C++ language report

Deadline 2019

[201601639 컴퓨터공학부 홍승현] e-mail : cx\_nf@naver.com Contact : 010 – 4019 - 2514

목차

[클래스 다이어그램 1](#_Toc24722381)

[Main.cpp 2](#_Toc24722382)

[코드사진 및 간략도 3](#_Toc24722383)

[bool exit 4](#_Toc24722384)

[GraphicEditor editor 4](#_Toc24722385)

[정리 4](#_Toc24722386)

[Shape, Line, Circle, Rect 5](#_Toc24722387)

[private 5](#_Toc24722388)

[int shape 5](#_Toc24722389)

[Shape\* next 5](#_Toc24722390)

[protected 5](#_Toc24722391)

[Virtual void draw() = 0; 5](#_Toc24722392)

[public 5](#_Toc24722393)

[Shape(const int shape) 5](#_Toc24722394)

[Virtual ~Shape() 5](#_Toc24722395)

[void paint() 5](#_Toc24722396)

[Shape\* add(Shape\* p) 6](#_Toc24722397)

[Shape\* getNext() const 6](#_Toc24722398)

[int getShape() const 6](#_Toc24722399)

[shape.h 코드 7](#_Toc24722400)

[정리 7](#_Toc24722401)

[GraphicEditor.h 8](#_Toc24722402)

[graphiceditor.h 코드 8](#_Toc24722403)

[private 9](#_Toc24722404)

[Sharp\* startNode 9](#_Toc24722405)

[Sharp\* peek 9](#_Toc24722406)

[Int size[4] = { 0, } 9](#_Toc24722407)

[bool Error(const string& errorcode, const int size\_start, const int size\_end) 9](#_Toc24722408)

[public 9](#_Toc24722409)

[GraphicEditor() 9](#_Toc24722410)

[~GraphicEditor() 9](#_Toc24722411)

[int Menu() 9](#_Toc24722412)

[void Input 9](#_Toc24722413)

[void Delete 9](#_Toc24722414)

[void AllShow() const 10](#_Toc24722415)

[void Statistic\_Information() 10](#_Toc24722416)

[정리 10](#_Toc24722417)

[GraphicEditor.cpp 알고리즘, 코드 설명 11](#_Toc24722418)

[graphiceditor\_cpp 코드 11](#_Toc24722419)

[정리. 11](#_Toc24722420)

[프로그램 간략도 12](#_Toc24722421)

[Menu코드 설명 및 간략화 12](#_Toc24722422)

[Input 코드 설명 및 간략화 12](#_Toc24722423)

[Delete코드 설명 및 간략화 13](#_Toc24722424)

[AllShow 코드 설명 및 간략화 14](#_Toc24722425)

[Statistic\_Information 코드 설명 및 간략화 15](#_Toc24722426)

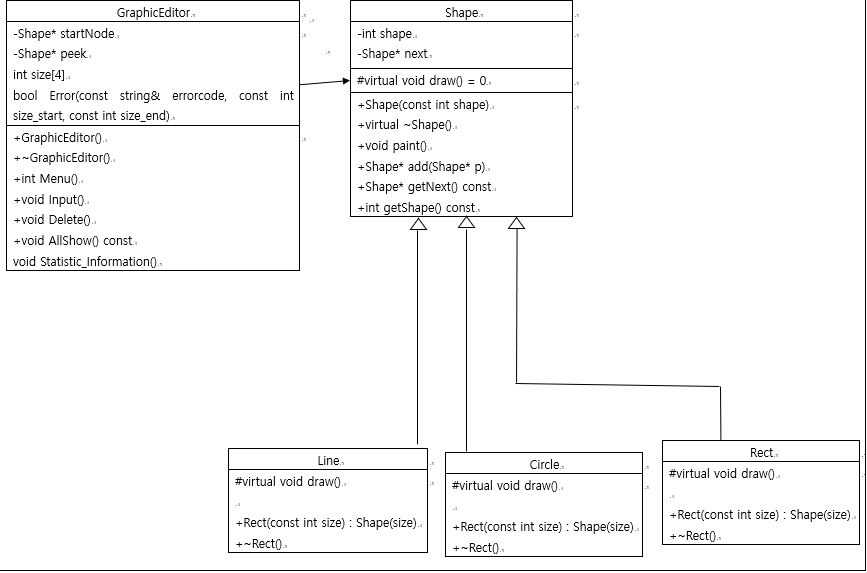
[Error 코드 설명 및 간략화 15](#_Toc24722427)

[출력결과 17](#_Toc24722428)

[이 과제를 함으로써 느꼈던 점 19](#_Toc24722429)

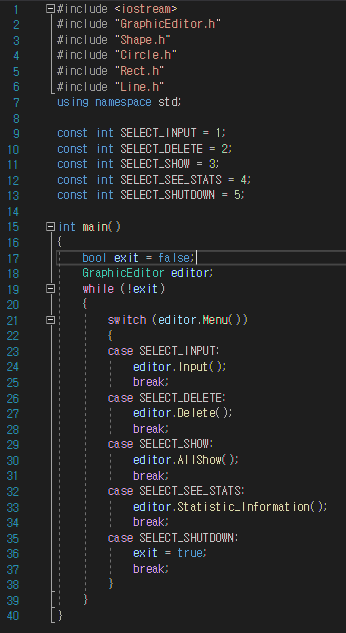
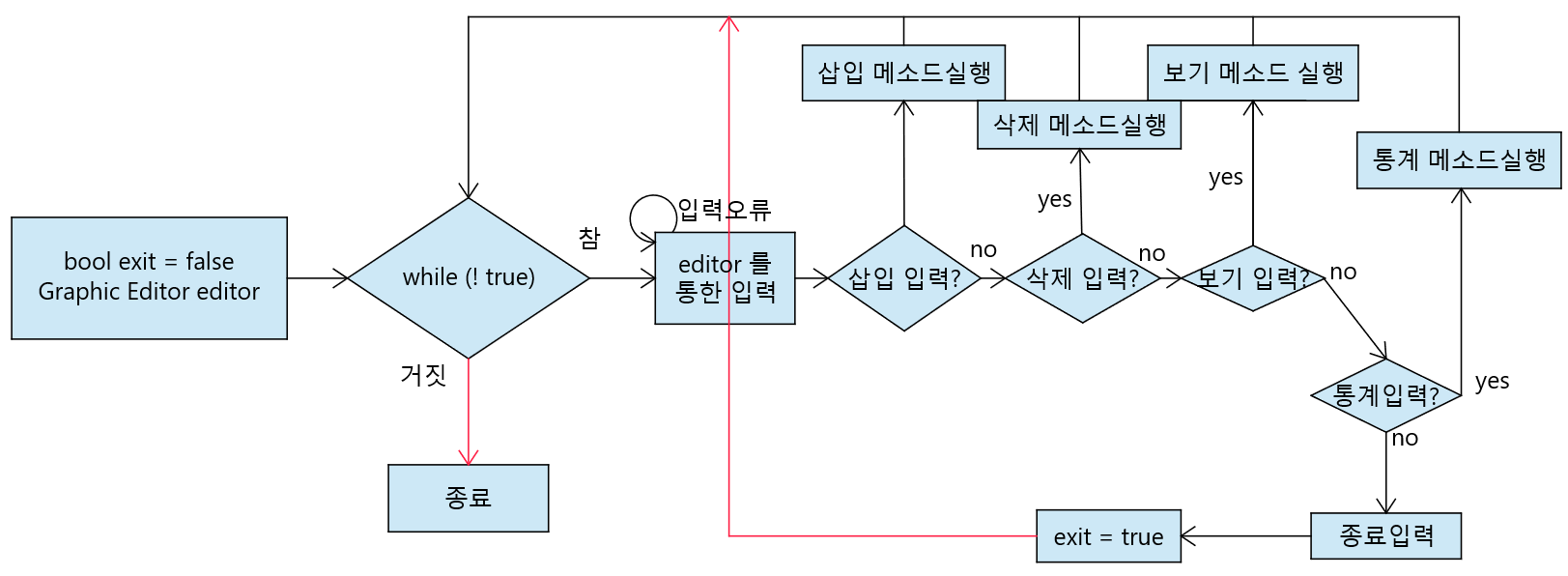
[재미있었고, 공부에 많은 도움이 되었다 19](#_Toc24722430)

# 클래스 다이어그램



# Main.cpp

## 코드사진 및 간략도

<코드 (왼쪽그림)> <간략도 (아래그림)>

### bool exit

후에 반복문을 빠져나가게 하는 조건문으로, 에디터 종료가 필요할 시, exit의 부울값을 바꾸어 반복문을 빠져나가게 한다.

### GraphicEditor editor

그래픽에디터의 객체로, 전반적인 코드구성이 GraphicEditor에 포진되어있다.

## 정리

간략도와 똑같이 실행되며, 코드에서 직관적인 코드 구성을 위해 const int형 변수를 전역으로 선언하였다.  
(Case 0: 보다 case SELECT\_INPUT: 이 보다 더 직관적이라 생각함.)  
해당 클래스에 대한 자세한 설명은 후에 게시하겠음.

# Shape, Line, Circle, Rect

## private

### int shape

어느 도형(선, 원, 사각형)인지를 결정해 주는 변수이다. 생성자를 통해 초기화가 되며, 그 이후로는 바뀌지 않는 변수이다.

### Shape\* next

도형 선언 후 다음 도형을 가리키기위한 shape 포인터형 변수.  
연결리스트를 통해 구현되었다.

## protected

### Virtual void draw() = 0;

도형을 출력하기 위한 순수가상함수이며, 각 상속받는 파생클래스에서 재정의해서 사용한다.

## public

### Shape(const int shape)

해당 클래스의 생성자이며, 다음 도형을 가리키는 next 의 주소값을 nullptr로 지정해주고, 도형을 가리키는 shape변수를 초기화해준다.

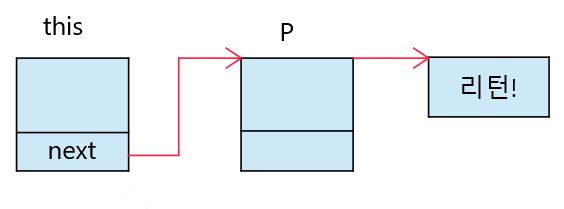
### Virtual ~Shape()

해당 클래스의 소멸자이며, 업캐스팅을 통한 올바른 소멸을 위해 가상 함수로 지정해주었다.

### void paint()

도형을 출력하기 위한 메소드로, draw 메소드 내에서 실행된다. 동적바인딩이 일어나는지 확인하기 위한 출력메소드이다.

### Shape\* add(Shape\* p)

본 클래스의 들어있는 next 변수를 p에 가리키게 만들고, 그것을 리턴시키는 메소드이다. 그림으로 설명하자면 다음과 같다.

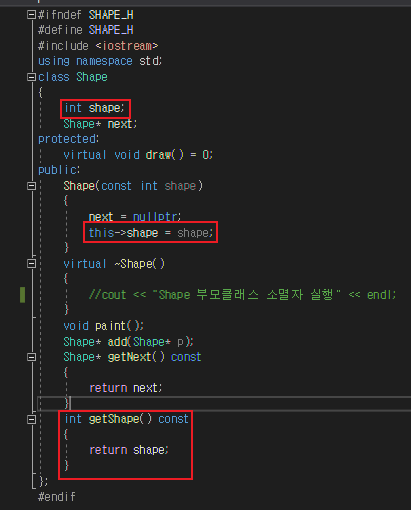
### Shape\* getNext() const

해당 클래스의 Shape\* 객체인 next를 불러온다.

### int getShape() const

해당 클래스를 업캐스팅한 도형이 무슨 도형인지를 불러온다.

## shape.h 코드

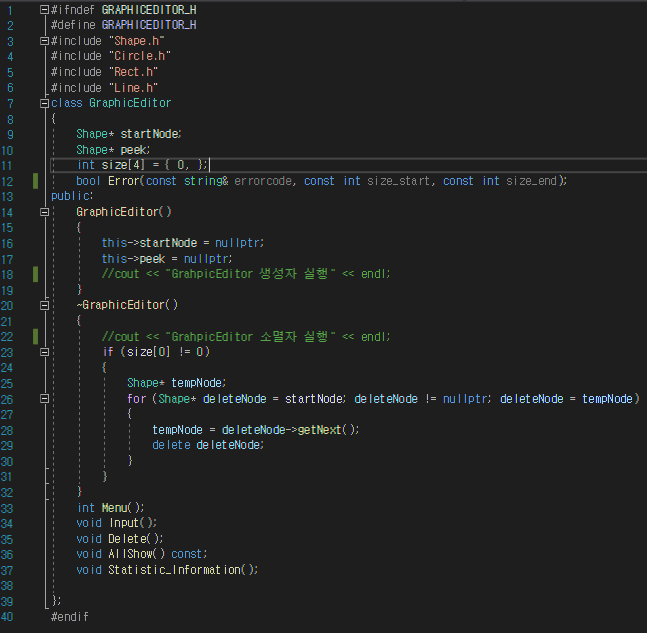


## 정리

나머지 모든 헤더랑 cpp는 과제 내 이미지랑 같이 draw를 오버라이딩 하였으며, 위 표시해둔 부분만 추가하였다.  
통계정보를 출력하기위해 shape 변수를 만들었으며, Line을 동적할당하면 shape 변수는 1, Circle이면 2, Rectangle이면 3으로 만든다.

# GraphicEditor.h

## graphiceditor.h 코드



## private

### Sharp\* startNode

연결리스트의 첫번째 노드이다.

### Sharp\* peek

연결리스트의 맨 마지막 노드를 가리킨다.

### Int size[4] = { 0, }

각 도형의 개수에 대한 정보를 저장하는 배열이다. 각 도형의 수는 3개이고 총 도형의 개수에 대한 합까지 해서 총 4개의 요소를 가지고 있다.

### bool Error(const string& errorcode, const int size\_start, const int size\_end)

과제 1에서 사용했다시피 입력에 대한 오류를 검출하는 메소드로, 시작과 끝을 파라미터로 입력받아 그 사이에 숫자가 입력되지 않으면 오류로 처리를 한다.

## public

### GraphicEditor()

해당 클래스의 생성자로, startNode와 peek를 nullptr로 초기화 시키는 역할을 한다.

### ~GraphicEditor()

해당 클래스의 소멸자로, 나열되어있는 도형이 모두 삭제되지 않고 종료할 시 동적할당된 도형들의 메모리를 해제시켜주는 작업을 한다.

### int Menu()

전체 매뉴에 대한 가시성을 출력하고, 그에 대한 입력을 받아 반환하는 메소드이다.

### void Input

해당 도형을 연결리스트로 넣을 때 발생하는 메소드이다.

### void Delete

연결리스트로 발생된 도형들 중 하나를 삭제하려고 할 때 실행되는 메소드이다.

### void AllShow() const

나열되어 있는 도형들을 순서대로 출력시키는 역할을 한다.

### void Statistic\_Information()

나열되어 있는 도형중에서 어느 도형인지, 총 도형의 개수는 몇 개인지를 보여주는 메소드이다.

## 정리

Private 공간에는 Shape 포인터 형의 변수 2개와 int형 배열 변수, 그리고 잘못된 입력을 방지하기위한 에러 메소드를 넣었으며, public 공간에는 메뉴선택창에서 입력을 받을 Menu() 메소드, 도형을 삽입하기 위한 Input() 메소드, 들어가 있는 도형을 삭제하기 위한 Delete()메소드, 현재 도형에 관련하여 보여주는 AllShow() 메소드, 어느 도형이 있는지 통계를 보여주는 Statistic\_Information() 메소드가 있다.

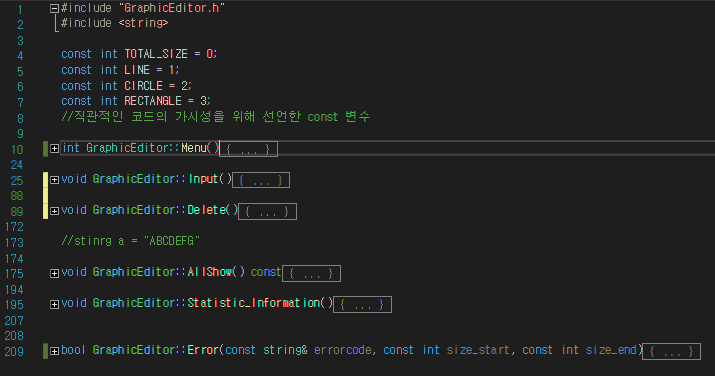
배열 size의 각 요소는 이렇게 된다.

1. 총 도형의 개수
2. 선의 개수
3. 원의 개수
4. 사각형의 개수

이는Input을 통해 각자의 동적할당에 대해 개수도 증가시키고, Statistic\_Information 메소드 내에서도 출력을 위해 사용하게 된다. 이는 뒤에 설명하겠다.

# GraphicEditor.cpp 알고리즘, 코드 설명

## graphiceditor\_cpp 코드



## 정리.

직관적 코드를 위해 const변수를 선언했으며 해당 cpp파일 내에 모든 곳에 사용된다.

Menu는 메뉴선택창을 출력하고 보기 중에서 하나를 선택한다 했을 때 입력 & 에러 검출 후에 올바른 값을 반환한다.

Input은 도형을 삽입한다 했을 때, 실행되는 메소드로 여러 도형을 순서대로 집어넣는 역할을 한다.

Delete는 나열되어있는 도형들을 삭제하려 할 때 실행되는 메소드로 첫번째 도형이든 중간 즈음의 도형이든 끝에 있는 도형이든 하나의 인덱스를 선택하여 삭제할 수 있다.

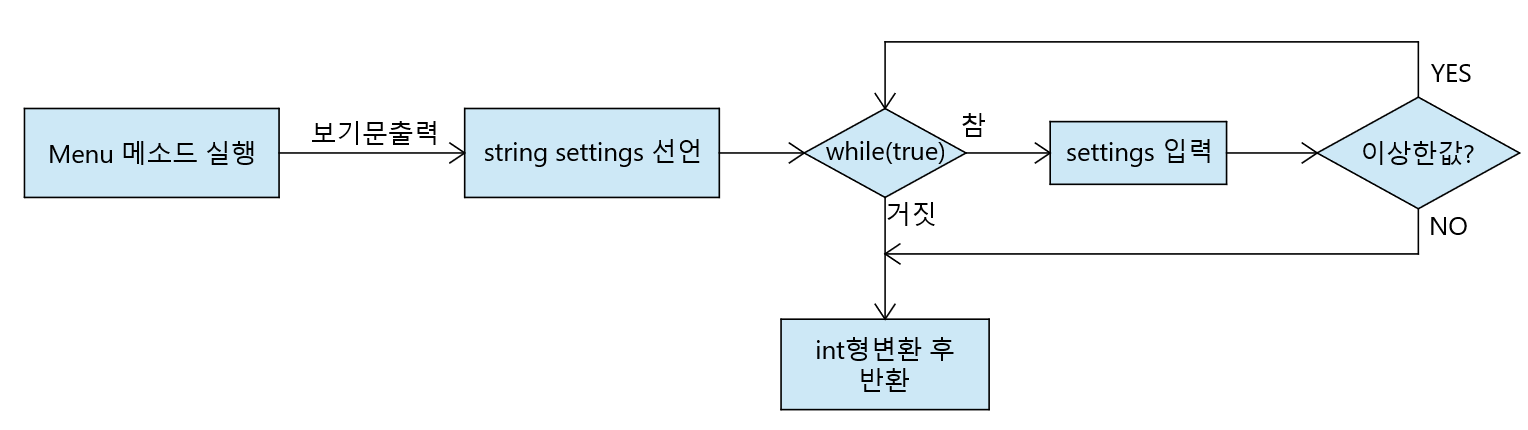
AllShow는 말 그대로 나열 되어있는 도형들을 순서대로 출력해주는 역할을 한다.

Statistic\_Information은 나열 되어있는 도형 중에서 어느 도형인지를 출력해주는 역할을 한다.

Error는 입력받은 값이 원하는 값인지 판별하는 메소드이며 원하는 값이 아닐 경우 에러가 검출된 것이므로 true를 반환하고, 원하는 값인 경우에는 에러가 발생되지 않았으므로 false를 반환한다.

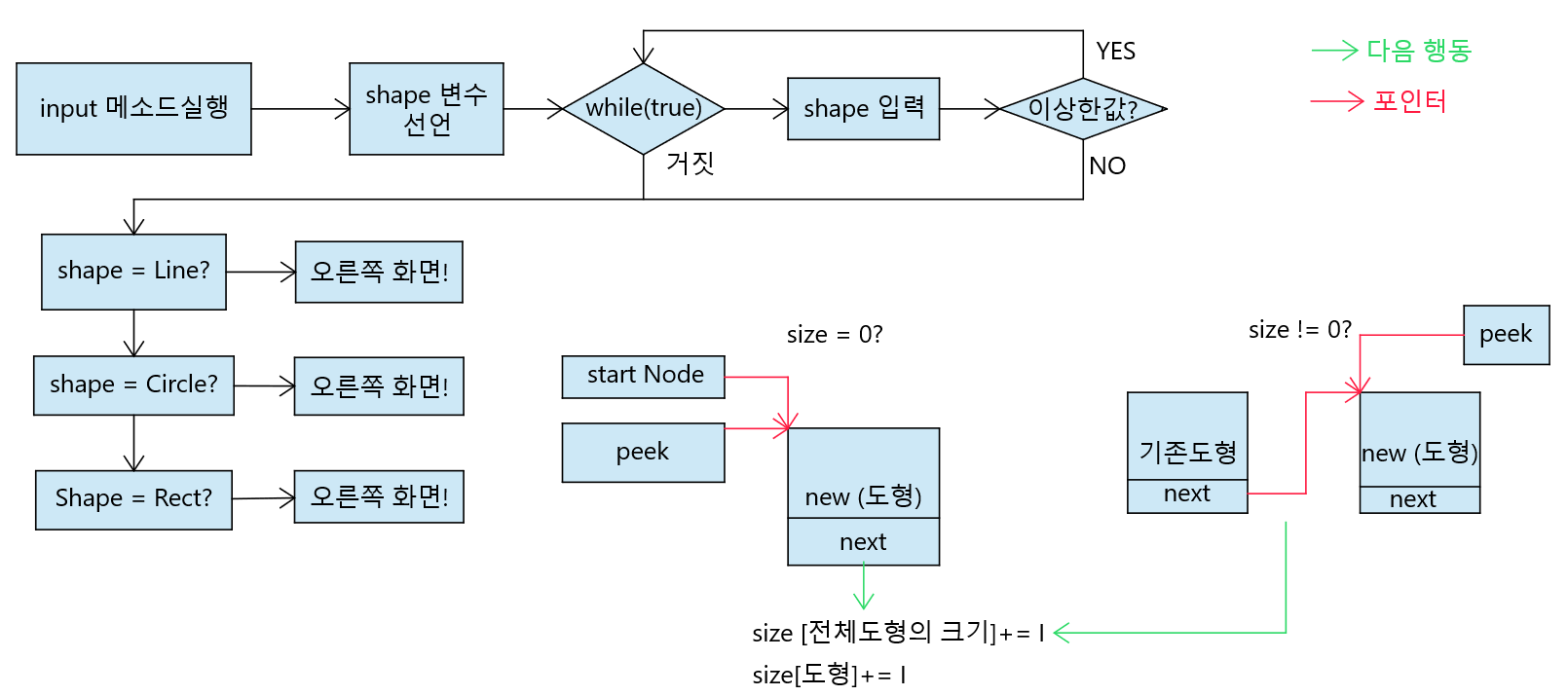
## 프로그램 간략도

### Menu코드 설명 및 간략화



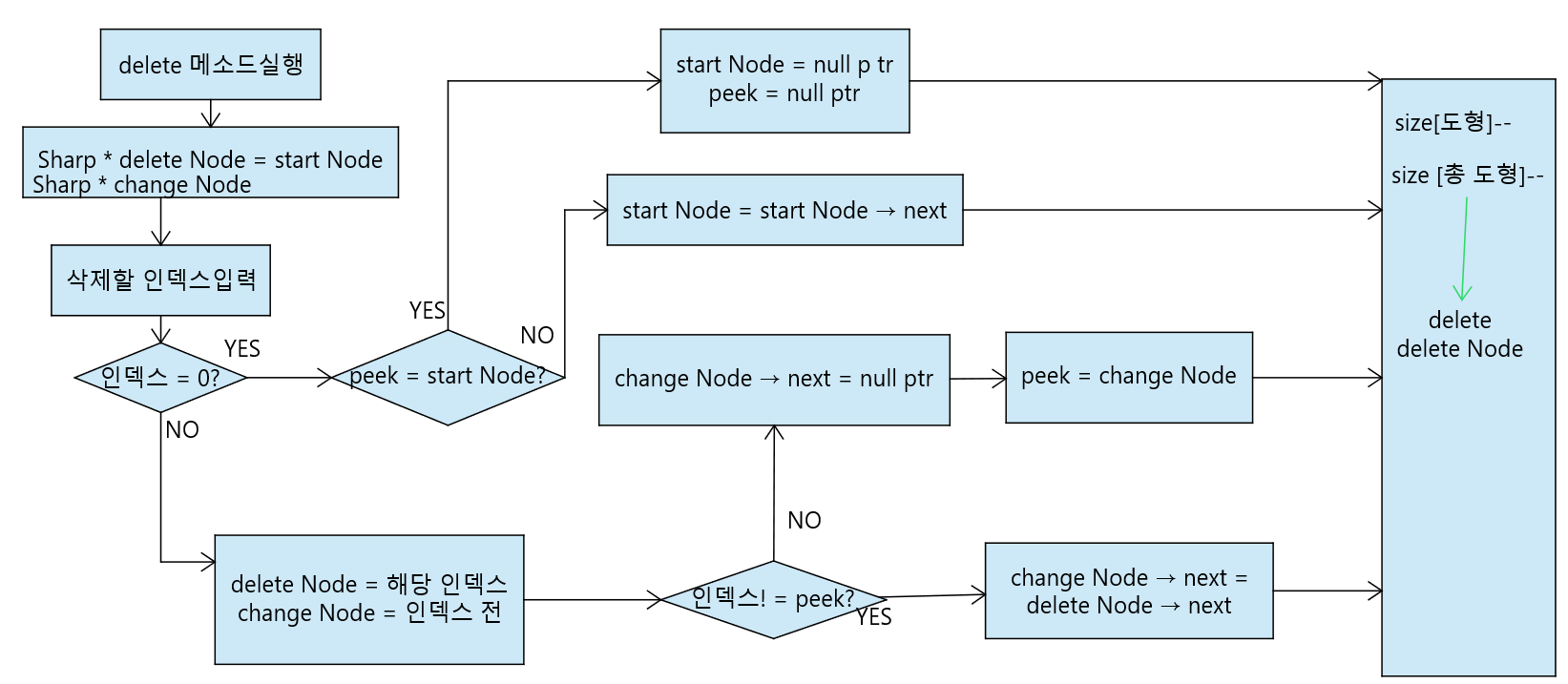
while문은 Error 메소드를 출력하는 에러검출 반복문이다. 그 외에는 코드랑 같다.

### Input 코드 설명 및 간략화



Input메소드를 통해 입력받으면, 도형이 선인지, 원인지, 사각형인지 판별 후에 동적할당으로 넣게된다. 이 때 도형이 나열되어 있지 않은 경우(size == 0)와 나열되어 있는 경우(size != 0)을 구분해서 보아야 한다. (각자 삽입하는 방식이 다르기 때문!)

### Delete코드 설명 및 간략화

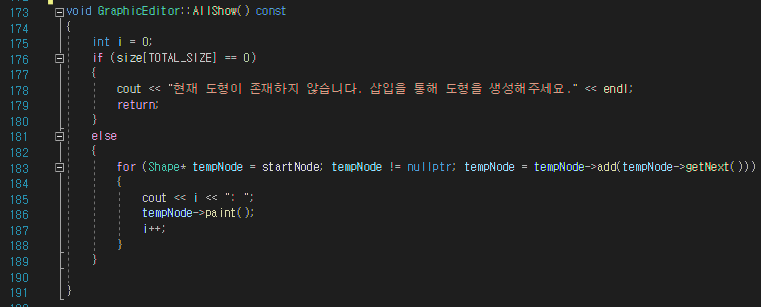


인덱스(이하 노드라고 하겠음)를 삭제할 때 고려해야할 요소가 4가지가 있다.

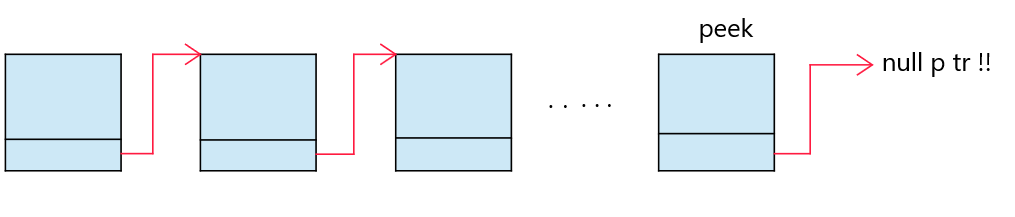
1. 삭제하려는 노드가 첫 노드인데, 첫 노드가 peek일 때
2. 삭제하려는 노드가 첫 노드이고 첫 노드가 peek이 아닐 때
3. 삭제하려는 노드가 첫 노드가 아니지만 peek일 때
4. 삭제하려는 노드가 첫 노드도 아니고 peek 도 아닐 때

이 4가지 요소에 따라 제거하는 형식도 위 간략도와 같이 달라진다. 이는 코드상에도 구현이 잘 되어 있다.

### AllShow 코드 설명 및 간략화

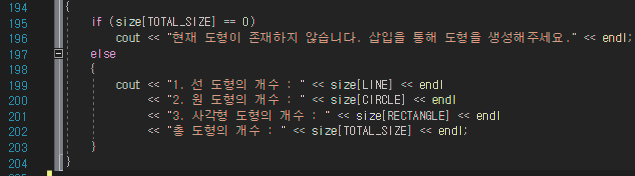


for문안에 초기값을 스타트 노드로 두고 next만큼 넘어가면서 nullptr를 만나기 전까지 반복한다. 이말은 즉,



위 간략도 처럼 첫 노드부터 peek까지를 for문을 통해 반복하겠다는 코드라고 볼 수 있다.

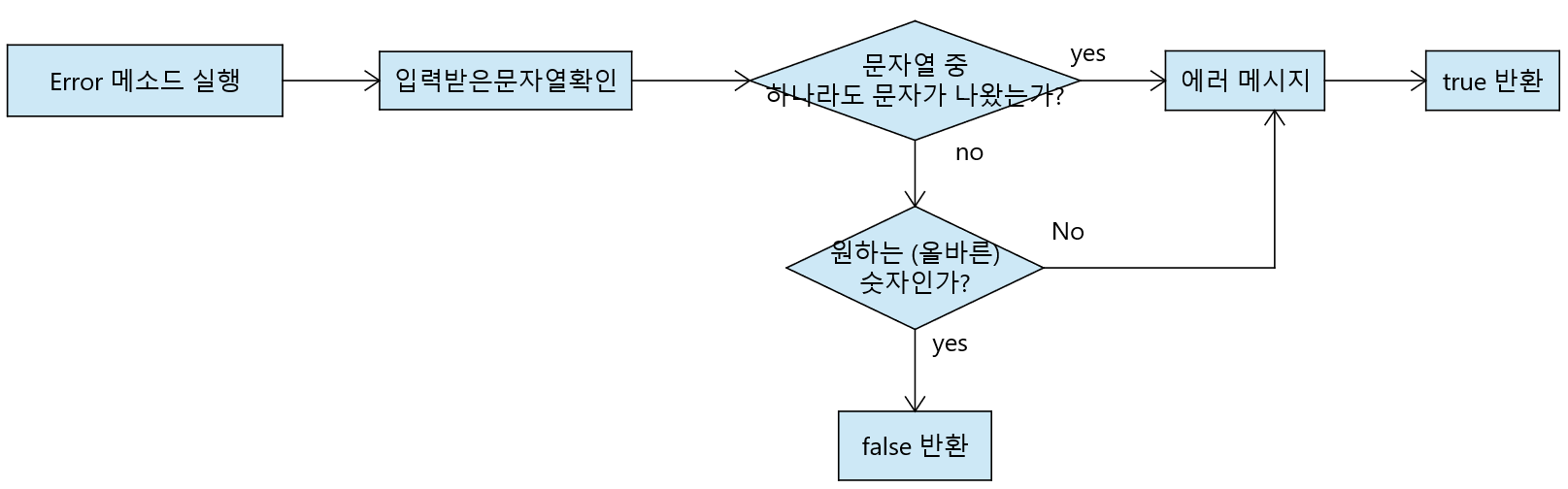
### Statistic\_Information 코드 설명 및 간략화



GraphicEditor설명에서 언급했던 대로 각 개수를 출력해주는 메소드이다. 간단한 메소드이므로 자세한 설명은 넘어가겠다.

### Error 코드 설명 및 간략화

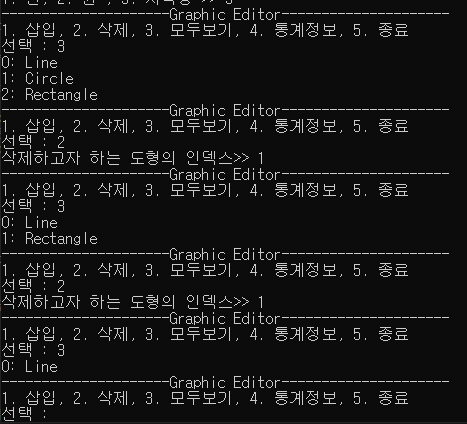
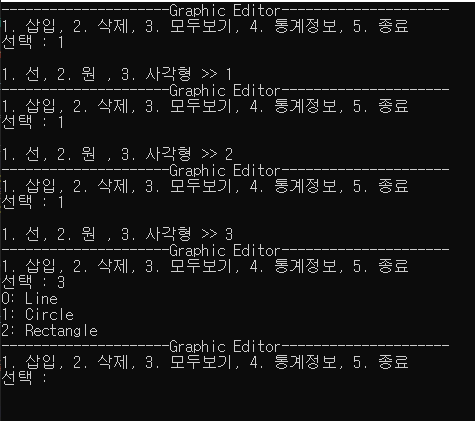


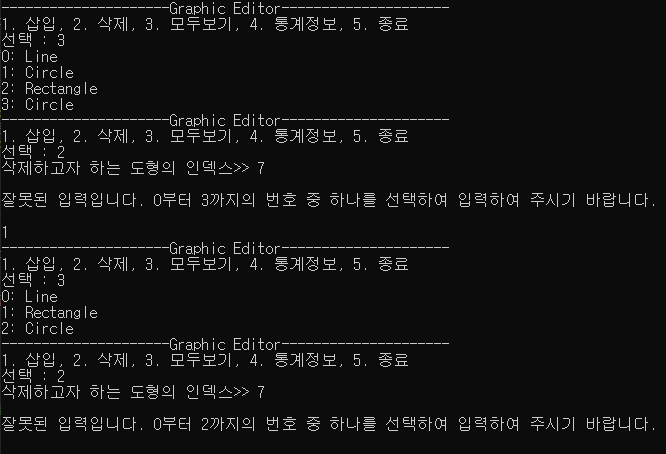
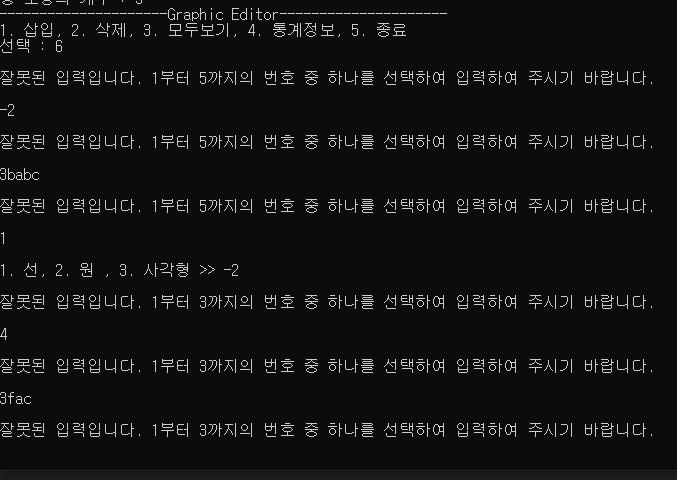
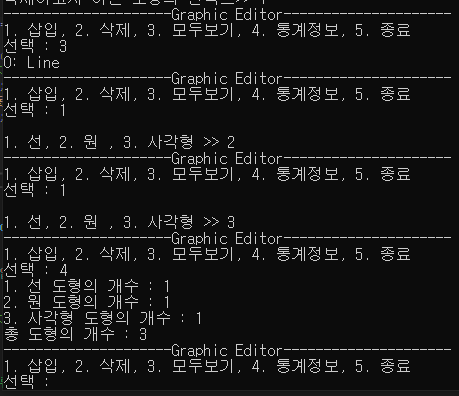


프로젝트1에서의 에러검출이랑은 살짝 다르게 풀어서 만들었다.

그 이유는 1부터가 아닌 0부터 입력을 받을 수도 있고, 10개 이상의 도형을 삽입시, 프로젝트 1에서의 에러검출로 하게 되면은, 무조건적으로 2자리수 이상의 문자는 에러가 검출되게끔 해놓았으므로 오류가 발생하게 된다. 따라서 Error코드에 하나의 매개변수를 더 넣어주어 자세한 입력이 가능하게끔 만들었다

## 출력결과





## 이 과제를 함으로써 느꼈던 점

### 재미있었고, 공부에 많은 도움이 되었다

1. 연결리스트 알고리즘을 통해 만드는 과제였다.

작성자에게 있어 정말 푸는게 좋았다고 느꼈는데, 복학 전부터 항상 알고리즘에 대해 고민하고, 공부했던 탓이라 그런가 알고있는 알고리즘이 나와서 많이 행복했었다 ^^.. 실제로 연결리스트를 통해 스택을 구현해보기도 하였기에, 과제를 하면서 복습 차원으로 하는 느낌이 들어 더더욱 연결리스트의 의미를 굳게 이해하게끔 만들었던 시간이 아니였다 싶다.

1. 항상 알고리즘에서는 예외처리가 참으로 많다고 느낀다.

몇 달 전 연결리스트로 스택을 구현하려고 했을 때 많이 애먹었던 느낌도 났었고, 그 때 기억을 토대로 예외처리를 해서 보다 쉽게 풀었던 감이 없지 않아 있다. (과거의 가르침을 현재에 적용시킨달까) 물론 예외가 있고 처리하지 못한 것이 있을 수도 있다. 하지만 이런 점을 생각하면서 찾으려고 노력하고, 찾아서 해결하는 그 행동에서 즐거움이 비롯된다고 느낀다.

참으로 교수님께서 과제는 재미나게 내주시는 것 같아 항상 코딩할 때마다 즐겁다. :D

프로젝트 2(과제 2) 보고서 종료

작성자 : 201601639 홍승현