Программа курса

ООП с использованием языка C++

Об'єктно-орієнтоване програмування з використанням мови C++
Object-Oriented Programming using C++

Для групп стационара. Версия 4.0.1

Продолжительность курса: 60 пар (30 дней)

Цель курса

Обучить слушателя разработке приложений с использованием объектно-ориентированного подхода, заложенного в язык программирования C++.
Научить выбирать правильные механизмы для решения той или иной задачи.
Ознакомить с тонкостями использования инкапсуляции, наследования, полиморфизма, динамических структур данных, библиотеки STL.

По окончании курса слушатель будет:

- понимать базовые и расширенные концепции ООП;
- реализовывать пользовательские конструкторы копирования;
- грамотно перегружать операторы методами-членами класса и внешними функциями;
- разбираться в тонкостях динамических структур данных;
- уметь проектировать иерархии классов;
- программировать с использованием шаблонов и виртуальных методов;
- использовать библиотеку стандартных шаблонов STL;
- перехватывать исключения и строить собственные иерархии пользовательских исключений;
- взаимодействовать с файловыми потоками и потоками данных.

По окончании данного курса студент сдает практический и теоретический экзамен по материалам курса. Для допуска к экзамену должны быть сданы все домашние и практические задания.

Перед началом данного предмета необходимо предоставить студентам доступ к следующим курсам Cisco Networking Academy:

- Programming Essentials in C++;
- Advanced Programming in C++.

Тематический план

Модуль 1

Введение в объектно-ориентированное программирование на C++

1. **Вступление.**
2. **История и этапы развития языка C++.**
3. **Сравнительный анализ языка C++ с другими языками программирования (C, PASCAL, BASIC).**
4. **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.
5. **Класс и объект.**
6. **Классы.**
   - Понятие класса.
   - Синтаксис объявления.
   - Спецификаторы доступа:
     - public;
     - private;
     - protected.
7. **Переменные-члены класса.**
8. **Методы-члены.**
   - Реализация тела метода внутри класса.
   - Вынос тела метода за класс.
9. **Практические примеры работы с классами.**
   - Использование спецификаторов доступа.
   - Реализация практических примеров (Студент, Прямоугольник, Точка, Машина и так далее).
10. **Понятие аксессора, инспектора, модификатора.**
    - Определение.
    - Реализация.

11. Встроенные (inline) методы в классах.

- Необходимость использования.
- Примеры объявления и использования.
- Ограничения при использовании inline методов.

12. Сравнительный анализ структур и классов.

13. 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).

14. 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).

15. Указатели на объекты.

16. Массивы объектов.

17. Инициализаторы.

- Синтаксис объявления.
- Примеры практического использования (инициализация поля класса, константы члена класса, инициализация внутреннего объекта).

18. Унифицированная инициализация объектов.

19. Инициализация членов класса.

20. Делегирование конструкторов.

21. 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.

Модуль 2

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.

Модуль 3

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.

Модуль 4

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:
         - цели и задачи перегрузки инкремента и декремента;
         - синтаксис перегрузки;
         - 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. **Применение переноса при перегрузке оператора присваивания.**
5. **Заданные по умолчанию методы (default) и удаленные методы (delete).**
6. **Специальные перегрузки.**
   - Перегрузка [].
   - Перегрузка ().

- Перегрузка оператора преобразования типов.
- Использование explicit для преобразований, определяемых классом.

7. **Список операторов, которые невозможно перегрузить.**
8. **Статический полиморфизм и перегрузка операторов как частный случай.**
9. **Перегрузка операторов дружественными и глобальными функциями.**
   - Перегрузка операторов глобальными функциями:
     - отличия синтаксиса;
     - примеры использования (классы Вектор, Матрица, Строка и так далее).
   - 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.

Модуль 5

Шаблоны классов, класс string

1. **Статический полиморфизм и шаблоны как частный случай.**
2. **Шаблоны классов.**
   - Шаблоны классов.
   - Полная специализация.
   - Частичная специализация.
   - Примеры создания шаблонов классов (например: Вектор, Матрица и так далее).
3. **Шаблоны с переменным числом аргументов.**

4. **Использование std::initializer_list.**
5. **Класс string.**
   - Что такое string.
   - Цели и задачи класса string.
   - Анализ устройства класса string.
   - Примеры использования класса string.

Модуль 6

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;
     - кольцевая очередь;
     - очередь с приоритетами;
     - примеры создания и использования очередей.
   - 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.
     - двусвязный список:
       - добавление элементов в список;
       - обход списка;

        - удаление элементов;
        - замена элементов;
        - показ элементов списка;
        - поиск элемента в списке;
        - примеры создания и использования списков;
        - сравнительный анализ типов списка.
    - Деревья:
        - понятие дерева;
        - бинарное дерево поиска;
        - сортирующее дерево;
        - красно-черное дерево;
        - операции, выполняемые над деревом:
            - добавление элемента;
            - получение значения элемента;
            - удаление элемента;
            - показ дерева;
            - поиск элемента;
            - уничтожение дерева;
            - примеры создания и использования бинарных деревьев поиска, сортирующих деревьев, красно-черных деревьев.
    - Сравнительный анализ изученных динамических структур данных.

Модуль 7

Агрегация, композиция и наследование

1. **Вложенный класс.**
    - Синтаксис объявления.
    - Цели и задачи вложенных классов.
    - Примеры использования (например: связанный список и так далее).
2. **Агрегация и композиция.**
    - Понятие агрегации.
    - Понятие композиции.
    - Отличие агрегации от композиции.
3. **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 (например: иерархии Человек-Студент, Человек-Милиционер и так далее).
- 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.
- Обсуждение плюсов и минусов наследования.
- Наследование шаблонов:
  - виртуальный базовый класс;
  - пример проблемы ромба;
  - спецификатор virtual и виртуальное наследование;
  - пример использования виртуального базового класса.

Модуль 8

Виртуальные методы

1. **Указатель на базовый класс.**
2. **Виртуальные методы.**
3. **Ранее и позднее связывание.**
4. **Статический и динамический полиморфизм.**
5. **Таблица виртуальных функций.**
6. **Использование спецификаторов override и final.**
7. **Примеры использования виртуальных методов.**

8. **Абстрактный класс.**
   - Чисто виртуальный метод.
   - Абстрактный класс.
9. **Виртуальный деструктор.**
10. **Чисто виртуальный деструктор.**

Модуль 9

Обработка исключительных ситуаций

1. **Понятие исключительной ситуации.**
2. **Необходимость обработки исключительных ситуаций.**
3. **Типы исключительных ситуаций.**
4. **Базовые понятие обработки исключительных ситуаций.**
   - Ключевое слово try.
   - Ключевое слово catch.
   - Ключевое слово throw.
   - Примеры использования обработки исключительных ситуаций.
5. **Понятие необработанного исключения.**
6. **Специальная форма catch(...).**
7. **Исключения и функции.**
8. **Описание списка исключений генерируемых функцией.**
9. **Раскрутка стека вызовов.**
10. **Повторная генерация исключения.**
11. **Построение иерархии пользовательских классов исключений.**
12. **Стандартный класс exception и его потомки.**
13. **Обработка ошибок при выделении памяти.**
14. **Обработка не пойманных и неожидаемых исключений.**

Модуль 10

Пространства имен

1. **Причины возникновения пространств имен.**
2. **Синтаксис объявления.**

3. **Оператор using.**
4. **Вложенные пространства.**
5. **Тонкости использования пространств имен.**

Модуль 11

Преобразования типов в C++

1. **Оператор typeid.**
2. **Преобразования типов в C++.**
   - dynamic_cast.
   - static_cast.
   - reinterept_cast.
   - const_cast.

Модуль 12

Работа с потоками в языке C++

1. **Понятие потока.**
2. **Виды потоков.**
3. **Ввод и вывод в языке C++.**
4. **Файловый ввод-вывод в C++.**
   - Класс ofstream.
   - Класс ifstream.
   - Класс fstream.
5. **Файловые операции.**
   - Открытие файла.
   - Закрытие файла.
   - Чтение данных.
   - Запись данных.
   - Позиционирование по файлу.
   - Перегрузка <<,>> для чтения, сохранения данных в файл.

Модуль 13

Умные указатели, работа со стандартной библиотекой C++, лямбда-функции

1. **Умные указатели.**
   - Что такое умный указатель.
   - Классы умных указателей:
     - auto_ptr;
     - shared_ptr;
     - unique_ptr;
     - концептуальные отличия классов умных указателей.
   - Особенности использования auto_ptr.
   - Особенности использования unique_ptr.
   - Особенности использования shared_ptr.
2. **Стандартная библиотека шаблонов (STL).**
   - Что такое STL.
   - История возникновения STL.
   - Цели и задачи стандартной библиотеки шаблонов.
3. **Основные понятия STL.**
   - Контейнер.
   - Итератор.
   - Алгоритм.
   - Функтор.
4. **Контейнер.**
   - Что такое контейнер.
   - Типы контейнеров.
   - Пример использования контейнера vector.
5. **Итератор.**
   - Что такое итератор.
   - Типы итераторов.
   - Почему так много типов итераторов.
   - Пример использования итераторов.
6. **Подробно о контейнерах.**
7. **Анализ и использование классов list, map, multimap.**

8. **Практические примеры использования классов контейнеров.**
9. **Использование функторов.**
10. **Использование алгоритмов.**
11. **Практические примеры использования функторов, алгоритмов.**
12. **Лямбда-функции.**
    - Что такое лямбда-функция.
    - Цели и задачи лямбда-функций.
    - Примеры использования.

Модуль 14

Экзамен