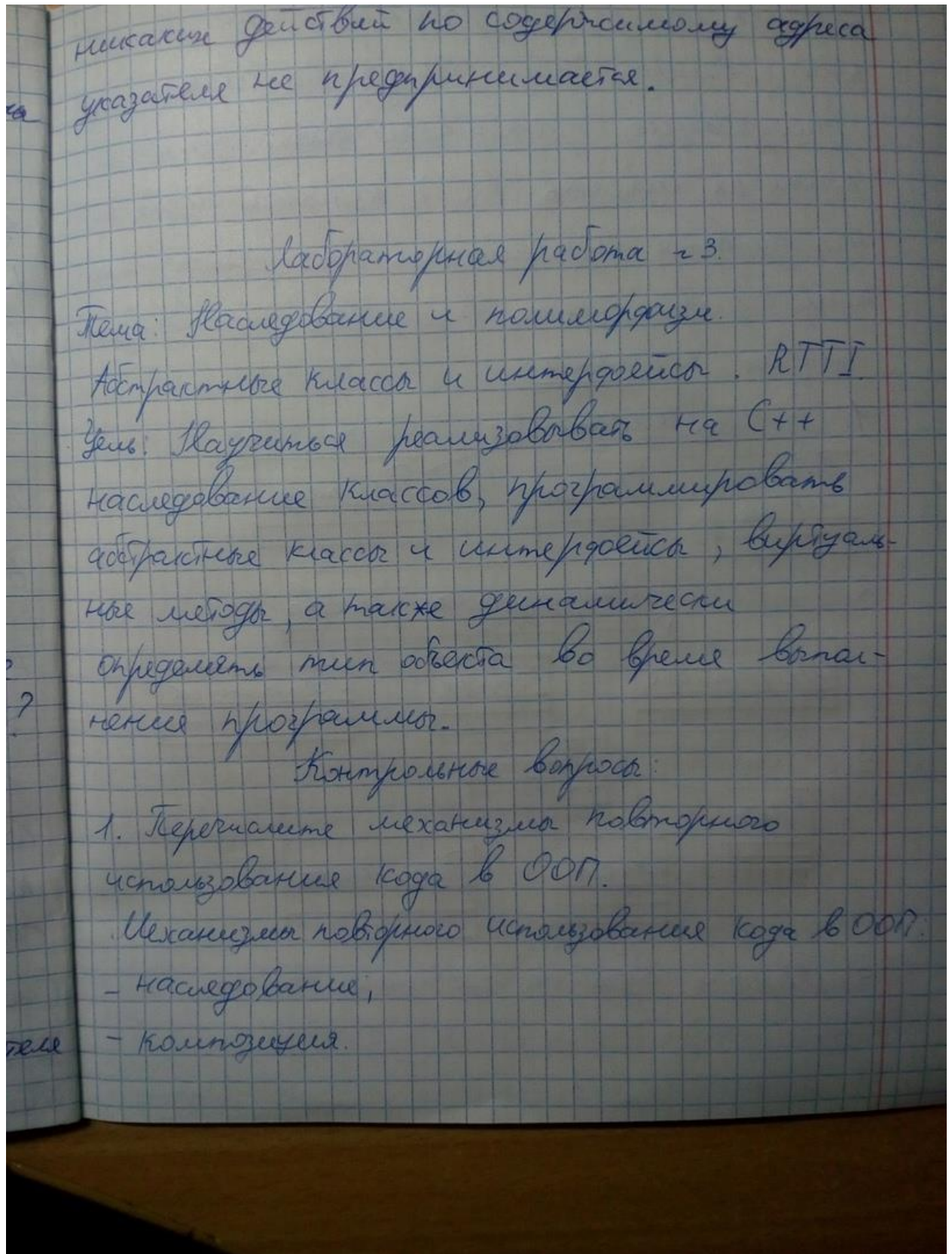
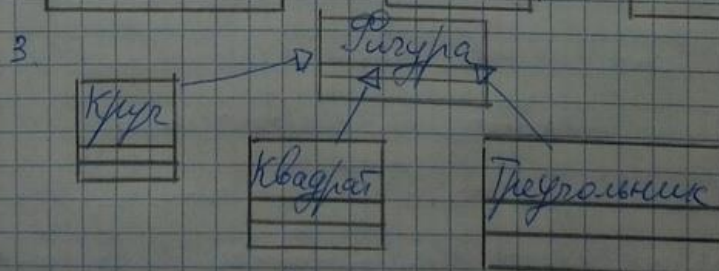
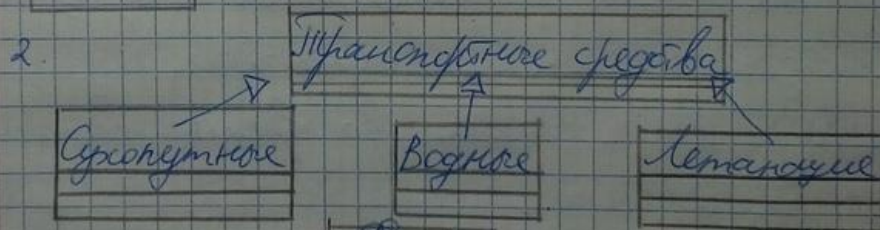


Контрольные вопросы:



2. Что такое наследование? Приведите три примера осеи наследования классов. Наследование представляет собой один из ключевых аспектов ООП, который позволяет наследовать функциональность одного класса или базового класса в другом - производном классе. Примеры:



3. Типы и виды наследования в C++.

Для спецификации доступа в C++ есть три уровня: private, protected и public. Аналогично. Типы наследования:

- private-наследование - данные protected и public базового класса доступны из методов производного класса, но недоступны извне, но есть они становятся private;

- protected-наследование - данные, ~~которые~~ protected и public базового класса становятся protected;

- public-наследование - публичное наследование наиболее употребительно. Типы from protected и public данные из базового класса остаются, соответственно protected и public в производимом классе.

	Исходный	наследующий	доступа
public-наследование	public	private	protected
	public	private	protected
private-наследование	private	private	private
protected-наследование	protected	private	protected

Виды наследования:

- одиночное (простое) наследование - порожденный класс наследует элементы одного базового класса;
- множественное наследование - порожденный класс наследует элементы более, чем от одного базового класса.

4. В чем заключается разница между ранним и поздним связыванием?

Раннее связывание означает, что компилятор может напрямую связать имя идентификатора (например, имя функции или переменной) с машинным адресом.

Позднее связывание - в некоторых программах невозможно знать наперед, какая функция будет вызываться первой. В языке C++ для выполнения позднего связывания используются указатели на функции.

Указатель на функцию - тип указателя, который указывает на функцию вместо переменной.

5. Что такое полиморфизм? Виртуальные методы

Полиморфизм - свойство, которое позволяет использовать одно и то же имя функции для решения двух и более схожих, но технически разных задач. Полиморфизм - возможное замещение методов объекта родителем

методами объекта - потомка, имеющих то же имя

Чтобы использовать полиморфизм, необходимо задать:

1) все классы - потомки являются наследниками одного и того же базового класса.

2) функции, реализующие метод, должны быть объявлены виртуальными в базовом классе.

Виртуальные функции - специальный вид

функций - членов класса. Виртуальная

функция отличается от обычной функцией тем,

что для обычной функции связывание

базовая функция с ее определением осу-

ществляется на этапе компиляции. Для

виртуальных функций это происходит во время

выполнение программы.

Для объявления виртуальной функции используется ключевое слово `virtual`.

Функция-член класса может быть объявлена как виртуальная, если

- класс, содержащий виртуальную функцию, базовый в иерархии порождения;
- реализация функции зависит от класса и будет различной в каждом порожденном классе.

Виртуальная функция - это функция, которая определяется в базовом классе, а любой порожденный класс может ее переопределить.

Виртуальная функция вызывается только через указатель или ссылку на базовый класс.

6. Что такое абстрактный класс? Чисто виртуальные методы.

Абстрактные классы - это классы, которые содержат или наследуют без переопределения хотя бы одну чисто виртуальную функцию.

Абстрактный класс определяет интерфейс для
предопределения производными классами.

В чисто виртуальное функции - функции,
которые не имеют определения. Чтобы

определить виртуальную функцию как
пустую, ее объявление завершается значением
"=0". Например:

```
class Figure
```

```
{  
public:
```

```
    virtual double getSquare()=0;
```

```
    virtual double getPerimeter()=0;
```

```
    virtual void ShowFigureType()=0;
```

```
};
```

7. Что из себя представляет класс-интерфейс?
Для чего он используется?

Интерфейс - это класс, который не имеет
переменных - данных и все методы которого
являются чисто виртуальными функциями.
Использование интерфейсов - один из вариантов

обеспечение полиморфизма в объектных
языках и средах. Все классы, реализующие
один и тот же интерфейс, с точки зрения
определенного типа поведения, ведут
себе внешне одинаково. Это позволяет писать
обобщенные алгоритмы обработки данных,
используя в качестве типов параметров
интерфейсы, и применять их к объектам
различных типов, всякий раз получая
предусмотренный результат. Интерфейсы позволяют
находить множественное наследование абстракций.

8. Порядок создания и удаления подклассов.

Зачем нужен виртуальный деструктор?

Деструкторы вызываются в том же
самом порядке, в каком классы создаются
один за другим в иерархии классов.

В противоположность этому деструктор
производного класса вызывается перед
деструктором базового класса, т.е. подкласс
создается, будет удален в обратном порядке.

Если в классе присутствует хотя бы одна виртуальная функция, деструктор также следует сделать виртуальным.

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

9. Что означает RTTI? Какие есть возможности RTTI в C++?

Динамическая идентификация типа данных (RTTI) - механизм, который позволяет определить тип данных переменной или объекта во время выполнения программы.

Для этого используется функция `typeid`.

Для использования этой функции необходимо включить заголовочный файл `typeid.h`.

Зачем `typeid`: `typeid(объект)`,

где объект является объектом, тип которого требуется определить. Функция `typeid`

возвращает ссылку на объект типа `typeid_t`, который описывает тип объекта.

Преобразование типов в C++:

- `static_cast<>` - для приведений, которое проверяется только во время компиляции; возвращает ошибку, если компилятор обнаруживает, что вы пытаетесь выполнить приведение типов, которое полностью невозможно.

- `dynamic_cast` $\langle \rightarrow \rangle$ - для безопасного выполнения приведения указателя на Base к типу, который проверяет среда. Оператор `dynamic_cast` является более безопасным `static_cast`, чем для типов, но проверка среды выполнения требует некоторых дополнительных издержек;

- `const_cast` $\langle \rangle$ - для приведения параметра `const` - значения хранилища переменной или `const` для преобразования неперемещивой значения `const`;

- `reinterpret_cast` $\langle \rangle$ - для приведения к типу неизвестным типам, таким как `fun` указание и `int`.