과제 5

202104183 김현중

문제정의)

1. 삽입, 2. 삭제, 3. 모두 보기, 4. 종료 4가지의 기능이 있는 그래픽 편집기를 만들어야한다. 1을 누르면 선, 원, 사각형 중 하나를 선택해야한다. 그리고 선택한 것을 3번을 누를 때 모두 나오도록 한다. 2번으로 지금까지 넣었던 도형을 삭제할 수 있다. 4번을 누르면 시스템이 종료된다.
각각의 Line, Circle, Rect 파일들은 Shape 파일에게서 상속받는다.
GraphicEditor는 UI, Shape, Line, Circle, Rect 모두에게서 상속받아야한다.

문제 해결 방법)

UI클래스의 경우 화면에 출력되는 기본적인 선택지들을 만들 것이고, shape 클래스에서는 각각의 도형들을 그리는 함수를 만들 것이다. Line Circle Rect 클래스들은 각각 라인, 원, 사각형을 그리고 GraphicEditor에서는 각각의 값들로 시스템을 구현할 것이다.

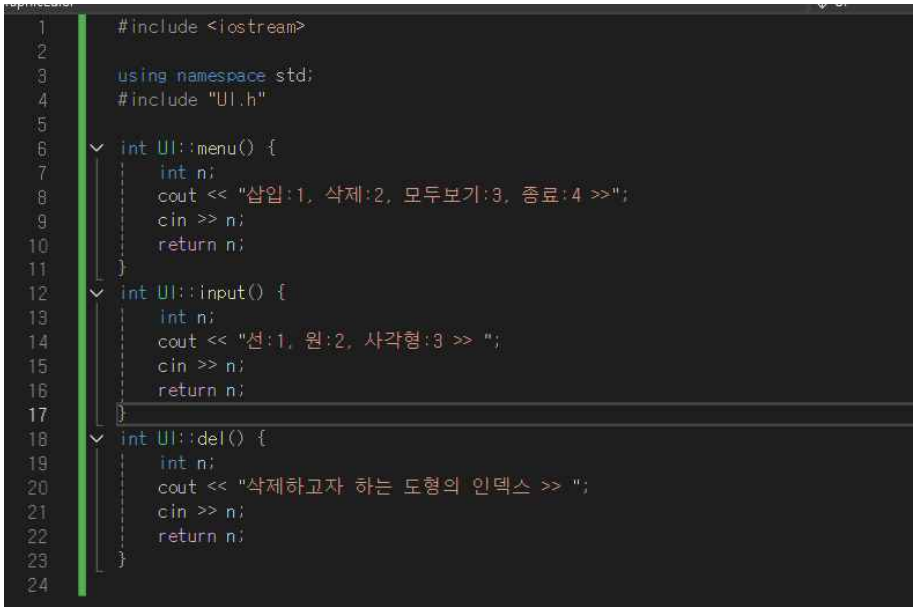
아이디어 평가)

위의 문제 해결 방법에서 말한 것을 토대로 만들면 이 도형들을 그린 것을 저장해야한다.
그러기 위해서는 무한 배열을 생성하여 저장하는 방법도 있지만 저번 수업 시간에서 배운 pStart와 pLast를 사용하여 도형들을 이어서 만들었다.

pStart에 도형을 만들어 Line을 집어넣으면 pStart는 Line, pLast도 Line이 된다. 그런데 여기에 add를 통해 주소 값을 바꾸어서 Circle을 생성해주어 pLast에 Circle을 저장한다. 그리고 Rect 또한 같게 생성을 한다.
쉽게 말해서 pStart는 처음 만든 Line의 주소 값이고 pLast는 마지막에 만든 주소 값이므로 주소를 next로 넘기면 처음부터 마지막까지의 도형을 출력할수 있는 것이다.

문제를 해결한 키 아이디어 또는 알고리즘 설명)

헤더파일, 선언부cpp, main.ccp파일로 나눠서 너무 파일이 많아 중요한 파일만 설명하겠다.



```cpp
#include <iostream>

using namespace std;
#include "UI.h"

int UI::menu() {
    int n;
    cout << "삽입:1, 삭제:2, 모두보기:3, 종료:4 >>";
    cin >> n;
    return n;
}
int UI::input() {
    int n;
    cout << "선:1, 원:2, 사각형:3 >> ";
    cin >> n;
    return n;
}
int UI::del() {
    int n;
    cout << "삭제하고자 하는 도형의 인덱스 >> ";
    cin >> n;
    return n;
}

```

우선 UI의 구현부이다. menu함수와 input함수 del함수가 있는데 간단히 설명하자면 menu함수는 맨 처음 삽입, 삭제, 모두보기, 종료를 화면에 띄우고 값을 입력받아 return해주는 것이다. 밑의 input과 del 또한 이 함수와 작동방식은 같다.

```
#pragma once

class Shape {
    Shape* next;
protected:
    virtual void draw();
public:
    Shape();
    virtual ~Shape() {}
    Shape* add(Shape* p) {
        this->next = p;
        return p;
    }
    Shape* getNext() {
        return next;
    }
    void paint();
    void setNext(Shape* p);
};
```

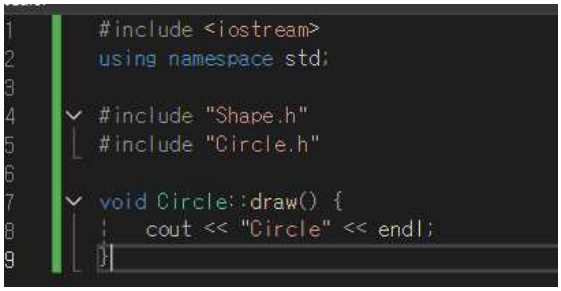
```
#include <iostream>
using namespace std;

#include "Shape.h"

void Shape::draw() {
    cout << "--Shape--" << endl;
}
Shape::Shape() {
    next = NULL;
}
void Shape::paint() {
    draw();
}
void Shape::setNext(Shape* p) {
    this->next = p->next;
}

```

Shape의 선언부와 구현부이다. 우선 paint()함수로 draw함수를 불러와서 각 도형을 화면에 띄워줄 것이다. 여기서 중요한 것이 setNext와 add인데 이것들은 밑에서 쓰이겠지만 설명을 하자면 add(주소) 이 주소를 넣어 add함수를 실행하면 next(주소값)을 p(새로운 주소)값으로 바꾸는 것이다. 자세한 것은 밑에서 설명하겠다.



```cpp
#include <iostream>
using namespace std;

#include "Shape.h"
#include "Circle.h"

void Circle::draw() {
    cout << "Circle" << endl;
}
```

Circle의 구현부이다. 간단히 Circle을 출력한다. 나머지 Line, Rect 또한 cout 안의 Circle이 Line, Rect로 변하는 것 이외에는 다를 것이 없으므로 Circle만 사진을 첨부하겠다.

```cpp
#include <iostream>
using namespace std;

#include "Shape.h"
#include "Circle.h"
#include "Line.h"
#include "Rect.h"
#include "UI.h"
#include "GraphicEditor.h"

GraphicEditor::GraphicEditor() {
    pStart = NULL;
    count = 0;
}

void GraphicEditor::create(int num) {
    switch (num) {
    case 1:
        if (count == 0) {
            pStart = new Line();
            pLast = pStart;
        }
        else
            pLast = pLast->add(new Line());
        count++;
        break;

    case 2:
        if (count == 0) {
            pStart = new Circle();
            pLast = pStart;
        }
        else
            pLast = pLast->add(new Circle());
        count++;
        break;

    case 3:
        if (count == 0) {
            pStart = new Rect();
            pLast = pStart;
        }
        else
            pLast = pLast->add(new Rect());
        count++;
        break;

    }
}
```

마지막으로 GraphicEditor의 구현부이다.

우선 create함수부터 보겠다. 이 함수는 UI의 1번을 구현하는 함수이다. 입력받은 num을 이용하여 switch case를 이용하여 1(Line),2(Circle),3(Rect)를 만들었다.

case 1의 경우 if문을 사용하여 count가 0이면 (처음으로 도형을 만드는 것이라면) pStart에 new Line()을 통해 Line을 만들어준다. 그러면 pStart에 Line이 저장되고 pLast 또한 Line

이 저장된다.

```
void GraphicEditor::indelete(int num) {
    Shape* p = pStart;
    Shape* del = pStart;

    if (num < count) {
        for (int i = 0; i < num; i++) {
            p = del;
            del = del->getNext();
        }
        if (num == 0)
            pStart = p->getNext();
        else
            p->setNext(del);
        count--;
        if (count == 1) {
            pLast = pStart;
        }
        delete del;

    }
    else
        cout << "인덱스를 잘못 입력하셨습니다." << endl;
}

void GraphicEditor::call() {
    bool exit = true;
    cout << "그래픽 에디터입니다." << endl;
    while (exit) {
        switch (menu()) {
        case 1:
            create(input());
            break;
        case 2:
            indelete(del());
            break;
        case 3: {
            Shape* p = pStart;
            for (int i = 0; i < count; i++) {
                cout << i << ": ";
                p->paint();
                p = p->getNext();
            }
            break;
        }
        case 4:
            exit = false;
            break;
        }
    }
}
```

다음으로는 indelete이다. UI의 2번에 해당하는 함수로 만든 도형을 삭제시키는 함수이다.
이 함수에서 p에 pStart del에 pStart를 넣는다. 그 후 if문에서 num이 < count 일 때 del을 삭제하는 것인데 I=0부터 num-1까지 반복시켜서 del을 삭제시키려는 도형의 주소까지 반복하여 getNext 해준다. 이 때 num이 0이면 pStart = p->getNext를 통해 pStart값을 다음 값으로 바꾸어 준다.(삭제되기 때문에 다음 값이 초기 값이 될 거이므로 다음 값으로 넘겨주는 것)
만약 num이 0이 아니라면 p=p->getNext()를 통해 삭제될 도형의 다음 도형으로 p를 설정하여 이전 도형들과 순번이 이어지게 만들어준다. 만약 이를 하지 않으면 삭제된 도형 이후 도형들 또한 연결이 다 끊어지게 됨.
그리고 count가 1일 경우 도형이 한 개이므로 pLast=pStart;로 초기의 값을 가르키게한다.
다음으로 call함수이다. 이 함수로 지금까지 만든 함수들을 사용하여 시스템을 구축할 것이다. 우선 cout로 그래픽 에디터라고 출력해주고 while문을 사용하여 exit이 false가 되면 깨지게 while문을 만들었다. 1번은 위의 input 함수를 사용하고 2번은 indelete를 썼다. 3번은 지금까지 만든 도형을 보여주는 함수로 p를 pStart으로 설정하여 처음으로 만든 p를 paint해준다. 그리고 getNext를 통해 다음 도형으로 넘어가서 count-1번 반복해 출력해준다.

4번은 exit을 false로 만들어 시스템을 종료한다.