Перечень теоретических вопросов по предмету «**Объектно-ориентированное программирование на языке С++**»

**Введение в объектно-ориентированное программирование на С++**

1. Три принципа объектно-ориентированного программирования:

• инкапсуляция. Определение, примеры использования в повседневной среде;

• полиморфизм. Определение, примеры использования в повседневной среде;

• наследование. Определение, примеры использования в повседневной среде;

2. Класс и объект:

• понятие класса;

• синтаксис объявления;

• спецификаторы доступа:

◉*◉* public;

◉*◉* private;

◉*◉* protected.

3. Переменные-члены класса и методы-члены:

• реализация тела метода внутри класса;

• вынос тела метода за класс.

5. Практические примеры работы с классами:

• использование спецификаторов доступа;

• реализация практических примеров (студент, прямоугольник, точка,

машина, и так далее).

4. Встроенные (inline) методы:

• необходимость использования;

• примеры объявления и использования.;

• ограничения при использовании inline методов.

5. Сравнительный анализ структур и классов.

6. Конструктор:

• проблемы, возникающие при использовании неинициализированных

переменных.;

• понятие конструктора;

• синтаксис объявления;

• конструктор по умолчанию;

• конструктор, принимающий параметры;

• перегруженные конструкторы;

• примеры использования (например, классы студент, точка, машина,

и так далее).

7. Деструктор:

• утечки ресурсов. причины их возникновения и плачевные последствия

данного явления;

• понятие деструктора;

• синтаксис объявления;

• примеры использования (например, классы студент, массив, строка,

и так далее).

8. Указатели на объекты.

9. Массивы объектов.

10. Инициализаторы:

• списочная инициализация;

• примеры практического использования (инициализация поля класса,

константы члена класса, инициализация внутреннего объекта).

11. Инициализация членов класса и делегирование конструкторов.

12. Статические переменные-члены и статические функции-члены класса:

• необходимость использования статических членов (показать на практическом примере, например подсчет количества объектов, и так далее);

• синтаксис объявления статических переменных-членов класса;

• синтаксис объявления статических функций-членов класса;

• отличие статистических функций-членов класса от функций-членов класса.

**Указатель this и конструктор копирования**

1. Указатель this:

• понятие указателя this;

• практические примеры использования указателя this.

2. Конструктор копирования:

• понятие побитового копирования;

• проблемы, связанные с побитовым копированием;

• проблемные ситуации, требующие конструктора копирования (передача по значению объекта, возврат объекта по значению, создание объекта в форме присваивания другого объекта);

• синтаксис конструктора копирования;

• примеры использования конструктора копирования (классы вектор, строка, матрица и так далее);

• обсуждение тонкостей конструктора копирования:

◉*◉* спецификатор const;

◉*◉* необходимость передачи по ссылке.

**Константные методы, explicit конструктора**

1. Константный метод:

• синтаксис объявления;

• особенности указателя this в константном методе;

• примеры использования.

2. Объявление конструктора с использованием ключевого слова explicit:

• примеры ситуации, иллюстрирующие неявное создание объекта;

• ключевое слово explicit и его использование;

• объявление конструктора с использованием ключевого слова explicit.

**Перегрузка операторов**

1. Необходимость использования перегрузки операторов:

• примеры кода (реализация классов, например, таких как дробь, матрица, через обычные методы члены типа sum, mult и так далее);

• логичность использования стандартных символов (+, - , >,< и так далее).

2. Перегрузка операторов:

• общие понятия перегрузки операторов:

◉*◉* классификация операторов на основании количества операндов (бинарные, унарные, триадный);

◉*◉* определение перегрузки операторов;

◉*◉* различные виды перегрузки (метод-член, функция-друг, глобальная функция);

• синтаксис перегрузки операторов методом – членом (унарный, бинарный вид);

• примеры перегрузки операторов:

◉*◉* перегрузка арифметических операторов:

▶*▶* перегрузка операторов +,-, \* и так далее;

▶*▶* перегрузка инкремента и декремента;

ӽ*ӽ* цели и задачи перегрузки инкремента и декремента;

ӽ*ӽ* синтаксис перегрузки;

ӽ*ӽ* отличия перегрузки постфиксной и префиксной формы;

◉*◉* перегрузка логических операторов;

◉*◉* возврат по ссылке;

◉*◉* перегрузка оператора присваивания;

3. Конструктор переноса:

• что такое конструктор переноса;

• цели и задачи конструктора переноса;

• примеры реализации.

4. Специальные перегрузки:

• перегрузка [];

• перегрузка ();

• перегрузка оператора преобразования типов;

• использование explicit для преобразований, определяемых классом.

5. Список операторов, которые невозможно перегрузить.

6. Перегрузка операторов дружественными функциями:

• дружественные функции:

◉*◉* понятие дружественной функции;

◉*◉* цели и задачи дружественных функций;

◉*◉* ключевое слово friend;

◉*◉* отличия дружественных функций от методов-класса;

◉*◉* примеры использования дружественных функций;

◉*◉* перегрузка операторов с использованием дружественных функций;

◉*◉* список операторов, которые невозможно перегрузить не методами-

членами классов;

• перегрузка ввода-вывода:

◉*◉* потоковые классы ostream и istream;

◉*◉* синтаксис перегрузки ввода-вывода;

◉*◉* примеры использования (классы Вектор, Матрица, Строка и так далее);

• дружественные классы:

◉*◉* цели и задачи;

◉*◉* синтаксис и примеры использования.

**Шаблоны функций и классов**

1. Шаблоны функций:

• понятие шаблона. Определение и объявление шаблона;

• синтаксис объявления шаблонной функции;

• ключевые слова class и typename;

2. Отличия обычной и шаблонной функции;

3. Примеры создания шаблонных функций (например, Максимум, Мини-

мум, Сортировка, Поиск и так далее);

4. Перегрузка шаблонных функций;

5. Шаблоны классов:

• полная специализация;

• частичная специализация;

• примеры создания шаблонов классов (например, Вектор, Матрица,

и так далее).

**Динамические структуры данных**

1. Понятие динамической структуры данных.

2. Стек:

• понятие стека;

• принцип lifo;

• пример создания и практического использования стека.

3. Очереди:

• понятие очереди;

• типы очередей:

◉*◉* обычная очередь. Принцип FIFO;

◉*◉* очередь с приоритетами;

4. Списки:

•понятие списка;

•односвязный список:

▶*▶* добавление элементов в список;

▶*▶*обход списка;

▶*▶* удаление элементов;

▶*▶* замена элементов;

▶*▶* показ элементов списка;

▶*▶* поиск элемента в списке;

▶*▶* примеры создания и использования списков;

5. Двусвязный список:

▶*▶* добавление элементов в список;

▶*▶*обход списка;

▶*▶* удаление элементов;

▶*▶* замена элементов;

▶*▶* показ элементов списка;

▶*▶* поиск элемента в списке;

6. Примеры создания и использования списков, сравнительный анализ типов списка;

7.Бинарные деревья;

◉*◉* понятие дерева;

◉*◉* операции, выполняемые над деревом:

▶*▶* добавление элемента;

▶*▶* получение значения элемента;

▶*▶* удаление элемента;

▶*▶* показ дерева;

▶*▶* поиск элемента;

▶*▶* уничтожение дерева;

▶*▶* примеры создания и использования бинарных деревьев;

8. Сравнительный анализ изученных динамических структур данных.

**Агрегация, композиция и наследование**

1. Вложенный класс:

• синтаксис объявления;

• цели и задачи вложенных классов;

• примеры использования (например, связанный список, и так далее).

2. Наследование:

• цели и задачи наследования;

• примеры использования наследования в окружающей среде;

• типы наследования;

• понятия базового и дочернего класса;

• одиночное наследование:

◉*◉* синтаксис одиночного наследования;

◉*◉* спецификатор доступа protected;

◉*◉* спецификаторы доступа при одиночном наследовании;

◉*◉* поведение конструкторов и деструкторов при одиночном наследовании;

◉*◉* примеры использования одиночного наследования;

3. Множественное наследование:

◉*◉* синтаксис множественного наследования;

◉*◉* спецификаторы доступа при множественном наследовании;

◉*◉* поведение конструкторов и деструкторов при множественном наследовании;

◉*◉* примеры использования множественного наследования;

◉*◉* недостатки использования множественного наследования;

4. Плюсы и минусы наследования;

• наследование шаблонов;

• виртуальный базовый класс:

◉*◉* пример проблемы ромба;

◉*◉* спецификатор virtual и виртуальное наследование;

◉*◉* пример использования виртуального базового класса.

**Виртуальные методы**

1. Указатель на базовый класс, виртуальные методы.

2. Статический и динамический полиморфизм.

3. Таблица виртуальных функцій, иИспользование спецификаторов override и final.

4. Примеры использования виртуальных методов.

5. Абстрактный класс:

• чисто виртуальный метод;

• абстрактный класс.

6. Виртуальный деструктор и чисто виртуальный деструктор.

**Обработка исключительных ситуаций**

1. Понятие исключительной ситуации, необходимость обработки исключительных ситуацій, типы исключительных ситуаций.

2. Базовые понятие обработки исключительных ситуаций:

• ключевое слово try;

• ключевое слово catch;

• ключевое слово throw;

• примеры использования обработки исключительных ситуаций.

3. Понятие необработанного исключения, специальная форма catch(…).

4. Исключения и функции.

5. Построение иерархии пользовательских классов исключений, стандартный класс exception и его потомки.

6. Обработка ошибок при выделении памяти.

**Пространства имен**

1. Причины возникновения пространств имен, синтаксис объявления, оператор using.

2. Вложенные пространства, тонкости использования пространств имен.

**Преобразования типов в C++**

1. Оператор typeid.

2. Преобразования типов в C++.

• dynamic\_cast;

• static\_cast;

• reinterept\_cast;

• const\_cast.

**Работа с потоками в языке С++**

1. Понятие потока, виды потоков, ввод и вывод в языке C++.

2. Файловый ввод-вывод в С++.

• класс ofstream;

• класс ifstream;

• класс fstream.

3. Файловые операции:

• открытие файла;

• закрытие файла;

• чтение данных;

• запись данных;

• позиционирование по файлу;

• перегрузка <<,>> для чтения, сохранения данных в файл.

**Класс string, умные указатели, работа со стандартной библиотекой C++, лямбда-функции**

1. Класс string:

• что такое string?;

• цели и задачи класса string;

• анализ устройства класса string;

• примеры использования класса string.

3. Стандартная библиотека шаблонов (STL).

• что такое STL?;

• история возникновения STL;

• цели и задачи стандартной библиотеки шаблонов.

4. Основные понятия STL.

• контейнер;

• итератор;

• алгоритм;

• функтор.

5. Контейнер.

• что такое контейнер?

• типы контейнеров;

• пример использования контейнера vector.

6. Итератор:

• что такое итератор?

• типы итераторов;

• почему так много типов итераторов?

• пример использования итераторов.

7. Подробно о контейнерах.

8. Анализ и использование классов list, map, multimap.

9. Практические примеры использования классов контейнеров.

10. Использование функторов.

11. Использование алгоритмов.

12. Практические примеры использования функторов, алгоритмов.

13. Лямбда-функции:

• что такое лямбда-функция;

• цели и задачи лямбда-функций;

• примеры использования.