자료구조 HW1

B711222 박조은
Hongik University
mrnglory@mail.hongik.ac.kr

October 12, 2019

I. List of Source Files

- hw1a
  - recta.h
  - recta.cpp
  - hw1a.cpp
- hw1b
  - rectb.h
  - rectb.cpp
  - hw1b.cpp

II. hw1a

i. recta.h

```
1  #ifndef RECTANGLE_H
2  #define RECTANGLE_H
3  class Rectangle{
4  public:
5          Rectangle(int, int, int, int);
6          void Print();
7          bool LessThan(Rectangle&);
8          bool Equal(Rectangle&);
9          int GetHeight();
10         int GetWidth();
11 private:
12         int xLow, yLow, height, width;
13 };
14 #endif
```

```
/*
 * 헤더파일 중복으로 발생할 수 있는 문제를 막기 위해 #ifndef #endif #define 전처리기 사용
 * Rectangle class에 public type의 멤버함수와 private type의 멤버변수 정의
 * class와 이름을 같게 하여 Rectangle 생성자 선언
*/
```

ii. recta.cpp

```
1  #include <iostream>
2  #include "recta.h"
3  using namespace std;
4
5  Rectangle::Rectangle(int x = 0, int y = 0, int h = 0, int w =
       ↪ 0)
6  : xLow(x), yLow(y), height(h), width(w) {}
7
8  void Rectangle::Print()
9  {
10         cout << " Position: " << xLow << " " << yLow <<
               ↪ "; Height = " << height << " Width = "
               ↪ << width << endl;
11 };
12
13 bool Rectangle::LessThan(Rectangle& s)
14 {
15         if (height * width < s.height * s.width)
16                 return true;
17         else
18                 return false;
19 }
20
21 bool Rectangle::Equal(Rectangle& s)
22 {
23         if (height * width == s.height * s.width)
24                 return true;
25         else
26                 return false;
27 }
28
29 int Rectangle::GetHeight()
30 {
31         return height;
32 }
33
34 int Rectangle::GetWidth()
35 {
36         return width;
37 }
```

```
/*
 * Rectangle class의 생성자에 접근하여 멤버변수를 초기화해주는 멤버이니셜라이저
 * Rectangle class의 instance에 해당되는 멤버변수들의 값을 출력해주는 함수
 * hw1a.cpp 의 main 함수에서 볼 수 있듯, Rectangle class의 instance로서 r과 s가 선언 되어 있으며, r에 대해 s가 가리키는 사각형의 너비 값을 비교하여, s instance에 해당되는 멤버변수간의 연산 값이 r instance의 그것보다 클 경우 참, 반대의 경우 거짓이라는 값을 반환하는 boolean형 함수이며, 비교대상으로 넘어오는 두번째 instance를 s라는 매개변수로 받아오는 것이다. 즉, hw1a.cpp에서 Rectangle class의 instance명으로 정의된 s와는 엄연히 다른 개념이다.
 * r instance 와 s instance 에 해당되는 멤버변수간의 연산 값이 같을 경우 참, 다를 경우 거짓이라는 값을 반환하는 boolean형 함수이며, 비교대상으로 넘어오는 두번째 instance를 s라는 매개변수로 받아오는 것이다. 즉, hw1a.cpp에서 Rectangle class의 instance명으로 정의된 s와는 엄연히 다른 개념이다.
 * 멤버변수 height의 값을 반환하는 int형 함수
 * 멤버변수 width의 값을 반환하는 int형 함수

*/
```

iii. hw1a.cpp

```
1  #include <iostream>
2  #include "recta.h"
3  using namespace std;
4  int main()
5  {
6          Rectangle r(2, 3, 6, 6), s(1, 2, 4, 6);
7
8          cout << "<rectangle r>"; r.Print();
9          cout << "<rectangle s>"; s.Print();
10
11         if(r.LessThan(s))
12                 cout << "s is bigger";
13         else if(r.Equal(s))
14                 cout << "Same Size";
15         else cout << "r is bigger";
16         cout << endl;
17 }
```

```
/*
 * Rectangle class로부터 r과 s instance 및 이에 해당하는 멤버변수 값을 정의
 * r instance 와 s instance 의 멤버변수 값들을 출력
 * recta.cpp 에서 정의된 LessThan() 함수의 반환값을 r instance 에 대하여 s instance 와 비교하여 출력
 * recta.cpp 에서 정의된 Equal() 함수의 반환값을 r instance에 대하여 s instance와 비교하여 출력

*/
```

iv. Results

iv.1 makefile

```
1 hw1a:hw1a.o recta.o
2         g++ -o hw1a hw1a.o recta.o
3 hw1a.o recta.o:recta.h
```

iv.2 compile

```
1 [B711222@localhost hw1d]$ ./hw1a
2 <rectangle r> Position: 2 3; Height = 6 Width = 6
3 <rectangle s> Position: 1 2; Height = 4 Width = 6
4 r is bigger
```

III. HW1B

i. rectb.h

```
1  #ifndef RECTANGLE_H
2  #define RECTANGLE_H
3  using namespace std;
4
5  class Rectangle {
6  public:
7          Rectangle(int, int, int, int);
8          bool operator < (Rectangle&);
9          bool operator == (Rectangle&);
10         int GetHeight();
11         int GetWidth();
12
13 friend ostream& operator << (ostream&, Rectangle&);
14
15 private:
16         int xLow, yLow, height, width;
17 };
18
19 #endif
```

```
/*
 * hw1b 는 hw1a 와 동일한 결과 값을 갖지만, hw1b 에서 relational operator overloading 을 구현했다는 것에서 차이점이 존재한다.
 * ' < ' 연산자와 ' == ' 연산자는 결과 값으로서 boolean 을 가져야 하는데, 실제로 hw1b 에서는, hw1b.cpp 에서의 ' r < s ' 라는 표현을 보면 알 수 있듯이, 피연산자로서 instance r 과 s 를 적어주었으며, 단순히 이러한 표현만으로는 boolean 값을 도출해낼 수 없다.
 * 허나 hw1b 에서의 r < s 의 의미는 instance r 과 s 가 갖는 멤버변수가 표현하는 사각형의 면적끼리의 비교 결과를 뜻한다.
 * 이는 rectb.h 와 rectb.cpp 에서 < 연산자의 overloading 을 가능하게 코드를 설계한 덕분이며, equal operator 또한 위와 동일한 연유로 ' r == s ' 라는 표현만으로 많은 의미를 함축시켜, < operator 의 경우와 같은 맥락으로서 원하는 결과 값을 도출해낼 수 있다.
 * 이처럼 앞 뒤 context 에 걸맞게 기존 operator 에 개발자의 입맛에 맞는 의미를 새로 부여하는 것을 operator overloading (연산자 오버로딩) 이라고 한다.
 * 13번째 line 의 경우, 전역 함수로서 출력 연산자를 overloading 하기 위해 friend 로서 선언이 되었다.
*/
```

ii. rectb.cpp

```
1  #include <iostream>
2  #include "rectb.h"
3  using namespace std;
4
5  Rectangle::Rectangle (int x = 0, int y = 0, int h = 0, int w
       ↪ = 0)
6  : xLow(x), yLow(y), height(h), width(w) {}
7  ostream& operator << (ostream& os, Rectangle& s)
8  {
9          os << " Position: " << s.xLow << " " << s.yLow
               ↪ << "; Height = " << s.height << " Width
               ↪ = " << s.width << endl;
10         return os;
11 }
12
13 bool Rectangle::operator < (Rectangle& s)
14 {
15         if (height * width < s.height * s.width)
16                 return true;
17         else
18                 return false;
19 }
20
21 bool Rectangle::operator == (Rectangle& s)
22 {
23         if (height * width == s.height * s.width)
24                 return true;
25         else
26                 return false;
27
28 }
29
30 int Rectangle::GetHeight()
31 {
32         return height;
33 }
34
35 int Rectangle::GetWidth()
36 {
37         return width;
38 }
```

```
/* hw1a 의 코드와 동일한 부분은 설명을 생략하도록 한다.
 * line 7: 같은 출력 기능을 갖는 연산자라 하더라도 overloading 을 해주어, ' s.xLow ' 등과 같이 객체의 멤버변수를 출력하는 것이 가능해진다.
```

* 이는 매개변수 s로 하여금 instance s 에 대한 멤버변수의 값들을 받아온다.
* line 13: instance r 과 s 의 멤버변수 height 와 width 가 갖고있는 값을 곱하고, 그 결과가 s 의 경우 더 크다면 참, 아니라면 거짓이라는 의미를 operator < 에 부여한다.
* line 21: line 13 과 마찬가지로 height * width 의 결과가 instance r, s 각각 같은 값을 갖는다면 참, 아니라면 거짓이라는 의미를 operator == 에 부여한다.

*/

iii. hw1b.cpp

```
1  #include <iostream>
2  #include "rectb.h"
3  using namespace std;
4
5  int main()
6  {
7          Rectangle r(2, 3, 6, 6), s(1, 2, 4, 6);
8
9          cout << "<rectangle r>" << r
10                 << "<rectangle s>" << s;
11
12         if (r < s)
13                 cout << "s is bigger";
14         else if (r == s)
15                 cout << "Same Size";
16         else cout << "r is bigger";
17         cout << endl;
18 }
```

/*
* line 12: instance s 가 가리키는 사각형의 면적이 r 의 그것보다 크다면 "s is bigger", 같다면 "Same Size", 작다면 "r is bigger" 이라는 문자열을 출력하는 조건문을 작성하였다.

*/

iv. results

iv.1 makefile

```
1  hw1b:hw1b.o rectb.o
2          g++ -o hw1b hw1b.o rectb.o
3  hw1b.o rectb.:rectb.h
```

iv.2 compile

```
1  [B711222@localhost hw1d]$ ./hw1b
2  <rectangle r> Position: 2 3; Height = 6 Width = 6
3  <rectangle s> Position: 1 2; Height = 4 Width = 6
4  r is bigger
```