Constructores

Constructores

En C++ una forma de asegurar que los objetos siempre con tengan valores válidos es escribir un constructor.

Un constructor es una función miembro especial de una clase que es llamada automáticamente siempre que se declara un objeto de esa clase. Su función es crear e inicializar un objeto de su clase

Constructores

En C++ la inicialización de los datos miembro del objeto no se puede realizar en el momento en que son declarados; sin embargo, existe una función miembro especial llamada constructor que permite inicializar objetos en el momento en que se crean. Un constructor es una función miembro especial que se llama exactamente igual que la clase y sirve para construir un nuevo objeto y asignar valores válidos a sus datos miembro.

Constructores

Tiene el mismo nombre que la clase a la que pertenece.

- No tiene tipo de retorno (incluyendo void), por lo tanto no retorna ningún valor.
- Puede admitir parámetros como cualquier otra función.
- No puede ser heredado.
- No puede ser declarado const ni static.
- Debe ser público.

Constructores

Cuando una clase tiene un constructor, éste será invocado automáticamente siempre que se cree un nuevo objeto de esa clase .

Se puede crear un objeto de cualquiera de las siguientes:

- Declarando un objeto global
- Declarando un objeto local u objeto temporal
- Invocando al operador new
- Llamando explícitamente a un constructor

Tipos de constructores

Tipos de constructores:

1) Constructor por omisión (implícito)

2) Constructor explícito con argumentos

3) Constructor explícito con argumentos por omisión

4) Constructor copia

Constructor por omisión (implícito)

Dado que los constructores son funciones miembro, admiten argumentos igual que éstas. Cuando en una clase no especificamos ningún constructor el compilador crea uno por omisión sin argumentos .

Un constructor por omisión de una clase, es un constructor que se puede invocar sin argumentos.

Constructor por omisión (implícito)

Si el objeto creado con el constructor por omisión es global, el constructor inicializa a ceros los datos miembro numéricos y a nulos los datos miembro alfanuméricos. Si el objeto creado es local, lo inicializa con valores indeterminados.

https://replit.com/join/wvveygwfzq-maximo-arielari

Constructor explícito con argumentos

Un constructor explícito con argumentos es aquel que define una lista de argumentos que le ayudarán a inicializar el objeto con valores válidos, en dicha lista de argumentos recibe los valores con los cuales inicializará el objeto que se va a construir.

Ejemplo

```
class CFecha {
        private: int dia, mes, anio;
        public: CFecha(int, int, int); //constructor explícito con argumentos
void asignarFecha( );
 void obtenerFecha( );
int fechaCorrecta( );
 };
```

Ejemplo

```
CFecha::CFecha(int dd, int mm, int aa)
 {
dia = dd;
 mes = mm;
 anio = aa; }
```

Importante

Una vez que se ha declarado un constructor, como en el caso del constructor explícito con argumentos, éste substituye al constructor por omisión que genera el compilador y ya no podemos declarar objetos sin enviar argumentos.

Constructor copia

Otra forma de inicializar un objeto es asignándole otro objeto de la misma clase en el momento de su declaración, esto es, cuando se crea .

Por ejemplo:

CFecha hoy(18,2,2014);

CFecha otroDia(hoy);

CFecha unDia=hoy;

Constructor copia

Un constructor que crea un nuevo objeto a partir de otro existente es denominado constructor copia. Un constructor de éstas características tiene un solo argumento, el cual es una referencia a un objeto constante de la misma clase.

Clase(const Clase &obj);

Ejemplo

Persona(const Persona &obPerson);

Donde:

- Persona es el nombre de la clase.
- const es palabra reservada que se emplea para designar constantes.
- & es el operador para hacer la referencia.
- obPerson es el nombre que le queremos dar al objeto.

Constructor copia

Si no especificamos un constructor copia, el compilador proporciona un constructor copia por omisión público, para cada clase definida. También cuando sea necesario podemos codificar nuestra propia versión del constructor copia.

NOTA: Es importante señalar que dentro de la clase, el constructor copia debe ir acompañado siempre por algún otro constructor, debido a que es necesaria la referencia a un objeto ya construido para poder realizar la copia.

Constructor Copia Ejemplo

```
Persona::Persona(const Persona &objPersona){
        strcpy(nombre,objPersona.nombre);
        for(int u=0;u<12;u++){
            ventas[u]=objPersona.ventas[u]; }
}
```

Destructor

Destructor

El destructor es un método de la clase que se usa para destruir objetos del tipo de la clase, no tiene parámetros de entrada ni valor de retorno. Para declarar un destructor se utiliza el caracter virgulilla (~) seguido del nombre la clase, es decir, el destructor también se denomina del mismo modo que la clase. Los destructores se ejecutan automáticamente justo cuando un objeto alcanza el límite de su tiempo de vida. Ese tiempo de vida está definido por el contexto (scope) donde se ha arrojado el objeto. Recuerde que un contexto (scope) está delimitado en C++ por las llaves {y}. El tiempo de vida para un objeto de almacenamiento estático es de toda la ejecución del programa. Mientras que el tiempo de vida de un objeto de almacenamiento automático depende enteramente de su contexto, si es global o local.

Destructor

```
class NewClass
{
    public:
    ~NewClass(); //Declaración del destructor de la clase
};
NewClass::~NewClass(){} //Definición del destructor de la clase
```

Bibliografía

https://www.codingame.com/playgrounds/50577/miembros-especiales-de-la-clase-en-c-practica-2/constructores

https://www.uaeh.edu.mx/docencia/P_Presentaciones/icbi/asignatura/constructores.pdf