

Sobrecarga de operadores

Operator Overloading

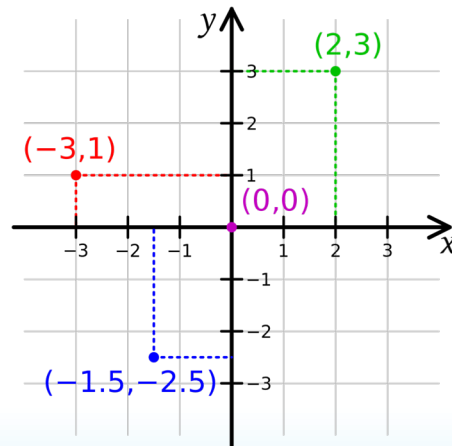
Sobrecarga de Op. – Conceptos Clave

- Es el concepto de tomar operadores tales como + , - , * entre otros y darles un significado para tipos de datos definidos por el usuario
- Ej: se pueden definir clases y que éstas se puedan presentar comportamientos usando operadores aritméticos
- Para clarificar esto se describe el siguiente escenario (a continuación)

Sobrecarga - Escenario

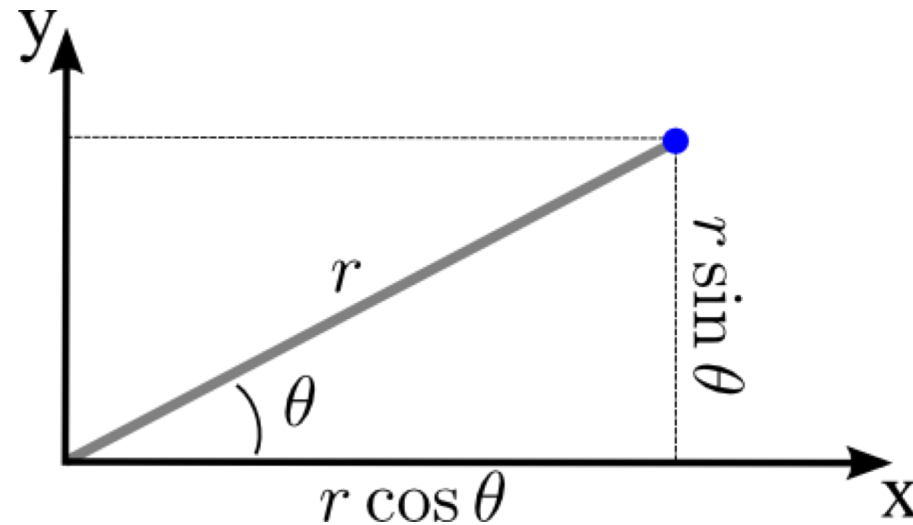
- Asumamos que se tiene dos clases que representan coordenadas polares y rectangulares

Las coordenadas rectangulares manejan puntos en un plano cartesiano en el formato (x, y)



Sobrecarga - Escenario

Las coordenadas polares es como se manejan puntos en un plano cartesiano usando ángulos y una dimensión, ej: (4 , 30°)



Sobrecarga - Escenario

- La suma y resta de coordenadas es mucho más sencillo de hacer en la forma rectangular
- La división y multiplicación es mucho más sencilla de hacer en el formato polar

Sobrecarga

- De no permitirse la sobrecarga de operadores, la suma de dos coordenadas (en formato rectangular) se podría hacer usando una de estas dos opciones

```
Rect r1(1,2);  
Rect r2(3,2);  
  
int x = r1.x + r2.x;  
int y = r1.y + r2.y;  
  
//o  
  
r1.sumar (r2);
```

← Mala práctica de desarrollo de software


← Válido, pero se puede mejorar

Sobrecarga

- Es mucho más claro escribir algo así

```
Rect r1 (1,2);  
Rect r2 (3,2);  
Rect r3;  
  
r3 = r1 + r2
```

Es claro que hay una suma
entre dos tipos de datos
definidos por el usuario



Quieres saber más?

<https://www.programiz.com/cpp-programming/operator-overloading>

Sobrecarga

- Mejor aún, es posible ocultar detalles como: qué tal que se desea sumar una coordenada polar con una rectangular

```
Rect r1(1,2);  
Pol p1(3,45);  
Rect r3;  
  
r3 = r1 + p1;
```

Completamente válido
Y el compilador se encarga
de realizar las instrucciones que
le proporcionemos:

Para este ejemplo p1 será
convertido a rectangular,
se hace la suma y
se regresa el valor esperado

Código de Rect (.h)

```
1  #ifndef RECT_H_INCLUDED
2  #define RECT_H_INCLUDED
3
4
5  class Rect
6  {
7
8  private:
9      int x;
10     int y;
11
12 public:
13
14     Rect();
15     Rect(int, int);
16
17     Rect operator+(const Rect& val)
18     {
19         Rect temp;
20         temp.x = this->x + val.x;
21         temp.y = this->y + val.y;
22         return temp;
23     }
24
25     //no la mejor opcion para imprimir sus valores solamente
26     int getX(); |
27     int getY();
28
29
30 };
31
32
33 #endif // RECT_H_INCLUDED
34
```

Notar que la palabra reservada operator seguida del símbolo para suma indica que se está sobrecargado la suma

Esta sobrecarga del “+” , permite que cuando dos tipos de datos Rect se sumen , su parte x se sumará con la parte x, la parte y con la parte y y se regresará la suma en un tipo de dato Rect

Código de Rect (.cpp)

```
1  #include "Rect.h"
2
3  Rect::Rect ()
4  {
5
6  }
7
8  Rect::Rect (int valX, int valY)
9  {
10     x = valX;
11     y = valY;
12 }
13 int Rect::getX ()
14 {
15     return x;
16 }
17
18 int Rect::getY ()
19 {
20     return y;
21 }
22
```

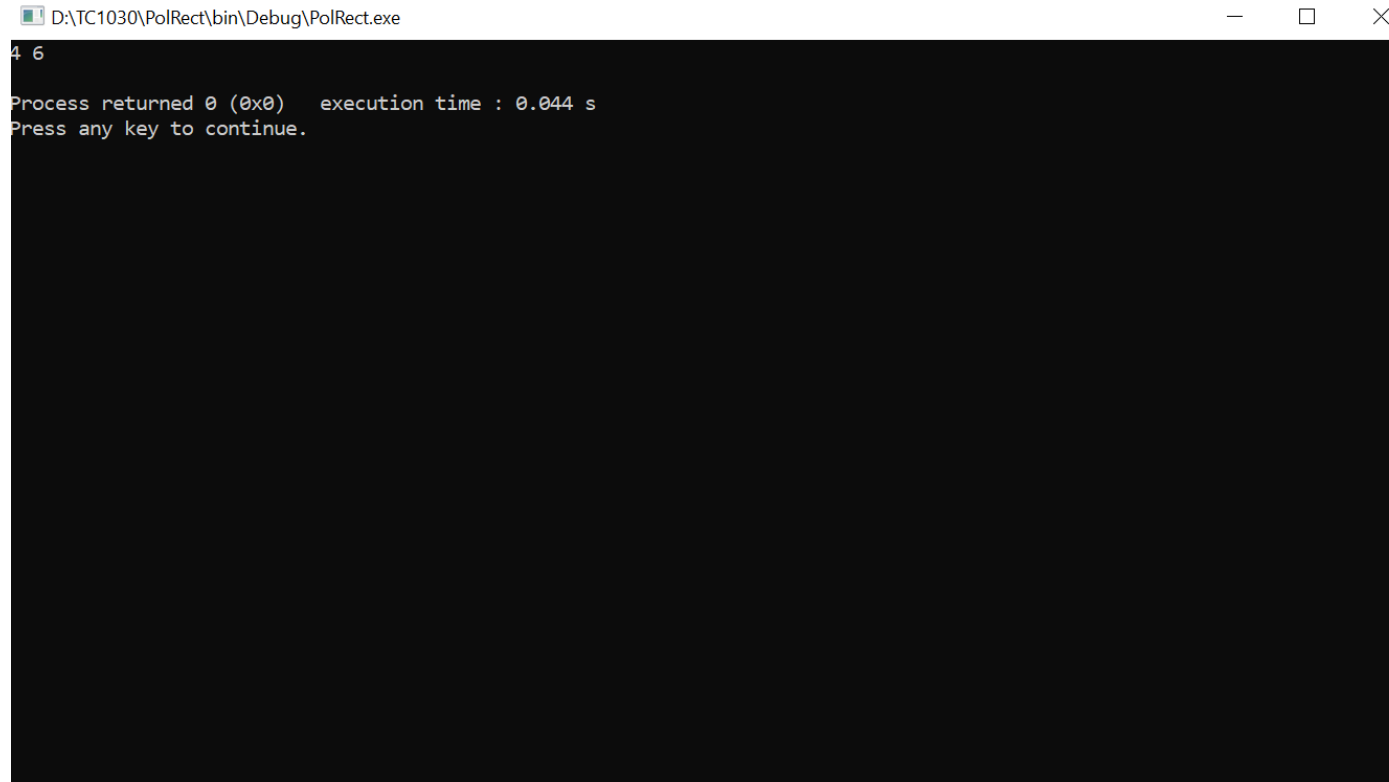
Código de main (.cpp)

```
1  #include <iostream>
2  #include "Pol.h"
3  #include "Rect.h"
4
5  using namespace std;
6
7  int main()
8  {
9
10     Rect r1(1,2);
11     Rect r2(3,4);
12     Rect r3;
13
14     r3 = r1 + r2;
15
16     cout << r3.getX() << " " << r3.getY() << endl;
17
18
19     return 0;
20 }
21
```

La sobrecarga del "+" permite que esto sea posible
De lo contrario, sería un error de compilación
(el compilador no sabría como interpretar
La suma de dos tipos de datos Rect)

Se podría hacer de otra forma, declarando un método
Suma, pero es mucho más claro y fácil de entender

Resultado



```
D:\TC1030\PolRect\bin\Debug\PolRect.exe
4 6
Process returned 0 (0x0) execution time : 0.044 s
Press any key to continue.
```

Como se puede observar, se imprime la suma de rect1 y de rect2

$$1 + 3 = 4$$

$$2 + 4 = 6$$

Práctica

- Crear la sobre carga de “-” para Rect
- Crear la sobre carga de “*” y “/” para Pol

Reto:

- Crear la sobre carga para Rect de + **PERO** recibiendo un Polar, esto es, si llega se diera la suma de un Rectangular con un Polar, es possible
- Hint: la sobrecarga de + para Rect con Pol debe recibir un Polar como parámetro, convertirlo a rectangular, hacer la suma, y regresarlo.