



Object-Oriented Programming

ASSIGNMENT 3

박태성 / 2015722031 / 2019 년 5 월 31 일

1.

Problem description and explanation

Write a program that simulates and shows the results of the battle between 5 humans and 5 monsters until there are no survivals in any camps.

Followings are constraints.

- Each camp(human and monster) has 5 instances. Human is consists of four classes, Peasant Army, Sword Master, Archer, and Warlock.
- Inherit abilities as Figure 1.
- Override action function as Figure 3.
- Instances for each camp have connection. Implement the connection with double linked list. Do not use any librarys related to double linked list. If health is lower than 0, delete from the list. Each instance has its own index.
- The battlefield is 5*5 array. Multiple instances can have the same coordinates. Intial state of each instance is like Figure 6.
- Each camp is given a turn to attack or move. Monster takes a turn after Human. All instance can attack or move in their turn.
- Instance attacks the closest opponents. If there are not any opponents in the attack range, move toward the closest opponets(move horizontally first and move cell by cell). Move toward the instace which has lower index, if there are one more instance to attack or move.
- If there are no more survivals in each camp, the battle ends.

I created a [distance calculation\(\)](#) for searching the closest opponent. I included header cmath to use [sqrt\(\)](#) and [pow\(\)](#). This function finds the monster or human at the nearest distance and stores it in the target variable. And target is used as an argument of action (). In action (), if the target enters the attack range, it attacks and moves to the monster or human if it does not enter the attack range. Every object compares the distance through the loop and finds the target.

I created a [displayDeathList\(\)](#) function in the class that stores dead objects in an array(deathlist) and prints them with the results of each round. The [deletedead\(\)](#) function implements a function that deletes objects whose health is less than zero from the list.

When I made double linked lists, I created tailnode and [*GetTail\(\)](#) function because it is useful to delete node.

Result screen

```

C:\Users\박태성\Desktop\광운대 2학년 1학기\OOE\Assignment3\예비\Assignment3-1\Assignment3-1\Debug\Assignment3-1.exe
###Result of round 1###
<Human>
1
2
3
4
5
<Monster>
A
B
C
D
E

###Result of round 2###
<Human>
2
4
<Monster>
A
B
C
D
E
Human 1 has died
Human 3 has died
Human 5 has died

###Result of round 3###
<Human>
4
<Monster>
A
B
C
E
Human 2 has died
Monster D has died

###Result of round 4###
<Human>
<Monster>
A
B
C
E
#### Final result of battle ####
Human has been defeated
계속하려면 아무 키나 누르십시오 . . .

```

Review

I was confused because I did not know the difference between [overloading](#) and [overriding](#). Overloading means that the function name is the same, but the type or number of arguments is wrong. Overriding means redefining a function of the same name in each instance. Overriding has been useful in redefining the action of warlock and archer.

I have implemented a [double-linked list](#), but I do not know the advantages of comparing it with a [single linked list](#). The reason for not finding this point in this assignment is that the order of attack starts from the object with low index.

The length of the code was too long to find a point where an error occurred. I used cout to find out where the error occurred, but this time I could use breakpoints and debugging to find out exactly where the error occurred.

Calculate the distance and designate the object with the closest distance as target. And they attack or move by inspecting whether they are within attack range. Through the loop, every object has an opportunity to attack or move. I used a loop and implemented all functions with the assumption that one target is one. So the implementation of warlock failed. A warlock can attack up to four monsters. Therefore, up to four targets can be accommodated. In my code, however, exceptions can not be processed because the loop is used. Through this assignment, I knew the fatal disadvantages of the loop. It is impossible to handle exceptions. Therefore, I think we can solve the problem by modifying the code as follows. First, the distance between all living objects is calculated and stored in vector pair format with index. Then the monster inside the attack range is saved as vector pair type. It would be better if it could be sorted based on distance length after storage. Next is action (). In the case of Warlock, the attack is performed from the first to the fourth nearest distance. The amount of code that needs to be modified is too large to fix. Here is the code in the direction I wanted to modify.

```
while (Current != NULL)
{
    double distance=CalculateDistance()

    p = make_pair(MonCurr->GetIndex(), distance);

    v.push_back(p);

    MonCurr = MonCurr->GetNext();
}

int clos = v[0].second; //closest distance
```

2.

Problem description and explanation

Write a program that implement a simple cache system which has a structure of single linked list composed of maximum five 'CacheNode' classes. Read words from the file "sentences.txt" using LRU algorithm. 'CacheNode' and 'CacheManager' classes must keep given constraints. Followings are caching rules.

- At the beginning, the cache is empty.
- When caching, words are not case sensitive.
- Words can be separated by one or more consecutive non-alphabet characters.

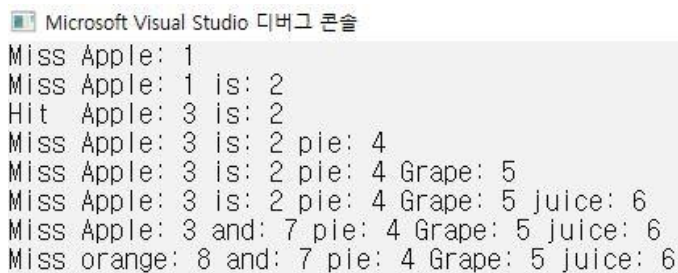
- Print message as 'Hit' and update variable 'timestamp' if the same word exists in linked list.
- If not, print message as 'Miss' and create an object of 'CacheNode'. If cache is not full, add it to the end. If cache is full, replace the node according to LRU algorithm.
- After every caching, print out the result.

I used the `strcpy ()` function and `c.str ()` to convert a string type character to a `char *` type. This is because the `strtok ()` function uses `char *` as an argument.

I have used the `tolower()` function to check whether a word is the same or not by changing all characters to lowercase to solve the case where the word is not case-sensitive. An alternative is the `toupper ()` function. The `toupper ()` function converts all characters to uppercase.

According to the LRU algorithm, the oldest stored word is deleted and a new word is stored. Therefore, i implemented `FineMIN()` function that finds the smallest word in the timestamp.

Result screen



```

Microsoft Visual Studio 디버그 콘솔
Miss Apple: 1
Miss Apple: 1 is: 2
Hit Apple: 3 is: 2
Miss Apple: 3 is: 2 pie: 4
Miss Apple: 3 is: 2 pie: 4 Grape: 5
Miss Apple: 3 is: 2 pie: 4 Grape: 5 juice: 6
Miss Apple: 3 and: 7 pie: 4 Grape: 5 juice: 6
Miss orange: 8 and: 7 pie: 4 Grape: 5 juice: 6

```

Review

The `strtok()` function uses only `char *` as an argument, so if you have entered a string in the file as a string type, converting is necessary.

Whether the maximum size of the list is greater than 5 or not depends on how it works. Therefore, it is a good idea to manage `listsize` by declaring it as a variable. I wanted to cut the word, create a node, and connect it to the old node. In the end, it was more intuitive to cut nodes and create nodes first.

I also found that there are many ways to implement `linked lists`. Some developers preferred to use tail nodes, and others did not. In this case, I did not use a tail node, but if the length of the node is very long, should use the tail node. This is because, in some cases, searching from the head may cause unnecessary repetition.

3. Problem description and explanation

Write a program that simulates the operation of nine neurons according to the following rules.

- Nine neurons initially construct a linked list as Fig.
- Each inter-layer connection can be broken with a 10% probability. Implement probability by using `srand()` and `rand()`. Seed of rand operation should be `srand(1)`.
- At final state, execute the calculation and print the result.
- Construct Neuron class as given rules.
- At first state, create nine neuron objects and link as Fig.
- Total three state updates are performed and connections can be broken during state update. Connection of the nodes is only accessible through head node.
- In the fourth state, execute calculation horizontally from each neuron of first layer until there is no connection. Do not consider the priority of operators. Discard the remainder of division. Print the expression and answer as Output Example.

I have stored the operator in string format for a given problem. Therefore, I implemented functions `conv()` that performs an operation and `oper()` that convert an operator. The above two functions are executed inside the `calcu()`.

When implementing `connbreak()`, I declared a `tmp` to store Head so that I can go down to Neuron when I reach Neuron on the far right.

Result screen

A screenshot of a Windows file explorer window. The address bar shows the path: C:\Users\박태성\Desktop\광운대 2학년 1학기\WOOE\Assignment3\Assignment. Below the address bar, there is a list of files. The first file is a text file with the following content: 9+0*6=54, 5/3=1, 1+7/4=2, and 계속하려면 아무 키나 누르십시오 . . .

Review

It is very difficult to generate a `random number` for a computer. Generally, if the seed value is not changed, the fixed value is output. Therefore, it is difficult to say that this is a random number generation. Thus, the computer uses the function called `time()` as a seed. However, even if you use a method that keeps the seed value constant, it generally looks random enough, but it can not be considered "true random". Since it is a value determined by an expression, regularity must occur, and prediction can also be made if it knows what a formula looks like.

I used `append()` between the `append()` and `insert()` function names. I heard that there are some developers who mix it. In this assignment, we only need to add the function to the last node, so we decided on the function name with `append()`.

Initially, I was trying to solve the problem using the loop statement. However, if the connection between the nodes is lost, the loop can not catch it and get the desired result. So I did not use the loop.

4.

Problem description and explanation

Implement the program that shows the following statement. “Flip the cards multiple of 2 at first, flip the cards multiple of 3, and keep doing this to the last cards”.

Followings are constraints.

- Implement with MFC.
- Display the cards by using 53 bmp files(52 front images and one back image).
- Width of card = 49, Height of card = 72.
- Display the front side of the 52 cards when the ‘Reset’ button is pressed.
- Flip the cards multiple of next step when the ‘1 Step’ button is pressed(repeat this process until multiple is 52). Do not delete the photo file, but overwrite it. Thus, some card can be overwritten many times.
- Display the final result(after flipping from multiple of 2 to multiple of 52) when the ‘Final’ button is pressed. Final result is same with the final result after ‘1 Step’ button is pressed 51 times

I used the `bitbit()` among the main graphics functions that refer to the bitmap GDI object to display the image. I transferred the bitmap block by making it compatible with WinDC and memDC. This is called high-speed copying. The bitmap does not output directly to the screen, but after displaying the image in memory, the bitmap is copied to the screen at a high speed and is visible to the user. **Memory DC** do not load on the screen because they only process all the output from the memory and copy them to the screen at high speed. The `Bitbit` function is a function to copy the screen drawn in memory DC to the screen DC at high speed. Since the original contents are copied, `SRCCOPY` is used as an argument. I printed the bitmap through the following procedure.

1. Declare the memory DC to use.
2. Create a memory DC.
3. Load the bitmap image.
4. Transfer the loaded bitmap to the memory DC.
5. Fast copy bitmap from memory DC to screen DC.
6. Release the bitmap object.

I declared variable `multiple` and array `state` in `CAssignment34View` class. The array ‘state’ is an array that stores the state of the card whether it is the front side or the back side. Either overwrite the front image or overwirte the back image depending on the values in the array.

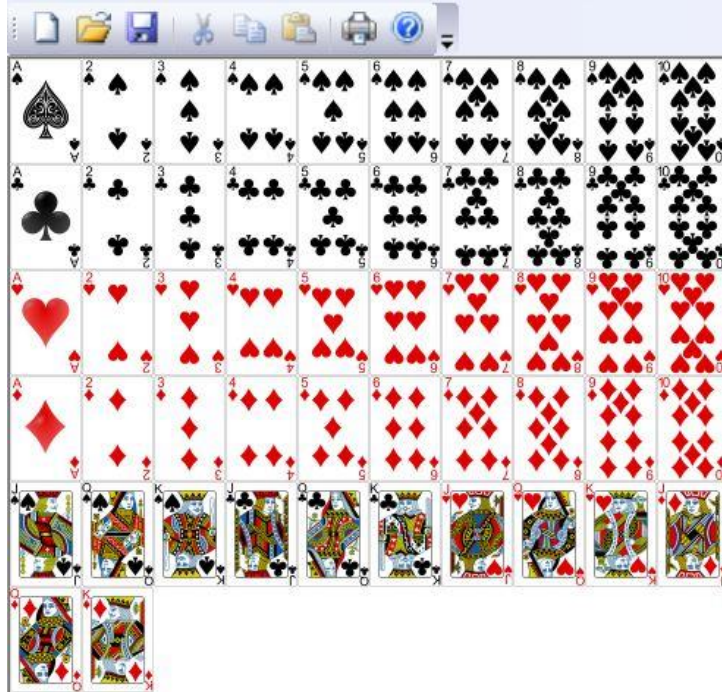
When you add a bitmap to a resource, it automatically defines the number. This was used as an argument in the loop statement.

```
#define idb_bitmap1      311
#define idb_bitmap2      312
#define idb_bitmap53     363
```

Result Screen

제목 없음 - Assignment3-4

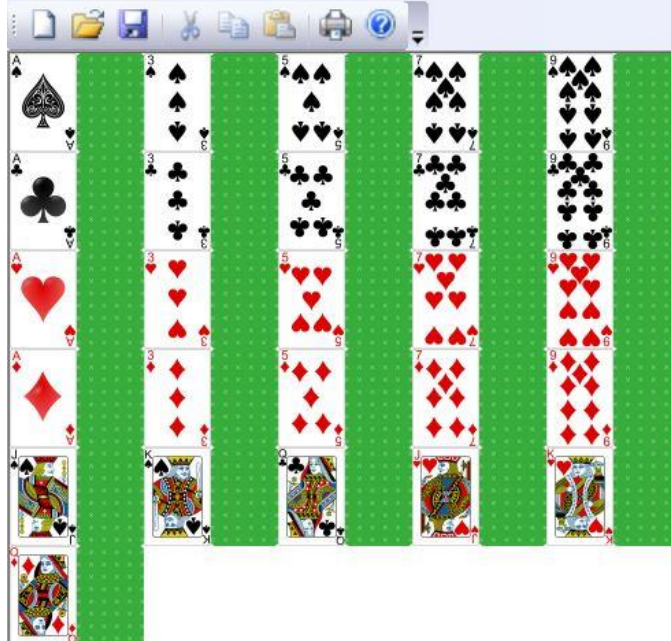
파일(F) 편집(E) 보기(V) 도움말(H) Reset 1Step Final



When 'Reset' button is pressed

제목 없음 - Assignment3-4

파일(F) 편집(E) 보기(V) 도움말(H) Reset 1Step Final



After flip the cards multiple of 2



After flip the cards multiple of 3



Final Result

Review

[Path of bitmap](#) is automatically saved when loading the bitmap. Therefore, if you

change the location of the cards folder, you may get errors when loading bitmaps. It seems to be a lot of errors related to resoure.h when working with MFC files relatively. So be careful not to modify the original [resource.h](#). I had to create a new file because I could not compile while writing the report because of the above problem. Unlike other assignments, there were many codes that were set at the default setting. I was unfamiliar with MFC, so I accidentally generated the code and caused a lot of problems.