UNIDAD 2

Introducción al Lenguaje "C++"

1.- **La programación Orientada a Objetos.**

La Programación Orientada a Objetos no es un concepto nuevo, data de hace unas dos decadas. El origen de la Programación Orientada a Objetos viene del Lenguaje de Programación Simula 67 y del Smalltalk desarrollado por Xerox, Palo Alto, en el Research Center en los inicios de 1970. Entonces. ¿Porqué el interés actual por esta metodología?. El interés creció en parte porque se cuenta con un nuevo lenguaje "C++", que de entrada, le dá al programador recursos para escribir Programas eficientes Orientados a Objetos de propósito general. Vemos que la Programación Orientada a Objetos difiere en mucho de nuestro estilo familiar de programación, por lo que se requiere un nuevo enfoque hacia el problema a resolver.

Cuando se desarrolla Programación Orientada a Objetos, el programadpor se pregunta **Qué** hacer con un Objeto, en vez de enfocarse a los aspectos procedurales convencionales de **Cómo** obtener algo. La programación Orientada a Objetos, simplemente se enfoca a la manipulación de Objetos. con este concepto se puede representar casi cualquier cosa; un número, una cadena, una estructura de pacientes en un hospital, o la construcción gráfica de un rectángulo, o cualquier otra figura geométrica. En esencia, un objeto contiene los datos o estructuras de datos necesarias para describir un objeto junto con un conjunto de operaciones a realizarse sobre los datos. Vamos a ver que un objeto no es más que una instancia particular de un tipo de dato abstracto que diseñamos acorde a un conjunto de reglas particulares.

Como ya se ha comentado, gran parte del valor de la Programación Orientada a Objetos es consecuencia del concepto de **Herencia**, es por ello que un programador puede iniciar con la construcción de una liberría de algunos tipos de Objetos, o clases, ampliandola para una nueva aplicación, agregando tipos de datos y operaciones para crear nuevas clases. En otras palabras, en vez de escribir una nueva aplicación desarrollando completamente el código, un **cliente** hereda datos y operaciones de alguna clase base, agregando nueva funcionalidad, cuando agrega nuevos datos o funciones, el programador no necesita modificar la clase base, consiguiendo con ello **código reusable**.

Otra característica muy importante en la Programación Orientada a Objetos es el concepto de **Enlace Dinámico** o **Enlace Tardio** (late binding), él que nos ayuda a crear programas más generales permitiendo que cada clase de un determinado grupo de clases relacionadas tenga una implementación diferente en una función en particular, En un programa el **cliente** puede aplicar la función a un abjeto sin necesidad de conocer la clase especifica del objeto. En tiempo de ejecución, el sistema determina la clase especifica del objeto e invoca la implementación particular de la función.

2.- El Lenguaje "C++".

El Lenguaje de programación C++, fué diseñado e implementado por Bjarne Stroustrup de la AT&T Bell Laboratories como el sucesor de "C". Mientras toma algunas ideas de Simula 67 y Algol 68, el lenguje de programación C++ mantiene la compatibilidad con programas en "C", así como su eficiencia. En la Figura # 1 se muestra su árbol genealógico.

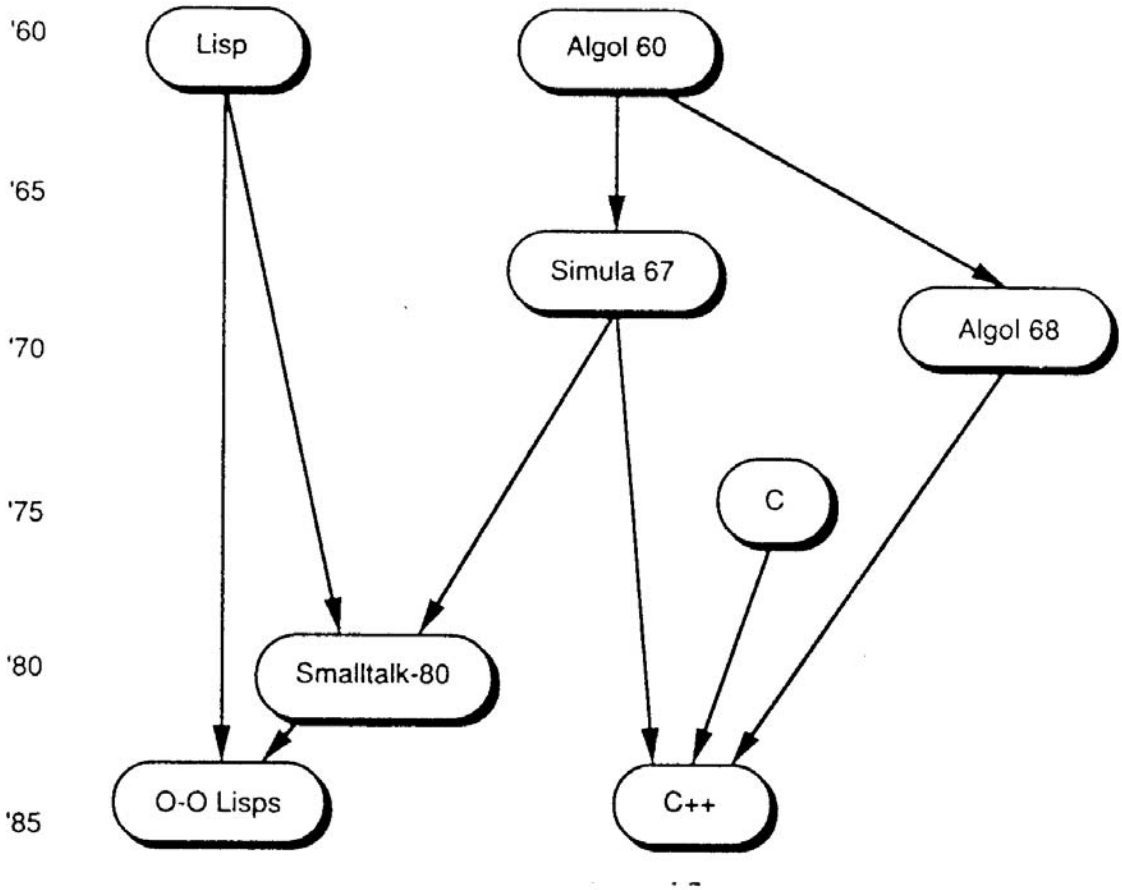

Figura # 1, La Herencia del C++

Al C++ se le agregan nuevas y potentes características, haciendolo útil para un gran número de aplicacionnes, desde manejo de dispositivos hasta inteligencia artificial, C++ es un lenguaje enfocado al desarrollo en serio de software; por su intima relación con "C", su potencial para el desarrollo de interfaces gráficas de usuario, para programación de sistemas, y por soportar el desarrollo de software a gran escala.

Los libros donde se encuentra la definición del C++, son:

El Lenguaje de Programación C++	The Annotated C++
Bjarne Stroustrup	Reference Manual
Addison Wesley	Ellis & Bjarne Stroustrup
Edición Especial, 2000	Addison Wesley 1990

Los cuales nos brinda una detallada explicación del Lenguaje, contienen muchos ejemplos y ejercicios, incluye un manual de referencia. y una definición más formal del Lenguaje.

3.- Diferencias de C++ con respecto a "C".

El Lenguaje "C" es de código compacto y eficiente, sin embargo su verificación de errores en tiempo de compilación es muy pobre, tanto para tipos como para argumentos de funciones, C++ solventa este problema y ofrece nuevas carcterísticas.

C++ cuenta con un mecanismo estricto de verificación de tipos, usa la declaración de prototipo de función para verificar los argumentos de las funciones, adicional a ésto nos permite declarar variables por referencia, entonces, cuando una función tiene una **referencia** como un argumento formal el compilador de C++ sabe que deberá pasar la dirección del argumento actual cuando la función es llamada.

4.- El C++ y la programación orientada a objetos.

Al hablar de C++, pueden surgir algunas preguntas;

* ¿Por qué C++ es importante como tema de discución al hablar de la Programación Orientada a Objetos?.
* ¿Por qué no planear el desarrollo de la Programación Orientada a Objetos en Lenguajes como "C", Pascal, u otro lenguaje familiar?.

Stroustrup en su artículo "What is Objet Oriented Programming", dice: "Un Lenguaje soporta un estilo de programación, si nos brinda facilidades que hacen su utilización conveniente (facil, segura y eficiente) para usarlo en dicho estilo.

Por otro lado, un lenguaje requiere que el programador cuente con cierta habilidad para escribir programas en un dado estilo, sin mucho esfuerzo, el lenguaje solo soporta el estilo. Por ejemplo es facil y eficiente escribir programas estructurados en Pascal, se puede también escribir programas estructurados en BASIC, pero es dificil.

C++ no solamente corrige la mayoria de las deficiencias de "C", también introduce nuevas cualidades, desarrolladas para un soporte adecuado de abstracción de datos y Programación Orientada a Objetos. Mostramos en seguida algunas:

clases Es el constructor básico del Lenguaje, el cual nos permite crear tipos de datos, llamados **tipos de datos abstractos**.

variable miembro Nos describen los datos como tipos abstractos.

funciones miembro Nos definen las operaciones permitidas sobre los datos miembro.

sobrecarga de operadores Nos permite dar un significado adicional a la mayoria de los operadores, de tal manera de poderlos usar con nuestros propios tipos de datos, de tal forma que sean faciles de manejar.

sobrecarga de funciones Similar a la sobrecarga de operadores, nos evitan excesivos nombres de funciones, haciendo el código facil de leer.

Control automático de conviersión de tipos Nos permite mezclar nuestros propios tipos con otros y con los proporcionados por el lenguaje C++.

clases dirivadas Heredan **variables miembro** y **funciones miembro** de su clase base, se diferencian de su clase base porque contienen otras variables miembro y otras funciones mienbro.

funciones virtuales Le permiten a una clase derivada redefinir funciones miembro heredadas de una clase base. Entonces podemos escribir programas muy generales, sin mencionar una clase especifica del objeto manipulado, usando el enlazado dinámico, el sistema en tiempo de ejecución (run time) va a escoger la función apropiada de una clase particular.

5.- ¿Por qué usar C++.

Algunas buenas razones son las siguientes:

1) C++ es compatible con "C".

2) Usa los tipos de datos fundamentales; int, float, etc. de la forma convencional.

3) Maneja el paradigma Orientado a Objetos de manera natural, el programador puede continuar escribiendo programas estructurados en "C" y aprovechar los beneficios de C++.

4) Los programas escritos en C++ usando el paradigma de programación orientada a objetos, resultan faciles de escribir, rastrear y una vez implementados, faciles de leer y mantener.