

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.

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

Сценарий использования для примера:

1. Создать репозиторий.
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 репозиторий
14. Отправить изменения в удалённый репозиторий, забрать изменения из удалённого репозитория.

Вопросы.

Что такое система управления версиями? Для чего она используется?

Что такое репозиторий (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.
- Кватернион
- Вектор
- 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.

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 класс? Что такое объект?

Что такое абстрагирование?

Что такое инкапсуляция? Что такое метод? Что такое конструктор?

Что такое принцип сокрытия?

Что такое «чёрный ящик»?

Что такое оператор?

Что такое поле класса?

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?

Задание 3. Класс «матрица»

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

Реализовать:

- доступ к отдельным элементам матрицы,
- *доступ к строкам матрицы?*
- сложение, вычитание
- умножение на число
- умножение матрицы на матрицу
- транспонирование,
- вычисление определителя,
- вычисление обратной матрицы.
- заполнение матрицы одним значением
- заполнение матрицы случайными числами,
- Создание диагональной матрицы.
- Операторы \ast , $-$, $+$?
- 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.

Вопросы

Какие бывают виды конструкторов?

Что такое оператор?

Объясните правило большой пятёрки

Как должен быть реализован оператор присваивания?

Что такое оператор? Как он определяется?

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 называются базовыми и производными?

Что такое множественное наследование?

Сколько предков может иметь класс?

Что такое перегрузка метода? Что такое переопределение метода?

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

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. 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++

Реализовать предыдущую лабораторную работу. Как минимум одно поле и метод должны быть переопределены. 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)*?

Задание 7. Калькулятор.

Калькулятор должен корректно обрабатывать любые входные данные. Сделать обработку исключительных ситуаций. Помимо арифметических операций и возведения в любую степень калькулятор должен вычислять функции: \sin , \cos , \tan , \ln , \exp .

Дополнительно: хранить историю вычислений.

Вопросы

Что такое исключительная ситуация?

Как работает механизм обработки исключительных ситуаций?

Что такое бизнес-логика?

В какой части программы должна быть реализована бизнес-логика?

Изобразите диаграмму классов для приложения.