Puntero this

Al iniciar el apunte de Introducción llamábamos la atención acerca del código siguiente:

```
void Articulo::Cargar(){
    cout<<"INGRESE EL CODIGO: ";
    cin>>codArt;
    cout<<"INGRESE LA DESCRIPCION: ";
    cin>>descripcion;
    cout<<"INGRESE EL PRECIO UNITARIO: ";
    cin>>pu;
    cout<<"INGRESE EL STOCK: ";
    cin>>stock;
}
```

Decíamos que era extraño que se pudieran usar variables sin que hayan sido declaradas. Y aclarábamos que era posible debido a que las funciones eran de las clases, y por lo tanto conocían a las variables de esa clase. Ahora bien, si hay varios objetos Articulo declarados, y todos llaman al método Cargar() ¿cómo hace el método Cargar() para no confundirse y escribir en las propiedades de cada uno de los objetos?.

Veamos esto con un poco más de detalle. Sabemos que para cada uno de los objetos de una clase se guarda una copia de los datos; también sabemos que sólo se almacena una copia de las funciones de la clase, por lo que cada objeto usará las funciones como si fueran propias, aunque en realidad las comparte con todos los objetos de su clase. Volviendo ahora a la pregunta del párrafo anterior, la respuesta es que Cargar() conoce la dirección de cada uno de los objetos que lo llama; como cualquier función sólo puede modificar el valor de sus variables locales, o de aquellas variables a las que accede por medio de la dirección.

Dentro de las funciones definidas en las clases entonces hay un mecanismo que permite diferenciar un objeto de otro. Analicemos el siguiente código:

```
int main(){
    Articulo obj;
    cout<<""DIRECCION DE obj: "<<&obj<<endl;

    return 0;
}
```

Al ejecutarlo podríamos obtener el siguiente resultado:

```
DIRECCION DE obj: 0x62fdf0
```

Agreguemos ahora a la clase el siguiente método:

```
void Articulo::MostrarDireccion(){
    cout<<this;
}
```

Y modifiquemos el programa anterior

```
int main(){
    Articulo obj;
    cout<<"DIRECCION DE obj: "<<&obj<<endl;
    obj.MostrarDireccion();

    return 0;
}
```

por pantalla veremos al ejecutar el programa

```
DIRECCION DE obj: 0x62fdf0
0x62fdf0
```

En este caso lo que se muestra son direcciones de memoria (por supuesto que éstas pueden cambiar en cada ejecución) coincidentes, por lo que podemos afirmar que

- **this** es algo que existe dentro de la clase (no dio error al utilizarlo)
- Al mostrarlo por pantalla se imprimió una dirección, por lo tanto es un puntero.
- La dirección que se imprimió en la pantalla es la misma que la del objeto que llamó al método.
- Entonces podemos concluir que **this es un puntero que contiene la dirección del objeto que llama a un método**.

Cada vez que un objeto llama a un método éste recibe la dirección del objeto que lo está llamando, y por lo tanto las acciones que se ejecuten se harán sobre ese objeto en particular.

Se dice que el puntero this es oculto, ya que no es necesario hacer referencia a él. Pero se lo puede utilizar de manera explícita en caso que sea necesario o útil. Podríamos entonces reescribir el método Cargar() de la siguiente manera:

```
void Articulo::Cargar(){
    cout<<"INGRESE EL CODIGO: ";
    cin>>this->codArt;
    cout<<"INGRESE LA DESCRIPCION: ";
    cin>>this->descripcion;
    cout<<"INGRESE EL PRECIO UNITARIO: ";
    cin>>this->pu;
    cout<<"INGRESE EL STOCK: ";
    cin>>this->stock;
}
```

En este caso el método funcionará de la misma manera que antes. El ejemplo es sólo para visualizar el puntero y probar su existencia.

Más adelante en el curso veremos otros ejemplos de aplicación del puntero this.

Nota: el operador -> permite a un puntero acceder a las propiedades y métodos definidos en la clase a la que pertenece. Reemplaza al operador de indirección *