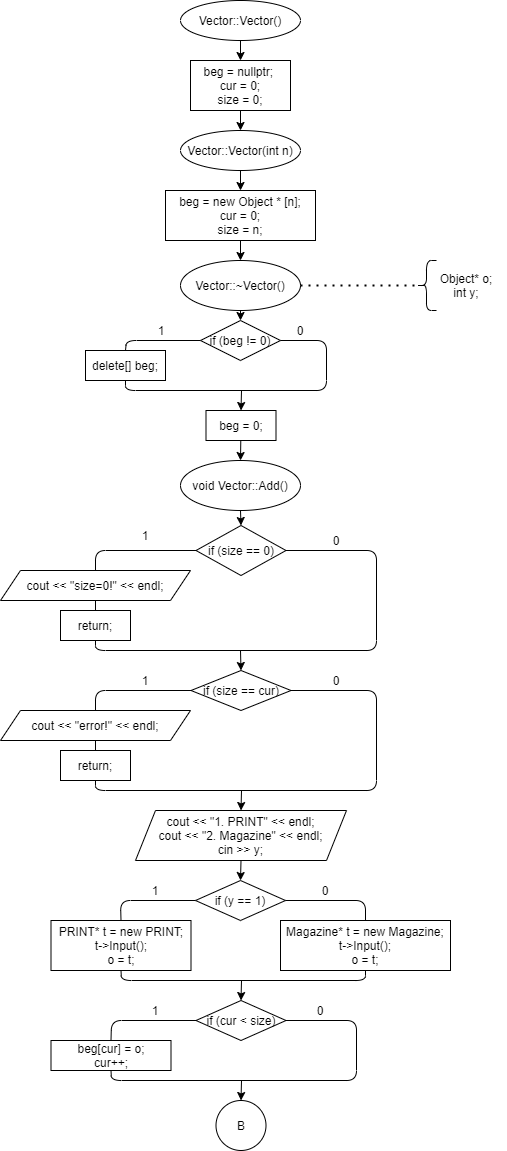
**Диаграммы классов**

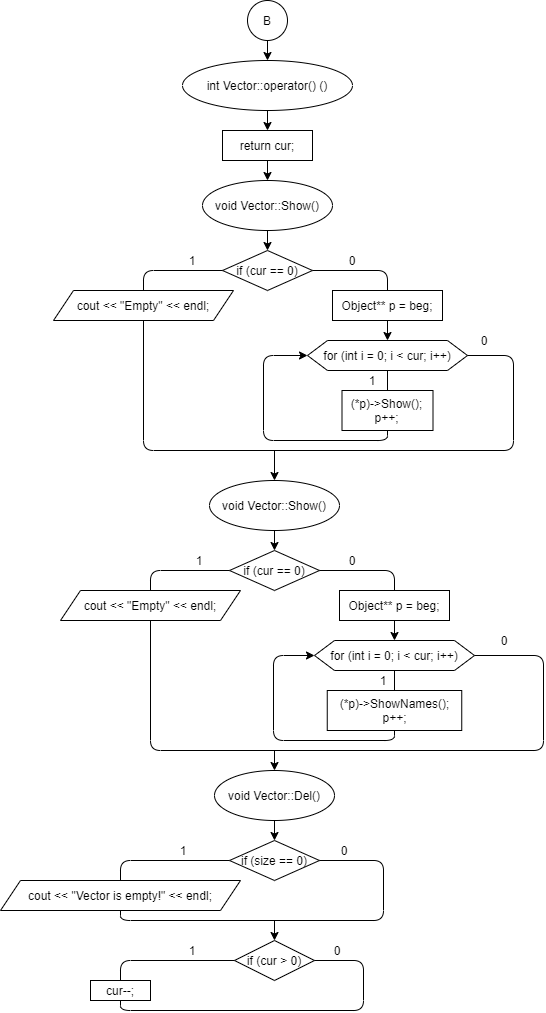


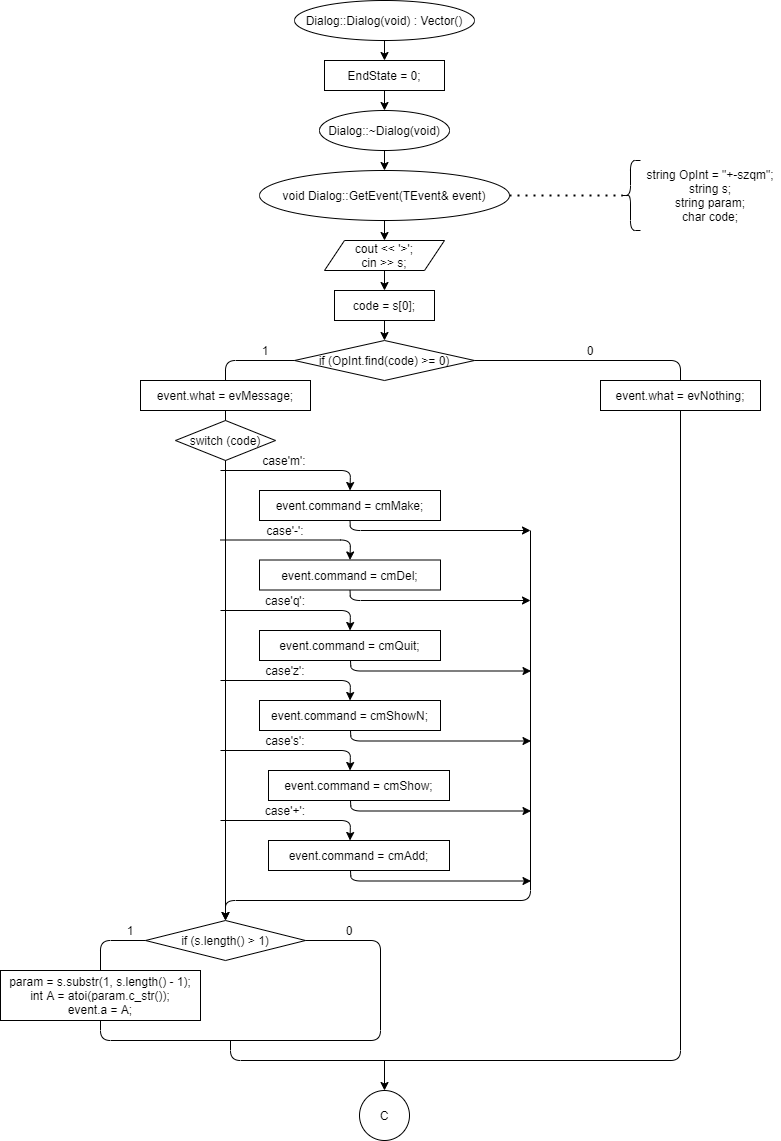
**Блок-схема**

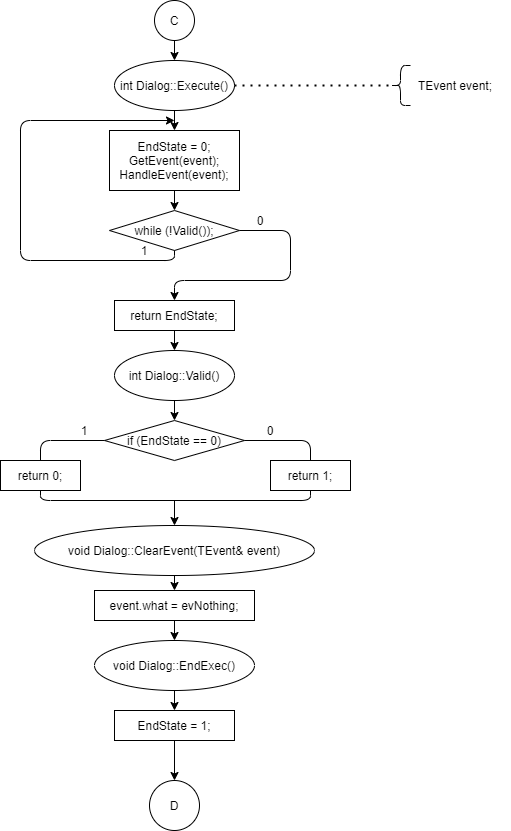


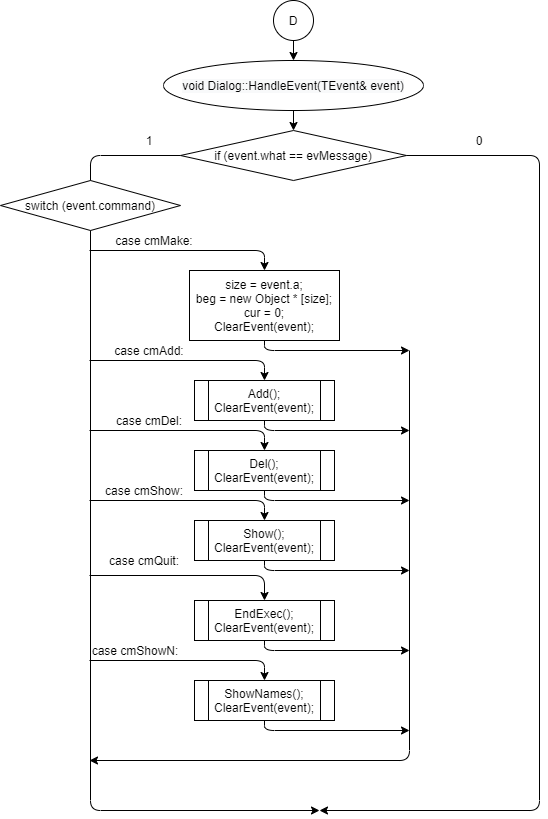


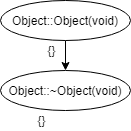


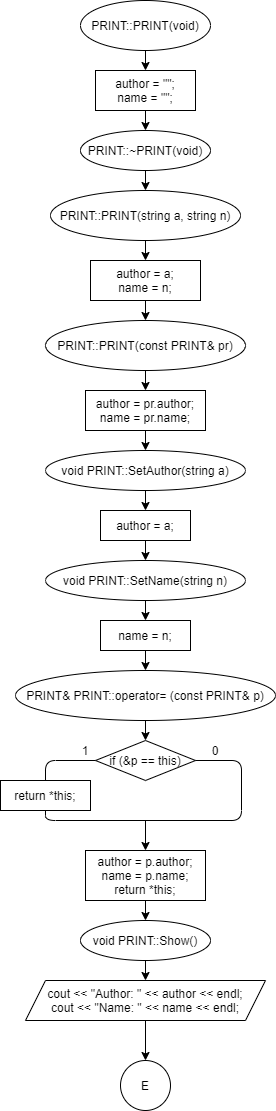


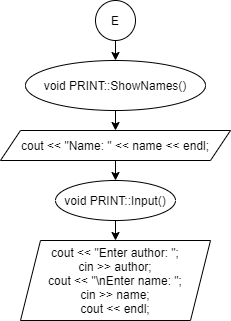


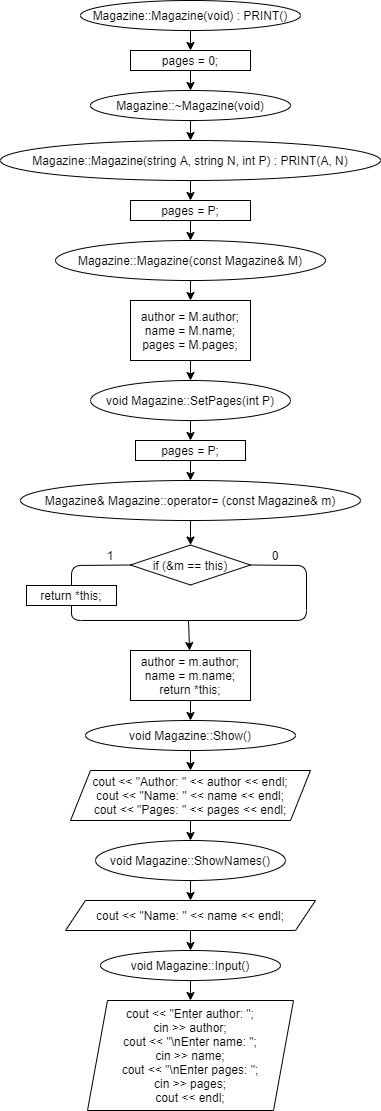






****

****

****

**Код**

**LaboratoryWork\_18.8**

#include <iostream>

#include"Dialog.h"

using namespace std;

int main()

{

srand(time(0));

setlocale(LC\_ALL, "rus");

cout << " m[number] : создать массив длиной number" << endl;

cout << " + : добавить элемент в массив" << endl;

cout << " - : удалить элемент из массива" << endl;

cout << " s : вывести весь массив на экран" << endl;

cout << " z : вывести названия произведений на экран" << endl;

cout << " q : выйти из программы" << endl << endl;

Dialog D;

D.Execute();

}

**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 = 5;

const int cmQuit = 101;

const int cmShowN = 6;

struct TEvent

{

int what;

union

{

int command;

struct

{

int message;

int a;

};

};

};

**Object.h**

#pragma once

#include<iostream>

using namespace std;

class Object

{

public:

Object(void);

virtual void Show() = 0;

virtual void Input() = 0;

virtual ~Object(void);

virtual void ShowNames() = 0;

};

#include "Object.h"

**Object.cpp**

Object::Object(void)

{

}

Object::~Object(void)

{

}

**Vector.h**

#include"Object.h"

#include"Magazine.h"

class Vector

{

public:

Vector();

Vector(int);

~Vector(void);

void Add();

void Del();

void Show();

void ShowNames();

int operator ()();

protected:

Object\*\* beg;

int size;

int cur;

};

**Vector.cpp**

#pragma once

#include<iostream>

#include "Vector.h"

#include"Magazine.h"

using namespace std;

Vector::Vector()

{

beg = nullptr;

cur = 0;

size = 0;

}

Vector::Vector(int n)

{

beg = new Object \* [n];

cur = 0;

size = n;

}

Vector::~Vector()

{

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

beg = 0;

}

//METHODS

void Vector::Add()

{

if (size == 0)

{

cout << "size=0!" << endl;

return;

}

if (size == cur)

{

cout << "error!" << endl;

return;

}

Object\* o;

cout << "1. PRINT" << endl;

cout << "2. Magazine" << endl;

int y;

cin >> y;

if (y == 1)

{

PRINT\* t = new PRINT;

t->Input();

o = t;

}

else

{

Magazine\* t = new Magazine;

t->Input();

o = t;

}

if (cur < size)

{

beg[cur] = o;

cur++;

}

}

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

void Vector::Show()

{

if (cur == 0)

{

cout << "Empty" << endl;

}

else

{

Object\*\* p = beg;

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

{

(\*p)->Show();

p++;

}

}

}

void Vector::ShowNames()

{

if (cur == 0)

{

cout << "Empty" << endl;

}

else

{

Object\*\* p = beg;

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

{

(\*p)->ShowNames();

p++;

}

}

}

void Vector::Del()

{

if (size == 0)

{

cout << "Vector is empty!" << endl;

}

if (cur > 0)

cur--;

}

**Print.h**

#include <iostream>

#include"Object.h"

using namespace std;

class PRINT : public Object

{

public:

PRINT(void);

public:

virtual ~PRINT(void);

void Show();

void ShowNames();

void Input();

PRINT(string, string);

PRINT(const PRINT&);

// selecters

string GetAuthor() { return author; }

string GetName() { return name; }

//modificators

void SetAuthor(string);

void SetName(string);

PRINT& operator=(const PRINT&);

protected:

string author, name;

};

**Print.cpp**

#include "Print.h"

//construtor

PRINT::PRINT(void)

{

author = "";

name = "";

}

//destructor

PRINT::~PRINT(void) {}

//constructor w parametres

PRINT::PRINT(string a, string n)

{

author = a;

name = n;

}

//copy constructor

PRINT::PRINT(const PRINT& pr)

{

author = pr.author;

name = pr.name;

}

//setters

void PRINT::SetAuthor(string a)

{

author = a;

}

void PRINT::SetName(string n)

{

name = n;

}

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

{

if (&p == this) return \*this;

author = p.author;

name = p.name;

return \*this;

}

void PRINT::Show()

{

cout << "Author: " << author << endl;

cout << "Name: " << name << endl;

}

void PRINT::ShowNames()

{

cout << "Name: " << name << endl;

}

void PRINT::Input()

{

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

cout << "\nEnter name: "; cin >> name;

cout << endl;

}

**Magazine.h**

#pragma once

#include"Print.h"

class Magazine : public PRINT

{

public:

Magazine(void);

public:

~Magazine(void);

void Show();

void ShowNames();

void Input();

Magazine(string, string, int);

Magazine(const Magazine&);

int GetPages() {return pages;}

void SetPages(int);

Magazine& operator= (const Magazine& );

protected:

int pages;

};

**Magazine.cpp**

#include "Magazine.h"

Magazine::Magazine(void) : PRINT()

{

pages = 0;

}

Magazine::~Magazine(void) {}

Magazine::Magazine(string A, string N, int P) : PRINT(A, N)

{

pages = P;

}

Magazine::Magazine(const Magazine& M)

{

author = M.author;

name = M.name;

pages = M.pages;

}

void Magazine::SetPages(int P)

{

pages = P;

}

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

{

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

author = m.author;

name = m.name;

return \*this;

}

void Magazine::Show()

{

cout << "Author: " << author << endl;

cout << "Name: " << name << endl;

cout << "Pages: " << pages << endl;

}

void Magazine::ShowNames()

{

cout << "Name: " << name << endl;

}

void Magazine::Input()

{

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

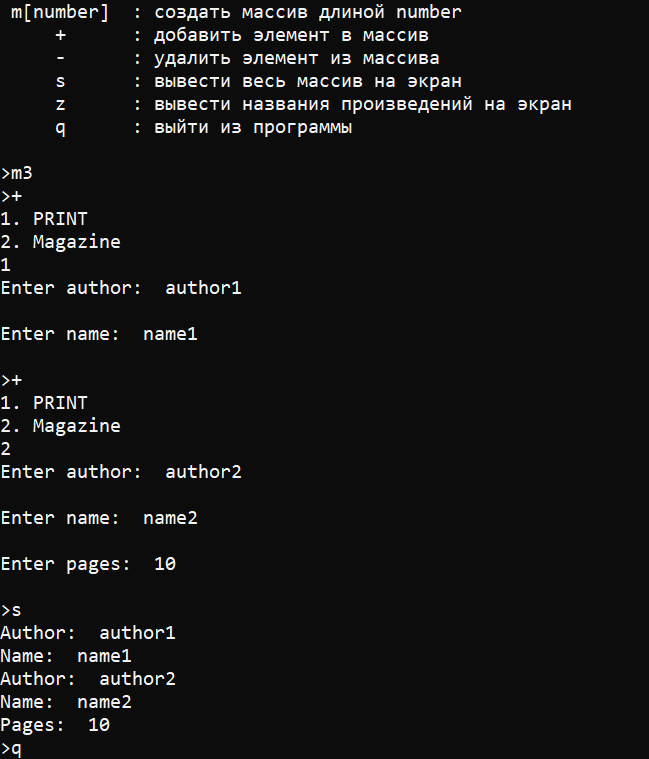
cout << "\nEnter name: "; cin >> name;

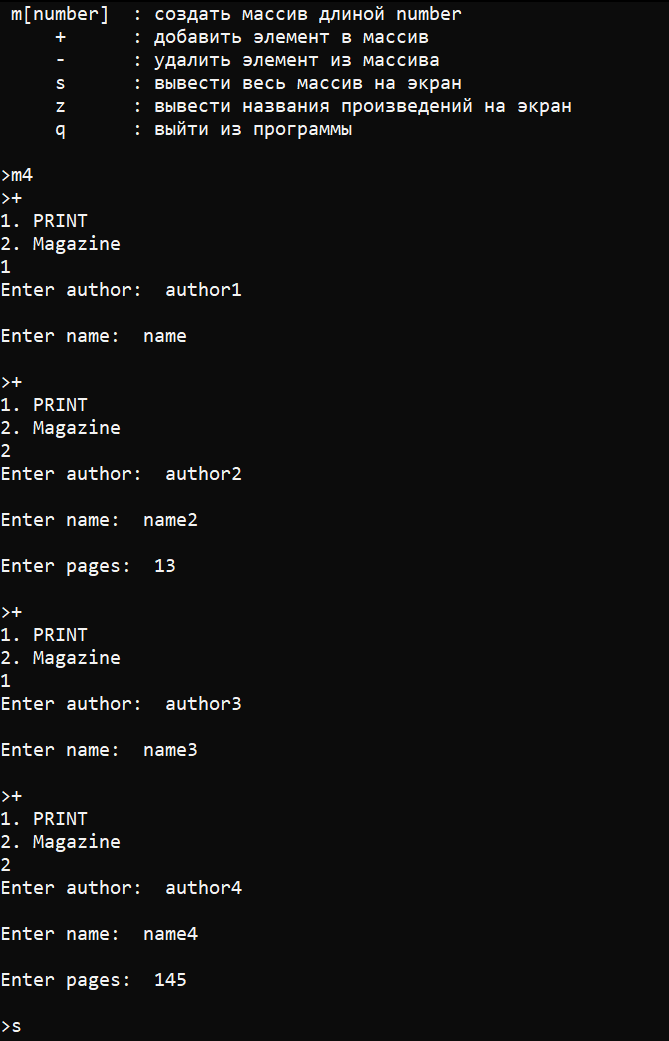
cout << "\nEnter pages: "; cin >> pages;

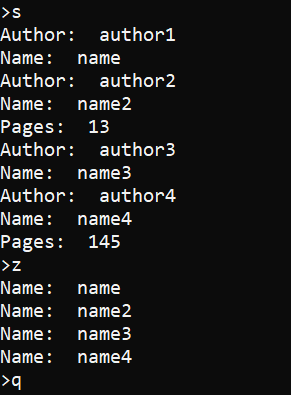
cout << endl;

}

**Скриншоты**

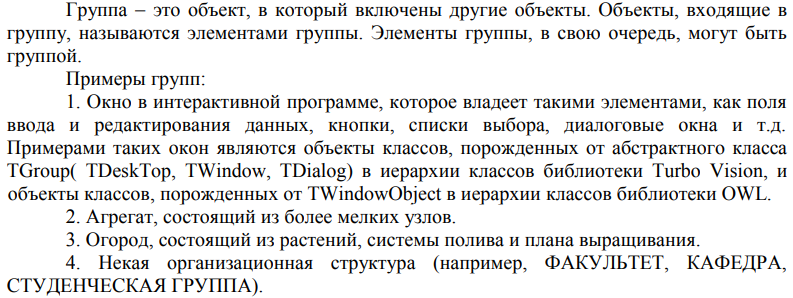




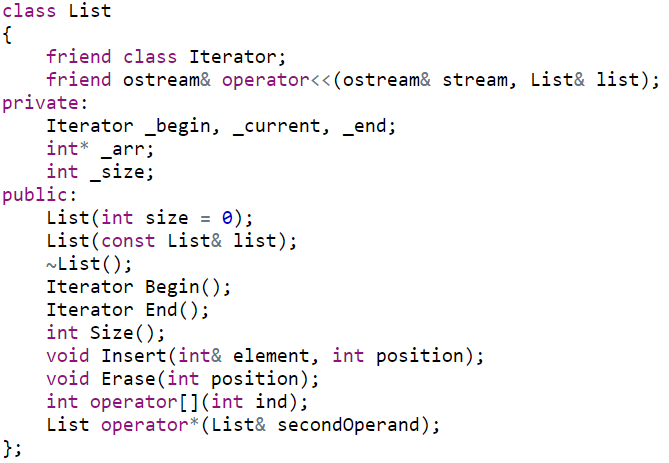


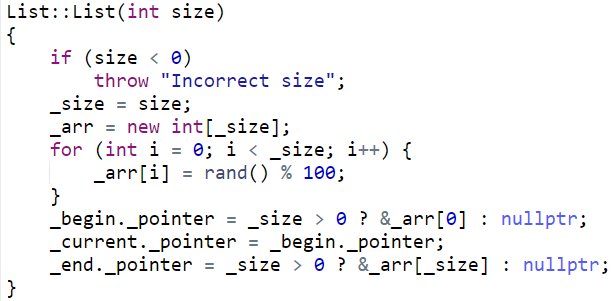
**Контрольные вопросы**

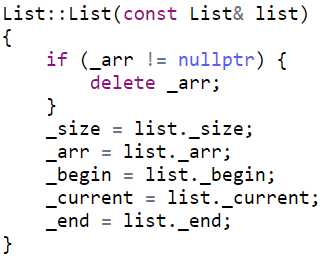


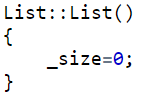
1. 



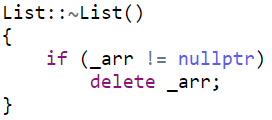
2. 

3. 

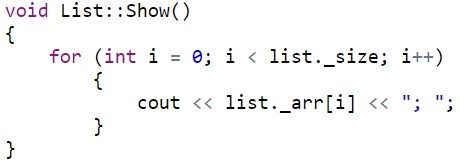




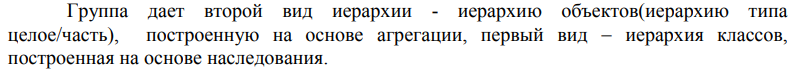


4. 



5. 



6. 

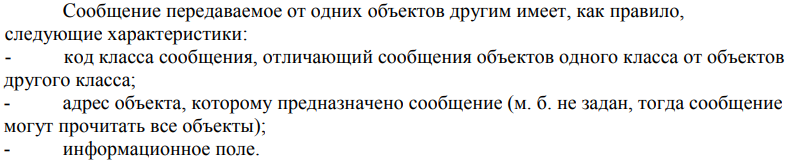


7. Для того, чтобы хранить ссылку на дочерний класс.

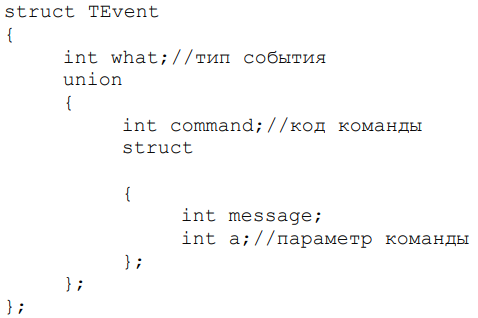


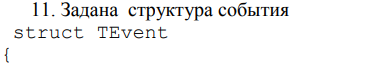
8. Событие – это пакеты информации, которыми обмениваются объекты и которые создаются объектно-ориентированной средой в ответ на те или иные действия пользователя.

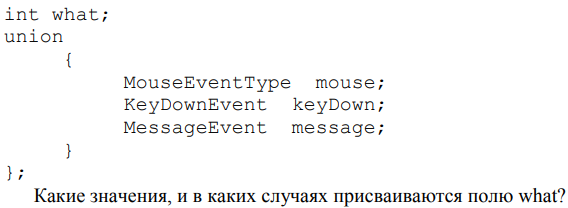


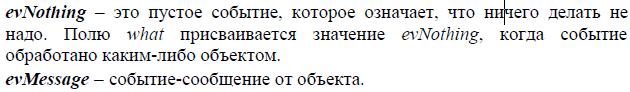
9. 

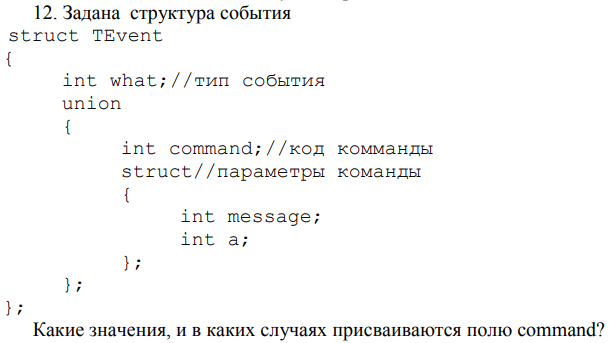


10. 

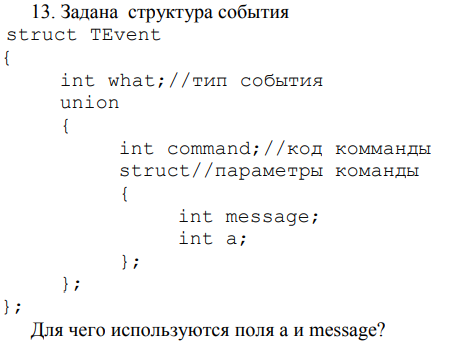




11. 

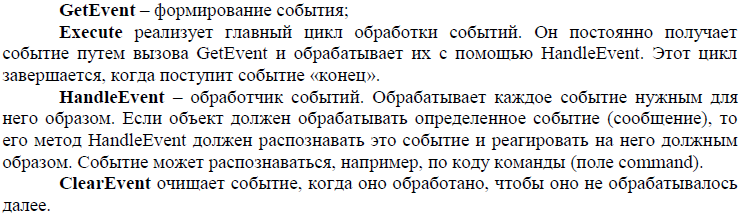


12. Код команды, которую необходимо выполнить при появлении некого события (через ветвление switch).

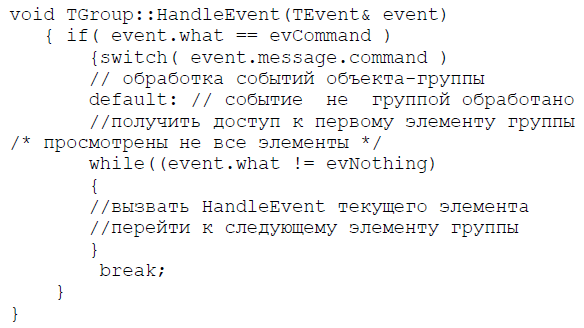


13. В поле a передаётся необходимый в дальнейшем параметр, а поле message содержит константу из переменной command.



14. 

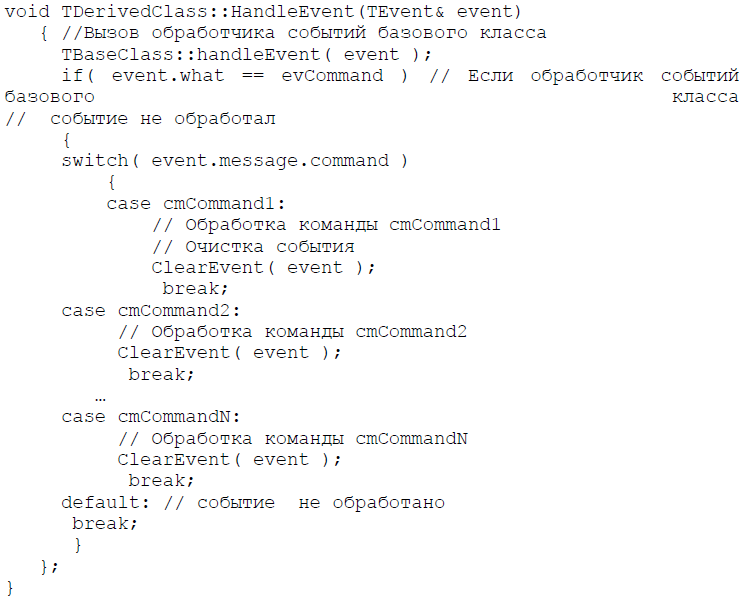


15. 



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.