Sobrecarga de operadores en C++

Escrito por adrianvaca

Domingo, 20 de Marzo de 2011 19:27 -

La sobrecarga de operadores, aunque puede ser una capacidad exótica, la mayoría de personas las usa implícita y regularmente se valen de los operadores sobrecargados.

Por ejemplo, el operador de suma (+) funciona de manera diferente sobre los enteros, puntos flotantes y dobles. No obstante dicho operador funciona muy bien con las variables int, float y double y varios otros tipos integrados han sido sobrecargados por el propio lenguaje C++.

Los operadores se sobrecargan escribiendo una definición de función (con su encabezado y cuerpo) de manera habitual, excepto que el nombre de la función ahora se vuelve la palabra clave operator, seguida por el símbolo del operador que se sobrecarga. Por ejemplo el nombre de la función operator+ sirve para sobrecargar el operador de suma (+).

Para utilizar un operador sobre objetos de una clase, dicho operador debe ser sobrecargado, con dos excepciones: el operador de asignación (=) puede utilizarse con cualquier clase, sin sobrecarga explícita.

El comportamiento predeterminado del operador (=) es una asignación a nivel de miembros de los datos miembro de la clase. El operador de dirección (&) también puede utilizarse sin sobrecarga con objetos de cualquier clase, simplemente devuelve la dirección de memoria del objeto.

La sobrecarga de operadores no es automática; el programador debe escribir funciones de sobrecarga de operadores que realicen las operaciones deseadas. A veces conviene que estas funciones se hagan funciones miembro, en otras ocasiones conviene que sean funciones friend, ocasionalmente puede hacerse funciones no miembro, no friend

Es posible llegar a los extremos de la sobrecarga, como sobrecarga, como sobrecargar el operador + para que realice operaciones tipo resta. Tales empleos de la sobrecarga hace que sea muy difícil entender el programa.

Una lista de operadores que pueden o no sobrecargarse es la siguiente:

Operadores que pueden sobrecargarse

+				-	
-	!	=	>		+=
/=	%=	A=	&=		!=
[]	()	new	delete		
new []	delete []				

Operadores que NO pueden sobrecargarse

Sobrecarga de operadores en C++

Escrito por adrianvaca

Domingo, 20 de Marzo de 2011 19:27 -

.

Los operadores `&`, `*`, `+` y `-` tiene versiones unarias y binarias, estas versiones unarias y binarias se pueden sobrecargar por separado.

No es posible crear nuevos operadores; sólo se pueden sobrecargar los operadores existentes, esto desgraciadamente, evita que el programador use notaciones como `**` como en BASIC para la exponenciación.

La sobrecarga de un operador de asignación y de uno de suma para permitir instrucciones como:

```
obj eto2          =          obj eto2          +
```

no implica que el operador `+=` también este sobrecargado para permitir instrucciones como:

```
obj eto2          +=          obj eto1          ;
```

tal comportamiento puede lograrse explícitamente sobrecargando el operador `+=` de dicha clase.

Las funciones de operador pueden ser funciones miembro o funciones no miembro, estas últimas con frecuencia se hacen friend por razones de desempeño. Las funciones miembro utilizan implícitamente el operador `this` para obtener uno de los argumentos de su objeto de clase.

Tal argumento de función puede debe listarse explícitamente en una llamada de función no miembro.

Cuando una función de operador se implemente como función miembro, el operador de la izquierda (o el único) debe ser un objeto de clase (o una referencia a un objeto de clase) de la clase del operador. Si el operador de la izquierda debe ser un objeto de una clase diferente o un tipo integrado, esta función operador debe implementarse como función no miembro.

Una función de operador no miembro debe ser friend si necesita acceder directamente a miembros `private` o `protected` la clase.

Las funciones miembro de los operadores de una clase específica se llaman sólo cuando el operando de la izquierda de un operador binario específicamente es un objeto de esa clase, o

Sobrecarga de operadores en C++

Escrito por adrianvaca

Domingo, 20 de Marzo de 2011 19:27 -

cuando el operando de un operador unario es un objeto de esa clase.

Ejemplo

Creación de una clase string y sobrecarga de la mayoría de sus operadores.

```
#include<iostream.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>
#include<assert.h>

enum bool { false, true }

class string {
    int size;
    char * ptr;
public:
    string (char []= &);
    string (string &); //destru
    ~string () { operator = (
    string &);
    bool operator == (string &);
    bool operator != (string &);
    bool operator ! (); //prueba si string esta vacia
    bool operator < (string &);
    bool operator > (string &);
    bool operator <= (string &);
    bool operator >= (string &);
    string & operator += (
    char operator [] (int); //operador de subíndice
    string operator () (int,
    int longitud (void);
    friend ostream & operator << (
    friend istream & operator >> (

};

string :: string (char
size = strlen (
ptr =new char [
if (ptr == NULL) {
strcpy (ptr,
}
```

Sobrecarga de operadores en C++

Escrito por adrianvaca

Domingo, 20 de Marzo de 2011 19:27 -

```
string      ::      string      (      string
    size    =      strlen     (
    ptr     =new   char        [
    ptr     ==     NULL       ){
    strcpy  (      ptr        ,
}

string      :: ~
    delete

string      string      (){
    []      ptr        ;
}

string      &          string      ::      operato
    if(&    s          !=       this      ){
    delete  []         ptr        ;
    size    =          s        .
    ptr     =new   char        [
    strcpy  (      ptr        ,
    }
    else    cout      "Intento de asignar un objeto a
    return (*this) ; //habilita asignaciones en casca
}

bool string  ::
    if(!    strcmp
    return (false
}

bool string  ::
    if(!(*  this
    return( ==
    false   );
}

bool string  ::
    if (    size
    return  false
}

bool string  ::
    if (    strcmp
    return  false
}

bool string  ::
    if (    s
    return  false
}

string      operator  == (      string
    ptr     ptr        ,
)

string      operator  != (      string
    s       s          )) return
)

string      operator  ! (){
    0       0          ) return
}

string      operator  < (      string
    ptr     ptr        ,
)

string      operator  > (      string
    this    this       ) return
```

Sobrecarga de operadores en C++

Escrito por adrianvaca

Domingo, 20 de Marzo de 2011 19:27 -

```
}

bool string      :: operator <= ( string
  if( !(         s          < *   this      ) ) retur
  return        false      ;
}

bool string      :: operator >= ( string
  if( !(         s          > *   this      ) ) retur
  return        false      ;
}

string           & string      :: operator
  char           *             =
  size           +=            .
  ptr            =new          [
  if(            ==            ) {
  strcpy         (             ptr
  strcat         (             ptr
  delete         []            ;
  return (*      this          //habilita llamadas en cascada
}

char string      :: operator [] ( int num
  assert         (             >=
  return (       ptr           num
}

//Devuelve subcadena que comienza en: inicio y de longitud: subszie
string string    :: operator () ( int inicio
  //asegura que inicio este en el rango y que subszie sea >=0
  assert         (             >=

  string        *             = new
  if(           subptr        == 0

  //determina la longitud de la subcadena
  if((          subszie       == 0
  subptr       ->            =

  else

  subptr       ->            =

  //asigna memoria para la subcadena
  delete subptr ->           ptr ;
  subptr       ->           ptr = new
  if(          subptr        -> ptr ==
```

Sobrecarga de operadores en C++

Escrito por adrianvaca

Domingo, 20 de Marzo de 2011 19:27 -

```

        //copia la subcadena a la nueva string
        strncpy ( subptr -> ptr [
        return (* subptr ); //devue
    }

int string :: longitud ( void
    return ( size );

ostream & salida operator << ( ostream
    return ( salida ); //habilita el proceso en cascada

istream & entrada operator >> ( istream
    return ( entrada ); //habilita proceso en cascada

void main ( void ){
    textcolor ( BLACK );
    textbackground ( WHITE );
    clrscr ();
    string s1 ( "hola" ),

        //probando operadores de igualdad y relacionales
        cout << "s1: " <<
        cout << endl <<
        << endl << "Resultado de s1!=s2: " << (
        << endl << "Resultado de s1> s2: " << (
        << endl << "Resultado de s1< s2: " << (
        << endl << "Resultado de s1>=s2: " << (
        << endl << "Resultado de s1<=s2: " << (

        //prueba operador sobrecargado (!)
        cout << endl <<
        if(! s3 ){
            cout << "s3 esta vacio, asignando s1 a
            s3 = s1 ;
            cout << "ns3: " <<
        }
    }

```

Sobrecarga de operadores en C++

Escrito por adrianvaca

Domingo, 20 de Marzo de 2011 19:27 -

```
        //probando operador sobrecargado de concatenacion
        cout          <<          endl          <<
        s1            +=          s2            ;
        cout          <<          "s1: "       <<
        //probando operador sobrecargado []
        cout          <<          ", s1[8]= "   <<

        //prueba del operador sobrecargado ()
        cout          <<          endl          <<

        //prueba el constructor de copiado
        string        *          s4            = new
        cout          <<          endl          <<

        //prueba del operador de asignacion =
        cout          <<          endl          <<
        *             =          s3            ;
        cout          << *       s4            "Longitu
        <<          endl          <<

        getch        ();

}
```