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 классов.

Студент гр. 3385 _____ Майдуров А.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы.

Изучить классы в объектно-ориентированном программировании на языке C++. Создать классы кораблей, менеджера кораблей и игрового поля, а также методы для работы с ними. Это является первым шагом в реализации проекта первой игры на языке программирования C++.

Задание.

a) Создать класс корабля, который будет размещаться на игровом поле. Корабль может иметь длину от 1 до 4, а также может быть расположен вертикально или горизонтально. Каждый сегмент корабля может иметь три различных состояния: целый, поврежден, уничтожен. Изначально у корабля все сегменты целые. При нанесении 1 урона по сегменту, он становится поврежденным, а при нанесении 2 урона по сегменту, уничтоженным. Также добавить методы для взаимодействия с кораблем.

b) Создать класс менеджера кораблей, хранящий информацию о кораблях. Данный класс в конструкторе принимает количество кораблей и их размеры, которые нужно расставить на поле.

c) Создать класс игрового поля, которое в конструкторе принимает размеры. У поля должен быть метод, принимающий корабль, координаты, на которые нужно поставить, и его ориентацию на поле. Корабли на поле не могут соприкасаться или пересекаться. Для игрового поля добавить методы для указания того, какая клетка атакуется. При попадании в сегмент корабля изменения должны отображаться в менеджере кораблей. Каждая клетка игрового поля имеет три статуса:

i) неизвестно (изначально вражеское поле полностью неизвестно),

ii) пустая (если на клетке ничего нет)

iii) корабль (если в клетке находится один из сегментов корабля).

Для класса игрового поля также необходимо реализовать конструкторы копирования и перемещения, а также соответствующие им операторы присваивания.

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 копирования нужно выполнять глубокое копирование

• При реализации перемещения, не должно быть лишнего копирования

• При выделении памяти делайте проверку на переданные значения

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

Выполнение работы

Класс перечисление SegmentStatus

Перечисление SegmentStatus может представлять 3 разных значения - информация о состоянии сегментов.

Unharmed=2 - сегмент невредим

Harmed=1 - сегмент поврежден

Destriyed=0 - сегмент уничтожен

Класс Segment

Класс Segment - сегмент корабля. Предоставляет основной функционал для работы с сегментами.

Поля класса Segment:

status(тип *SegmentStatus*) - хранит состояние сегмента.

Методы класса Segment:

Segment() - конструктор класса. Не принимает аргументов, т.к. нет смысла создавать уже поврежденный сегмент. Соответственно сегмент всегда создается целым.

void *takeDamage*(int *damage_value*) - наносит урон сегменту.

SegmentStatus *getStatus*() -возвращает статус сегмента.

void *printStatus*() - выводит статус сегмента соответсвующей строкой в терминал.

Класс перечисление ShipOrientation

Перечисление ShipOrientation может представлять 2 разных значения - ориентация корабля.

Horizontal - горизонтальная

Vertical - вертикальная

Класс Ship

Класс моделирует объект корабля обладающий определенными характеристиками, которые хранятся в его полях. Также в классе предоставлен функционал для взаимодействия с кораблями.

Поля класса Ship:

size(тип *unsigned*) - хранит размер корабля.

death_status(тип *bool*) - хранит информацию о том уничтожен корабль полностью или нет. Но при этом поле может хранить неактуальную информацию. Это обусловленно функционалом некоторых полезных методов класса. В классе есть метод обновляющий эту переменную. Также переменная объявлена с ключевым словом mutable это сделано для того, чтобы вышеупомянутые методы могли выдавать актуальную информацию о константных объектах класса.

segs(тип *std::vector*) - вектор хранящий сегменты, из которых корабль состоит.

pos(тип *std::array<unsigned, 2>*) - хранит координаты корабля. Оформлено в виде массива, чтобы впоследствии можно использовать как ключ в ассоциативном массиве. Также массив размера 2, в отличии от вектора, подчеркивает, что координаты должно быть ровно 2.

orientation(тип *ShipOrientation*) - хранит ориентацию корабля на поле.

Методы класса Ship:

Ship(unsigned *size*) - конструктор класса. Создает сегменты в количестве *size.* Поле *death_status* устанавливается в *false, pos* устанавливается в 0, 0, *orientation* устанавливается в *Horizontal*

bool *wasItDeath*() - возвращает неактуальную информацию о поле death status.

bool *isItDeath*() - обновляет информацию о поле *death_status* и возращает ее.

bool *wasItDestroyedJustNow*() - обновляет информацию о поле *death_status*. Затем сравнивает его с предыдущим значением. Т.о. если корабль был уничтожен только что вернет *true*, в противном случае *false.*

void *setOrientation*(ShipOrientation *orientation*) - сеттер поля *orientation*

void *setPos*(unsigned *x*, unsigned *y*) - сеттер координат

ShipOrientation *getOrientation*() - геттер ориентации корабля

void *printShipInfo*() - выводит всю основную информацию о корабле в терминал.

unsigned *getX*() - геттер x координаты

unsigned *getY*() - геттер y координаты

unsigned *getSize*() - геттер размера корабля.

Segment *&operator*[](unsigned *index*) - переопределение оператора []. Возвращает сегмент по индексу.

Класс ShipManager

Класс создает и хранит корабли, и соответственно управляет временем их жизни.

Поля класса ShipManager:

number_of_ships(тип *unsigned*) - хранит количество кораблей.

ships(тип *std::vector<Ships>*) - хранит корабли

Методы класса ShipManager:

ShipManager(unsigned *size_of_vector,* const std::vector<unsigned> *&sizes*) - конструктор класса. Создает вектор кораблей,

инициализированных от чисел из *sizes*. Размер вектора, предположительно, *size_of_vector*. Вектор принимается по ссылке во избежание лишнего копирования.

Ship &*getUnplacedShip*(unsigned *n*) - возвращает корабль хранящийся по индексу *n*.

unsigned *getNumberOfShips*() - геттер количества кораблей.

Класс перечисление CellStatus

Перечисление CellStatus может представлять 3 разных значения - информация о клетке поля.

Hidden - клетка пока что скрыта

Empty - клетка пустая

Occupied - клетка занята сегментом какого-то корабля

Класс CellOfField

Класс моделирует клетку игрового поля.

Поля класса CellOfField:

status(тип *CellStatus*) - статус клетки.

near_of_ship(тип *bool*) - флаг показывающий есть ли рядом с этой клеткой или в самой клетке сегменты. Понадобится для отслеживания пересечения или касания кораблей.

segment_ptr(тип *Segmnet**) - хранит указатель на сегмент, который находится в этой клетке(если такой есть, иначе *nullptr*).

Методы класса CellOfFiled:

CellOfField(CellStatus *status*) - конструктор класса. Создает клетку со статусом *status*. *near_of_ship* устанавливается в *false,* а *segment_ptr* в *nullptr*. Выбор статуса клетки пригодится при выборе того кому принадлежит поле - игроку или его противнику.

CellStatus *getStatus*() - геттер статуса клетки.

void *setStatus*(CellStatus *status*) - сеттер статуса клетки.

Segment **getSegment*() - геттер сегмента(по указателю). Также имеет обычную и константые версии.

void *setSegment*(Segment **segment_ptr*) - сеттер сегмента.

bool *isNearOfShip*() - геттер *near_of_ship*.

void *shipIsNear*() - сеттер *near_of_ship*. Не принимает аргументов, т.к. случай, когда сегмент пропадает с клетки невозможен. Исключение составляет конец игры. Поэтому данный сеттер всегда устанавливает поле в *true.*

Класс перечисление FieldStatus

Перечисление FieldStatus может представлять 2 разных значения - информация о принадлежности поля игроку.

EnemyField - поле принадлежит противнику

PlayerField - поле принадлежит игроку

Класс BattleField

Класс представляющий игровое поле, игровое поле состоит из клеток поля и хранит расставленные по нему корабли. Также класс предоставляет функционал для взаимодействия между полем и другими объектами.

Поля класса BattleField:

width(тип *unsigned*) - ширина поля.

height(тип *unsigned*) - высота поля.

status(тип *FieldStatus*) - статус поля - вражеское или нет .

placed_ships(тип *std::map<std::array<unsigned, 2>, const Ship **>) - ассоциативный массив кораблей, где ключ - координаты. Сделан для

оптимизации некоторых методов и для возможности пользоваться информацией о кораблях - но не менять ее.

field(тип *FieldStatus std::vector<std::vector<CellOfField> >*) - двумерный вектор клеток - по сути само поле.

Методы класса BattleField:

BattleField(unsigned *width*, unsigned *height*, FieldStatus *status*) - создает поле размером *width* на *height,* каждая клетка, которого инициализируется в зависимости от *status*. Если поле - вражеское, то клетки создается скрытыми, если поле принадлежит игроку - то пустыми.

BattleField(BattleField const *&other*) - конструктор копирования

BattleField *&operator*=(const BattleField &other) - оператор присваивания копирования

BattleField(BattleField *&&moved*) - конструктор перемещения

BattleField *&operator*=(BattleField *&&moved*) - оператор присваивания копирования

void *placeShip*(Ship *&ship*, unsigned *x*, unsigned *y*, ShipOrientation *orientation*) - ставит корабль на поле, на клетку с координатами *x*, *y* с заданной ориентацией, обновляя его поля. Первый сегмент корабля закрепляется за указанной клеткой, остальные закрепляются за последующими клетками поля, расположенными ниже или правее стартовой(зависит от ориентации). Добавляет корабль в ассоциативный массив. Отслеживает пересечение или касание кораблей и выбрасывает исключение в таких случаях.

void *attackCell*(unsigned *x*, unsigned *y*, int *damage_value*) - атакует клетку поля с заданными координатами, нанося сегменту, если он там есть, *damage_value* урона. Также обновляет статус клетки: скрытые клетки меняют статус на *Empty* или *Occupied*

bool *areAllShipsDestroyed*() - с помощью ассоциативного массива узнает, все ли корабли на поле уничтожены.

bool *wasAnyShipDestroyedJustNow*(unsigned *x*, unsigned *y*) - *x, y* - предположительно клетка, по которой недавно была совершена атака. С помощью этих данных, ассоциативного массива и метода класса Ship *wasItDestroyedJustNow* выясняет был ли после этой атаки уничтожен какой-либо корабль.

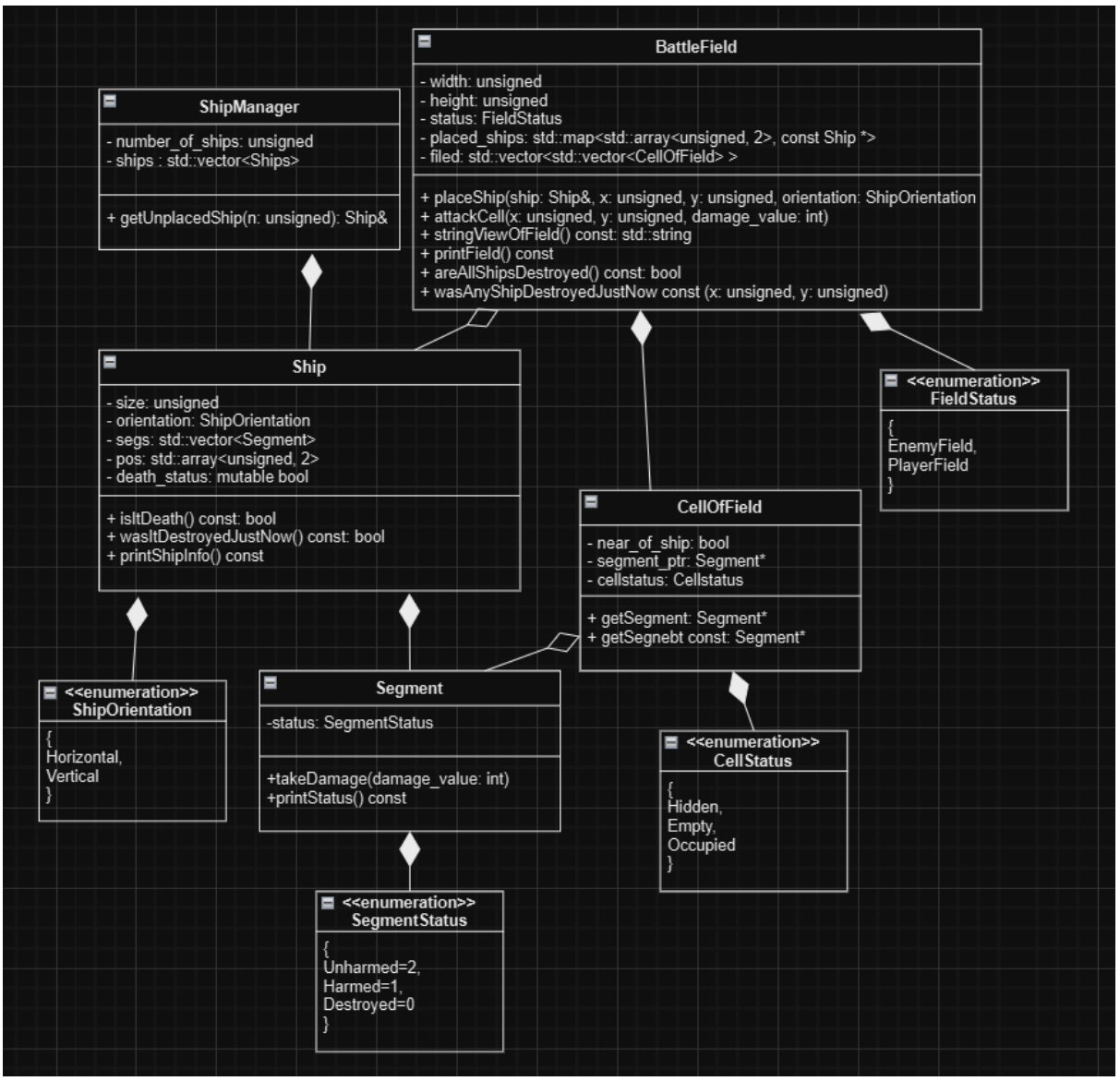
unsigned *getWidth*() - геттер ширины

unsigned getHeight() - геттер высоты

std::string *stringViewOfField*() - возвращает поле в виде строки - нужно для отрисовки поля.

void *printField()* - отрисовывает поле в терминале.

UML-диаграмма классов



Выводы.

Было изучено объектно-ориентированное программирование в C++. Созданы классы кораблей, менеджера кораблей и игрового поля, а также методы для работы с ними. Сделан первый шаг в реализации проекта игры на языке программирования C++.