Programación orientada a objetos

Nombre: Oscar Alejandro Penilla Skakievich

Tarea: Investigación 2 _ parcial 3

Fecha: 17/05/22

Grupo:4C1

Investigación sobrecarga de operadores en C++

¿Que es sobrecarga de operadores en C++?
Es redefinir los operadores para que actuen de diferente manera usando los objetos de una clase determinada

Ventajas y desventajas
- Ahorrarse problemas y trabajo para unir clases
- Uso de arreglo dinamico
- Simplifica Codigo
- mas intuitivo
- Muy facil de usar

- Se complican las definiciones de clases
- Además que se puede caer en ambigüedad
- Se puede modificar su definicion pero no su gramatica
- Es necesario que un operando sea un objeto de la clase sobrecargada

Tipos de operadores

Operador	nombre	Operador	nombre
,	Coma	---	Decremento
!	NOT	--=	Asig y resta
!=	Desigual	-->	Selec de miembro
%	Módulo	-->*	Selec de puntero a miembro
%=	Asignacion/ modulo	/	division
&	Direccion de	/=	Asig y div
&&	AND	<	menor que
&	AND bit a bit	<=	Menor igual que
&=	Asignacion AND bit a bit	=	Asignacion
()	Llamada de funcion	==	Igual
()	Conversion	>	Mayor que
*	Multiplicacion	>=	Mayor igual que
*	Desreferencia de puntero	[]	Matriz
*=	Mult. y Asignacion	^	XOR
+	Suma	^=	Asig XOR
+	Unario mas	|	OR bit a bit
++	incremento	|=	Asig OR
+=	Asig y suma	||	OR logico
-	resta	~	Complemento a uno
-	Negacion Unaria		

¿Cuáles operadores no pueden ser sobrecargados?

Operador	nombre
.	Selección de miembro
.*	Selección de puntero a miembro
::	Resolución de ámbito
?:	Condicional
#	Convertir a una cadena
##	Concatenación de preprocesador
Sizeof	tamaño de

Sintaxis y ejemplos completos del operador: +, %, ==

+
```
#include <iostream>
using namespace std;

class Complex{
private:
    int real, imag;
public:
    Complex(int r=0, int i=0){real=r; imag=i;}

    Complex operator + (Complex const &obj){
        Complex res;
        res.real = real + obj.real;
        res.imag = imag + obj.imag;
        return res;
    }
    void print(){cout<< real<< " + i"<<imag<<endl;}
};
int main(){
    Complex c1(10,5), c2(2,4);
    Complex c3 = c1 + c2;
    c3.print();
}
```

Sintaxis

```
tipo operator + (lista de parametros);
tipo operator (operador) (lista de parametros);
```

```
#include <iostream>
using namespace std;

class Distance {
  private
    int feet;
    int inches;
  public:
    Distance() {
      feet = 0;
      inches = 0; }
    Distance(int f, int i) {
      feet = f;
      inches = i; }
    Distance operator - () {
      feet = -feet;
      inches = -inches;
      return Distance(feet, inches); }
    bool operator == (const Distance& d) {
      if (feet == d.feet) {
        return true;
      }
      if (feet == d.feet && inches == d.inches) {
        return true; }
      Return false;
      }
};
int main() {
    Distance D1(11, 10), D2(11, 10);
    if (D1 == D2) {
      cout << "Son iguales\n";
    }else {
      cout << "No son iguales\n"; }

    return 0;
}
```

```
#include <iostream>
using namespace std;

class Parcija {
  public:
    float a, b;
  Parcija operator % (const float a, const float b) {
      a = a;
      b = b;
  };

int main() {
  Parcija A(50, 75);
  Parcija B(150, 175);
  Parcija C = A % B;
  Return 0;
}
```