**2022 Spring OOP Assignment Report**

과제 번호 : ASSN4\_prob1

학번 : 20210716

이름 : 최대현

Povis ID : daehyeonchoi

**명예서약 (Honor Code)**

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

프로그램을 하다 보면 결정해야 할 세부 사항이 많은데, 이러한 세부 사항을 처리한 방법과 이유를 보고서에 쓰십시오.

독창적인 아이디어와 추가 기능은 보너스 점수를 받을 수 있으므로, 보고서에 명확히 기재하십시오.

문제가 여러 개인 경우, 각 문제별로 정리해서 작성합니다.

1. **프로그램 개요**

개발환경: Visual Studio 2022

Template을 통한 overriding으로 generic programming을 구현하는 과제이다. 이 문제의 input data의 경우 (유효한 입력) Int와 float이 들어올 수 있는데, 지금까지는 int와 float 입력에 대한 함수를 따로 구현해줘야 했다면 이번 과제에서는 기존에 배웠던 template을 이용해서 여러 datatype에 대해 일반화할 수 있도록 구현하였다. 또한, 이번 과제에서는 Stack, Queue, Vector 등 모든 STL 을 사용할 수 있었고, 그 중에서 2번째 과제에서 직접 구현해보기도 했던 Vector를 직접 사용해보았다.

1. **프로그램의 구조 및 알고리즘**

.

**Vector 자료형**

C++의 vector는 C++ 표준 라이브러리(Standard Template Library)에 있는 컨테이너로 사용자가 사용하기 편하게 정의된 class이다. vector를 생성하면 메모리 heap에 생성되며 동적할당된다. 시간적인 측면에서 array에 비해 성능은 떨어지지만 메모리를 효율적으로 관리하고 예외처리가 쉬우며, 템플릿으로 정의되어서 다양한 자료형에 대해 사용할 수 있어서 빈번하게 사용된다.

1. **Geometry.h (Geometry class, Rectangle class, Point class)**

1번 문제에서는 2가지의 도형이 입력으로 들어올 수 있다. Point, rectangle로, 각각에 대해 다른 함수를 호출해줘야 한다. 이것을 virtual function인 draw2canvas로 geometry class에 구현해두고, 후에 이것을 상속받은 rectangle, point class에서 재정의하여 사용하였다.

**Geometry class:** 후에 point class와 rectangle class가 이것을 상속받음.

텍스트이(가) 표시된 사진

자동 생성된 설명

상속한 클래스에서 편리하게 사용하기 위해 모두 public class에 구현하였다.

**geometry():** 기본 생성자, 별도로 구현하지는 않았다.

**virtual ~geometry():** virtual로 구현한 가사 소멸자

**virtual void draw2canvas(float\*\* board, int \*\*isfill, int height, int width) = 0:**

virtual로 구현한 가상 함수, canvas 속 board에 값을 표시해주는 데 사용된다.

**float dimarr[5]={}:** 차원값이 2 이상 5 이하이므로 크기가 5인 배열로 만들어준 뒤 0으로 초기화해줬다.

**int dims = 5;**

텍스트이(가) 표시된 사진

자동 생성된 설명

**Point class**

**point<T, N>():** 기본 생성자, 별도로 구현하진 않았음.

**~point():** 소멸자

**point <T, N>(const T\*):** 차원 배열을 가져와서 값을 넣어주는 복사 생성자

**void draw2canvas(float \*\*board, int \*\*isfill, int height, int width):** 앞서 geometry class에서 가상함수로 구현했던 draw2canvas 함수로, board와 isfill이라는 2개의 2차원 배열을 가져와서 값을 넣어준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**Rectangle class**

**rectangle<T>(const T&, const T&, const T&, const T&):** 생성자, 각각 x1,y1,x2,y2 순으로 받아와서 저장한다. 이 때, 사각형을 만들 수 있는 2가지 케이스가 있는데, 나중에 위치관계를 나타내는 함수를 표현하기 위해 항상 두 x값 중 작은 값을 x1에 넣어주었고, 두 y값 중 작은 값을 y1에 넣었다.

**rectangle<T>():** 기본 생성자, 별도로 구현하진 않음

**~rectangle():** 소멸자

다음 5개의 함수는 exclusive mutual한 관계에 있는 함수들로 두 사각형의 관계에 대해 5가지의 함수 중 1개만 참이 될 수 있다.

**bool is\_equal(const rectangle<T>&):** 두 사각형이 완전히 같은지 나타내는 함수. 4가지 매개변수 값이 모두 같으면 true를 리턴했다.

**bool is\_intersect(const rectangle<T>&):** 두 사각형이 겹치는지 나타내는 함수, x\_overlap 함수값과 y\_overlap 함수값이 모두 참이면 true를 리턴했다.

**bool is\_line\_overlap(const rectangle<T>&):** 두 사각형이 한 변을 공유하는지 나타내는 함수이다.

**bool is\_point\_match(const rectangle<T>&):** 두 사각형이 한 점을 공유하는지 나타내는 함수

**bool is\_disjoint(const rectangle<T>&):** 두 사각형이 만나지 않음을 나타내는 함수. 위의 4가지 함수를 모두 호출해서 참인 경우 false를 return하고, 모두 참이 아닌 경우 true를 return했다.

**void draw2canvas(float \*\*board, int \*\*isfill, int height, int width)** 앞서 geometry class에서 가상함수로 구현했던 draw2canvas 함수로, board와 isfill이라는 2개의 2차원 배열을 가져와서 값을 넣어준다.

**bool x\_overlap(const rectangle<T>& other):** this와 other이 x축에 대해 겹치는지 판정하는 함수이다. 즉, this의 x1 | x2 가 other의 x1보다 크고 x2보다 작은 경우 true를 return한다. 이 함수가 참이라고 해서 겹치는 것은 아니다. Y축 부분도 똑같이 보아야 하기 때문 (x축과 수직하게 직선을 그었을 때 그것이 overlap 관계에 있을 수는 있지만 실제로는 disjoint 할 수 있다)

**bool y\_overlap(const rectangle<T>& other):** this와 other이 y축에 대해 겹치는지 판정하는 함수이다. 즉, this의 y1 | y2 가 other의 y1보다 크고 y2보다 작은 경우 true를 return한다. 이 함수가 참이라고 해서 겹치는 것은 아니다. Y축 부분도 똑같이 보아야 하기 때문 (y축과 수직하게 직선을 그었을 때 그것이 overlap 관계에 있을 수는 있지만 실제로는 disjoint 할 수 있다)

1. **Canvas2d.h**

텍스트이(가) 표시된 사진

자동 생성된 설명

**canvas2d(int width, int height, char c\_empty);**

**~canvas2d():** 기본 소멸자, 메모리 관리를 위해 멤버 변수인 board와 isfill의 동적할당 해제를 구현했다.

**void draw2stdout():** geometry 포인터 vector를 상속받기 때문에, 이것을 시작값부터 끝까지 받아와서 차례대로 draw2canvas 함수를 호출한다.

**int width:** 보드의 width

**int height:** 보드의 height

**char c:** 보드의 빈 공간을 채울 문자

**float\*\* board:** 차원값을 담을 실제 보드, 출력할 때 형변환을 통해 int로 출력할 수 있으므로 더 포괄적인 datatype인 float로 구현함.

**int\*\* isfill:** 해당 인덱스에 값이 채워져있는지를 나타내는 배열

텍스트이(가) 표시된 사진

자동 생성된 설명

1. **Parser class**

**canvas2d parse\_canvas(const std::string&):** string 쪼개서 board 그리는 함수. <sstream>을 include하여 관련 클래스인 istreamstring 자료형을 사용했고, getline 함수로 ‘,’ 단위로 문자열을 쪼개서 간단하게 해결했다.

**geometry\* parse\_geometry(const std::string&):** string을 쪼개서 point, rectangle 등 geometry를 상속받는 class를 만들어 리턴하는 함수. 마찬가지로 <sstream>을 include하여 관련 클래스인 istreamstring 자료형을 사용했고, getline 함수로 ‘,’ 단위로 문자열을 쪼개서 간단하게 해결했다. Rectangle일 때와 point일 때를 다르게 만들어야 하기 때문에 if문 안에서 지역적으로 리턴을 해야 했고, 따라서 메모리를 stack에 저장하는 것이 아니라 동적할당을 통해 heap에 저장했다.

1. **토론 및 개선**

2차원 배열을 동적할당하고, 이를 소멸자에서 할당 해제해주면서 메모리 사용을 효율적으로하기위해 노력했다. 보드가 채워져있는지를 확인하는 isfill 배열을 bool형으로 구현했으면 더 c++스러운 프로그램이 되었을 것 같다. 또한, 배열을 2개 선언해서 불필요하게 메모리를 많이 사용했다는 느낌이 드는데, 보드를 1개로 만들어서 할 수 있는 방법에 대해 조금 더 생각해보고 싶다.

Class 상속과 Virtual function을 이용한 함수의 overriding을 실제로 사용해보고, stl 중 하나인 vector의 사용버베 대해 익힐 수 있었다. 또한 sstream.h를 include하여 parser을 구현할 때 string을 간단하게 잘라서 사용할 수 있었다.

**문제3> canvas2d class 처럼 STL vector를 상속받을때 예상되는 단점을 간단하게 논하시오.**

어떤 클래스를 상속받을 때, 상속을 당하는 부모클래스에서는 자신을 상속하는 자식 클래스를 배려하여 virtual function을 자주 만든다. 예를 들어, 이번 과제에서도 geometry class 속 draw2canvas 함수를 virtual로 구현했다. 그 뒤 point와 rectangle 클래스에 대해서 draw2canvas함수를 재정의해서 overriding했다. Virtual function으로 만들어야 하는 중 대표적인 함수가 바로 소멸자인데, STL Vector의 소멸자는 virtual function이 아니다. 따라서 메모리 누수가 필수불가결적으로 발생한다. 즉, Vector을 상속했을 때 그 클래스를 ‘new’ keyword로 동적 할당했을 경우 소멸자를 부를 때 vector에 대한 메모리 할당 해제가 발생하지 않는다. 따라서, Vector을 포함한 STL을 상속하는 것은 그리 좋지 않은 문법이다.

**4. 참고 문헌**