**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Санкт-Петербургский колледж телекоммуникаций им. Э.Т. Кренкеля**

**Отделение**  *Информационных технологий*

**Цикловая комиссия** *Информатики и программирования в компьютерных системах*

**Отчет о выполнении лабораторной работы**

№ 14. Обработка событий. Виртуальные функции и абстрактные

классы

Вариант 2

Выполнили: Обучающиеся 3 курса 500 группы,

Воронов Никита Павлович,

Паршиков Сергей Витальевич.

Проверил: преподаватель Баталов Д.И.

Санкт-Петербург

2021

1. **Цель**

Получить навык создания классов с использованием принципа наследования и

реализации полиморфного поведения классов.

1. **Заданий**

**Сылка на гит: https://github.com/Cyberslayer30000/SPbCT\_ParshikovSV**

Написать программу с оконным интерфейсом, которая выполняла бы задания в

зависимости от номера варианта. Оконный интерфейс этой программы должен содержать

обязательно кнопку, при нажатии на которую следует начинать вычисления. Элементы

обрабатываемых последовательностей можно выводить как в само окно, так и в другие

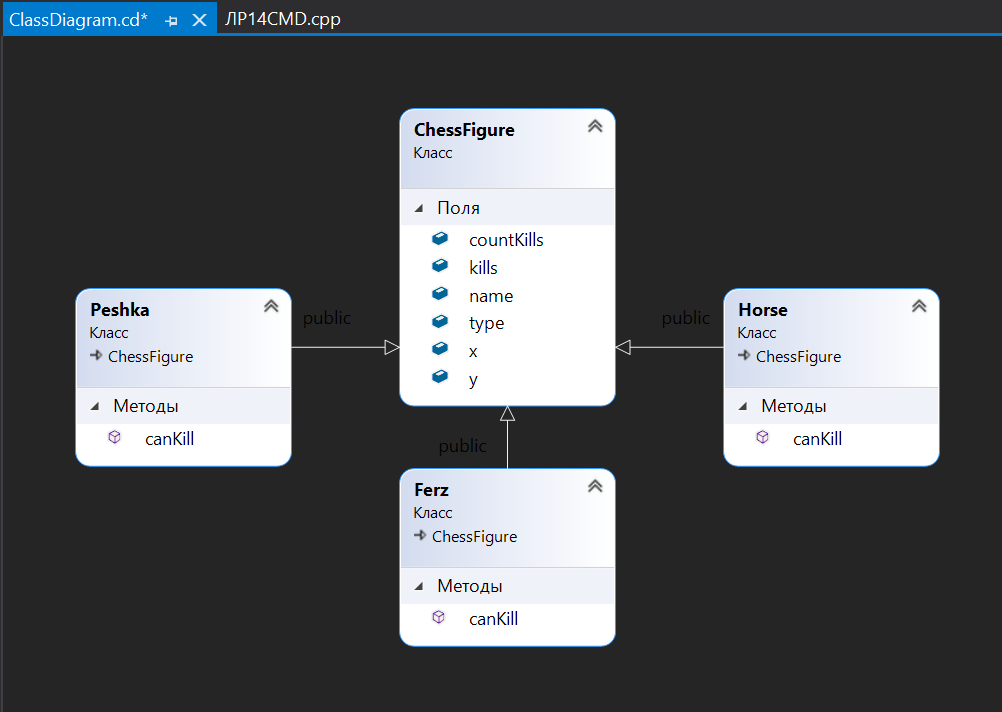
элементы управления, например строку редактирования и т.п.

Определить класс «Шахматная фигура» и унаследовать от него классы,

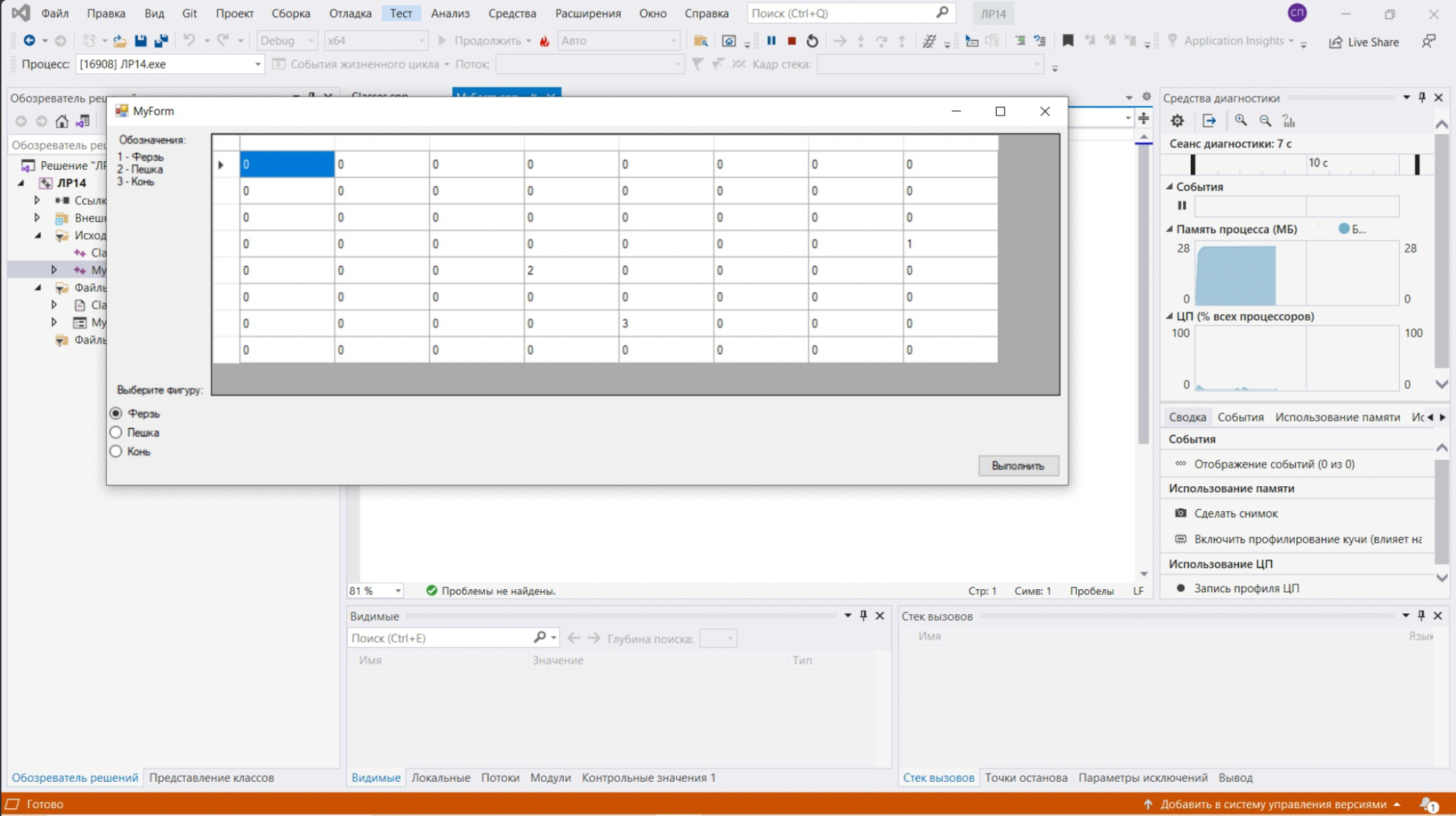
соответствующие шахматным фигурам «Ферзь», «Пешка», «Конь». Задать позицию и

определить список фигур, которые может убить некоторая выбранная.

**Диаграмма классов UML.**



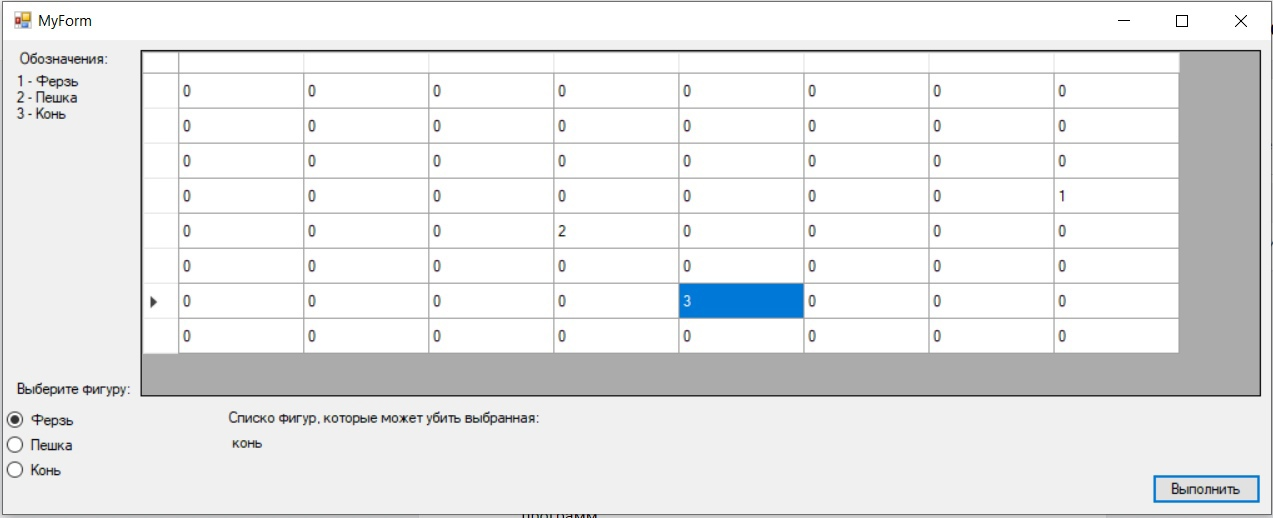
**Форма**

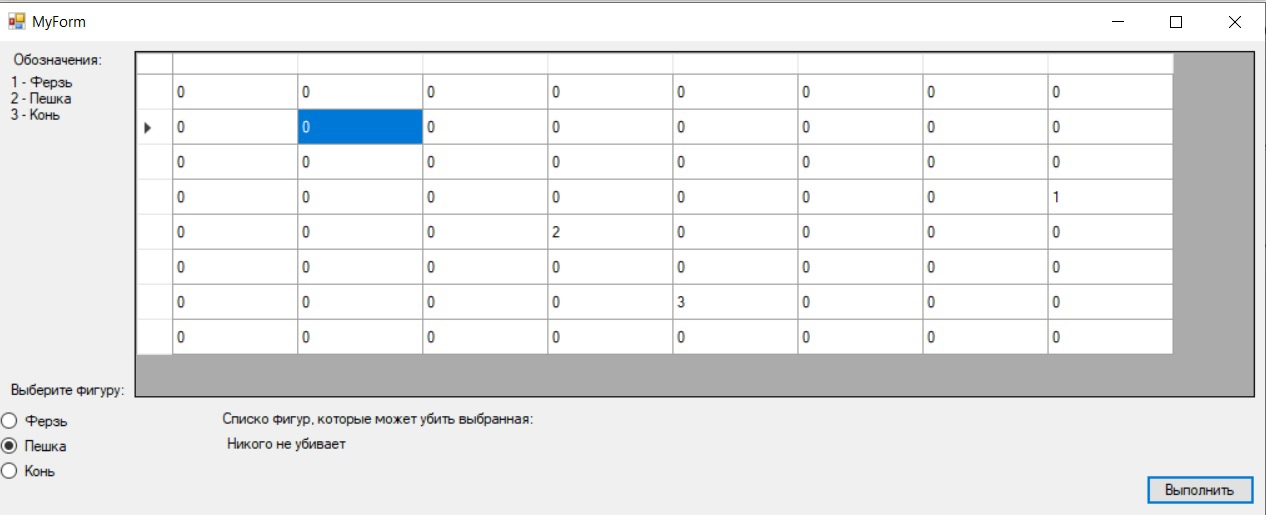


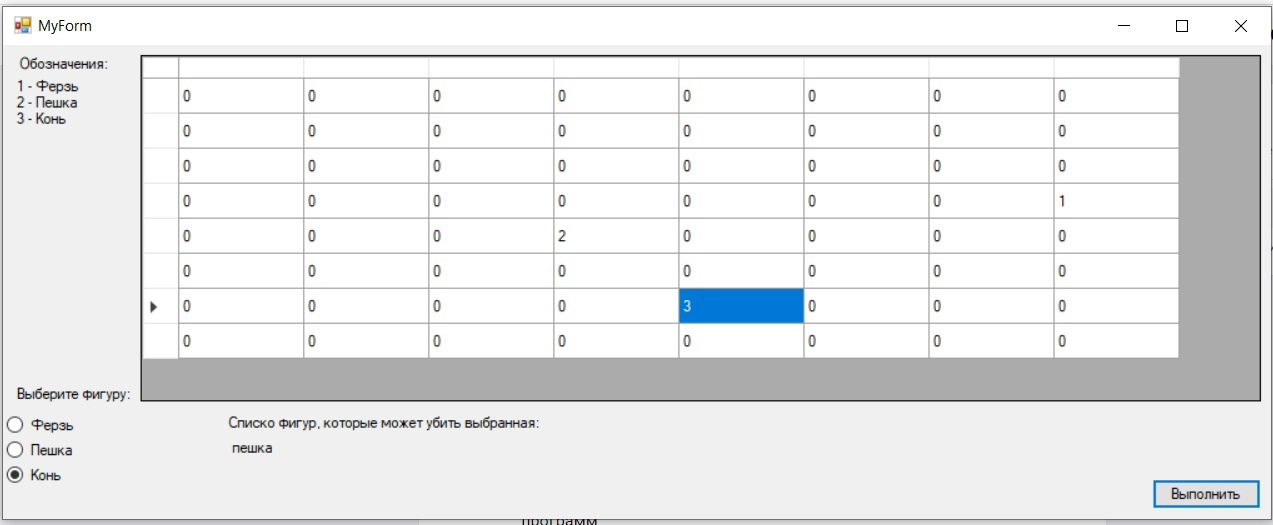
**Код**



**Результат работы программы.**







1. **Ответы на контрольные вопросы**

**1. В чем заключается механизм наследования? Какие преимущества он предоставляет в ООП?**

Наследование отражает отношения между классами «это есть». Примеры наследования: автомобиль есть транспортное средство, клиент банка есть человек, прямоугольник есть геометрическая фигура. Таким образом, при наследовании базовый и производный классы выступают как, соответственно, обобщение и конкретизация некоторого объекта реального мира. В случае наследования новый класс в буквальном смысле создаётся на основе ранее объявленного класса, наследует, а возможно и модифицирует его данные и функции. Объявленный класс может служить основой (базовым классом) для новых производных классов. Производные классы наследуют данные и функции своих базовых классов и добавляют собственные компоненты.

Построение систем классов с использованием механизма наследования позволяет точнее описать в программе предметную область поставленной задачи, быстрее модифицировать код программы при необходимости, ускорить процесс проектирования и программирования.

**2. Как реализуется механизм наследования в программах на объектно-ориентированном языке программирования?**

*class имя\_класса: список\_базовых\_классов*

{

//определение собственных компонент

//переопределение унаследованных компонент базовых классов

};

Здесь *список\_базовых\_классов* – это перечень (через запятую) тех классов, от

которых будет унаследован определяемый класс. Эти классы к моменту определения

производного класса должны быть определены. После подобного объявления все

общедоступные и защищенные компоненты базовых классов становятся компонентами

производного класса без дополнительного определения. В некоторых источниках базовый

класс называют суперклассом, а производный - подчиненным классом.

Общие правила порождения классов:

1) количество базовых классов в списке порождения может быть любым;

2) один и тот же класс не может быть задан в списке порождения дважды;

3) базовый класс к моменту определения производного должен быть определен или

описан;

4) ни базовый, ни порожденный класс не могут быть определены с помощью

ключевого слова union;

**3. От чего зависит доступность унаследованного члена класса в производном классе в языке С++?**

1) области видимости компонента в базовом классе;

2) способа определения производного класса (через class или struct);

3) спецификации доступа, указанной в списке базовых классов при объявлении

наследования.

Пояснения требует третий пункт: при объявлении списка наследования для каждого базового класса можно указать спецификатор доступа в виде уже известных нам ключевых слов public, protected, private. В этом случае список наследования может выглядеть, например, следующим образом:

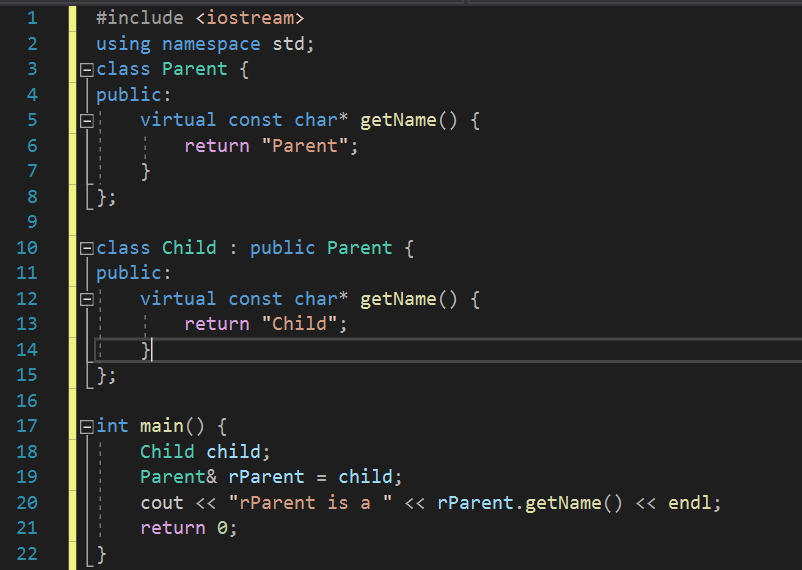
class A: public B, private C, protected D

{…};

**4. Что такое виртуальная функция? Приведите свой пример эффективного**

**полиморфного поведения функций.**

Виртуальная функция — это функция, объявленная с ключевым словом virtual в базовом классе и переопределенная в одном или в нескольких производных классах. Это особый тип функции, которая, при её вызове, выполняет «наиболее» дочерний метод, который существует между родительским и дочерними классами. Это свойство еще известно, как полиморфизм. Дочерний метод вызывается тогда, когда совпадает сигнатура (имя, типы параметров и является ли метод константным) и тип возврата дочернего метода с сигнатурой и типом возврата метода родительского класса.



Результат: rParent is a Child

Поскольку rParent является ссылкой на родительскую часть объекта child, то, обычно, при обработке rParent.getName() вызывался бы Parent::getName(). Тем не менее, поскольку Parent::getName() является виртуальной функцией, то компилятор понимает, что нужно посмотреть, есть ли переопределения этого метода в дочерних классах. И компилятор находит Child::getName()!

**5. Почему для классов с объявленными виртуальными методами в языке C++**

**необходимо объявлять виртуальным и деструктор?**

Только так обеспечивается корректное разрушение объекта производного класса через указатель на соответствующий базовый класс.

**6. Что такое абстрактный класс С++? Какие ограничения существуют при его**

**использовании? Для чего подобные классы определяются в программе?**

Абстрактные классы - это классы, которые содержат или наследуют без переопределения хотя бы одну чистую виртуальную функцию. Абстрактный класс определяет интерфейс для переопределения производными классами.

Абстрактные классы невозможно использовать для следующих элементов:

* переменных и данных членов;
* типов аргументов;
* типов возвращаемых функциями значений;
* типов явных преобразований

Другое ограничение состоит в том, что при вызове конструктором абстрактного класса чистой виртуальной функции (прямо или косвенно) результат будет неопределенным. Однако конструкторы и деструкторы абстрактных классов могут вызывать другие функции-члены.

Для абстрактных классов можно определять чистые виртуальные функции, но вызывать их можно только непосредственно с использованием следующего синтаксиса:

abstract — имя класса::Function-Name()

Абстрактные классы используются в качестве обобщенных концепций, на основе которых можно создавать более конкретные производные классы. Невозможно создать объект типа абстрактного класса; однако можно использовать указатели и ссылки на типы абстрактного класса.

**7. Чем реализация механизма наследования в языке С# отличается от реализации в С++?**

Синтаксически механизм реализации наследования в С# во многом схож с языком С++. Унаследовать класс от уже существующего можно с помощью объявления вида:

class имя\_класса: базовый\_класс

{

//определение собственных компонент

};

При этом в списке базовых классов может быть указан лишь один класс, множественное наследование реализаций классов в C# запрещено. Этим разработчики языка радикально решили вопрос разрешения имен для одноименных компонент при наследовании.

**8. Что такое интерфейс С#? Как он определяется и используется в программе?**

Интерфейс – это декларация той функциональности, которую должен обеспечивать

реализующий интерфейс класс. Часто говорят, что интерфейс является контрактом, который должен исполнять класс, объявивший себя наследником класса и тем самым вызвавшийся этот контракт исполнять. Определяется интерфейс с помощью ключевого слова interface и содержит лишь набор абстрактных членов:

interface имя

{

тип имя\_метода1 (список\_параметров);

…

тип имя\_свойства{set; get;}

…

}

Как видно из примера, в определение интерфейса помещаются лишь прототипы методов и свойств. Тела методов и аксессоров свойств не могут быть определены в интерфейсе, они задаются лишь в классах, наследующих интерфейс.

В теле интерфейса не указываются модификаторы доступа к членам, нельзя

использовать для методов интерфейса ключевые слова virtual и static.

**9. Почему нельзя создавать экземпляры интерфейсов в программах на языке С#?**

Понятие "экземпляр интерфейса (instance of an interface)" как таковое не корректно! Интерфейс не может содержать в себе реализацию, не является объектом и служит для того что бы задавать поведение объекту (классу).

**10. Как концепцию интерфейса можно реализовать в языке С++, в котором не**

**определен подобный тип данных?**

C++ не имеет встроенных концепций интерфейсов. Можно реализовать это, используя абстрактные классы, которые содержат только чисто виртуальные функции. Поскольку он допускает множественное наследование, можно наследовать этот класс, чтобы создать другой класс, который затем будет содержать в себе этот интерфейс (объектный интерфейс).