Course project

CSC 2110. Version 2021-0w-1.1

# Updates in version 1.1

* One learning objective was removed.
* Bag’s data definition must be an array of pointers (Go to Hero class).
* A mutator for defense power was added to Hero class. (Go to Hero class).
* The operator overloading was added to the Hero class diagram. (Go to Hero class).
* The operator overloading in Monster class was removed.
* Add “decrease retreat count” in the retreat action in the pseudocode of the game. (Go to Game’s business rules).
* Add more information about “use item”, “pick up item”, and “move” processes. (Go to Game’s business rules).
* Item’s business rules were added. (Go to Item’s business rules).
* The definition of the roomMonster and roomItem’s data type was corrected. Also, the const keyword was added to 2 accessors. (Go to Room class).
* The returned data type of getName was corrected (Go to Room class).
* A recommendation about how rooms are connected was added. (Go to Room’s business rules).
* Another section about possible new classes and functions was added. (Go to Other classes and functions inside a class).

# Information

**Due date:** Sunday, April 18, 2021

**Points:** 20

**Learning objectives:**

* Understand user defined classes and be able to apply them in C++ programs.
* Understand inheritance and composition and be able to apply them in C++ classes.
* Understand pointer data type and become aware of the features of classes with

pointers member variables.

* Understand polymorphism conceptually and be able to create C++ classes using

polymorphism.

* Understand operators overloading and be able to apply them in C++ programs.
* Use data structures (linked list) and standard template library (vector type).
* Apply problem solving strategies to analyze, design, implement, document, and test

a software application.

# Instructions

## Project

Create a Visual Studio C++ project. Name it using the following format:

firstname\_lastname\_project

(e.g. my name is Alberto Cahdez, then the project name is alberto\_cahdez\_project).

## Game

The objective of the project is to implement a simple game. In this game, the player moves a character (hero) through multiple rooms to get to the exit. In some of the rooms, the hero may find other characters (monsters) and items. The hero will battle the enemies and take the times.

The player wins when the hero reaches the exit.

The player loses when hero’s health points (HP) are zero (0).

## Character abstract class

|  |
| --- |
| Character |
| #hp: int  #attackPower: int |
| +setHP(int): void  +getHP() const: int  +setAttackPower(int): void  +getAttackPower() const: int |

Character class is an abstract class i.e., all functions are pure virtual functions.

### Character’s business class

hp cannot be less than zero.

## Hero class

|  |
| --- |
| Hero |
| -defensePower: int  +bag: Item\*[3]  -retreatCount: int |
| +getDefensePower() const : int  +decreaseDefensePower(): void  +setDefensePower(int): void  +getRetreatCount() const: int  +decreaseRetreatCount(): void  +anyItems() const: bool  +isAlive() const: bool  +operator\*(Monster& m): int  +Hero(int,int,int,int) |

Hero class is a derived class from Character class.

The arguments of the Hero’s constructor are hp, attack power, defense power, and retreat count. This constructor initializes the corresponding member variables and the bag with 3 null pointers.

The decreaseRetreatCount function decreases retreatCount by 1. The decreaseDefensePower function decreases defensePower by 1. The anyItems function returns false when all pointers in bag are NULL; otherwise, returns true. The isAlive function returns true when hp is greater than 0; otherwise, returns false.

### Hero’s business rules

retreatCount cannot be less than zero.

defensePower cannot be less than 1.

Overload the multiplication operator when multiplying Hero \* Monster, and use the following pseudocode

Monster.hp = Monster.hp - hero.attackPower < 0 ? 0 : Monster.hp-hero.attackPower

hero.hp = hero.hp - Monster.attackPower/hero.defensePower

Monster.decreaseAttackPower

hero.decreaseDefensePower

if hero.hp == 0 // hero died

return -1

else if Monster.hp == 0 // monster died

return 1

else // no character died

return 0

## Monster class

|  |
| --- |
| Monster |
| +decreaseAttackPower(): void  +Monster(int,int) |

Monster class is a derived class from Character class.

The arguments of the Monster’s constructor are hp, and attack power.

The decreaseAttackPower function decreases attackPower by 1.

### Monster’s business rules

attackPower cannot be less than 1.

## Item class

|  |
| --- |
| Item |
| -type: int // 0-hp, 1-defense  -value: int |
| +isHP() const: bool  +isDefense() const: bool  +getValue() const: int  +Item(int,int) |

The arguments of the Item’s constructor are type, and value.

### Item’s business rules

When a 0-type item is used, the hero’s hp is increased by the item’s value.

When a 1-type item is used, the hero’s defense is increased by the item’s value.

## Room class

|  |
| --- |
| Room |
| -name: string  -east: Room\*  -west: Room\*  -north: Room\*  -south: Room\*  -roomMonster: Monster\*  -roomItem: Item\*  -exit: bool |
| +setName(string): void  +getName() const: string  +setEastRoom(Room\*): void  +setWestRoom(Room\*): void  +setNorthRoom(Room\*): void  +setSouthRoom(Room\*): void  +getEastRoom() const: Room\*  +getWestRoom() const: Room\*  +getNorthRoom() const: Room\*  +getSouthRoom() const: Room\*  +setRoomMonster(Monster\*): void  +getRoomMonster() const: Monster\*  +setRoomItem(Item\*): void  +getRoomItem() const: Item\*  +isExit(): bool  +Room(string, Room\*,Room\*,Room\*,Room\*,Monster\*,Item\*)  +Room(string, Room\*,Room\*,Room\*,Room\*,Monster\*)  +Room(string, Room\*,Room\*,Room\*,Room\*,Item\*)  +Room(string, Room\*,Room\*,Room\*,Room\*,bool) |

### Room’s business rules

A map is a set of rooms connected through the pointers (east, west, north, and south).

A map does not have cycles. As a cycle example, see the left figure below where A, B, C, and D are connected rooms. As an example without cycles, see the right figure below.

C

D

C

D

B

A

B

A

Map without a cycle

Map with a cycle

All rooms in the map must be connected to at least another room. It is recommended that if room X is connected to room Y, room Y is connected to the room X in the opposite direction i.e. if X.east = Y, then Y.west = X.

A start room is a room which has no monster and no items. The game begins with the hero in the start room. The start room cannot be an exit. There is only one start room per game.

An exit room is a room which has no monster and no items. The game ends if the hero reaches the exit room. The exit room cannot be a start room. There is only one exit room per game.

All other rooms configurations are:

* Room with no monster and no item.
* Room with a monster only.
  + In this room, the hero will battle the monster unless the hero retreats.
* Room with an item only.
  + In this room, the hero can pick up the item.
* Room with a monster and item.
  + In this room, the item can only be picked if the monster is defeated.

## Game class

|  |
| --- |
| Game |
| -currentRoom: Room\*  -HeroName: Hero |
| +run(): void  -createMap(): void  Game() |

The Game’s constructor will initialize the Hero characteristics and create the map of the game.

The createMap function will define the monsters, items, and rooms of the game with dynamic variables. The rooms must be connected, and the monsters and items will be located in rooms. The current room must be positioned in the start room.

### Game’s business rules

The map must have at least 7 rooms, 2 monsters, and 3 items.

The run function will start the game following this pseudocode:

*PreviousRoom = null*

*While (hero.alive and !currentRoom.isExit){*

*Show room information (name, monster’s info, item’s info).*

*If Retreat? { // ask user if wants to retreat, assuming retreat count >0*

*Move to previousRoom*

*Decrease retreat count*

*}*

*If hero.gotItems? {*

*Want to use it? {*

*Execute “use an item” process*

*}*

*}*

*if currentRoom.gotMonster? {*

*battle! // repeat battle until monster or hero wins*

*}*

*if hero.alive? {*

*if currentRoom.gotItem? {*

*Want to pick it up? {*

*execute “pick up item” process*

*}*

*}*

*execute “move” process (and update previousRoom pointer)*

*}*

*}*

*if currentRoom.isExit? {*

*display "Congratulations"*

*} else {*

*display “Game over”*

*}*

The “use an item” process will display the items in the hero’s bag and ask the player to choose one to use. Only one item can be used in this iteration. A HP-item will increase hero’s hp by item’s value. A defense-item will increase hero’s defense power by item’s value. Once an item is used, it will be removed from memory.

The “pick up item” process will display the contents in the hero’s bag and ask the player to select one position to store the new item. If the selected position is not empty, delete the current item from memory, then store new item’s reference in that position. Once the item is stored in the bag, remove the reference to it in the room.

The “move” process will display of possible movements (east, west, north, south) and ask the user to select one. Also, the previous room pointer will be updated with the current room before moving to the new position.

### Class files

Classes must be separated in a .h file (header), and a .cpp (source code).

### Other classes and functions inside a class

You can create other class that help you in the development. These new classes cannot substitute (completely or partially) the classes which are already described on this document.

You can create other functions inside a class that help you in the development. These new functions cannot substitute (completely or partially) the functions which are already described on this document.

# Personalize your game

Add at least one unique feature to your game. Examples:

* Add another type of Item to the game e.g.
  + a retreat token to increase retreatCount.
  + an item to increase attack power
* Add 1 or more subclasses of monster.
* “Draw” the map and the current position of the hero.
* The monster uses a battle cry when attacking.
* Part of the hero’s story is revealed as the hero gets closer to the exit.
* Other features which do not change the core of the game. If needed, we can discuss your ideas in office hours.

# The main program

In the main program create a game object and execute run.

# Submission

Include the appropriate files in the VS project to compile and execute it.

Compress your project folder in a ZIP folder and submit it on Canvas.