Объектно-ориентированное программирование

Лабораторные работы (очная форма обучения)

Вместо лабораторных работ (кроме второй) можно разрабатывать собственный проект по договорённости с преподавателем. Индивидуальный проект должен предполагать самостоятельный процесс объектно-ориентированного проектирования.

Рекомендуется использовать систему контроля версий.

Семестр I (2018-2019)

Задание 1. Работа с Git.

(не обязательное, максимальная оценка на экзамене — 3, если задание не сдано)

Продемонстрировать работу с системой управления версиями git на примере одного из проектов (программ) созданных на лабораторных занятиях.

Сценарий использования для примера:

1. Создать репозитарий.
2. Добавить файлы к отслеживанию.
3. Сделать коммит (зафиксировать изменения).
4. Исправить предыдущий коммит.
5. Создать новую ветку.
6. Создать worktree для ветки master.
7. Переключится на неё.
8. Внести изменения. Посмотреть разницу (diff)
9. Зафиксировать изменения
10. Посмотреть что содержится в файлах на ветке master.
11. Объединить ветки.
12. Решить конфликт.
13. Клонировать удалённый репозитарий
14. Отправить изменения в удалённый репозитарий, забрать изменения из удалённого репозитария.

Вопросы.

Что такое система управления версиями? Для чего она используется?

Что такое репозитарий (локальный и удалённый)?

Какие файлы следует добавлять к отслеживанию, а какие нет?

В каких случаях создавать ветку?

Что такое конфликт? Как исправить?

Ссылки

- http://uleming.github.io/gitbook/index.html
- https://inoyakaigor.ru/blog/85

Задание 2. Простой класс

Классы на выбор:

- Время. Сложение, вычитание. Добавление минут, секунд, часов и т. п. Перевод времени в секунды, часы, минуты. Конвертирование в строку.
- Дата. Реализовать то же самое, что и для времени.
- Комплексное число. Операторы сложения, вычитания, умножения (на комплексное и действительное число). Вычисление аргумента и модуля.
- Кватернион. Аналогично комплексному числу.
- Вектор. Здаётся своими компонентами. Вычисление длины, углов между осями; операторы сложения и вычитания, умножения на число.
- Другой класс по согласованию с преподавателем.

1. Описать АДТ.

2. Представить класс в виде UML диаграммы

3. Описать класс на C++. Реализовать методы для доступа и изменения данных, контроль постоянства, конструктор с параметрами. Операторы и генерирование исключительных ситуаций если необходимо

4. Продемонстрировать работу с классом, с основными методами, операторами. Создать массив из объектов. Записать состояние объектов в файл, загрузить из файла.

5. Для класса привести документацию описав его назначение, принципы использования, смысл методов и их параметров. Если необходимо привести пример использования класса в документации.

Вопросы

Что такое АДТ?

Что такое предусловия? Для чего нужны? Что такое постусловия?

Что такое класс? Что такое объект?

Что такое абстрагирование?

Что такое инкапсуляция? Что такое метод? Что такое поле класса? Что такое конструктор?

Что такое принцип сокрытия? Что такое «чёрный ящик»?

Что такое оператор?

Как вызвать метод конкретного объекта находящегося в массиве?

Чем отличаются обращения к методам в C++ с использованием объекта, ссылки на объект и указателя на объект?

Что такое равенство объектов? Когда объекты идентичны?

Что такое поведение? Что такое состояние?

Пример: https://github.com/VetrovSV/OOP/tree/master/simple_class

Задание 3. Класс «матрица»

(не обязательное, -1 к максимальной оценке, если не сдана)

Создать класс представляющий матрицу. В качестве основы использовать класс vector.

Реализовать:

- доступ к отдельным элементам матрицы,
- сложение, вычитание
- умножение на число
- умножение матрицы на матрицу
- транспонирование,
- заполнение матрицы одним значением
- заполнение матрицы случайными числами,
- создание диагональной матрицы.
- Дополнительно:
  - вычисление определителя
  - вычисление обратной матрицы
  - доступ к строкам матрицы
  - Операторы *=, -=, +=?
  - применения функции к элементам матрицы
- Наглядно продемонстрировать работу всех методов. Недопустимые или невозможные операции над матрицами обрабатывать с помощью механизма генерации исключений.
- Рекомендуется использование системы контроля версий при разработке.

  Пример: QGenericMatrix - http://doc.qt.io/qt-5/qgenericmatrix.html

Для класса привести документацию описав его назначение, принципы использования, смысл методов и их параметров. Если необходимо привести пример использования класса в документации.

Возможна замены темы задания по согласованию с преподавателем.

Вопросы

Какие бывают виды конструкторов?

Объясните правило большой пятёрки

Как должен быть реализован оператор присваивания?

Что такое оператор? Как он определяется?

Когда оператор следует определять как метод, а когда как дружественную функцию?

Чем отличается класс vector от класса list?

Задание 4. Наследование

1. Создать UML диаграмму из 3-х (или более) классов имеющих отношение типа наследование.
2. Реализовать классы на C++. Продемонстрировать работу с ними.

Пример классов для задания: геометрическая фигура на плоскости, квадрат, круг, прямоугольник.

Для классов привести документацию описав их назначение, принципы использования, смысл методов и их параметров. При необходимости приведите пример использования классов в документации.

Вопросы

Что такое наследование?

Какие классы называется базовыми и производными?

Сколько предков может иметь класс?

Что такое множественное наследование? Чем оно опасно? Когда его можно использовать?

Что такое перегрузка метода? Что такое переопределение метода?

Как работает преобразование типов связанных наследованием?

Что такое интерфейс (ООП)?

Как влияют области видимости внутри класса на наследуемые методы и поля?

Задание 5. Диаграмма классов

Составить UML диаграмму классов. Не менее 4 классов. Каждый взаимодействует хотя бы с одним другим. Взаимодействие не должно быть только последовательным. Объекты не должны быть однотипными. Должно быть как минимум по одному отношению: ассоциация, агрегация (композиция), наследование. Указать мощность связей. Диаграмму оформить в электронном и твёрдом формате. Сохранение диаграммы исключительно в формат растровых изображений не допускается. Возможна презентация схемы на доске или проекторе, обсуждение в группе.

Диаграмма классов должна быть целостной: полностью описывать предметную область, не иметь лишних сущностей, свойств или отношений.

Вопросы

Какие отношения возможны между классами?

Какие отношения возможны между объектами?

Что такое мощность отношения?

Как в C++ может быть реализовано отношение *ассоциация (связь)?*

Чем отличается агрегация от композиции?

Как в C++ может быть реализовано отношения агрегации и композиции?

Когда следует использовать наследование, а когда агрегацию (композицию)?

Задание 6. Реализовать лабораторную 5 на C++

(не обязательное, +1 к оценке на экзамене)

Реализовать предыдущую лабораторную работу. Как минимум один метод должен быть переопределён. Каждый класс должен располагаться в отдельном модуле.

Для классов привести документацию описав их назначение, принципы использования, смысл методов и их параметров. При необходимости приведите пример использования классов в документации.

Представить состояние и изменение объектов наглядно.

Рекомендации: вести разработку «сверху вниз», использовать систему контроля версий.

Вопросы

Какие отношения возможны между классами?

Какие отношения возможны между объектами?

Что такое мощность отношения?

Как в C++ может быть отображено отношение *ассоциация (связь)?*