МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Тема: Создание игры с помощью классов

Студент гр. 4381 Ишамчурин Д.И.

Преподаватель Жангиров.Т.Р.

Санкт-Петербург

2026

Цель работы.

Разработать консольную игру на C++, реализовав классы игрока, врага и игрового поля с конструкторами копирования и перемещения, где игрок перемещается по полю и сражается с врагами, управляемыми компьютером.

Задание.

1. Создать класс игрока, который должен хранить информацию об игроке (его жизни, урон, очки, и т.д. - студент сам определяет необходимые для работы характеристики). Объект класса игрока должен перемещаться по карте. Если у игрока кончаются жизни, то происходит конец игры.
2. Создать класс врага, который хранит параметры жизней и урона. Объектами класса врага управляет компьютер. При перемещении, если враг пытается перейти на клетку с игроком, то перемещение не происходит, и игроку наносится урон.
3. Создать класс квадратного/прямоугольного игрового поля, по которому перемещаются игрок и враги. Игровое поле не должно быть меньше 10 на 10 клеток, и не больше 25 на 25 клеток. Размеры поля задаются через конструктор. Рекомендуется для хранения информации об отдельных клетках поля создать отдельный класс.
   Реализовать конструкторы перемещения и копирования для поля, а также соответствующие операторы присваивания с копированием и перемещением (должна происходить глубокая копия).

Примечания:

Не забывайте для полей и методов определять модификаторы доступа

Для обозначения переменной, которая принимает небольшое ограниченное количество значений, используйте enum

Не используйте глобальные переменные

При реализации перемещения, не должно быть лишнего копирования

У поля не должно быть методов возвращающих указатель на поле в явном виде, так как это небезопасно

Информация о координатах игрока не должна дублироваться у игрока и у поля

Выполнение работы.

ОСНОВНАЯ СТРУКТУРА КЛАССОВ

Разработанная программа состоит из четырех основных классов, каждый из которых выполняет свою четко определенную роль в игровой механике.

Класс Cell является самым простым по своей структуре и предназначен для представления одной клетки игрового поля. В нем определены два закрытых логических поля hasPlayer и hasEnemy, которые хранят информацию о том, находится ли на данной клетке игрок или враг соответственно. Конструктор по умолчанию инициализирует эти поля значением false, что означает пустую клетку. Для изменения состояния клетки предусмотрены методы placePlayer, removePlayer, placeEnemy и removeEnemy, которые устанавливают или сбрасывают соответствующие флаги. Также класс содержит методы containsPlayer и contains Enemy для проверки наличия игрока или врага на клетке без возможности изменения состояния. Такая реализация позволяет инкапсулировать состояние клетки и предоставлять только необходимый интерфейс для работы с ней.

Класс Player отвечает за хранение и управление состоянием игрока. В его закрытой части объявлены переменные health для хранения количества жизней игрока, damage для определения урона, который игрок наносит врагам при атаке, а также x и y для хранения текущих координат игрока на поле. Конструктор класса принимает начальные значения здоровья и урона и устанавливает начальные координаты в ноль. Метод setPosition позволяет изменить координаты игрока при перемещении, а методы getX и getY предоставляют доступ к текущему положению. Для получения параметров игрока реализованы методы getHealth и getDamage. Метод takeDamage уменьшает здоровье игрока на переданное значение, что происходит при столкновении с врагом. Метод isAlive возвращает true, если здоровье игрока

больше нуля, и false в противном случае, что используется для определения окончания игры.

Класс Enemy по своей структуре очень похож на класс игрока, что логично, поскольку оба этих персонажа обладают схожими характеристиками. В закрытой части класса объявлены переменные health и damage для хранения параметров врага, а также x и y для его координат на поле. Конструктор принимает начальные значения здоровья и урона и устанавливает координаты в ноль. Метод setPosition позволяет компьютеру управлять положением врага на поле, а методы getX и getY предоставляют доступ к этим координатам. Метод getDamage возвращает величину урона, который враг наносит игроку при атаке. Метод takeDamage уменьшает здоровье врага при получении урона от игрока и предотвращает уход здоровья в отрицательную область. Метод isAlive проверяет, жив ли враг, что используется для удаления его с поля после гибели.

Класс GameField является центральным и наиболее сложным классом программы, который управляет всем игровым процессом и координирует взаимодействие между остальными классами. В его закрытой части объявлены переменные width и height для хранения размеров игрового поля, вектор cells типа Cell, который представляет собой двумерное поле, вытянутое в одномерный массив для удобства работы, вектор enemies для хранения всех врагов, находящихся на поле, и указатель player на объект игрока. Конструктор класса принимает ширину и высоту поля, инициализирует вектор клеток соответствующим размером и устанавливает указатель на игрока в nullptr. Также в конструкторе инициализируется генератор случайных чисел для обеспечения случайности перемещения врагов.

Для класса GameField реализованы специальные методы, обеспечивающие правильное управление ресурсами. Конструктор копирования выполняет глубокое копирование всех данных, включая вектор клеток и вектор врагов, а также копирует указатель на игрока. Оператор присваивания с копированием реализует аналогичную логику с проверкой на самоприсваивание. Конструктор перемещения переносит данные из временного объекта, обнуляя

указатель на игрока у исходного объекта, а оператор присваивания с перемещением выполняет аналогичные действия с освобождением текущих ресурсов. Такая реализация соответствует правилу пяти и гарантирует корректное поведение класса при копировании и перемещении.

В классе GameField также реализован вспомогательный закрытый метод getIndex, который преобразует двумерные координаты в индекс одномерного вектора клеток. Метод placePlayer размещает игрока на указанной клетке, сохраняя указатель на объект игрока, устанавливая его координаты и помечая соответствующую клетку как занятую игроком. Метод placeEnemy создает копию переданного врага, устанавливает его координаты, добавляет в вектор enemies и помечает соответствующую клетку как занятую врагом.

Метод movePlayer реализует логику перемещения игрока по полю. При вызове метода текущая клетка игрока очищается от флага присутствия игрока, затем на основе переданного направления вычисляются новые координаты. Если новые координаты выходят за границы поля, игрок возвращается на исходную позицию. Если на новой клетке находится враг, происходит атака: программа находит соответствующего врага в векторе enemies, наносит ему урон, и если враг погибает, удаляет его из вектора и очищает клетку. При этом игрок не перемещается на клетку с врагом, а остается на своем месте. Если клетка свободна, игрок перемещается на новую позицию.

Метод enemyTurn реализует ход компьютера, управляющего всеми врагами. Для каждого живого врага генерируется случайное направление движения от нуля до трех, соответствующее сдвигу по четырем сторонам света. Вычисляются новые координаты, и если они выходят за границы поля, враг пропускает ход. Если на новой клетке находится игрок, враг атакует его, нанося урон, и не перемещается на эту клетку. Если клетка свободна и не занята другим врагом, враг перемещается на новое место, при этом старая клетка очищается от флага врага, а новая помечается.

Метод draw отвечает за отображение текущего состояния игры на экране. Он очищает консоль и построчно выводит игровое поле, где символом P

обозначается игрок, символом E обозначаются враги, а точкой обозначаются пустые клетки. После отображения поля выводится текущее значение здоровья игрока. Метод allEnemiesDead является вспомогательным и возвращает true, если вектор enemies пуст, что сигнализирует о победе игрока.

Программа также содержит файл main.cpp, в котором реализована основная логика игрового процесса. В функции main создается игровое поле размером десять на десять клеток, создается игрок с шестью единицами здоровья и уроном два. Игрок размещается в левом верхнем углу поля с координатами ноль ноль. На поле помещаются два врага также с шестью здоровья и уроном два в координатах три пять и шесть семь. Затем запускается игровой цикл, который выполняется, пока игрок жив. На каждой итерации отображается текущее состояние поля, проверяется, не убиты ли все враги, и если убиты, выводится сообщение о победе и цикл прерывается. Затем программа запрашивает у пользователя направление перемещения с клавиатуры, обрабатывает введенный символ и вызывает соответствующий метод movePlayer. После хода игрока вызывается метод enemyTurn, реализующий ход врагов, и снова проверяется состояние игрока. Если игрок погиб, выводится сообщение о конце игры и программа завершается.

ВЗАИМОДЕЙСТВИЕ КЛАССОВ

Взаимодействие классов в программе построено вокруг центрального класса GameField, который координирует работу всех остальных объектов. Игровое поле хранит вектор объектов Cell, каждый из которых с помощью логических флагов сообщает о наличии на конкретной клетке игрока или врага. Класс Player и класс Enemy никак не взаимодействуют друг с другом напрямую, вся коммуникация между ними происходит через посредничество GameField. Когда игрок пытается переместиться на клетку, GameField проверяет состояние соответствующего объекта Cell с помощью методов containsPlayer и containsEnemy. Если клетка занята врагом, GameField обращается к конкретному объекту Enemy в векторе enemies, вызывает его метод takeDamage

для нанесения урона, а затем проверяет его состояние методом isAlive для решения вопроса об удалении с поля. Аналогично при ходе врагов GameField проверяет наличие игрока на соседних клетках через Cell и при обнаружении вызывает метод takeDamage у объекта Player. Таким образом, классы Cell выступают в роли информаторов о занятости пространства, Player и Enemy хранят свои параметры и предоставляют методы для их изменения, а GameField выступает в роли диспетчера, который опрашивает клетки, извлекает данные об игроке и врагах и организует их взаимодействие в соответствии с игровыми правилами.

ИГРОВОЙ ПРОЦЕСС

Игровой процесс начинается с создания игрового поля размером десять на десять клеток, после чего на поле размещаются игрок в левом верхнем углу и два врага в заданных координатах. Игровой цикл продолжается до тех пор, пока игрок жив и не уничтожены все враги. На каждом ходу сначала отображается текущее состояние поля, где символом P обозначается игрок, символом E обозначаются враги, а точками пустые клетки, также выводится текущее значение здоровья игрока. Затем программа ожидает ввода пользователя, который может нажать клавиши W, A, S или D для перемещения игрока вверх, влево, вниз или вправо соответственно. Если игрок пытается переместиться на пустую клетку, он просто меняет свою позицию на поле. Если игрок пытается переместиться на клетку с врагом, то происходит атака здоровье врага уменьшается на значение урона игрока, и если здоровье врага становится равным нулю, враг удаляется с поля, при этом игрок остается на своем месте. После завершения хода игрока наступает ход врагов, управляемых компьютером, каждый враг случайным образом выбирает направление для перемещения, и если соседняя клетка свободна, враг перемещается на нее, если же на соседней клетке находится игрок, враг атакует его, уменьшая здоровье игрока на свою величину урона, при этом враг не перемещается на клетку с игроком. Игра завершается победой, когда все враги уничтожены и вектор enemies становится пустым, либо поражением, когда здоровье игрока падает до

нуля и метод isAlive возвращает false. В конце игры выводится соответствующее сообщение о победе или проигрыше, после чего программа завершает свою работу.

UML диаграмма: