

1. Project Description of 1

The first problem is to design a linked list using the class so that you can change the information by selecting the connections, deletions, inserts, and lists in the desired order.

The larger framework is organized and understanding how variables and functions in the class are used and understanding of the linked list makes troubleshooting easy.

The functions used consist of the function to add a node, the function to delete a node, the function to find and replace the desired node with the desired data, and the function to delete all nodes.

2. Action Results

```
GA. C:\Windows
Initialization
insertion (1)
insertion (3)
insertion (4)
insertion (6)
Size of linked list = 4
-----
idx: 0, data: 1
idx: 1, data: 3
idx: 2, data: 4
idx: 3, data: 6
-----
Test insertion
insertion (1, 2)
insertion (4, 5)
Size of linked list = 6
-----
idx: 0, data: 1
idx: 1, data: 2
idx: 2, data: 3
idx: 3, data: 4
idx: 4, data: 5
idx: 5, data: 6
-----
Test pop
pop (0)
pop (3)
pop ()
Size of linked list = 3
-----
idx: 0, data: 2
idx: 1, data: 3
idx: 2, data: 4
-----
Test get and set function
linkedList.get(1) = 2
linkedList.set(1, 50)
linkedList.get(1) = 50
Test clear
Size of linked list = 0
This linked list is empty.
계속하려면 아무 키나 누르십시오 . . .
```

Initialization

insertion (5)

insertion (4)

insertion (2)

insertion (6)

Size of linked list = 4

idx: 0, data: 5

idx: 1, data: 4

idx: 2, data: 2

idx: 3, data: 6

Test insertion

insertion (2, 10)

insertion (4, 12)

Size of linked list = 6

idx: 0, data: 5

idx: 1, data: 4

idx: 2, data: 10

idx: 3, data: 2

idx: 4, data: 12

idx: 5, data: 6

Test pop

pop (1)

pop (2)

pop ()

Size of linked list = 3

idx: 0, data: 5

idx: 1, data: 10

idx: 2, data: 12

Test get and set function

linkedList.get(2) = 10

linkedList.set(1, 25)

linkedList.get(1) = 25

Test clear

Size of linked list = 0

This linked list is empty.

계속하려면 아무 키나 누르십시오 . . . █

3. Review

This is the first problem that I have encountered a linked list since the second semester of first grade.

The concept of a linked list was well understood in the first grade when we were working on a project, but using the class and template made moving variables and functions less familiar and time-consuming than before.

After studying the usage of the class again, I learned more about the concept and easily solved the problem by linking the list like I used in the first grade.

1. Project Description of 2

The second challenge was the application of the first lesson.

The function consisted of two main components, the first function to rearrange the original list in reverse order, the next function to rearrange the function in reverse order, and then to change the code to the original list.

2. Action Results

```
C:\Windows
Initialization
insertion (1)
insertion (3)
insertion (4)
insertion (6)
-----
idx: 0, data: 1
idx: 1, data: 3
idx: 2, data: 4
idx: 3, data: 6
-----
After reverse function
-----
idx: 0, data: 6
idx: 1, data: 4
idx: 2, data: 3
idx: 3, data: 1
-----
```

```
After sort function
-----
idx: 0, data: 1
idx: 1, data: 3
idx: 2, data: 4
idx: 3, data: 6
-----
계속하려면 아무 키나 누르십시오 . . .
```

```
Initialization
insertion (1)
insertion (10)
insertion (5)
insertion (7)
-----
idx: 0, data: 1
idx: 1, data: 10
idx: 2, data: 5
idx: 3, data: 7
-----
After reverse function
-----
idx: 0, data: 7
idx: 1, data: 5
idx: 2, data: 10
idx: 3, data: 1
-----
After sort function
-----
idx: 0, data: 1
idx: 1, data: 5
idx: 2, data: 7
idx: 3, data: 10
-----
계속하려면 아무 키나 누르십시오 . . .
```

3. Review

The code for arranging the list in reverse order was easily implemented.

We could do this simply by replacing the nodes that the connected lists point to with the nodes behind them, and then by changing the head pointer to the tail pointer at the end.

Secondly, we designed the code to compare values of data and rearrange the data.

We designed the code by comparing the data of each node by using the sorting method that we used a lot and replacing only the data between the nodes when the condition is satisfied.

This is a simpler function than I thought.

If it is a chance, it is better to arrange not only data but also nodes.

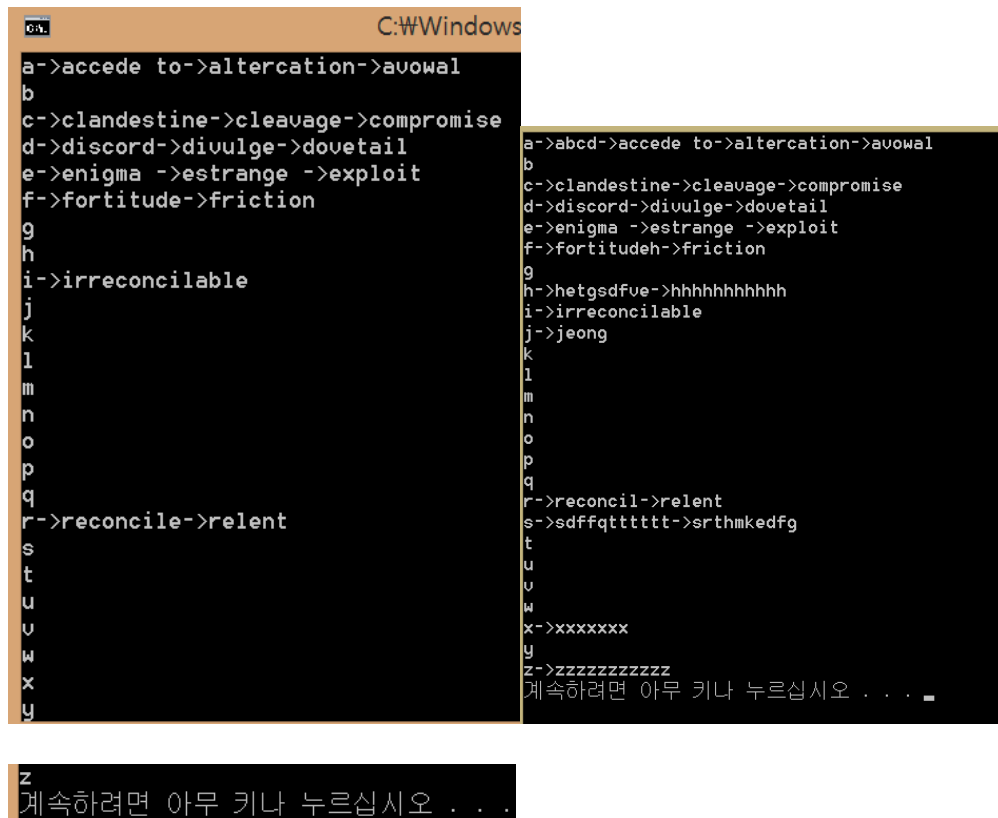
I think the exchange between data is much more efficient than the array of nodes themselves.

1. Project description of 3

The third task was to take the words from the Notepad using the File I/O function, arrange them alphabetically, and arrange them alphabetically.

Since there was no input, and as I learned the input and output of files through the second task, I am aware of and understand the concepts of using the class, and if I can apply them, it is not too difficult.

2. Action Results



```
C:\Windows
a->accede to->altercation->avowal
b
c->clandestine->cleavage->compromise
d->discord->divulge->dovetail
e->enigma ->estrangle ->exploit
f->fortitude->friction
g
h
i->irreconcilable
j
k
l
m
n
o
p
q
r->reconcile->relent
s
t
u
v
w
x
y

a->abcd->accede to->altercation->avowal
b
c->clandestine->cleavage->compromise
d->discord->divulge->dovetail
e->enigma ->estrangle ->exploit
f->fortitude->friction
g
h->hetgsdfue->hhhhhhhhhhh
i->irreconcilable
j->jeong
k
l
m
n
o
p
q
r->reconcil->relent
s->sdfqttttt->srthmkedfg
t
u
v
w
x->xxxxxxx
y
z->zzzzzzzzzz
계속하려면 아무 키나 누르십시오 . . .
```

```
z
계속하려면 아무 키나 누르십시오 . . .
```

3. Review

I was a little embarrassed because I wasn't given a very formal frame on the third assignment.

I know the concept of a linked list, but I didn't know well even if I studied the class, so it took me a long time to solve the assignment.

First, I had no problem connecting the alphabet with the letter, but I had a lot of trouble connecting words.

The word " according to " contains a space.

so, Problem solved by passing word by using getline function to receive word to recognize an opening as end of sentence.

We also solved the problem slowly by debugging the problem because we had to take the task into account the connection of words and alphabet connections, rather than just the one way the list was linked.

1. Project Description of 4

The fourth challenge is to design a program that gets the size of your card and allows you to select a menu to pull, erase and view all your cards.

First, when the program starts, Place the menu with the size of the card you can pull the first menu is the function of pulling out a card. The cards must be random in shape and number, and no duplicate cards should be used for exception processing. The second function is to delete a card.

However, you must be able to delete the card using the first-in-first-out data method.

The third menu shows the card that has been selected so far, and the fourth function functions to close the program.

2. Action Results

Queue Size : 2 ----- Queue Size : 2 1. Generate a card 2. Delete a card 3. Show all cards 4. End ----- Select menu : 3 Queue is Empty ----- Queue Size : 2 1. Generate a card 2. Delete a card 3. Show all cards 4. End ----- Select menu : 1 ----- Queue Size : 2 1. Generate a card 2. Delete a card 3. Show all cards 4. End ----- Select menu : 3 ♠6	Queue Size : 2 ----- 1. Generate a card 2. Delete a card 3. Show all cards 4. End ----- Select menu : 2 ♠6 is popped Queue is Empty ----- Queue Size : 2 1. Generate a card 2. Delete a card 3. Show all cards 4. End ----- Select menu : 1 ----- Queue Size : 2 1. Generate a card 2. Delete a card 3. Show all cards 4. End ----- Select menu : 1 ----- Queue Size : 2 1. Generate a card 2. Delete a card 3. Show all cards 4. End ----- Select menu : 3 ♠6 / ♥7	----- Queue Size : 2 1. Generate a card 2. Delete a card 3. Show all cards 4. End ----- Select menu : 2 ♠6 is popped ----- Queue Size : 2 1. Generate a card 2. Delete a card 3. Show all cards 4. End ----- Select menu : 1
--	---	--


```

Queue Size : 55 -----
Input Queue Size again(1~52) : 10 Queue Size : 10
-----
1. Generate a card
2. Delete a card
3. Show all cards
4. End
-----
Select menu : 1
-----
Queue Size : 10
1. Generate a card
2. Delete a card
3. Show all cards
4. End
-----
Select menu : 1
-----
Queue Size : 10
1. Generate a card
2. Delete a card
3. Show all cards
4. End
-----
Select menu : 1
-----
Queue Size : 10
1. Generate a card
2. Delete a card
3. Show all cards
4. End
-----
Select menu : 3
♠11 / ♠7 / ♥4 / ♦1 / ♦3 / ♣7
-----
Queue Size : 10
1. Generate a card
2. Delete a card
3. Show all cards
4. End
-----
Select menu : 2
♠11 is popped

```

```

-----
Queue Size : 10
1. Generate a card
2. Delete a card
3. Show all cards
4. End
-----
Select menu : 2
♠7 is popped
-----
Queue Size : 10
1. Generate a card
2. Delete a card
3. Show all cards
4. End
-----
Select menu : 4
End the program
계속하려면 아무 키나 누르십시오 . . .

```

3. Review

It was the most interesting task in this project.

You should use a random function for your priorities, and if you change the seed value, you can write programs more efficiently.

In this task, the linked list was a link between the cards, and each card points to the following cards with random numbers and shapes.

To prevent duplicate cards from being issued from the card extraction function, the selected nodes have been compared to the already selected nodes so that the card extraction can be repeated when the data is identical.

We also created the program by changing the head pointing to the front of the list to the next node and deleting the node that was the original head because the card must be deleted in a queue.

Since there is not much to do with exceptions, we have applied the questions 1 and 2 so that we can erase the cards and show the selected cards visually.

1. Project Description of 5

The task of 3-5 is to create the Snake Game.

Although result values were usually displayed using the console window, this is the first time that the MFC is used.

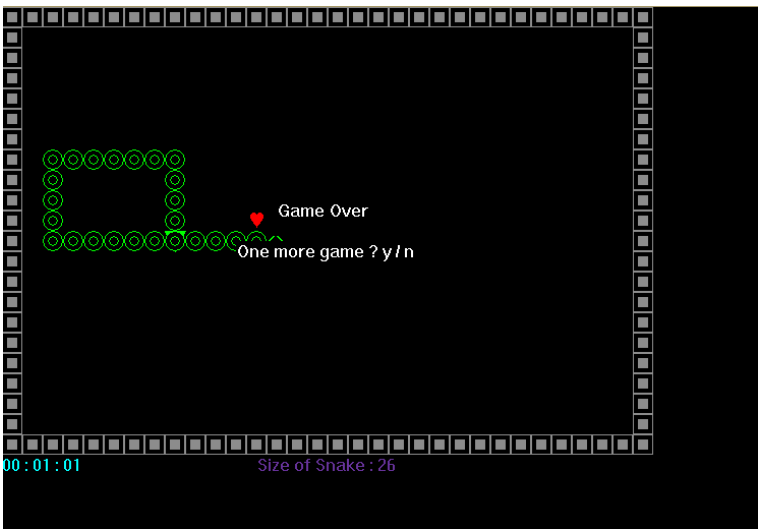
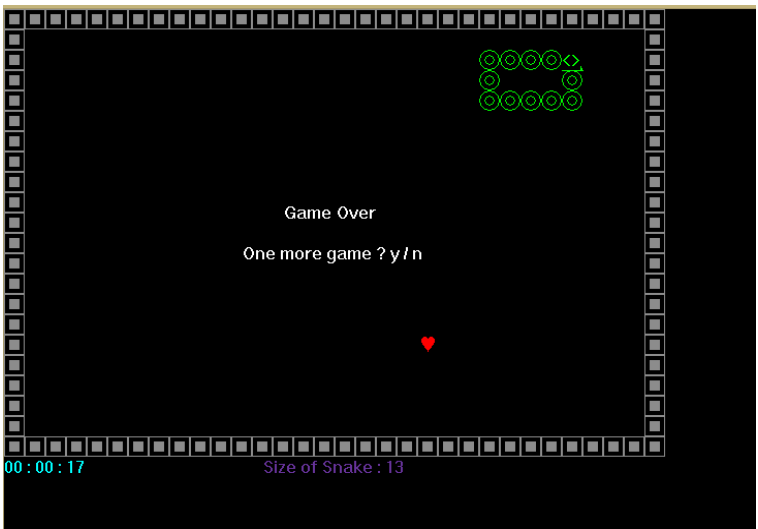
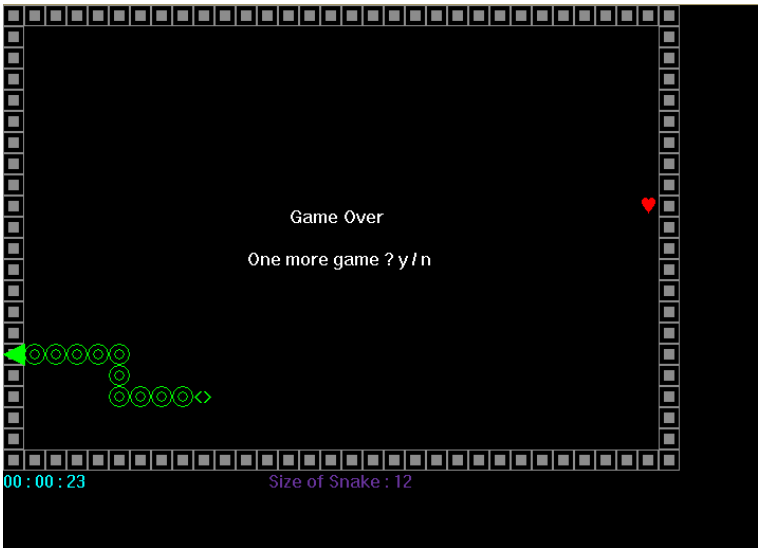
The program function is a program that creates a field for the game, automatically moves the snake, and changes its direction by receiving input from the keyboard.

There is also a timer over time and if eaten, the snake's length will increase, and its size should be visible.

The shape of the field and the shape of the snake's head, trunk, and tail prey are also defined, and cannot move backwards while on the move, and the current game is ended if it hits a wall or fits into its own body.

When the game is over, it lets you choose whether to play or end the game again.

2. Action Results



3. Review

It was an unfamiliar task when I first used MFC.

We did this by finding a lot of information on Google and referring to the algorithms used by people.

First, it wasn't your usual console window, and because there was no main function, I didn't know how the program works.

I repeated the references and learned how they function until I got used to them.

As a problem with the task, a problem occurred where KE UP is used but KEYDOWN is not used.

As a solution, we need to reset the other keys through search, so we can add the function by referring to the function that functions.

There was also a process in which a timer had to be created, and when the function ran over time and the keyboard was pressed, it was coded to run, and there was a time gap.

So, I solved the problem by using two timers and setting them in different times.

Because the entire Snake Game is licensed on Google a lot, we have written code with much reference.

This is a task to know that MFC is experienced and used and there are various functions.