**Assessment Cover Sheet**

This Assessment Cover Sheet is only to be attached to

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ASSESSMENT DETAILS** | | | | | | |
| Unit title | | Data Structure and Pattern | Tutorial /Lab Group | 1 | | Office use only |
| Unit code | | COS30008 | Due date | 18 Nov 2022 | |  |
| Name of lecturer/tutor | | Dr Mark Tee Kit Tsun | | | |  |
| Assignment title | | Programming Project | | | | Faculty or school date stamp |
| **STUDENT(S) DETAILS** | | | | | | |
| Student Name(s) | | | | | Student ID Number(s) | |
| (1) | Alex Ngie Guan Ming | | | | 102765770 | |
| (2) |  | | | |  | |
| (3) |  | | | |  | |
| (4) |  | | | |  | |
| (5) |  | | | |  | |

hard copy submission of assessments.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DECLARATION AND STATEMENT OF AUTHORSHIP** | | | | |
| 1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment. 2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made. 3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/tutor concerned. 4. I/we have not previously submitted this work for this or any other course/unit. 5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.   I/we understand that:   1. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited. | | | |
| **Student signature/s**  I/we declare that I/we have read and understood the declaration and statement of authorship. | | | |
| (1) | Alex | (4) |  |
| (2) |  | (5) |  |
| (3) |  | (6) |  |

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **https://www.swinburne.edu.my/current-students/manage-course/exams-results-assessment**

Copies of this form can be downloaded from the Student Forms web page at **https://www.swinburne.edu.my/current-students/manage-course/exams-results-assessment/how-to-submit-work.php**

Contents

[Introduction 4](#_Toc119673367)

[Youtube Video Link 4](#_Toc119673368)

[Inheritance and Derived Class 5](#_Toc119673369)

[Task Description 5](#_Toc119673370)

[Concept 5](#_Toc119673371)

[Implementation and Output 6](#_Toc119673372)

[Discussion 9](#_Toc119673373)

[Array 10](#_Toc119673374)

[Task Description 10](#_Toc119673375)

[Concept 10](#_Toc119673376)

[Implementation and Output 10](#_Toc119673377)

[Discussion 11](#_Toc119673378)

[Struct 12](#_Toc119673379)

[Task Description 12](#_Toc119673380)

[Concept 12](#_Toc119673381)

[Implementation and Output 13](#_Toc119673382)

[Discussion 15](#_Toc119673383)

[Singly Linked-List 16](#_Toc119673384)

[Task Description 16](#_Toc119673385)

[Concept 16](#_Toc119673386)

[Implementation and Output 18](#_Toc119673387)

[Discussion 21](#_Toc119673388)

[Doubly Linked-List 22](#_Toc119673389)

[Task Description 22](#_Toc119673390)

[Concept 22](#_Toc119673391)

[Implementation and Output 23](#_Toc119673392)

[Discussion 24](#_Toc119673393)

[Stack 25](#_Toc119673394)

[Task Description 25](#_Toc119673395)

[Concept 25](#_Toc119673396)

[Implementation and Output 25](#_Toc119673397)

[Discussion 27](#_Toc119673398)

[Queue 28](#_Toc119673399)

[Task Description 28](#_Toc119673400)

[Concept 28](#_Toc119673401)

[Implementation and Output 30](#_Toc119673402)

[Discussion 31](#_Toc119673403)

[Iterator 32](#_Toc119673404)

[Task Description 32](#_Toc119673405)

[Concept 32](#_Toc119673406)

[Implementation and Output 32](#_Toc119673407)

[Discussion 33](#_Toc119673408)

[Polymorphism 34](#_Toc119673409)

[Task Description 34](#_Toc119673410)

[Concept 34](#_Toc119673411)

[Implementation and Output 35](#_Toc119673412)

[Discussion 36](#_Toc119673413)

[Appendix 37](#_Toc119673414)

[Player.h 37](#_Toc119673415)

[Food.h 38](#_Toc119673416)

[FoodList.h 38](#_Toc119673417)

[Ingredient.h 39](#_Toc119673418)

[Inventory.h 39](#_Toc119673419)

[Monster.h 40](#_Toc119673420)

[MonsterList.h 40](#_Toc119673421)

[NPC.h 41](#_Toc119673422)

[NPCList.h 41](#_Toc119673423)

[Player.h 42](#_Toc119673424)

[Main.cpp 43](#_Toc119673425)

[Character.cpp 48](#_Toc119673426)

[Food.cpp 49](#_Toc119673427)

[FoodList.cpp 50](#_Toc119673428)

[Ingredient.cpp 51](#_Toc119673429)

[Inventory.cpp 52](#_Toc119673430)

[Monster.cpp 53](#_Toc119673431)

[MonsterList.cpp 55](#_Toc119673432)

[NPC.cpp 56](#_Toc119673433)

[NPCList.cpp 57](#_Toc119673434)

[Player.cpp 60](#_Toc119673435)

# Introduction

In this game, player allowed to select 3 gaming mode that’s easy, normal and hard. Both of 3 start from different initial hp and gold. Easy is HP200 with 1000gold, Normal Hp150 with 500gold and Hard start with HP100 and 200gold. Once player select the mode, game will start. In this game, player going to collect ingredients from monster and used it to cook for customer. After that, earn gold from each customer and upgrade the status of player to make adventure more easier.

Inside this game, player need go to adventure, defeated the monster to get the materials drop from the monster. Every monster has specified what materials they have and list out. Number of item drop for each materials is random, from range 1 to 4. Each type of monster only can defeat by player once per time in adventure. If want to challenge same monster, need to go back restaurant, so the monster state will be reset. Beside this, during the adventure, if player defeat by monster, it will directly return back to the restaurant, life – 1 and recover hp to full. Type in “Exit” will manually back to restaurant.

Once player already collect materials from monster, player can select customer they wish to serve by entering the customer’s name. player able to view customer in list or one by one. When player enter the name of customer wish to serve, if the ingredients is enough for that food, serving will success and current gold increase. After that server customer will be removed and add on the new customer. Other than that, if ingredient is not enough to produce the order of customer, message will pop up and list out which ingredient is no enuf. Player is able to check all the food recipe inside the game.

In this game design, don’t have the ending game, player is able to play continuously until quit the game manually. But once the life is used up during the adventure, the game is considered as GAME OVER.

# Youtube Video Link

<https://www.youtube.com/watch?v=R3CU7P-sggQ>

# Inheritance and Derived Class

## Task Description

Derived class (NPC, Player, Monster) will inherit all the function and variable from the base class (Character) to reduce the code redundancy.

## Concept

Diagram

Description automatically generated

Inheritance is applied in this session. Class Player, Monster and NPC is derived class and Class Character is base class. The reason I’m applying the inheritance for those classes is because all of them have the similar variable and action to take. To avoid the code redundancy and rewriting same code in every class, inheritance allow derived class to inherit the code from the base class, and can modify or just simply add more functionality, calculation or etc by implementing the function.

Example shown in the diagram (output) cmd1, cmd2 and cmd3, printing status of them by function of character. They have the same content to share, but all of them have other details.

## Implementation and Output

**Character.h (base class)**

Graphical user interface, text, application, email

Description automatically generated

**Monster.h (derived class)**

Graphical user interface, text, application, email

Description automatically generated

**NPC.h(derived class)**

Graphical user interface, text, application, email

Description automatically generated

**NPC.cpp**

Text

Description automatically generated

**Player.h (derived class)**

Graphical user interface, text, application

Description automatically generated

**Output**

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated with low confidence

## Discussion

No error or issues encountered at this part. But tried to inherit the class in different type, public and private. Public allowed derived class directly to call the function from the base class once the class is declared. But derived as private class, it cannot directly call the function by the class declared, must create another function in the derived class and call the inherited class inside the function created.

# Array

## Task Description

Initialize the ingredient class with 20 ingredients predefined in the class.

## Concept

Diagram

Description automatically generated

This is an array to store the information of ingredient, after that retrieve anytime when necessary to use it. In this array store 20 type of ingredient.

Reason that I use array to store those information is because this is only a simple information. No need complicated design for this class.

## Implementation and Output

**Ingredient.h**

A picture containing text

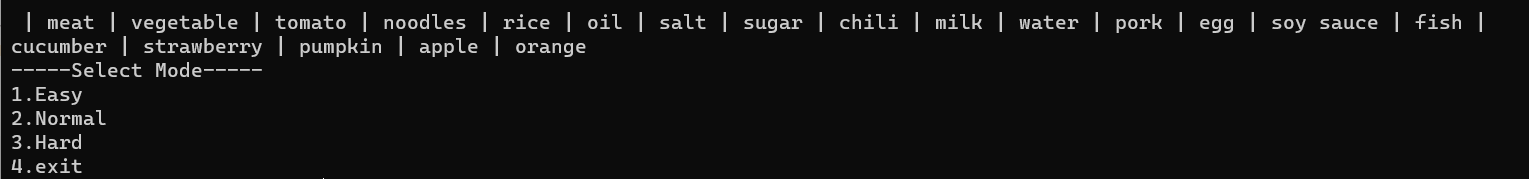
Description automatically generated

**Ingredient.cpp**

Graphical user interface, text, application

Description automatically generated

**Output**



## Discussion

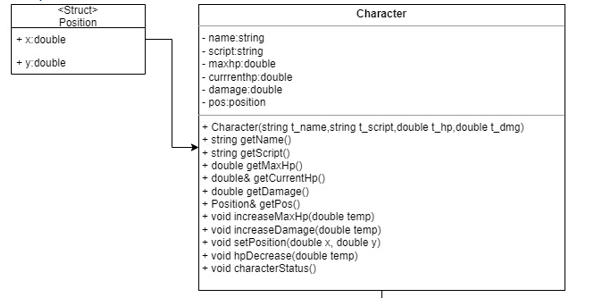
Seem like c++ doesn’t have return type lie “string[] getName()”, so I implemented operator for this class to solve this problem.

# Struct

## Task Description

Group relevel variable together

## Concept



X and y are represented to the coordinate of the character in the map. According to their relationship, add it as a member of Position struct. By using this method, inside class no need to declare a lot of variables representing to position. Just need to declare Position struct inside our class and call the x and y inside the position struct. This allows us more convenience to manage our coding in the future, code is also more readable.

## Implementation and Output

**Character.h**A picture containing background pattern

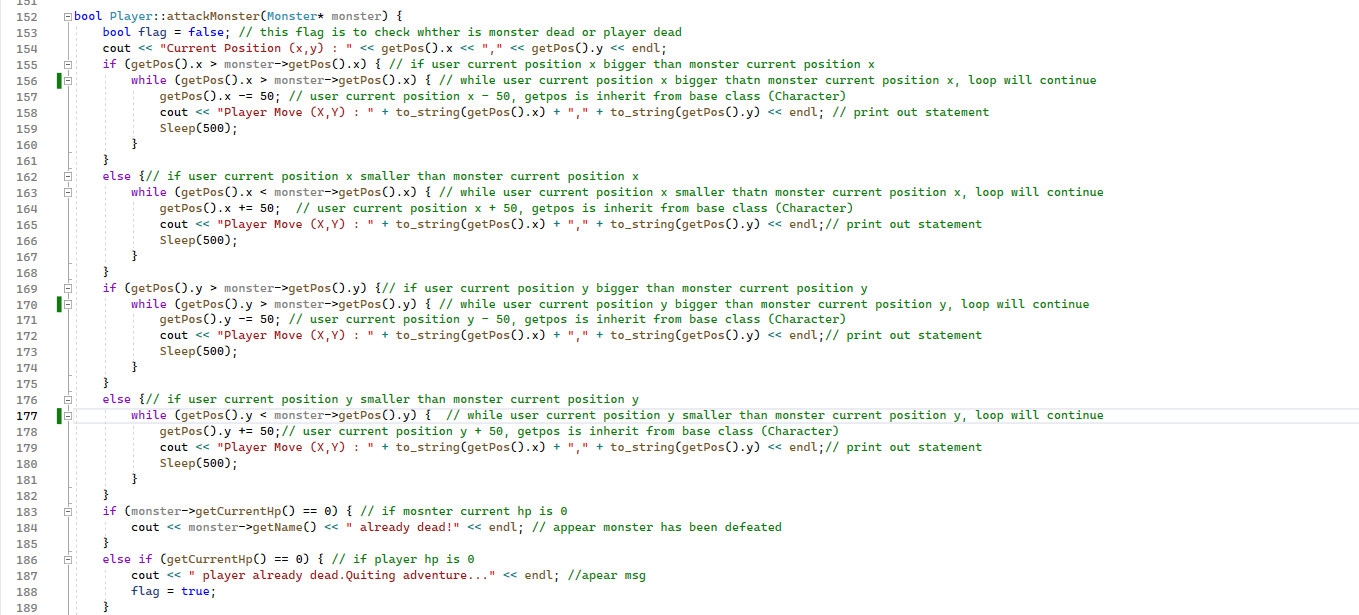
Description automatically generated

**Character.cpp**Graphical user interface, text, application

Description automatically generatedA picture containing company name

Description automatically generatedText

Description automatically generated



Text

Description automatically generated

**Output**

Text

Description automatically generated

## Discussion

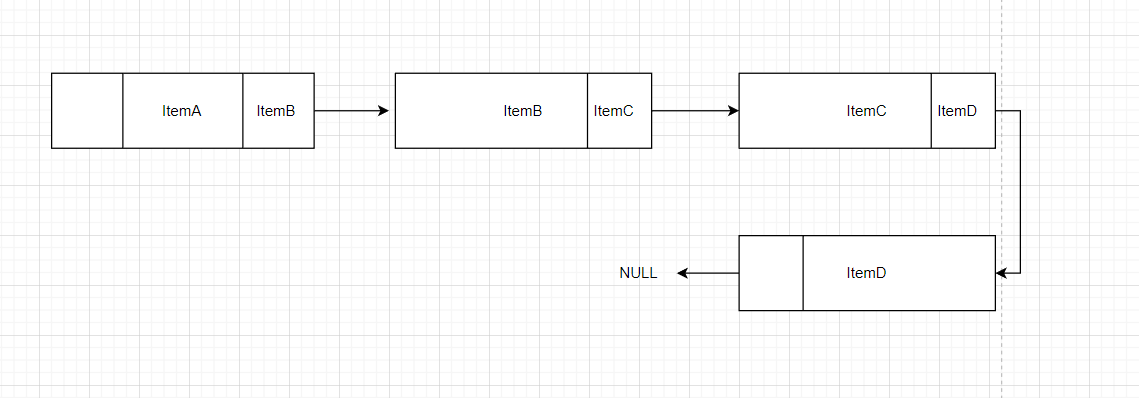
No problem is encountered in this part.

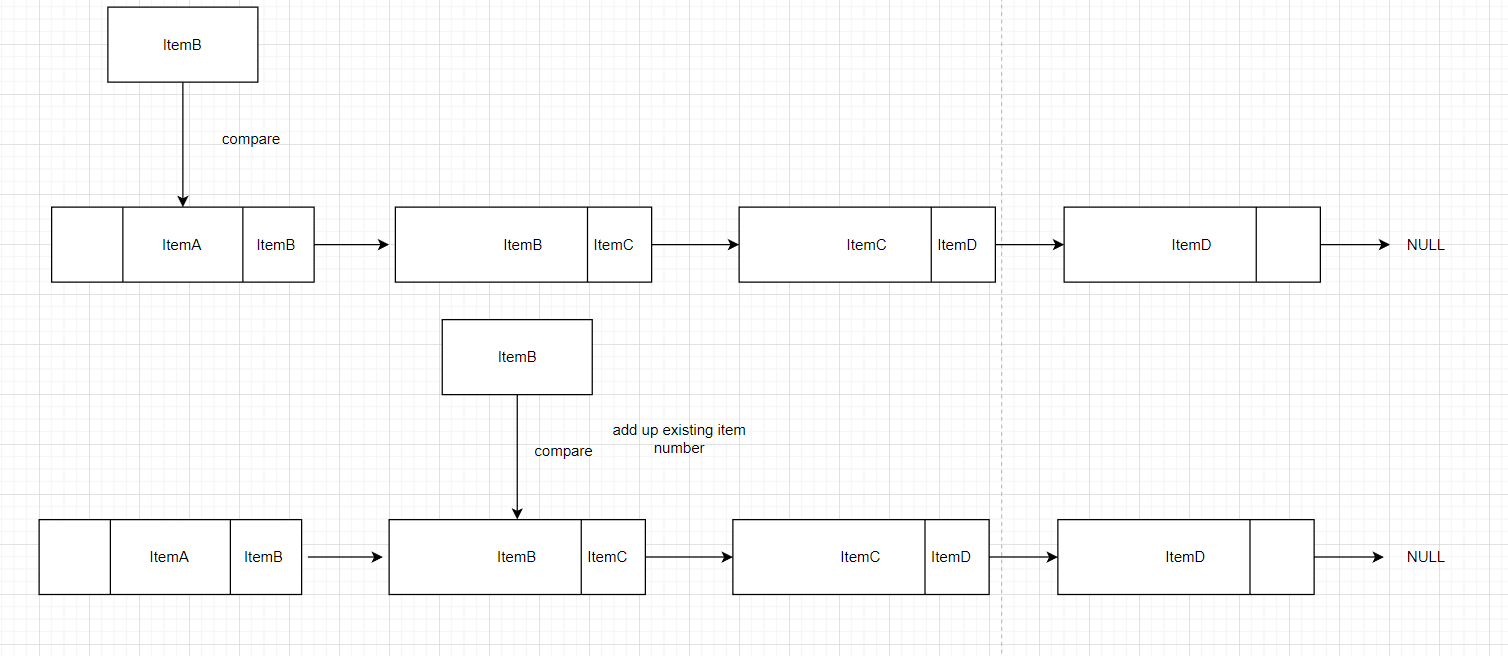
# Singly Linked-List

## Task Description

Store the item in inventory using single link list

## Concept





Chart

Description automatically generated with medium confidence

Chart, box and whisker chart

Description automatically generated

In the inventory in player and monster, just add the new item at the end of the single linked list.

For player, when defeated the monster, the item of monster inventory will add into player inventory. To perform this process, it will pass the item one by one to the inventory of player to search it already exist or not. If existed , add up the total number , if not exist will add one more item at the end of single linked list with it information.

## Implementation and Output

**Inventory.h**

Graphical user interface, text, application, email

Description automatically generated

**Monster.h**

Text

Description automatically generated

**Player.cpp**

Text

Description automatically generated

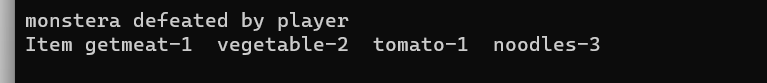
Text

Description automatically generated

**Output**

Text

Description automatically generated



Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

## Discussion

No error encountered in this part

# Doubly Linked-List

## Task Description

Allow player to view customer one by one

## Concept

Calendar

Description automatically generated with low confidence

Doubly linked list applied to allow user to view the customer list one by one. When input “prev”, it will check the previous of current pointer pointed item is null or exists, if exists pointer will move to the previous item, else appear msg to mention user is at the end. Same process when input “next”.

In the actual game, a restaurant theme game is impossible that only have 3 customers in once, so based on this sense, doubly linked list is necessary because it able to move in both sides.

## Implementation and Output

NPCList.h

Graphical user interface, text, application

Description automatically generated

NPCList.cpp

Graphical user interface, text

Description automatically generated

**Output**

Text

Description automatically generated

## Discussion

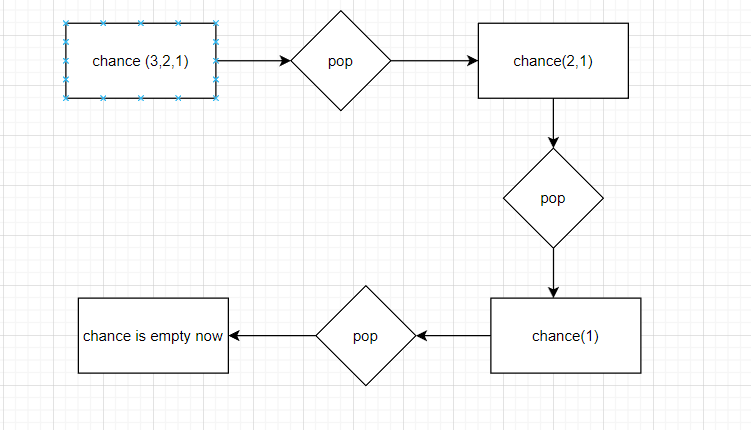
Have to declare another pointer to perform this next and previous action. Directly using head and tail will lost the pointer of head and tail

# Stack

## Task Description

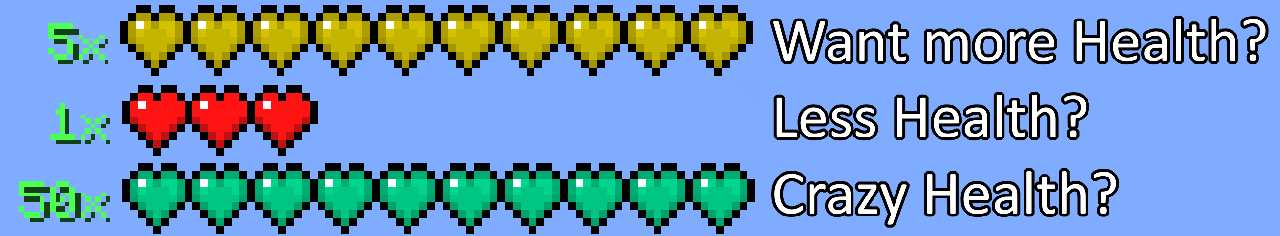
Storing life remaining of the player

## Concept



Picture stack perform

stack always pop up the top item inside it follow the first in first out rule. So in early will add number into the stack according 1 -> 2 -> 3. When player is dead, pop up the top item, that’s 3, and item inside the stack is 1-> 2. When all the item is pop up and stack is empty, it will return back to the main menu.



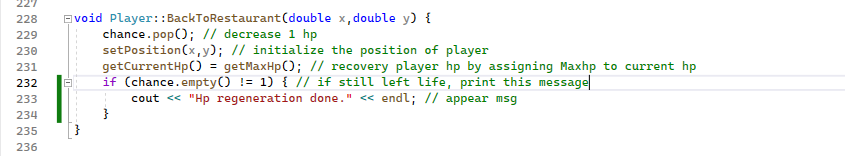
The reason applied this concept is because when decrease the chance of play, we doesn’t need to do any calculation, it doesn’t like player HP, minus depends on the damage received. Just need to simply pop up item from stack. For example at the top, let say we have 5 heart, when chance -1, just need to pop up 1 item from stack and refresh in real time.

## Implementation and Output

A picture containing text

Description automatically generated

code initialize life of player



code pop up 1 life if die

Text

Description automatically generated

Picture start with 2 life

Text

Description automatically generated

Picture die 1 time, life -1

Text

Description automatically generated

Picture left 1 life

Text

Description automatically generated

Picture life -1 , return to main menu

## Discussion

No error encountered in this part

# Queue

## Task Description

Insert ability learn in order into a queue by splitting the input of user

## Concept

A picture containing text, sky

Description automatically generated

Picture create queue

Diagram

Description automatically generated

Picture use queue

Diagram

Description automatically generated

Picture Flow of stack

This is a player status upgrade (hp and damage). First user has to enter a list of string with comma example (1,1,2,2,1) as much as u can. In this process , input enter will split by function and insert into queue. After that pass the queue into UpdateStatus() function and go through it 1 by 1 follow the first in first out rule.

This is considered like a training plan, so the first training in training plan will process first, after that second, third etc. Completely follow the first in first out rule.

## Implementation and Output

Text

Description automatically generated

code Enter ability list Example (1,1,2,2,2)

Text

Description automatically generatedcode Upgrade ability by queue

Text

Description automatically generated

cmd output of ability learn

## Discussion

Function that split a string by delimiter

Javapoint,2011, how to split string in C++? / https://www.javatpoint.com/how-to-split-strings-in-cpp, viewed 14 Dec 2022

# Iterator

## Task Description

Use of iterator to modify the original iterator function

## Concept

Diagram

Description automatically generated

Figure process in operator[] comparing

amending of iterator allow us to modify or implemented what we want. In this case , to prevent out of index error occur and stop my program. Operator[] is modified, if the index pass into is no valid in range, it will auto retrieve back the last item of the array.

## Implementation and Output

Graphical user interface, text, application

Description automatically generated

code declare of operator

Text

Description automatically generated

code operator[] function body

Graphical user interface, text, application

Description automatically generated

code test code in main body

## Discussion

No error encountered in this part.

# Polymorphism

## Task Description

It decides how user back to the restaurant. There are two situations, first is hp become zero when player during adventure, and the other one is type in “exit” keyword during adventure.

## Concept

Diagram

Description automatically generated

method overloading is applied in this part. By parameter passing to function BackToRestaurant(), it will automatically detect which function should be call.

Function overloading allow using same function name in the same class by passing different parameter to process similar action. In this case I applied this method in this function name “BackToRestaurant()". They both perform process back to the restaurant, but situation is different.

First is BackToResturant(double x, double y) with 2 double parameter. This function use when player dead during adventure, hp become zero. It directly set player position x and y to zero. After that recover hp and display appropriate message.

Other BackToRestaurant(string msg) with only 1 parameter. This function use when player type in “exit” in command. After that, a bunch of message will display to simulate player move from the current position back to restaurant. After that recover hp and display appropriated message.

Example in diagram code12 and code 13

## Implementation and Output

Graphical user interface, text, application, email

Description automatically generated

code BackToRestaurant overloadding function

Text

Description automatically generated

code back to restaurant when player died

Text

Description automatically generated

code back to restaurant when user input exit

## Discussion

No error encountered when implementing overloading function.

# Appendix

## Player.h

#pragma once

#include <iostream>

#include <string>

#include <cstdlib>

#include <Windows.h>

using namespace std;

typedef struct {

double x;

double y;

}Position;

class Character { // base class

private:

string name;

string script;

double maxHp;

double currentHp;

double damage;

Position pos;

public:

Character(string t\_name,string t\_script,double t\_hp,double t\_dmg);

string getName();

string getScript();

double getMaxHp();

double& getCurrentHp();

double getDamage();

Position& getPos();

void increaseMaxHp(double temp);

void increaseDamage(double temp);

void setPosition(double x, double y);

void hpDecrease(double temp);

void characterStatus();

};

## Food.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Food {

private:

string name;

double cost;

string ii;

int total;

Food\* next;

public:

Food(string tempName,double tempCost,string temp);

Food\* getNext();

void setNext(Food\* temp);

void ingrediantDetail();

string getII();

double getCost();

string getName();

};

## FoodList.h

#pragma once

#include "Food.h"

class FoodList {

private:

Food\* list;

public:

FoodList();

void add(Food\* temp);

Food\* getFood();

void PrintAll();

Food\* searchIng(string temp);

};

## Ingredient.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Ingredient {

private:

string name[20] = { "meat","vegetable","tomato","noodles","rice","oil","salt","sugar","chili","milk","water",

"pork","egg","soy sauce","fish","cucumber","strawberry","pumpkin","apple","orange" };;

int index;

int maxSize;

public:

Ingredient();

string getName();

void operator++();

void operator--();

string operator[](int position);

bool operator==(Ingredient& temp);

void printAll(); // extra function to prove array is work

};

## Inventory.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Inventory { // single link list

private:

string itemName;

int itemNumber;

Inventory\* next;

public:

Inventory(string t\_itemName,int t\_itemNumber);

Inventory\* getNext();

int& getItemNumber();

string getItemName();

void setNext(Inventory\* temp);

void setItemName(string temp);

void setItemNumber(int temp);

};

## Monster.h

#pragma once

#include <cstdlib>

#include "Character.h"

#include "Inventory.h"

class Monster :public Character { // monster class is derived class of character class

private:

Inventory\* inventory;

Monster\* next;

public:

Monster(string name, string script, double hp, double damage,double x ,double y);

~Monster();

void add(string itemName);

void setNext(Monster\* temp);

Monster\* getNext();

Inventory\* getItem();

void printInventory();

void emptyInventory();

};

## MonsterList.h

#pragma once

#include "Monster.h"

class MonsterList {

private:

Monster\* list;

public:

MonsterList();

void add(Monster\* temp);

Monster& find(string name);

void printAll();

};

## NPC.h

#pragma once

#include <iostream>

#include <string>

#include <cstdlib>

#include "Character.h"

using namespace std;

class NPC :Character{ // NPC class is derived class of character class

private:

string order;

NPC\* next;

NPC\* prev;

public:

NPC(string temp\_order,string name,string script,double hp,double damage);

void Status();

string getOrder();

void setNext(NPC\* temp);

void setPrev(NPC\* temp);

NPC\* getNext();

NPC\* getPrev();

string getNName();

};

## NPCList.h

#pragma once

#include "NPC.h"

#include "FoodList.h"

#include "Player.h"

class NPCList {

private:

NPC\* npclist;

NPC\* head;

NPC\* tail;

string strings[8];

int remark; // record number of customer generated

public:

NPCList();

void add(NPC\* temp);

void remove(string order, FoodList\* ing,string cmd);

void GenerateNPC(FoodList\* ing);

void AllNPC();

void CheckOneByOne(string temp);

void displayData();

bool Serve(string name,Player\* player,FoodList \*fl);

int len(string str);

void split(string str, char seperator);

};

## Player.h

#pragma once

#include "Character.h"

#include "Inventory.h"

#include <Queue>

#include <dos.h>

#include "Monster.h"

#include <stack>

class Player:Character { // player class is derive class of character class

private:

Inventory\* inventory;

double money;

stack<int> chance; // used to store life

public:

Player(string name, string script, double hp, double damage,double temp\_m);

string getMyScript();

void add(string t\_itemName, int t\_itemNumber);

void remove(string t\_itemName, int t\_itemNumber);

bool validOfUse(string t\_itemName, int t\_itemNumber);

void readyToUse(string t\_itemName, int t\_itemNumber);

Inventory& search(string t\_itemName);

void addMoney(double temp);

void MyInventory();

void status();

void StatusUpgrade(queue<string>& temp);

bool attackMonster(Monster\* monster);

void BackToRestaurant(double x, double y);

void BackToRestaurant(string temp);

stack<int> getChance();

};

## Main.cpp

#pragma once

#include "NPCList.h"

#include "Player.h"

#include "Inventory.h"

#include "Ingredient.h"

#include <queue>

#include "FoodList.h"

#include <sstream>

#include <stdlib.h>

#include "MonsterList.h"

#include <SFML/Audio.hpp>

static string GameMenu();

string InGame();

string CustomerServe(NPCList &nl, FoodList\* fl,Player& pl);

FoodList\* createFood();

MonsterList\* createMonster(MonsterList\* mm);

string AdventureMenu(MonsterList\* mm);

int main() {

sf::SoundBuffer buffer;

buffer.loadFromFile("song.wav");

sf::Sound sound(buffer);

sound.play();

static FoodList\* fl = createFood();

Player\* player = nullptr;

Ingredient ingredient;

MonsterList\* monsterList = new MonsterList();

queue<string> upgrade;

string s1, s2,s3,s2\_3;

while (s1 != "4") {

s1 = GameMenu();

s2 = "";

system("CLS");

NPCList nl = NPCList();

nl.GenerateNPC(fl);

if (s1 == "1") {

player = new Player("Alex", "This is Easy Mode", 200, 10, 1000);

}

else if (s1 == "2") {

player = new Player("Alex", "This is Normal Mode", 150, 10, 500);

}

else if (s1 == "3") {

player = new Player("Alex", "This is Hard Mode", 100, 10, 200);

}

while (s2 != "6") {

s2 = InGame();

if (s2 == "1") {

s3 = "";

while (s3 != "4") {

s3 = CustomerServe(nl,fl, \*player);

}

}

else if (s2 == "2") {

player->MyInventory();

}

else if (s2 == "3") {

s2\_3 = "";

bool flag = false;

monsterList = new MonsterList();

monsterList = createMonster(monsterList);

while (s2\_3 != "exit") {

s2\_3 = AdventureMenu(monsterList);

bool flag = false;

if (s2\_3 != "exit") {

flag = player->attackMonster(&monsterList->find(s2\_3));

if (flag) {

player->BackToRestaurant(0,0);

s2\_3 = "exit";

if (player->getChance().empty() == 1) {

s2 = "6";

}

}

}

else if (s2\_3 == "exit") {

player->BackToRestaurant("Back to Resturant");

}

}

}

else if (s2 == "4") {

string t1 = "";

while (t1 != "3") {

cout << "\n---Status---" << endl;

cout << "1.Upgrade Status" << endl;

cout << "2.Check Status" << endl;

cout << "3.exit" << endl;

cout << "Selection (Enter Number) : "; getline(cin, t1);

if (t1 == "1") {

string t2,t3;

cout << "\n----Upgrade Status[100 per upgrade]----" << endl;

cout << "1.Running [Max HP + 10]" << endl;

cout << "2.Push Up [Damage + 5]" << endl;

cout << "3.exit" << endl;

cout << "Selection (1,2,1,2,1)(3 for exit) : "; getline(cin,t2); // enter selection

if (t2 == "3") { // if 3 == exit

t1 = "3"; // set t1 to 3 and quit

}

else {

stringstream X(t2); // convert input into stringstream

while (getline(X, t3, ',')) { // split 1 by 1 by delimeter ","

upgrade.push(t3); // push into queue

}

player->StatusUpgrade(upgrade); //pass into statusUpgrade()

}

}else if (t1 == "2") {

player->status();

}

}

}

else if (s2 == "5") {

cout << "\n------Ingredient List-------" << endl;

fl->PrintAll();

}

else if (s2 == "6") {

s1 = "";

}

}

}

return 0;

}

string AdventureMenu(MonsterList\* mm) {

string temp;

cout << "\n------Move And Attack------" << endl;

mm->printAll();

cout << "[exit] for quit" << endl;

cout << "Enter Monster Name:"; getline(cin, temp);

return temp;

}

MonsterList\* createMonster(MonsterList\* mm) {

Monster\* monsterA = new Monster("monstera", "Im Monster A", 100, 10, 50, 50);

monsterA->add("meat");

monsterA->add("vegetable");

monsterA->add("tomato");

monsterA->add("noodles");

Monster\* monsterB = new Monster("monsterb", "Im Monster B", 100, 10, 100, 200);

monsterB->add("rice");

monsterB->add("oil");

monsterB->add("salt");

monsterB->add("sugar");

Monster\* monsterC = new Monster("monsterc", "Im Monster C", 100, 10, 300, 450);

monsterC->add("chili");

monsterC->add("milk");

monsterC->add("water");

monsterC->add("pork");

Monster\* monsterD = new Monster("monsterd", "Im Monster D", 100, 10, 250, 350);

monsterD->add("egg");

monsterD->add("soy sauce");

monsterD->add("fish");

monsterD->add("cucumber");

Monster\* monsterE = new Monster("monstere", "Im Monster E", 100, 10, 500, 650);

monsterE->add("strawberry");

monsterE->add("pumpkin");

monsterE->add("apple");

monsterE->add("orange");

mm->add(monsterA);

mm->add(monsterB);

mm->add(monsterC);

mm->add(monsterD);

mm->add(monsterE);

return mm;

}

string GameMenu() {

string temp;

cout << "\n-----Select Mode-----" << endl;

cout << "1.Easy" << endl;

cout << "2.Normal" << endl;

cout << "3.Hard" << endl;

cout << "4.exit" << endl;

cout << "selection (Enter Number): "; getline(cin, temp);

return temp;

}

string InGame() {

string temp;

cout << "\n-----Select Action-----" << endl;

cout << "1.Serve Customer" << endl;

cout << "2.Inventory" << endl;

cout << "3.Adventure" << endl;

cout << "4.Status" << endl;

cout << "5.Check Ingredient" << endl;

cout << "6.exit" << endl;

cout << "Selection (Enter Numeber) : "; getline(cin, temp);

return temp;

}

string CustomerServe(NPCList &nl,FoodList\* fl,Player& pl) {

string temp;

cout << "\n----Customer Serve----" << endl;

cout << "1.Customer List" << endl;

cout << "2.Serve Customer" << endl;

cout << "3.Remove " << endl;

cout << "4.Exit" << endl;

cout << "Selection (Enter Number) : "; getline(cin, temp);

if (temp == "1") {

nl.AllNPC();

}

else if (temp == "2") {

string x; bool flag = false;

while (x != "exit") {

nl.displayData();

cout << "Enter Name of Customer (prev | next | exit) : "; getline(cin, x);

if (x == "prev" || x == "next") {

nl.CheckOneByOne(x);

}

else {

nl.Serve(x, &pl, fl);

}

}

}

else if (temp == "3") {

string x;

cout << "----Remove Customer----" << endl;

nl.AllNPC();

cout << "Enter Name of Customer : "; getline(cin, x);

nl.remove(x,fl,"rejected");

}

return temp;

}

FoodList\* createFood() {

FoodList\* fl = new FoodList();

Ingredient i;

string temp = i[7] + "-" + i[9] +"-" + i[19];

Food\* orangejuice = new Food("Orange Juice", 20,temp);

fl->add(orangejuice);

string temp1 = i[7] + "-" + i[9] + "-" + i[18];

Food\* applejuice = new Food("Apple Juice", 30, temp1);

fl->add(applejuice);

string temp2 = i[7] + "-" + i[9]+ "-" + i[15];

Food\* pumpkinjuice = new Food("Cucumber Juice", 25, temp2);

fl->add(pumpkinjuice);

string temp3 = i[4] + "-" + i[5] + "-" + i[6] + "-" + i[8] + "-" + i[11] ;

Food\* riceA = new Food("riceA", 100, temp3);

string temp4 = i[4] + "-" + i[5] + "-" + i[6] + "-" + i[8] + "-" + i[12] + "-" + i[14];

Food\* riceB = new Food("riceB", 100, temp4);

fl->add(riceB);

string temp5 = i[3] + "-" + i[5] + "-" + i[6] + "-" + i[12] + "-" + i[13] ;

Food\* noodleA = new Food("noodleA", 100, temp5);

fl->add(noodleA);

string temp6 = i[0] + "-" + i[1] + "-" + i[3] + "-" + i[11] + "-" + i[13] ;

Food\* noodleB = new Food("noodleB", 100, temp6);

fl->add(noodleB);

string temp7 = i[1] + "-" + i[2] + "-" + i[10] + "-" + i[16] + "-" + i[17] ;

Food\* desertA = new Food("desertA", 100, temp7);

fl->add(desertA);

return fl;

}

## Character.cpp

#include "Character.h"

Character::Character(string s\_name,string s\_script,double t\_hp,double t\_dmg) {

name = s\_name;

script = s\_script;

maxHp = t\_hp;

currentHp = t\_hp;

damage = t\_dmg;

}

string Character::getName() {

return name;

}

string Character::getScript() {

return script;

}

double Character::getMaxHp() {

return maxHp;

}

double& Character::getCurrentHp() {

return currentHp;

}

void Character::hpDecrease(double temp) {

currentHp -= temp;

}

void Character::setPosition(double x, double y) { // set position of character

pos.x = x;

pos.y = y;

}

void Character::characterStatus() {

cout << "\n----------Status-----------" << endl;

cout << "Name : " << name << endl;

cout << "MaxHP/HP : " << maxHp << "/" << currentHp << endl;

cout << "Damage : " << damage << endl;

}

void Character::increaseMaxHp(double temp) {

maxHp += temp;

currentHp = maxHp;

}

void Character::increaseDamage(double temp) {

damage += temp;

}

double Character::getDamage() {

return damage;

}

Position& Character::getPos() { // return position of character

return pos;

}

## Food.cpp

#include "Food.h"

Food::Food(string tempName, double tempCost,string temp) {

name = tempName;

cost = tempCost;

ii = temp;

}

Food\* Food::getNext() {

return next;

}

void Food::setNext(Food\* temp) {

next = temp;

}

void Food::ingrediantDetail() {

cout << ii << endl;

}

string Food::getName() {

return name;

}

string Food::getII() {

return ii;

}

double Food::getCost() {

return cost;

}

## FoodList.cpp

#include "FoodList.h"

FoodList::FoodList() {

list = nullptr;

}

void FoodList::add(Food\* temp) {

Food\* f = list;

if (f == nullptr) {

list = temp;

}

else {

while (f != nullptr) {

if (f->getNext() == nullptr) {

break;

}

f = f->getNext();

}

f->setNext(temp);

}

}

Food\* FoodList::getFood() {

Food\* temp = list;

int num = rand() % (8 - 0 + 1) + 0;

for (int i = 0; i < num ; i++) {

if (temp->getNext() == nullptr) {

break;

}

temp = temp->getNext();

}

return temp;

}

void FoodList::PrintAll() {

Food\* f = list;

while (f != nullptr) {

cout << "Name : " << f->getName() << endl;

cout << "Need : ";

f->ingrediantDetail();

if (f->getNext() == nullptr) {

break;

}

f = f->getNext();

}

}

Food\* FoodList::searchIng(string temp) {

Food\* f = list;

while (f != nullptr) {

if (f->getName() == temp) {

break;

}

f = f->getNext();

}

return f;

}

## Ingredient.cpp

#include "Ingredient.h"

Ingredient::Ingredient() {

index = 0;

maxSize = 20;

}

string Ingredient::getName() {

return name[index];

}

void Ingredient::operator++() {

index++;

}

void Ingredient::operator--() {

index--;

}

string Ingredient::operator[](int position) {

if (position < 0 || position >= maxSize) { // if position is small than 0 or larger than max size

return name[maxSize - 1]; // return back last index item

}

return name[position]; // normal return

}

bool Ingredient::operator==(Ingredient& temp)

{

return (temp.index >= 0 && temp.index <= maxSize);

}

void Ingredient::printAll() {

for (int i = 0; i < 20; i++) {

cout << " | " << name[i];

}

}

## Inventory.cpp

#include "Inventory.h"

Inventory::Inventory(string t\_itemName, int t\_itemNumber) {

itemName = t\_itemName;

next = NULL;

itemNumber = t\_itemNumber;

}

Inventory\* Inventory::getNext() {

return next;

}

string Inventory::getItemName() {

return itemName;

}

int& Inventory::getItemNumber() {

return itemNumber;

}

void Inventory::setItemName(string temp) {

itemName = temp;

}

void Inventory::setItemNumber(int temp) {

itemNumber = temp;

}

void Inventory::setNext(Inventory\* temp) {

next = temp;

}

## Monster.cpp

#include "Monster.h"

Monster::Monster(string name, string script, double hp, double damage,double x,double y) :Character(name, script, hp, damage) {

inventory = NULL;

setPosition(x, y);

next = nullptr;

}

Monster::~Monster() {

inventory = NULL;

}

void Monster::add(string itemName) {

Inventory\* temp = inventory; // create temporary inventory

int num = (1 + rand() % (4 - 1 + 1)); // random a number from 1 to 4

if (temp == nullptr) { // if inventory is null

inventory = new Inventory(itemName, num); // create and assign to inventory

}

else { // if invnetory is not empty

while (temp != nullptr) { // loop until the pointer is null

if (temp->getNext() == nullptr) { // next pointer is null

break; // break while lopp

}

temp = temp->getNext(); // get next pointer of current pointer

}

Inventory\* new\_i = new Inventory(itemName,num); // create new inventory pointer

temp->setNext(new\_i); // set it to next of current pointer

}

}

void Monster::setNext(Monster\* temp) { // set next pointer of current mosnter

next = temp;

}

Monster\* Monster::getNext() { // get next pointer of current mosnter

return next;

}

Inventory\* Monster::getItem() { // set inventory of current mosnter

return inventory;

}

void Monster::printInventory() { // print inventory detail

Inventory\* temp = inventory;

while (temp != nullptr) {

cout << temp->getItemName() << "-" << temp->getItemNumber() << " ";

if (temp->getNext() == nullptr) {

break;

}

temp = temp->getNext();

}

}

void Monster::emptyInventory() { // empty the inventory when monster die

Inventory\* temp = inventory;

while (temp != nullptr) {

temp->setItemNumber(0);

if (temp->getNext() == nullptr) {

break;

}

temp = temp->getNext();

}

}

## MonsterList.cpp

#include "MonsterList.h"

MonsterList::MonsterList() {

list = nullptr;

}

void MonsterList::add(Monster\* temp) {

Monster\* t = list;

if(t == nullptr){

list = temp;

}

else {

while (t != nullptr) {

if (t->getNext() == nullptr) {

break;

}

t = t->getNext();

}

t->setNext(temp);

}

}

void MonsterList::printAll() {

Monster\* temp = list;

while (temp != nullptr) {

temp->characterStatus();

//cout << temp->getName() << " Hp: " << temp->getCurrentHp() << "/" << temp->getMaxHp() << endl;

cout << "Item : ["; temp->printInventory(); cout << "]" << endl;

if (temp->getNext() == nullptr) {

break;

}

temp = temp->getNext();

}

}

Monster& MonsterList::find(string name) {

Monster\* temp = list;

Monster\* a;

while (temp != nullptr) {

if (temp->getName() == name) {

a = temp;

break;

}

if (temp->getNext() == nullptr) {

break;

}

temp = temp->getNext();

}

return \*temp;

}

## NPC.cpp

#include "NPC.h"

NPC::NPC(string temp\_order, string name, string script, double hp, double damage) :Character(name,script,hp,damage) {

order = temp\_order;

}

void NPC::setNext(NPC\* temp) { // set next NPC

next = temp;

}

void NPC::setPrev(NPC\* temp) { // set prev NPC

prev = temp;

}

NPC\* NPC::getNext() { // get next NPC

return next;

}

NPC\* NPC::getPrev() {// get prev NPC

return prev;

}

void NPC::Status() { // status of NPC

characterStatus();

cout << "Order : " << order << endl;

}

string NPC::getOrder() { // get order of NPC

return order;

}

string NPC::getNName() { // get name of npc

return getName(); // inherit from base class

}

## NPCList.cpp

#include "NPCList.h";

NPCList::NPCList() {

npclist = nullptr;

head = nullptr;

tail = nullptr;

remark = 0;

}

void NPCList::remove(string order,FoodList\* ing,string cmd) {

NPC\* temp = head;

bool flag = false;

while (temp != nullptr) {

if (temp->getNName() == order) {

flag = true;

break;

}

if (temp->getNext() == nullptr) {

break;

}

temp = temp->getNext();

}

if (flag) {

if (temp->getNext() != nullptr && temp->getPrev() != nullptr) {

temp->getNext()->setPrev(temp->getPrev());

temp->getPrev()->setNext(temp->getNext());

head = temp->getPrev();

tail = temp->getNext();

}

else {

cout << "here2" << endl;

if (temp->getNext() == nullptr) {

temp->getPrev()->setNext(nullptr);

tail = temp->getPrev();

}

if (temp->getPrev() == nullptr) {

temp->getNext()->setPrev(nullptr);

head = temp->getNext();

}

}

cout << temp->getNName() + " successfully " + cmd << endl;

NPC\* temp1 = new NPC(ing->getFood()->getName(), "c" + to\_string(remark), "Im " + to\_string(remark) + " Customer", 100, 0);

add(temp1);

remark++;

}

else {

cout << "No such Customer found" << endl;

}

}

void NPCList::GenerateNPC(FoodList\* ing) {

for (int i = 0; i < 3; i++) {

NPC\* temp = new NPC(ing->getFood()->getName(), "c" + to\_string(remark), "Im " + to\_string(remark) + " Customer", 100, 0);

add(temp);

//cout << i << ing->getFood()->getName() << endl;

remark++;

}

}

void NPCList::add(NPC\* temp) {

NPC\* npc = npclist;

if (npc == nullptr) {

npclist = temp;

head = npclist;

tail = npclist;

}

else {

head->setPrev(temp);

temp->setNext(head);

head = temp;

npclist = temp;

}

}

void NPCList::AllNPC() {

NPC\* npc = npclist;

cout << "\n----Customer List-----" << endl;

while (npc != nullptr) {

npc->Status();

if (npc->getNext() == nullptr) {

break;

}

npc = npc->getNext();

}

}

bool NPCList::Serve(string name, Player\* player,FoodList\* fl) {

bool x = false;

NPC\* temp = npclist;

while (temp != nullptr) {

if (temp->getNName() == name) {

x = true;

break;

}

if (temp->getNext() == nullptr) {

break;

}

temp = temp->getNext();

}

if (x) {

bool ok = true;

char separator = '-';

Food\* f = fl->searchIng(temp->getOrder());

string temp\_ing = f->getII();

split(temp\_ing, separator);

for (int i = 0; i < 8; i++) {

if (strings[i].empty() == false) {

bool fff = player->validOfUse(strings[i], 1);

if (fff == false) {

ok = false;

cout << strings[i] << " no enuf, please check inventory" << endl;

}

}

else {

break;

}

}

if (ok) {

for (int i = 0; i < 8; i++) {

if (strings[i].empty() == false) {

player->readyToUse(strings[i], 1);

}

else {

break;

}

}

player->addMoney(f->getCost());

cout << "Earned " << f->getCost() << endl;

remove(temp->getNName(), fl," served");

}

}

return x;

}

int NPCList::len(string str){

int length = 0;

for (int i = 0; str[i] != '\0'; i++)

{

length++;

}

return length;

}

void NPCList::split(string str, char seperator) {

int currIndex = 0, i = 0;

int startIndex = 0, endIndex = 0;

while (i <= len(str))

{

if (str[i] == seperator || i == len(str))

{

endIndex = i;

string subStr = "";

subStr.append(str, startIndex, endIndex - startIndex);

strings[currIndex] = subStr;

currIndex += 1;

startIndex = endIndex + 1;

}

i++;

}

}

void NPCList::CheckOneByOne(string temp) {

if (temp == "prev") { // if input is prev

if (npclist->getPrev() != nullptr) { // if prev is not a null pointer

npclist = npclist->getPrev(); // get prev pointer

}

else { // if empty prev pointer

cout << "\nthis is the first customer in list" << endl; // display messsage

}

}

else if (temp == "next") { // if input is next

if (npclist->getNext() != nullptr) { // if next is not a null pointer

npclist = npclist->getNext(); // get next pointer

}

else { // if next is null pointer

cout << "\nthis is the last customer in list" << endl; // display messsage

}

}

}

void NPCList::displayData() {

npclist->Status();

}

## Player.cpp

#include "Player.h"

Player::Player(string name,string script, double hp,double damage,double temp\_m):Character(name,script,hp,damage) {

inventory = NULL;

money = temp\_m;

for (int i = 1; i < 3; i++) {

chance.push(i);

}

}

stack<int> Player::getChance() {

return chance;

}

void Player::add(string t\_itemName, int t\_itemNumber) {

Inventory\* t\_inv = inventory;

bool flag = true; // true is new item , false == exists

if (t\_inv == nullptr) { // if inventory is empty

inventory = new Inventory(t\_itemName,t\_itemNumber); // create new inventory

}

else {

while (t\_inv != nullptr) { // if inventory pointer not null

if (t\_inv->getItemName() == t\_itemName) { // if itemname is exist in inventory

flag = false; // flag set to false

t\_inv->getItemNumber() += t\_itemNumber; //add the number of item

break; // break while loop and quit

}

if (t\_inv->getNext() == nullptr) { // if next pointer == null

break; // break loop

}

t\_inv = t\_inv->getNext(); // get next inventory

}

if (flag) { // true when item is no exist in inventory of player

Inventory\* temp = new Inventory(t\_itemName, t\_itemNumber); // add new item

t\_inv->setNext(temp); // set new item to the end of the single link list

}

}

}

void Player::remove(string t\_itemName, int t\_itemNumber) {

Inventory\* t\_inv = inventory;

bool flag = true;

while (t\_inv != nullptr) { //decrease item number by searching name of item

if (t\_inv->getNext() == nullptr || t\_inv->getItemName() == t\_itemName) {

t\_inv->setItemNumber(t\_inv->getItemNumber() - t\_itemNumber);

flag = false;

break;

}

t\_inv = t\_inv->getNext();

}

}

Inventory& Player::search(string t\_itemName) {

Inventory\* t\_inv = inventory;

bool flag = true;

while (t\_inv != nullptr) { //decrease item number by searching name of item

if (t\_inv->getNext() == nullptr || t\_inv->getItemName() == t\_itemName) {

return \*t\_inv;

}

t\_inv = t\_inv->getNext();

}

}

bool Player::validOfUse(string t\_itemName, int t\_itemNumber) {

Inventory\* t\_inv = inventory;

if (t\_inv == nullptr) {

return false;

}

else {

while (t\_inv != nullptr) {

if (t\_inv->getItemName() == t\_itemName) {

if (t\_inv->getItemNumber() > 0) {

return true;

}

else {

return false;

}

}

else if (t\_inv->getNext() == nullptr) {

return false;

}

t\_inv = t\_inv->getNext();

}

}

}

void Player::readyToUse(string t\_itemName, int t\_itemNumber) {

Inventory\* t\_inv = inventory;

while (t\_inv != nullptr) {

if (t\_inv->getItemName() == t\_itemName) {

t\_inv->getItemNumber() -= t\_itemNumber;

break;

}

if (t\_inv->getNext() == nullptr) {

break;

}

t\_inv = t\_inv->getNext();

}

}

string Player::getMyScript() {

return getScript();

}

void Player::MyInventory() {

Inventory\* x = inventory;

while (x != nullptr) {

cout << x->getItemName() << " : " << x->getItemNumber() << endl;

x = x->getNext();

}

if (inventory == nullptr) {

cout << "\nInventory is Empty" << endl;

}

cout << "\n";

}

void Player::status() { // print status of character

characterStatus(); // function inherit from base class (Character class) to print hp, name , position etc

cout << "Life : " << chance.top() << endl;

cout << "Pos X/Y : " << getPos().x << "/" << getPos().y << endl;

cout << "Gold : " << money << endl;

}

void Player::StatusUpgrade(queue<string> &temp) { // upgrade status of player, max hp and damage

cout << "\n---Upgrading...----" << endl; // upgrade status per time cost 100

if (money >= 100) { // if player current money > 100

while (temp.empty() == false){

if (money < 100) { // if player current money > 100

cout << "Money no enuf" << endl;

temp.pop();

}

else if(temp.front() == "1") { //

increaseMaxHp(10); // max hp increase 10

cout << "HP Increate 10 -- ";

cout << "Max/Current HP : " << getMaxHp() << "/" << getCurrentHp() << endl;

money -= 100; // money - 100

temp.pop(); // pop up 1 item from queue

Sleep(500);

}

else if (temp.front() == "2") {

increaseDamage(5); // damage increase 5

cout << "Damage Increase 5" << endl;

cout << "Damage : " << getDamage() << endl;

money -= 100; // money - 100

temp.pop(); // pop up 1 item from queue

Sleep(500);

}

}

}

else {

cout << "Money mo enuf" << endl;

}

}

bool Player::attackMonster(Monster\* monster) {

bool flag = false; // this flag is to check whther is monster dead or player dead

cout << "Current Position (x,y) : " << getPos().x << "," << getPos().y << endl;

if (getPos().x > monster->getPos().x) { // if user current position x bigger than monster current position x

while (getPos().x > monster->getPos().x) { // while user current position x bigger thatn monster current position x, loop will continue

getPos().x -= 50; // user current position x - 50, getpos is inherit from base class (Character)

cout << "Player Move (X,Y) : " + to\_string(getPos().x) + "," + to\_string(getPos().y) << endl; // print out statement

Sleep(500);

}

}

else {// if user current position x smaller than monster current position x

while (getPos().x < monster->getPos().x) { // while user current position x smaller thatn monster current position x, loop will continue

getPos().x += 50; // user current position x + 50, getpos is inherit from base class (Character)

cout << "Player Move (X,Y) : " + to\_string(getPos().x) + "," + to\_string(getPos().y) << endl;// print out statement

Sleep(500);

}

}

if (getPos().y > monster->getPos().y) {// if user current position y bigger than monster current position y

while (getPos().y > monster->getPos().y) { // while user current position y bigger than monster current position y, loop will continue

getPos().y -= 50; // user current position y - 50, getpos is inherit from base class (Character)

cout << "Player Move (X,Y) : " + to\_string(getPos().x) + "," + to\_string(getPos().y) << endl;// print out statement

Sleep(500);

}

}

else {// if user current position y smaller than monster current position y

while (getPos().y < monster->getPos().y) { // while user current position y smaller than monster current position y, loop will continue

getPos().y += 50;// user current position y + 50, getpos is inherit from base class (Character)

cout << "Player Move (X,Y) : " + to\_string(getPos().x) + "," + to\_string(getPos().y) << endl;// print out statement

Sleep(500);

}

}

if (monster->getCurrentHp() == 0) { // if mosnter current hp is 0

cout << monster->getName() << " already dead!" << endl; // appear monster has been defeated

}

else if (getCurrentHp() == 0) { // if player hp is 0

cout << " player already dead.Quiting adventure..." << endl; //apear msg

flag = true;

}

else { // else player and mosnter hp > 0

while (monster->getCurrentHp() > 0) { // continuous lopp when mosnter hp > 0

if (monster->getCurrentHp() - getDamage() > 0) {

monster->getCurrentHp() -= getDamage(); // decrease monter hp by player damge

}

else {

monster->getCurrentHp() = 0;

}

cout << "Player attack " << monster->getName() << endl; // display player hp

cout << monster->getName() + " HP : " << monster->getCurrentHp() << " / " << monster->getMaxHp() << endl << endl; // display monster hp

Sleep(500);

if (monster->getCurrentHp() == 0) { // if monster hp == 0

break; // break loop

}

getCurrentHp() -= monster->getDamage(); //decrease player hp by monster damage

cout << monster->getName() << " attack player " << endl; // appear msg

cout << getName() + " HP : " << getCurrentHp() << " / " << getMaxHp() << endl << endl; // dusplay plater current hp

Sleep(500);

if (getCurrentHp() == 0) { // if player hp is 0

cout << "Player already dead.Quiting adventure..." << endl; // appear msg

flag = true; // set flag to true

break;

}

}

if (flag == false) { // flag == false mean player win

cout << monster->getName() + " defeated by player" << endl; // display msg

cout << "Item get : "; monster->printInventory(); cout << endl; // display msg item get

Inventory\* temp = monster->getItem(); //get item from monster

while (temp != nullptr) {

add(temp->getItemName(),temp->getItemNumber()); // add it into player inventory

if (temp->getNext() == nullptr) { // if no next pointer, break

break;

}

temp = temp->getNext(); // get next pointer

}

monster->emptyInventory(); // empty inventory of monster

}

}

return flag; // return true or false, false means monster defeat, true is player dead

}

void Player::BackToRestaurant(double x,double y) {

chance.pop(); // decrease 1 hp

setPosition(x,y); // initialize the position of player

getCurrentHp() = getMaxHp(); // recovery player hp by assigning Maxhp to current hp

if (chance.empty() != 1) { // if still left life, print this message

cout << "Hp regeneration done." << endl; // appear msg

}

else {

cout << "Game End!" << endl;

}

}

void Player::BackToRestaurant(string temp) {

cout << "\nCurrent Position (x,y) : " << to\_string(getPos().x) + "," + to\_string(getPos().y) << endl;

while (getPos().x > 0) {

getPos().x -= 50;

cout << "Player Move (X,Y) : " + to\_string(getPos().x) + "," + to\_string(getPos().y) << endl;

Sleep(500);

}

while (getPos().y > 0) {

getPos().y -= 50;

cout << "Player Move (X,Y) : " + to\_string(getPos().x) + "," + to\_string(getPos().y) << endl;

Sleep(500);

}

getCurrentHp() = getMaxHp();

cout << temp << endl;

}

void Player::addMoney(double temp) {

money += temp;

}