|  |  |  |
| --- | --- | --- |
| **C++. Контрольный урок № 2 базовый модуль** | | |
| **Ученик:** | **Уроков пройдено:** | **Дата:** |
| **Ссылка на профиль:** | | |

|  |  |  |
| --- | --- | --- |
| **Теория (устный опрос, 1-2 вопроса из раздела, в случае затруднения, можно задать еще вопрос из раздела)** | | Отметка  +/- |
| **Тема 1 ООП, классы, экземпляры классов** (весь проект Эпичный лабиринт) | |  |
| При создании проекты мы используем объектно-ориентированное программирование.Чем удобен это подход? | При создании программы мы концентрируемся на определении объектов и их способностей с помощью создания классов. |  |
| Как создать свой класс? | Написать в код  **class имя\_класса**  **{**  **//закрытые поля и методы класса**  **public:**  **//открытые методы класса**  **};** |  |
| Что такое поля класса? | Это переменные, которые позволяют описать свойства проектируемого объекта. |  |
| Что такое методы класса? | Это функции, которые позволяют описать действия проектируемого объекта. |  |
| Как создать экземпляр класса? | **имя\_класса имя\_экземпляра;**  или с использованием конструктора:  **имя\_класса имя\_экземпляра( параметр1, параметр2, … );** |  |
| **Тема 2 Модификаторы доступа, геттеры, сеттеры, конструктор** (весь проект Эпичный лабиринт) | |  |
| Что такое конструктор класса? | Это специальный метод класса, его имя совпадает с именем класса, он не имеет типа возвращаемого значения. |  |
| Когда вызывается конструктор класса? | Он вызывается при создании экземпляра класса. |  |
| Зачем нужен конструктор класса? | Конструктор класса устанавливает обязательные поля класса, выполняет подготовительные действия, необходимые для существования экземпляра. |  |
| Для чего нужно делать закрытыми (private) поля класса? | Мы делаем поля закрытыми, чтобы никто кроме методов класса не мог случайно поменять данные в объекте класса. |  |
| Как получать доступ к закрытым членам класса в программе? | Нужно предоставить инструменты для работы с полями класса - открытые методы установки и получения значений, их обычно называют сеттерами и геттерами **set(), get()** |  |
| Как сделать методы класса открытыми? | Указать модификатор **public** перед объявлением методов. |  |
| **Тема 3 Организация проекта** (весь проект Эпичный лабиринт) | |  |
| Почему удобно разделить свой проект на части? | Если проект сложный и кода в нем достаточно много, то в нем легко запутаться. |  |
| Зачем создавать заголовочные файлы? | Для описания каждого класса лучше выделить отдельный заголовочный файл (.h), а реализацию методов каждого класса вынести в файл исполняемого кода (.cpp). Таким образом, мы можем отделить объявление от реализации. |  |
| Если код распределен по разным файлам, то как подключить заголовочный файл? | Использовать  **#include “имя\_файла”** |  |
| Как избежать повторного подключения файлов? | Использовать **#pragma once** |  |
| **Тема 4 Проектирование классов движка, персонажа, карты** (Эпичный лабиринт. Настройка проекта и создание персонажа, Создание игровой карты) | |  |
| Как устроен движок нашей программы? (class Engine) | Этот класс имеет поля: окно, карта, нажатие на клавиши, камера; методы: конструктор, инициализация, игровой цикл, рисование и обновление окна. |  |
| Какие поля имеет класс персонажа? почему они необходимы? (class Player) | * положение необходимо, чтобы знать, где расположить персонажа * текстура нужна, чтобы мы могли загрузить картинку * спрайт нужен, чтобы отрисовать героя * скорость необходима, чтобы управлять движением героя * ширина и высота нужны, чтобы использовать их при отрисовке * текущий кадр для смены анимации |  |
| Какие методы есть у класса персонажа? (class Player) | Конструктор, метод инициализации, обновление, метод, который не позволяет уйти за границы карты, геттеры и сеттеры полей |  |
| Какие поля есть у класса карты? (class Map) | Массив, из которого отрисовывается карта, картинка, текстура, спрайт, указатели на игрока и монетку. |  |
| Какие методы есть у класса карты? (class Map) | Конструктор, методы инициализации, обновления, рисования, получения указателя на персонаж |  |
| **Тема 5 Движение персонажа, игровая камера** (Эпичный лабиринт. Передвижение персонажа по карте. Игровая камера. Ограничение передвижения) | |  |
| Как реализовано движение персонажа? (class KeyPressedAction) | Логика отслеживания нажатия клавиш и движения реализована в отдельном классе KeyPressedAction, который имеет указатель на игрока и основной метод action, который вызывается из метода обновления движка и содержит всю логику движения при нажатии на клавиши и анимации. |  |
| Зачем понадобился переход к указателям при реализации движения персонажа? | Возникла необходимость передавать экземпляр игрока в класс движения для изменения, если передавать не по указателю, то мы будем менять копии экземпляра, а не его самого. |  |
| Что такое указатель this? | Это скрытый указатель, он указывает на текущий объект класса. |  |
| Как реализована игровая камера? (class Camera) | Камера имеет поле вида и указатель на игрока, методы установки центра, чтобы следить за игроком, методы обновления и получения вида. |  |
| Почему лучше “волшебные числа”: ширину, высоту окна, ширину, высоту карты, границы камеры - определять как константы? | Когда числа имеют названия, то проще реализовывать логику игры, находить ошибки, кроме того, при необходимости изменения этого числа достаточно поменять значение константы. |  |
| **Тема 6 Проектирование класса монетки, переход к наследованию** (Эпичный лабиринт. Передвижение персонажа по карте. Игровая камера. Ограничение передвижения) | |  |
| Какие поля и методы есть у класса монетки? (class Money)  Какие из этих полей и методов похож на поля и методы игрока? | Поля: положение, текстура,  спрайт, ширина, высота.  Методы: конструктор, метод инициализации, геттеры и сеттеры полей |  |
| Что такое наследование? Почему удобно применить его в нашем проекте? | Это свойство, позволяющее создать новый класс-потомок на основе уже существующего, при этом все характеристики класса родителя присваиваются классу-потомку.  В нашем проекте игрок и монетка имеют одинаковые поля и методы, удобно выделить базовый класс, в который мы сможем поместить общие поля и методы. |  |
| Как унаследовать один класс от другого? | **class имя\_класса\_наследника: модификатор имя\_класса\_родителя**  **{**  **//закрытые поля и методы класса**  **public:**  **//открытые методы класса**  **};** |  |
| В чем особенности модификатора доступа **protected**? | Этот модификатор делает члены класса доступными только для наследников. |  |
| Что такое виртуальная функция? | Это особый тип функции, которая, при её вызове, выполняет «наиболее» дочерний метод, который существует между родительским и дочерними классами. |  |
| **Итого (максимум 29)** | |  |

|  |  |
| --- | --- |
| **Практика (выполнение практических заданий)** |  |
| Добавил поле скорости в класс камня (Stone.h) |  |
| Добавил метод установки скорости в класс камня (Stone.h) |  |
| Добавил метод обновления в класс камня (Stone.h) |  |
| Реализовал метод установки скорости (Stone.cpp) |  |
| Добавил вызов метода установки скорости в конструктор камня (Stone.cpp) |  |
| Реализовал метод обновления камня (Stone.cpp) |  |
| Подключил заголовочный файл карты для проверки границы карты (Stone.cpp) |  |
| Добавил массив указателей на камни (Map.h) |  |
| Подключил заголовочный файл камня, чтобы можно было в карте создавать камни (Map.h) |  |
| В цикле заполнил массив камнями, расположив их равномерно по карте (Map.cpp) |  |
| Указал модификатор public, чтобы сделать методы камня доступными (Stone.h) |  |
| Добавил рисование каждого камня из массива (Map.cpp) |  |
| Уменьшил масштаб спрайта камня при инициализации (Stone.cpp) |  |
| Добавил обновление каждого камня из массива (Map.cpp) |  |
| **Итого (максимум 14)** |  |

**Оценка отношения ученика к обучению (по шкале от 1 до 5):**

Интерес –

Мотивации –

Вовлеченность –

Успеваемость –

***Во время занятий ученик проявил себя как (выбрать 2-3):***

* Внимательный
* Усидчивый
* Любознательный
* Трудолюбивый
* Креативный, новатор,
* Изобретатель
* Свой вариант

***Качества, требующие дальнейшего внимания и развития (выбрать 2-3):***

* Внимательность,
* Усидчивость
* Трудолюбие
* Выполнение домашних заданий
* Математические расчеты
* Умение презентовать-рассказать и логически объяснить
* Творчество
* Инициативность

***Темы, особенно интересные ученику:***

* Программирование
* Рисование
* Сам игровой процесс

***Рекомендации от преподавателя:***

* Продолжить курс
* Сменить курс
* Увеличить количество самостоятельных занятий до…
* Уменьшить количество самостоятельных занятий до…