Пермский национальный исследовательский политехнический университет

Кафедра «Информационные технологии и автоматизированные системы»

Лабораторная работа 8

**Программа, управляемая событиями**

Выполнил

Студент гр. РИС-24-3б Рунтов Иван Андреевич

Проверил

Доцент кафедры ИТАС Ольга Андреевна Полякова

г. Пермь, 2025

**Постановка задачи**

1. Определить иерархию пользовательских классов (см. лабораторную 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.

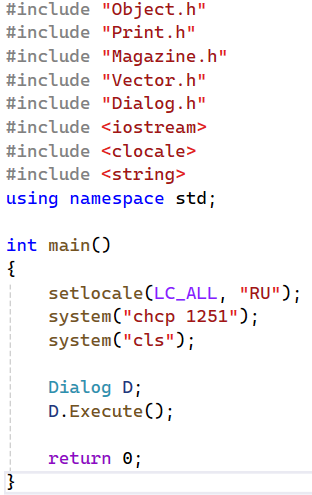
7. Написать тестирующую программу

8. Нарисовать диаграмму классов и диаграмму объектов

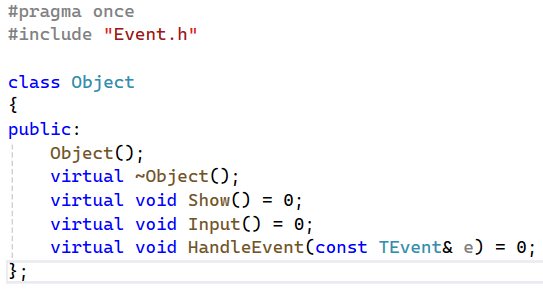
**Вариант 15**

Базовый класс: ПЕЧАТНОЕ\_ИЗДАНИЕ(PRINT) Название– string Автор – string Производный класс ЖУРНАЛ (MAGAZIN) Количество страниц - int Группа – Дерево (Tree). 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 о названиях всех элементов группы (формат команды : z) Конец работы (формат команды: q)

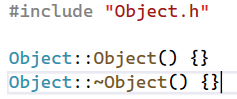
**Исполняемый файл main**



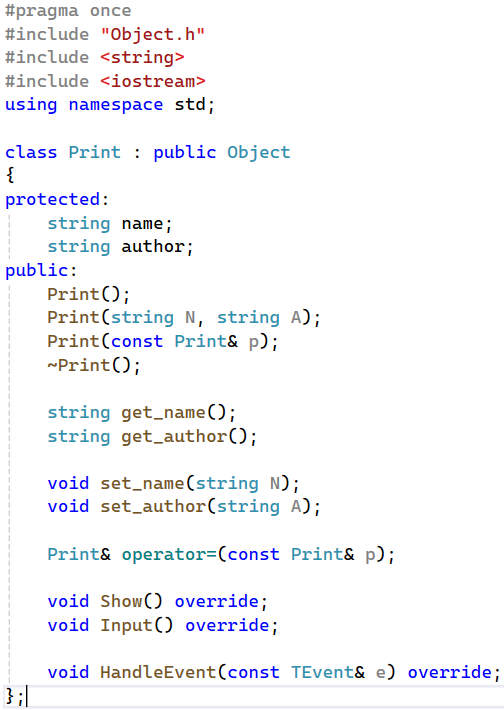
**Описание абстрактного класса Object**



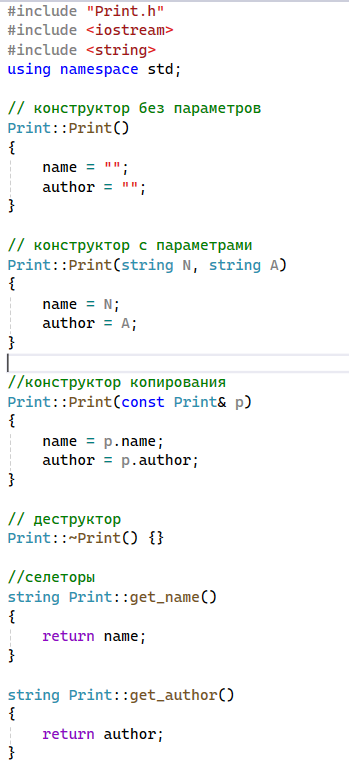
**Описание класса Object**

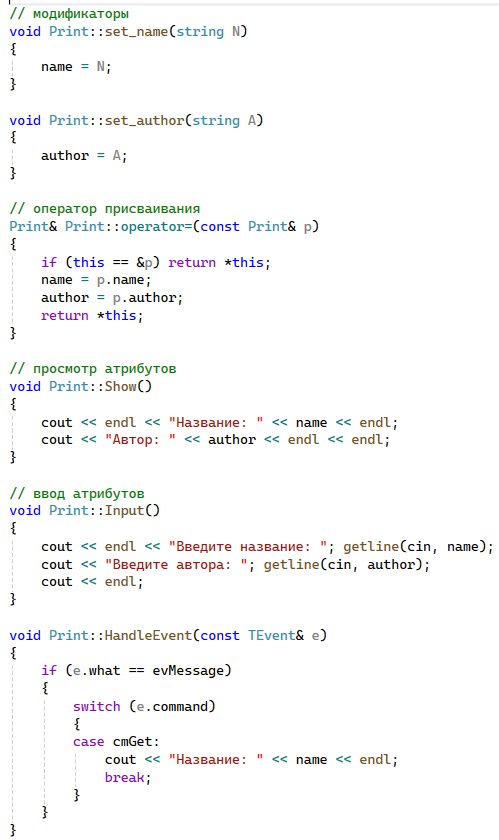


**Определение класса Print**

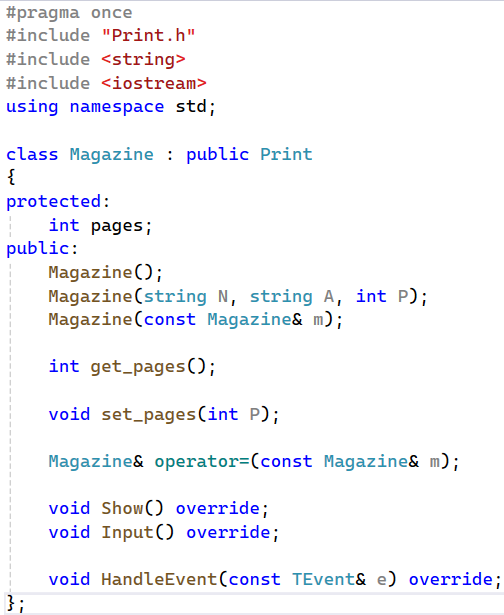


**Описание методов класса Print**

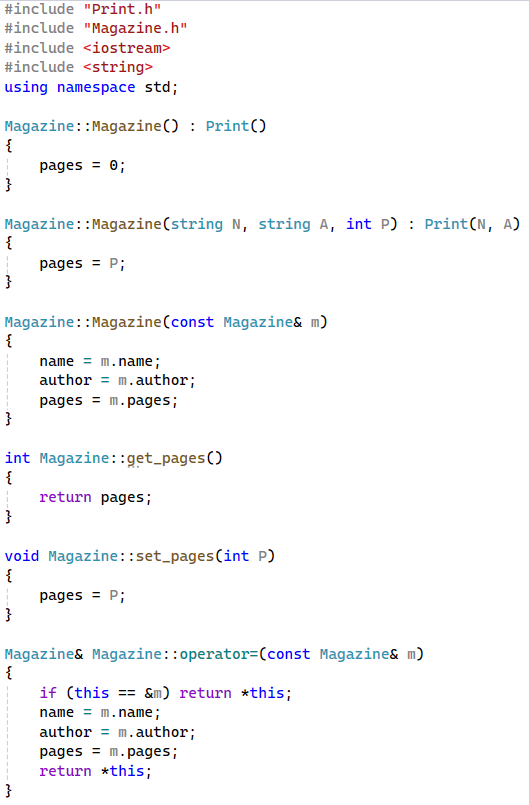


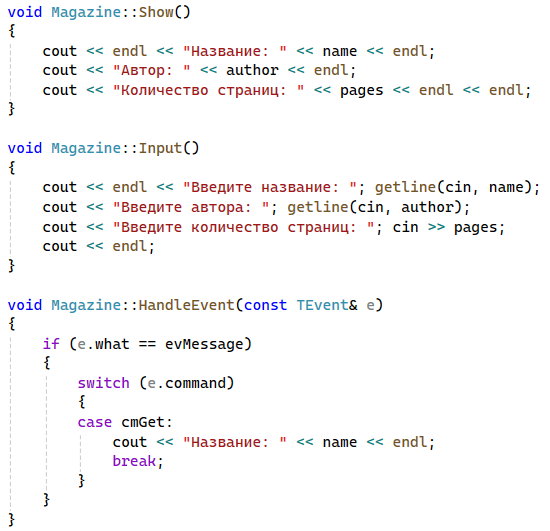


**Определение класса Magazine**

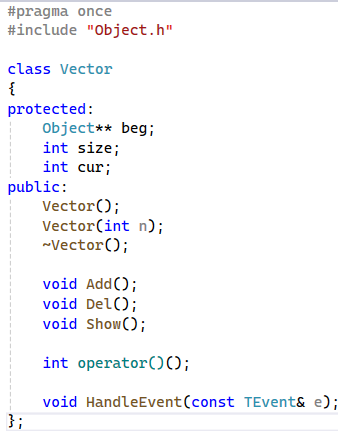


**Описание методов класса Magazine**

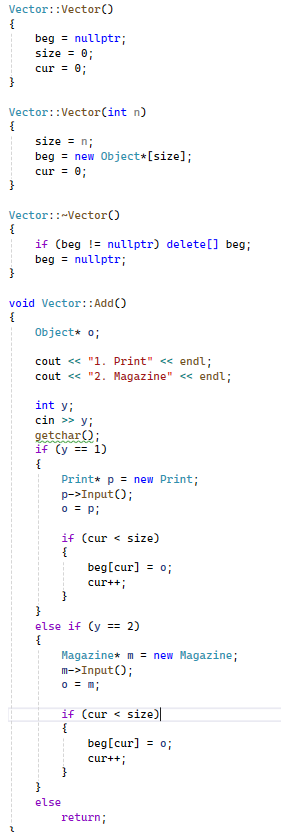


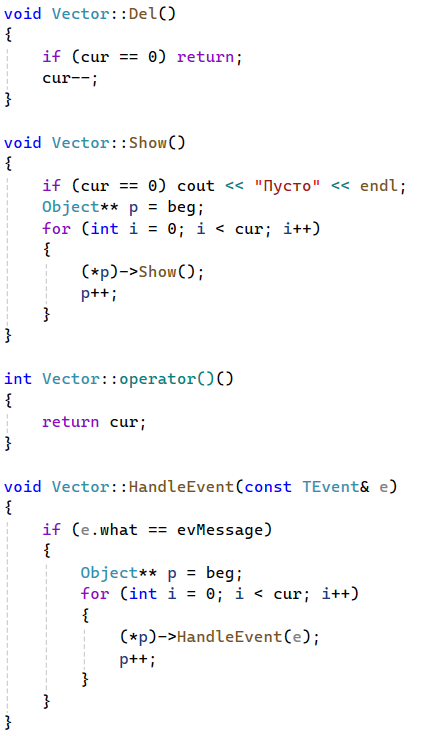


**Определение класса Vector**

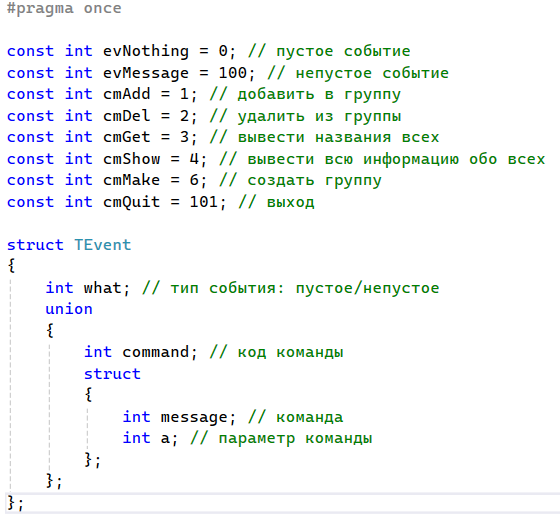


**Определение методов класса Vector**

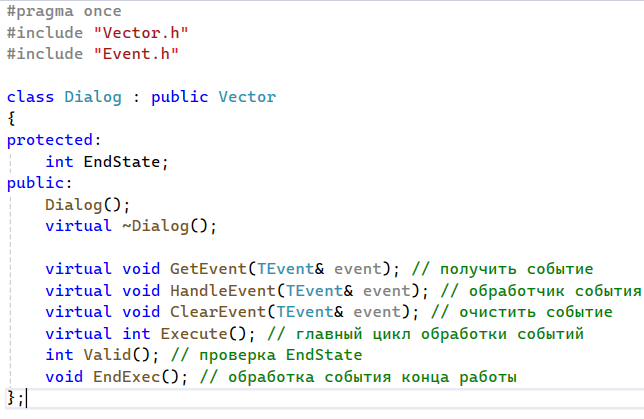




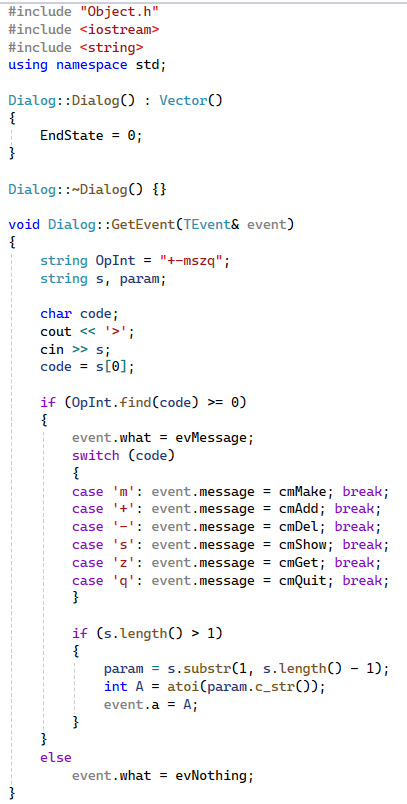
**Структура TEvent**

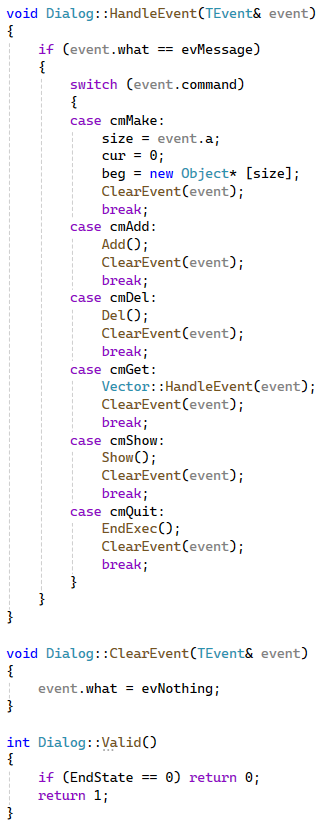


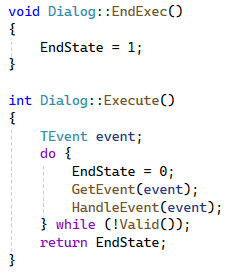
**Описание класса Dialog**



**Определение методов класса Dialog**







**Ответы на контрольные вопросы**



Группа – объект, в который заключены другие объекты.

Пример:

окно в интерактивной программе;

огород из растений, системы полива;

студенческая группа.



class List

{

private:

Node\* head;

Node\* tail;

int size;

};

struct Node

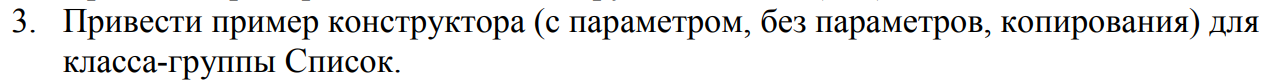
{

Node\* prev;

Node\* next;

Object\* data;

};



List()

{

head = nullptr;

tail = nullptr;

size = 0;

}

List(int n, Object\* k)

{

size = n;

Node\* node = new Node;

node->data = k;

head = tail = node;

for (int i = 1; i < size; i++)

{

Node node = new Node;

cin >> k;

node->data = k;

node->prev = tail;

tail->next = node;

tail = node;

}

}

List(const List& a)

{

Node\* tmp = a.head;

while (tmp != nullptr)

{

push\_back(tmp->data);

tmp = tmp->next;

}

}



~List()

{

while (size != 0)

{

Node\* tmp = head;

head = head->next;

delete tmp;

size--;

}

}



T& operator[](int index)

{

if (index >= 0 || index < size)

{

Node\* tmp = head;

for (int i = 0; i != index; i++)

tmp = tmp->next;

return tmp->data;

}

else

{

cout << "Неверный индекс!" << endl;

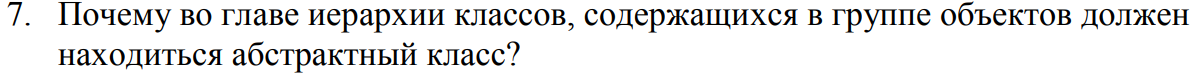
exit(1);

}

}



Группа дает вид иерархии - иерархию объектов(иерархию типа целое/часть), построенную на основе агрегации.



Это нужно для хранения классов потомков абстрактного класса, чтобы над объектами можно было выполнять одни действия.



Событие – информация, которой обмениваются объекты в ответ на одни или другие действия пользователя.



Сообщение передаваемое от одних объектов другим имеет, как правило, следующие характеристики:

- 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



struct Event

{

int what;

union

{

int message;

struct

{

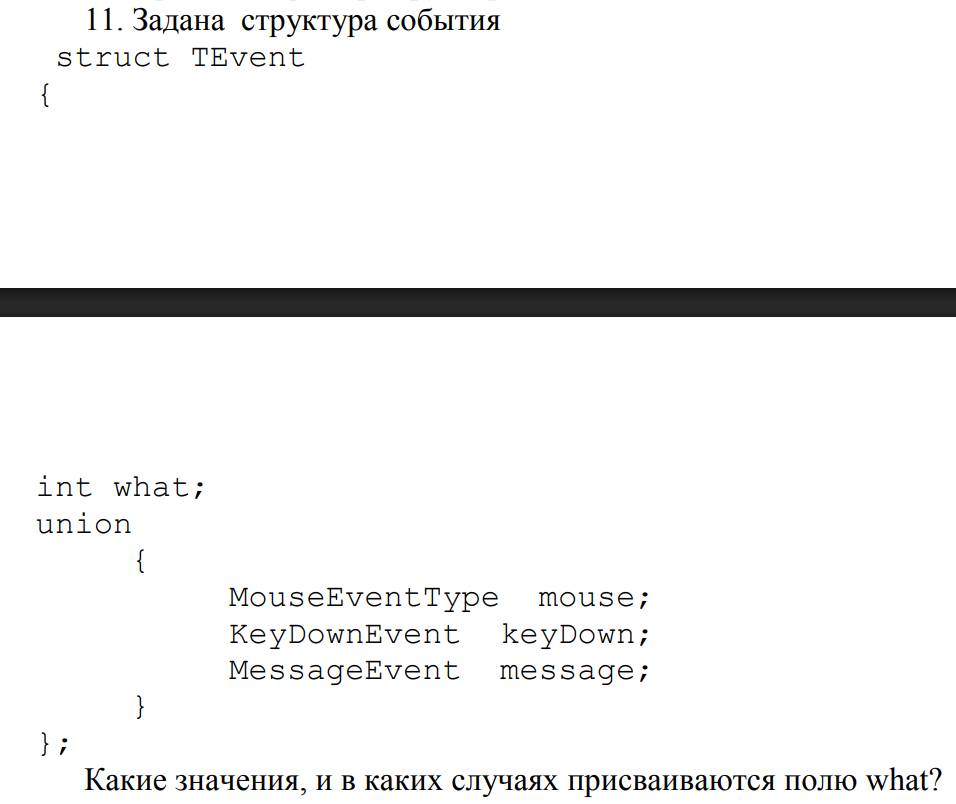
int command;

int a;

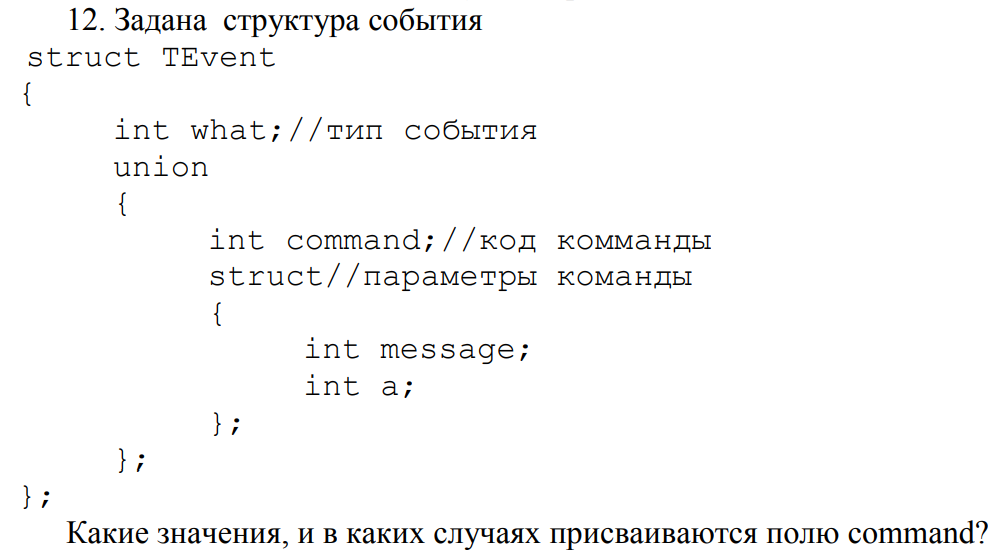
};

};

};



В случае действия мышью полю what присваивается evMouse – действие мышью, в случае нажатия стрелки вниз evKeyDown – нажатие стрелки вниз, в случае передачи сообщения от объекта evMessage – событие сообщение.



const int evNothing = 0; // пустое событие

const int evMessage = 100; // непустое событие

const int cmAdd = 1; // добавить в группу

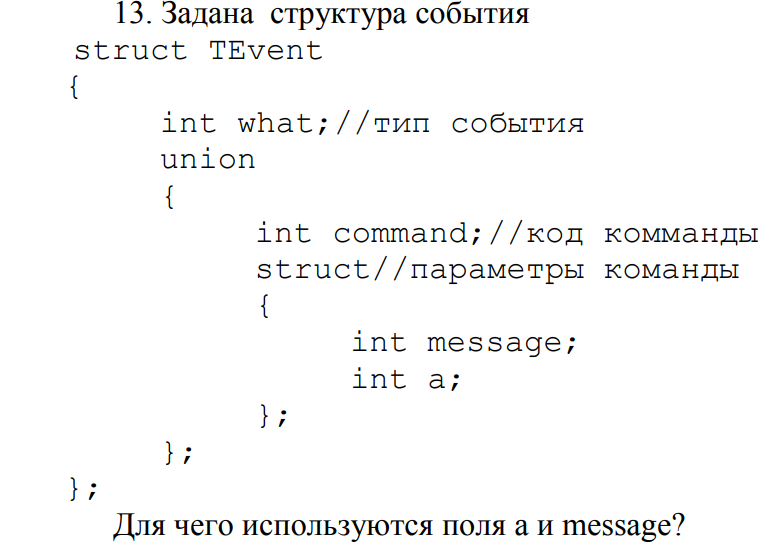
const int cmDel = 2; // удалить из группы

const int cmGet = 3; // вывести названия всех

const int cmShow = 4; // вывести всю информацию обо всех

const int cmMake = 6; // создать группу

const int cmQuit = 101; // выход



message; // команда

int a; // параметр команды



GetEvent – создание события

HandleEvent – обработчик события

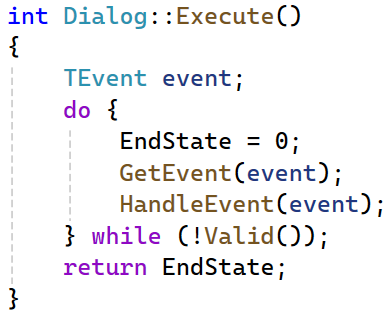
ClearEvent – очистка события

Valid – метод для проверки остановки программы

EndExec – метод для завершения

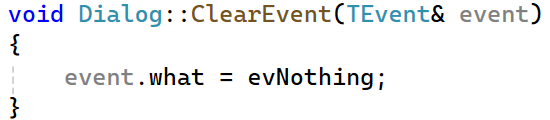
Execute – главный цикл получения и обработки сообщений





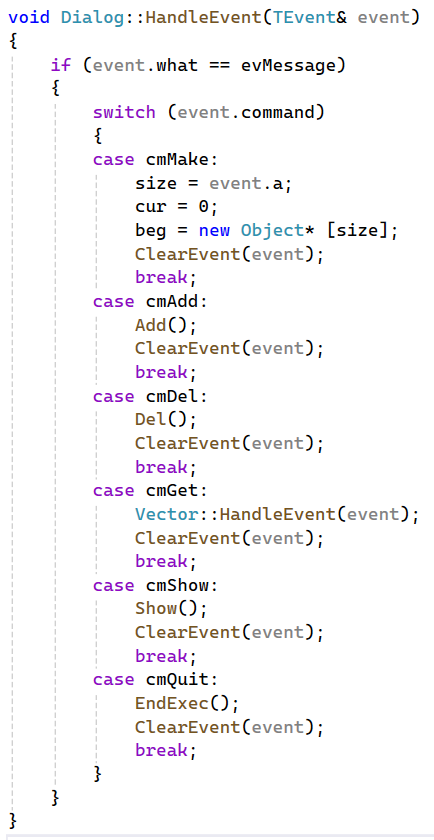


Очищает событие, заменяя тип сообщения на пустое



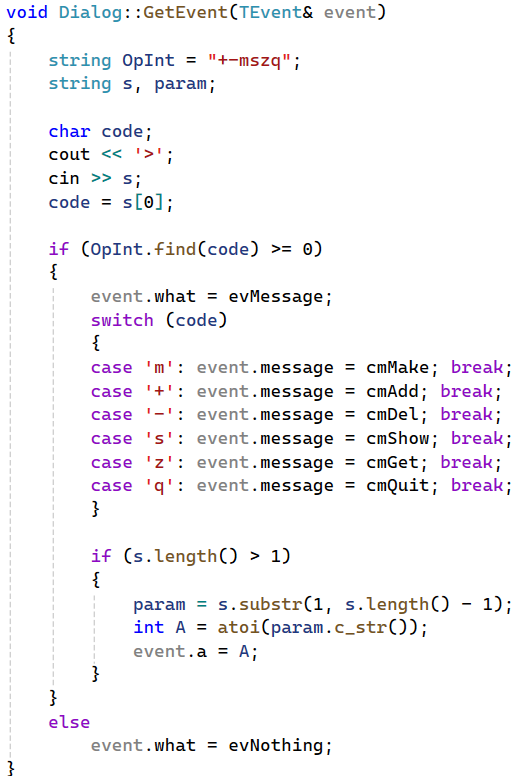


Обрабатывает событие по коду команды





Этот метод создает событие





Поле EndState содержится в классе Dialog и служит для прекращения главного цикла обработки.



Она используется для проверки EndState для установления конца работы главного цикла.

Ссылка на github: <https://github.com/RuntovIvan/Informatika>