**Sobrecarga**

En C++ es posible tener varios métodos dentro de la misma clase con el mismo nombre, a esto se le conoce como sobrecarga.

Es posible sobrecargar un método declarando otro con el mismo nombre pero diferente número o tipo de parámetros. C++ sabe

cual método debe ejecutarse utilizando la cantidad y tipo de parámetros que se colocan al mandar llamar a dicho método.

La sobrecarga no es exclusiva de funciones miembro de una clase, también es posible sobrecargar funciones libres; es decir,

funciones que no pertenecen a ninguna clase.

**Constructores**

Es muy común que algunas partes del objeto requieran ser inicializadas antes de poder ser utilizadas. C++ permite que los objetos

sean inicializados cuando se crean, a través de un constructor. Un constructor es una función especial que es miembro de la

clase, que tiene el mismo nombre de la clase y que se manda llamar automáticamente cuando se crea una instancia u objeto de

una clase.

En C++ un constructor tiene las siguientes características:

- Su nombre es el mismo que el de la clase
- No tiene definido un valor de retorno

Un constructor puede o no tener parámetros, al constructor sin parámetros se le conoce como el constructor Default.

Cuando se declara una instancia de una clase que tiene un constructor con parámetros, para que se llame a dicho constructor se

requiere el siguiente **formato:**

NombreClase  NombreVariable(listaParametros);

donde listaParametros son los parámetros actuales que se enviarán a los parámetros formales del constructor que se quiere llamar.

Por ejemplo, supongamos que vamos a declarar dos constructores para la clase Circulo como se muestra a

continuación:

```
Circulo::Circulo()
{
    radio = 0;
}

Circulo::Circulo(int ra)
{
     radio = ra;
}
```

podemos declarar instancias de la clase Círculo de la siguiente forma:

```
Circulo a;          // se llama al constructor default, porque no se especifícan
parámetros.

Circulo b(10);    // se llama al constructor que recibe como parámetro el
radio, porque es el que   coincide con los parámetros que se colocan al crear
el objeto.
```

Nota que si se va a "llamar" el constructor default, al declarar la variable no se colocan los paréntesis.

Si el programador no crea un constructor explicitamente, C++ crea uno "vacío" que se encarga de realizar algunas acciones

necesarias al crear un objeto.

**Destructores**

Un destructor es una función que se ejecutada automáticamente cuando se destruye un objeto, por ejemplo cuando un objeto es

local a un método y se termina la ejecución de dicho método. El destructor por si mismo no destruye el objeto, pero es útil para

liberar memoria dinámica que se había reservado al crear el objeto, o para cerrar un archivo que se abrió al crear el objeto, etc.

Un destructor en C++ tiene las siguientes características:

- Tiene el mismo nombre de la clase, pero precedido por el caracter ~.
- No tiene definido un valor de retorno

Una clase puede tener solamente un destructor, el cual no debe recibir parámetros ni tener valor de retorno. Si el programador no

crea un destructor C++ crea uno "vacío" que se encarga de realizar algunas acciones necesarias al destruir un objeto.

**Ejemplo completo**

El siguiente ejemplo muestra la declaración de una clase (la clase Pila) y una aplicación que hace uso de ella.

```
#include <iostream.h>

const int SIZE = 100;

//Definición de la clase Pila
class Pila {
          int pila[SIZE];
          int tope;
    public:
          Pila(); // constructor
          ~Pila(); // destructor
          void push(int i);
          int pop();
        };

// constructor de la clase pila
Pila::Pila()
{
    tope = 0;
    cout<<"Se inicializa la pila"<<endl;   // este letrro se utiliza solo para
entender el ejemplo
}


// destructor de la clase pila
Pila::~Pila()
{
    cout<<"se destruye la pila"<<endl;   // este letrero se utiliza solo para
hacer más claro el ejemplo
}


void Pila::push(int valor)
{
    if (tope ==SIZE)
    {
        cout<<"La pila esta llena"<<endl;
        return;
    }
    pila[tope] = valor;
    tope++;
}

int Pila::pop()
{
    if (tope == 0)
    {
        cout<<"No hay nada en la pila"<<endl;
        return -1;
    }
    tope--;
    return pila[tope];
}


int main()
{
    Pila a, b;

    a.push(1);
```

```
    b.push(2);
    a.push(3);
    b.push(4);

    cout<<a.pop()<<endl;
    cout<<a.pop()<<endl;
    cout<<b.pop()<<endl;
    cout<<b.pop()<<endl;

    return 0;
}
```

## Ejercicio

Copia los ejemplos que se incluyen en el material y pruébalos.

Define la clase Fecha, de acuerdo con las siguientes especificaciones:

Atributos:

- dia
- mes
- año

Constructores:

- que inicialice con una fecha fija que tu definas
- que reciba como parámetro los valores para inicializar la fecha

Métodos públicos:

- que permita modificar el valor de la fecha
- que muestre en la pantalla la fecha usando el formato dia / mes / año
- que muestre en la pantalla la fecha poniendo el mes con palabras
- que permita verificar si una fecha es válida; este método debe ser utilizado por el constructor y el método que modifica el

  valor de la fecha, si el usuario trata de inicializar con una fecha inválida se inicializará con el valor fijo que tu hayas definido.

Realiza después una aplicación para probar tu clase, debe al menos crear 2 objetos de tipo Fecha, utilizando cada uno de los

constructores y después mostrar las fechas correspondientes en la pantalla. ver solución

| Ligas sugeridas |
| --- |
| http://www.cplusplus.com/doc/tutorial/<br><br>http://www.cs.wustl.edu/~schmidt/C++/ |

***Regresar***

***Siguiente módulo***