Homework#06

문제 정의

STL 컨테이너의 종류 중 하나인 vector 컨테이너를 사용하여 이전에 코딩했던 9장 10번 문제를 변화시킨다.

문제 해결 방법

1. 원소의 저장, 삭제, 검색 등을 할 수 있는 클래스인 <vector>를 먼저 #include 시킨다

```
#include <iostream>
#include <string>
#include "Shape.h"
#include "Circle.h"
#include "Rect.h"
#include "Line.h"
#include <vector>
```

2. UI 클래스를 만들어 이전에 사용했던 메뉴, 삽입, 삭제에 대해 cin으로 값을 입력 받고 return으로 값을 도출하기 위해 cout, cin, return을 사용해 UI를 구현한다.

```
class UI {
    static int n;
public:
    static void start();
    static int menu();
    static int insert();
    static int del();
};
int UI::n = 0;
void UI::start() {
    cout << "그래픽 에디터입니다." << endl;
}
int UI::menu() {
    cout << "삽입:1, 삭제:2, 모두보기:3, 종료:4 >> ";
    cin >> n;
    return n;
}
int UI::insert() {
    cout << "선:1, 원:2, 사각형:3 >> ";
    cin >> n;
    return n;
}
int UI::del() {
    cout << "삭제하고자 하는 도형의 인덱스 >> ";
    cin >> n;
    return n;
}
```

3. GraphicEditor에 객체의 포인터를 관리하는 동적 배열인 v를 선언하고 STL 컨테이너의 요소를 순회 및 접근하는 요소인 Iterator it을 선언 해 v 벡터의 각 요소를 가리키도록 함.

```
class GraphicEditor {
    vector<Shape*> v;
    vector<Shape*>::iterator it;
```

4. 이제 삽입, 삭제, 모두보기를 구현해보자.(9장 10번 문제에 GraphicEditor에 기능이 쓰였던 그대로이니 뭐가 필요한지는 설명하지 않겠습니다)

4-1. 모든 구현은 vector 클래스의 주요 멤버와 연산자 표를 보며 구현했는데, push_back(element)로 마지막 벡터에 요소를 밀어넣을 수 있다는 걸 알고 switch case문으로 구현한 대신 if elif 문으로 구현할 수 있었다.

```
void insert() {
    int n = UI::insert();
    if (n == 1)
        v.push_back(new Line());
    else if (n == 2)
        v.push_back(new Circle());
    else if (n == 3)
        v.push_back(new Rect());
    else cout << "입력 에러" << endl;
}
```

4-2. 삭제 부분인데 이는 erase(Iterator it) (벡터에서 it을 가리키는 원소 삭제, 삭제 후 자동 배열 조정)을 이용해 v.begin()으로 첫 원소에 대한 참조 리턴부터 for 반복문으로 끝까지 돌린 후 n과 일치하는 인덱스의 원소 값을 삭제 시키는 것으로 간단하게 구현할 수 있었다.

```
void deleteShape() {
    int n = UI::del();
    Shape* del;
    it = v.begin();
    for (int i = 0; i < n; ++i)
        ++it;
    del = *it;
    it = v.erase(it);
    delete del;
}
```

4-3. 모두보기 파트인데 이도 연산자에 size() (벡터에 들어있는 원소의 개수를 리턴)를 이용해 vector는 가변 길이 배열을 구현하는 클래스이므로 0부터 v.size()-1까지 반복문을 돌려 paint()로 무엇을 그렸는지 도출시켰다.

```
void showAll() {
    for (int i = 0; i < v.size(); ++i) {
        cout << i << ": ";
        v[i]->paint();
    }
}
```

문제 해결을 위한 아이디어 도출 및 결과

솔직히 이번 문제는 아이디어라 할 것도 없고 <vector> 클래스가 하나부터 열까지 편하게 해줬던 문제였던것 같다. 아이디어 도출이 아닌 이전에 했던 코드와 비교라고 한다면..
도형의 삽입도 그렇고 삭제, 모두보기 모든 기능의 구현이 모두 편하게 이루어졌다.
하나씩 말해보자면,

1. 삽입에선 노드의 pStart를 도형으로 바꾸고 top을 pStart로 바꾸는 귀찮은 작업을 했다면 vector 클래스에서는 push_back만 해도 벡터의 마지막 원소를 추가시킨다. 이 말인 즉슨 굳이 노드를 한칸 더 움직이지 않아도 알아서 top을 조정한다는 얘기가 되므로 이전 문제의 3~4줄의 코드가 1줄로 사라지는 경험을 했다.

2.삭제에서도 마찬가지이다. 여기서도 인덱스 값을 넘겨받아 해당 인덱스에 대한 도형을 지우고 사이에 있던 인덱스 값이 지워지면 마지막에 있던 노드를 앞으로 불러오는 코드를 짰었다면 erase(Iterator it) 하나로 원소 삭제 및 자동 벡터 조절까지 시켜주니 엄청 편했다.

3.모두보기 파트에서는 그렇게 달라질건 없었지만 간편해지긴 했다.
원래 포인터 주소를 한 칸 한 칸 옮겨야하는 작업을 했다면 size()연산자를 활용해 굳이 포인터의 주소를 다음으로 넘기지 않아도 for 반복문으로 구현할 수 있는것이 편했다.