Visual Basic.Net to C++ Code Translator

Traductor de Visual Basic.NET a Código C++

Marcos Meriño, Mauricio Pacheco y Ana Serrano
{Cormamer, elmao4, ana_luisa170}@hotmail.com
Universidad Simón Bolívar
Barranquilla - Atlántico

Palabras clave:
Análisis Léxico, Análisis Sintáctico, Compilador, Intérprete, Traductor.

Resumen
En la realización de este programa traductor de lenguaje de visual Basic.Net a C++, se aplicaron los conceptos fundamentales de la teoría de compiladores, como son el análisis léxico y sintáctico, en estas fases es donde se realizará el inicio del programa traductor que se construyó. En el análisis léxico se verifica la correcta escritura de las variables y constantes, mientras que en la parte de sintáctica se evalúa que los ciclos y el condicional esté escrito correctamente
Después de la aplicación de cada una de las fases antes mencionadas se tendrá como resultado un software capaz de traducir código del lenguaje visual Basic. Net a código en el lenguaje C++. Cabe resaltar el beneficio que presentará esta utilidad a los estudiantes que conozcan poco de C++, pero manejen bien Visua.Net. Así como a cualquier otra persona que no maneje C. Dicho beneficio podría darse codificar un programa en C++ empleando menos tiempo dado el hecho que no lo maneja muy bien como se dijo anteriormente, o si desconoce su forma de codificar como lo son las palabras reservada o la forma de escritura de las variables

I. INTRODUCCIÓN

Este documento fue realizado con el fin de dar a conocer la creación de un software que consiste en un traductor de lenguajes de programación en esta oportunidad se trabajará con los lenguajes Visual Basic.Net y C++.

Como es de saber con el paso del tiempo los lenguajes de programación han ido evolucionando por así decirlo, son más fáciles de entender y además proporcionan una interfaz grafica ideal para el usuario, pero a su vez encontramos lenguajes que son totalmente opuestos a los modernos en estos aspectos. Es el caso de la universidad Simón bolívar donde el paso de un lenguaje a otro trae consecuencias negativas para algunos estudiantes y sobre todo en el momento de codificar en C++ ya que por lo general se encontraban trabajando en VB que se presume es mucho más fácil de aprender y en el cual se optimiza mucho el tiempo a la hora de la escritura del código, para encontrarse con un lenguaje como lo es C++ donde la escritura del código se hace mucho mas extensa. A esto se le agrega el poco tiempo que se tiene para el aprendizaje de un nuevo lenguaje.

Esta situación conlleva al inconformismo por parte de algunos estudiantes que se ven en la obligación de aprender un nuevo lenguaje ya que no existe una herramienta alterna que los ayude en la codificación y a hacer su tiempo mas optimo, por tal motivo se hace necesario la utilización de una herramienta que ayude a las escritura del código en C++. Es por esto, que se construyo el software, que ayudara a los estudiantes en su proceso de adaptación al nuevo lenguaje, dicho programa presenta la opción de escribir código en visual Basic.net y después de un proceso de compilación permitir traducirlo a C++.

Para poder realizar este software se aplicarán los conceptos básicos de la teoría de compiladores el cual dictamina la utilización de un análisis léxico y un análisis sintáctico.

En el primer análisis que es el léxico se trabajó en la estructura de las variables y constantes en cada uno de los dos lenguajes; el visual Basic.net y el C++ y se definió además la declaración, la asignación y la lectura de cada una de ellas.

Luego se siguió con el análisis sintáctico en esta fase se agrupan jerárquicamente todos los componentes léxicos en frases gramaticales que el compilador utiliza para sintetizar la salida. Aquí se trabajó con los ciclos y condicionales propios de cualquier código y por supuesto se estudió su estructura dentro de cada lenguaje y la manejabilidad que tiene; los ciclos y condicionales que fueron estudiados, utilizados e implementados fueron el FOR, IF y WHILE.

La integración de estas fases nos dará como resultado el objetivo propuesto que es la creación del software traductor del lenguaje visual Basic .Net al lenguaje C++.

II. DESARROLLO

A. Antecedentes

Actualmente no encontramos muchos programas dedicados a la traducción de lenguajes de programación, pero realizando una búsqueda exhaustiva encontramos uno, el BCX® que es un traductor del lenguaje "BASIC" a lenguaje "C" que tiene una sintaxis bien parecida a "Qbasic" y "PowerBasic", y también tiene cierto parecido a Visual Basic. El lenguaje es fácil de aprender, al igual que el código que traduce. Este programa es distribuido de manera gratuita y no solo es bueno traduciendo para el lenguaje C sino que también el código que traduce ha sido compilado con PellesC, Borland C++ 5.5, LCC-Win32 C compiler, Open Watcom C++ 1.3, MinGW32 C++ compiler y hasta en Microsoft Visual C++ Toolkit 2003.

B. Traductor

Un traductor es un programa que recibe como entrada código escrito en un cierto lenguaje y produce como salida código en otro lenguaje. Generalmente el lenguaje de entrada es de más alto nivel que el de salida. Ejemplos de traductores son los ensambladores y los compiladores.

Es un programa que acepta otros programas escritos en el lenguaje en cuestión y que, o los ejecute directamente, o los transforma en una forma adecuada para su ejecución. Un traductor que ejecuta un programa directamente se conoce como interprete, y un traductor que produce un programa equivalente en una forma adecuada para su ejecución se conoce como compilador. También es posible tener traductores intermedios entre intérpretes y compiladores: un traductor puede traducir un programa fuente a un lenguaje intermedio y después interpretarlo. Estos traductores podrían llamarse pseudointerprete, ya que se ejecuta el programa sin producir un programa objetivo; más bien, procesan la totalidad del programa fuente antes de que se inicie la ejecución [1].

1. Tipos de Traductores

Intérpretes: Un intérprete es un programa informático capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel.

Generalmente existe un subprograma para cada posible acción y que ejecuta dicha acción. Por lo tanto para interpretar un programa habrá que llamar a los subprogramas en la secuencia apropiada. Para ser más exactos, un intérprete es un programa que ejecuta repetidamente la secuencia siguiente: obtener la sentencia siguiente a ejecutar, determinar las acciones que han de ser ejecutadas, ejecutar las acciones [2].

Compilador: Es un programa que lee un programa escrito en un lenguaje; el lenguaje fuente, y lo traduce aun lenguaje equivalente es decir un lenguaje objeto. Como parte importante de este proceso de traducción, el compilador informa a su usuario de la presencia de errores en el programa fuente.

2. Etapas del proceso de traducción

A. Análisis Léxico

El análisis léxico constituye la primera fase, aquí se lee el programa fuente de izquierda a derecha y se agrupa en componentes léxicos (tokens), que son secuencias de caracteres que tienen un significado. El análisis lineal se llama análisis léxico o exploración. Por ejemplo en el análisis léxico los caracteres de la proposición de asignación.

Posición:=inicial + velocidad *60

Se agruparían en los componentes léxicos siguientes:

- El identificador posición.
- El símbolo de asignación.
- El identificador inicial.
- El signo suma.
- El identificador velocidad.
- El signo de multiplicación.
- El número 60.

Es de saberse que todos los espacios en blanco, líneas en blanco, comentarios y demás información innecesaria se elimina del programa fuente. También se comprueba que los símbolos del lenguaje (palabras clave, operadores,...) se han escrito correctamente. En este programa se manejan los nombres de las variables y constantes, así como sus restricciones sobre tamaño y conformación, a través de expresiones regulares, las cuales se construye a partir de expresiones regulares más simples utilizando un conjunto de reglas definitorias. Cada expresión regular r representa un lenguaje (r). Las reglas de definición especifican como se forma L(r) combinando de varias maneras los lenguajes representados por las subexpresiones de r.

Las siguientes son reglas que definen las expresiones regulares del alfabeto Σ. Asociada a cada regla hay una especificación del lenguaje representada por la expresión regular que se está definiendo.

- ε es una expresión regular definida por $\{\varepsilon\}$; es decir, el conjunto que contiene la cadena vacía.
- Si a es un símbolo Σ, entonces a es una expresión regular designada por {a}; por ejemplo, el conjunto que contiene la cadena a. Aunque se usa la misma notación para las tres, técnicamente, la expresión regular a es distinta de la cadena a o del símbolo a. El c contexto aclarará si se habla de a como expresión regular, cadena o símbolo.
- Suponiendo que r y s sean expresiones regulares representadas por los lenguajes L(r) y L(s), entonces,

  a. (r) | (s) es una expresión regular representada por L(r) u L(s).

  b. (r) (s) es una expresión regular representada por L(r) (s).

  c. (r) º es una expresión regular representada por (L(r)) º.

  d. (r) es una expresión regular representada por L(r) [2] [5].

B. Análisis Sintáctico

Es el proceso de determinar si una cadena de componentes léxicos puede ser determinada por una gramática. Un analizador sintáctico deberá poder construir el árbol de otro modo no se puede garantizar que la traducción sea correcta.

Un árbol de análisis sintáctico indica gráficamente como el símbolo inicial de a una gramática deriva una cadena del lenguajes, el no Terminal A tiene una producción A-> XYZ, entonces el árbol de análisis sintáctico puede tener un nodo interior etiquetado con A y tres hijos etiquetados con X, Y, Z, de izquierda a derecha. ver figura 1.

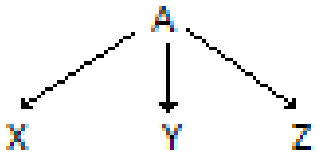

Figura 1.

Formalmente en una gramática independiente del contexto, un árbol de análisis sintáctico es un árbol con las siguientes propiedades:

- La raíz esta etiquetada con el símbolo inicial.
- Cada hoja esta etiquetada con un componente léxico o con vació.
- Cada nodo interior esta etiquetado con un no Terminal.
- Si A es el no Terminal que etiqueta a un nodo interior y X1, X2…, Xn son las etiquetas de los hijos de ese nodo, de izquierda a derecha, entonces A-> X1, X2…, Xn es una producción [3].

Gramática independiente del contexto (GIC): Una gramática libre de contexto (GIC), también llamada gramática no-conceptual o gramática de tipo 2, es una cuádrupla, G= (V, Σ, S, P) por:

- Un alfabeto V cuyos elementos se llaman variables o símbolos no-terminales.
- Un alfabeto Σ cuyos elementos se llaman símbolos terminales.
- Una variable especial S V, llamada símbolo inicial de la gramática.

La denominación "independiente de contexto" proviene del hecho de cada producción o regla de re-escritura A->w se aplica A independientemente del contexto en el que aparece A.

Ej. Sea G una gramática (V, Σ, S, P) dada por:

V= {S}

Σ = {a}

P= {S->aS, S-> λ}

Se tiene S=> λ y

S=>aS => a… aS =>a…a.

Por consiguiente, L (G)=a*.

De manera más simple, podemos presentar una gramática GIC listando sus producciones y separando con el símbolo | las producciones de una misma variable [4].

Gramática LL: El analizador sintáctico LL es un analizador sintáctico descendente, por un conjunto de gramática libre de contexto. Éste analizador las entradas son de izquierda a derecha. La clase de gramática que es analizable por éste método es conocida como gramática LL.

Un analizador LL es llamado un analizador LL (k) si usa k tokens cuando el analizador va hacia delante de la sentencia. Si existe tal analizador para cierta gramática y puede analizar sentencias de ésta gramática sin marcha atrás, entonces es llamada una gramática LL (k).

El trabajo del analizador sobre una cadena de gramática particular consiste en:

- Una búfer de entrada, una cadena de gramática.
- Una pila sobre la cual se almacenan los símbolos terminales y no terminales de la gramática aún sin analizar.
- Una tabla de análisis.

Lenguaje de programación: Un lenguaje de programación es un lenguaje artificial, diseñado

para representar expresiones e instrucciones de forma inteligible para las computadoras.

Los lenguajes de programación son como idiomas que constituyen el sistema de comunicación entre el hombre y el ordenador, mediante el cual se transmiten a este las instrucciones e información en un formato comprensible para la maquina. Los lenguajes que se aproximan al código máquina (ceros y unos) se denominan de alto bajo nivel, mientras que los que se parece al de los usuarios se denomina de alto nivel [6].

Lenguaje C++: C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales [7].

Visual Basic.Net: Es un lenguaje de programación desarrollado por Alan Cooper para Microsoft. El lenguaje de programación es un dialecto de BASIC, con importantes añadidos. Su primera versión fue presentada en 1991 con la intención de simplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y en cierta medida también la programación misma.

Visual Basic constituye un IDE (entorno de desarrollo integrado o en inglés Integrated Development Enviroment) que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código (programa donde se escribe el código fuente), un depurador (programa que corrige errores en el código fuente para que pueda ser bien compilado), un compilador (programa que traduce el código fuente a lenguaje de máquina), y un constructor de interfaz gráfica o GUI (es una forma de programar en la que no es necesario escribir el código para la parte gráfica del programa, sino que se puede hacer de forma visual). El Entorno de Desarrollo recibe el nombre de Entorno de Desarrollo de Microsoft Visual Studio .NET. Este entorno es personalizable y contiene todas las herramientas necesarias para construir programas para Microsoft Windows [8].

III. METODOLOGÍA

La investigación cuantitativa es un método de investigación basado en los principios metodológicos de positivismo y neopositivismo y que adhiere al desarrollo de estándares de diseño estrictos antes de iniciar la investigación.

Además de la metodología antes mencionada también tenemos otra metodología la cual es utilizada para el desarrollo de un sistema software el

cual es llamado modelo en cascada, que sugiere un enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma tal que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

Además de estas metodologías, se tuvo en cuenta para la puesta en marcha del software las fases de un compilador como son la de análisis léxico y sintáctico, en las cuales basamos el software de traductor de lenguajes.

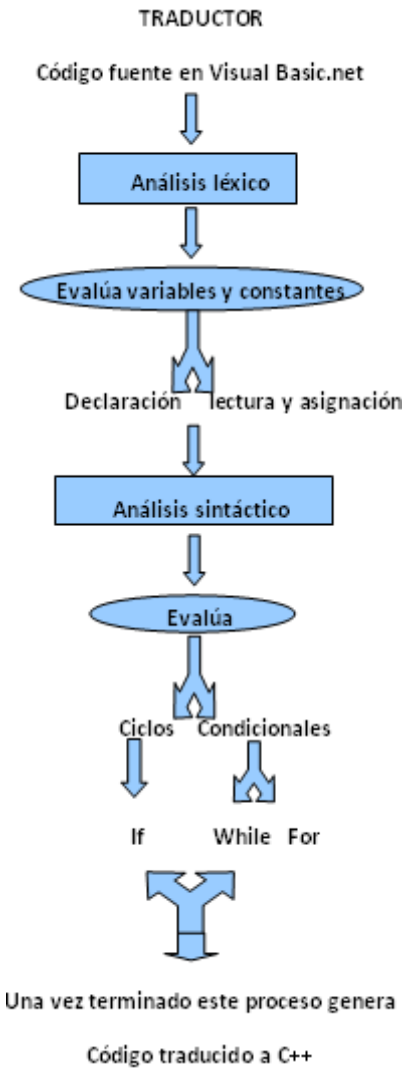

Figura 2. Descripción de las partes del traductor, y sus componentes más relevantes

En la siguiente figura (ver figura 2) se muestran algunos de los componentes o partes relevantes de este traductor, la parte exacta donde fueron utilizados los componentes léxico y sintáctico. En el análisis léxico se tuvo en cuenta las variables y constantes de ambos lenguajes el visual Basic.net y el C++, además fue muy importante saber cuales son las reglas que deben cumplir cada una de ellas para así poder realizar la declaración, la asignación, y la lectura de dichas variables y constantes.

Por otra parte en el análisis sintáctico se trabajo con los ciclos y condicionales los cuales son pieza clave en la formación de cualquier código, por supuesto se estudió su estructura y la forma en que se trabaja en cada uno de los dos lenguajes, los ciclos y condicionales en los que se trabajó fueron el IF, WHILE, y FOR.

IV. RESULTADOS

A esta a altura del proyecto se tiene parte o gran parte por así decirlo de los resultados totales del proyecto, aproximadamente un 99% de dicho proyecto, se tiene como primer término la traducción de la parte de declaración de variables de todo tipo, enteras, cadenas de caracteres, vectores y constantes, continuamos con la lectura y escritura de la variables, asignación de variables y operaciones aritméticas, siguiendo un orden estructural tenemos las condiciones o comparaciones con los operadores lógicos y relacionales. En base a esto se realizó la estructura del condicional if, while y for. Además de esto ya es un hecho el manejo de los comentarios dentro del código a traducir A continuación, en las figuras 3, 4, 5, 6, se presentan los pantallazos del software realizado aquí se muestra la declaración, asignación y lectura de variable y constantes. En la parte superior se escribe la codificación en visual Basic.net y después de verificar que todo este correcto se pasa al proceso de traducción en C++, el cual se muestra en la parte de abajo del pantallazo.

En las gráficas 3, 4, 5, 6 se muestran traducciones de líneas de código en el cual se introduce el código en visual Basic.Net que se va a traducir, como siguiente paso tenemos la opción de verificar el código para

comprobar que esté correcto, para esto hacemos clic en el botón verificar o en su defecto en el icono de verificar que se encuentra en el menú, después si podemos proceder a traducir el código, es vital que el código a traducir no presente errores porque entonces el programa no nos dará la opción de traducirlo.

En base a estos resultados tenemos que el proyecto esta dando una solución al problema mencionado al comienzo de este escrito.

Declaración

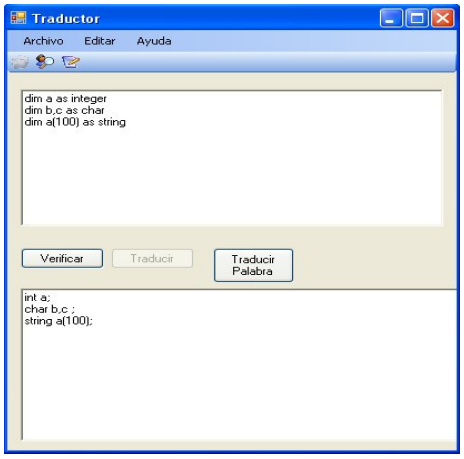

Figura 3. Declaración de variables

En la parte superior se encuentra el código en visual .Net y debajo la traducción en C++.

Escritura y Lectura

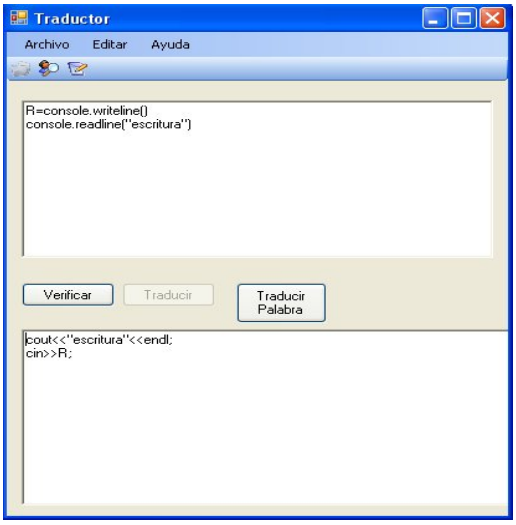

Figura 4. Escritura y lectura de una variable

Asignación.

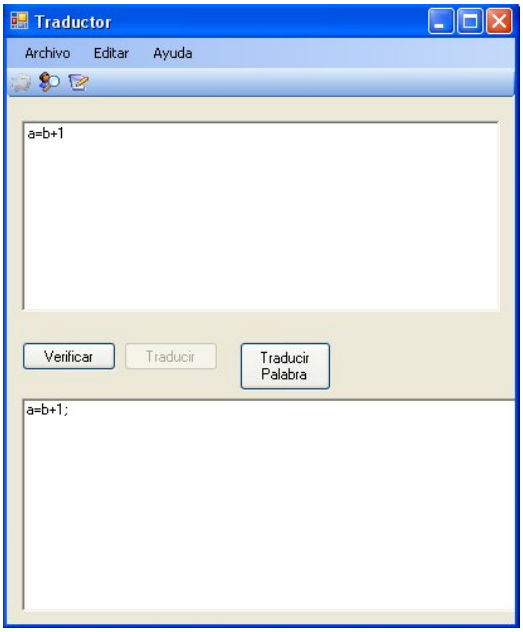

Figura 5. Asignación de variables

Condicional y Ciclos

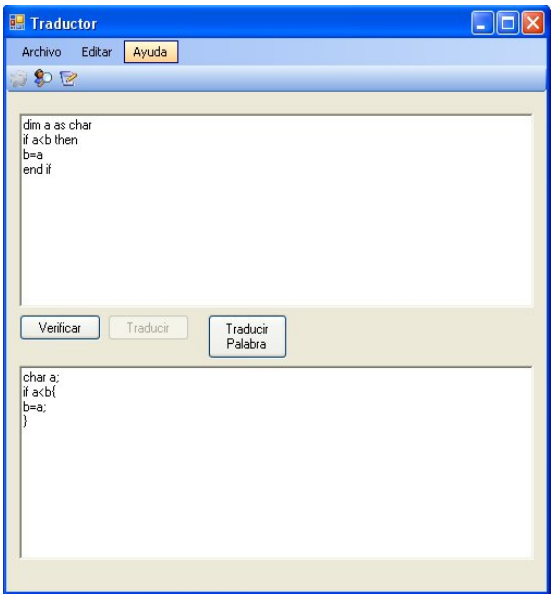

Figura 6. Condicional IF

V. CONCLUSION

Podemos concluir que fueron muy importantes los conocimientos adquiridos acerca de la teoría de compiladores ya que nos proporcionó una base indispensable para la realización del software. Además de la adquisición de nuevos conocimientos, este software ayudará a muchas personas tanto

estudiantes como programadores que deseen utilizarlo, ya que puede darse el caso que el alumno o el programador necesiten realizar un programa empleando C++, pero si no manejan este lenguaje pueden llevarlo a cabo en visual basic.Net y a través de este software traducir el código a C++ que es donde lo necesitan. Para la parte final de este proyecto no se hicieron pruebas piloto al programa con los estudiantes debido a que estos ya no se encontraban en horas de clase si no en parciales finales. Sin embargo estas pruebas se realizarán mas adelante, a un grupo de estudiantes de la asignatura de estructura de datos, que es en donde se maneja c++ actualmente, y se efectuarán en una segunda parte de este proyecto que se llevara a cabo por parte de los autores, con el fin de mejorar y corregir el programa en las deficiencias que este pueda presentar.

REFERENCIAS

[1] Kenneth C. Lounden. Lenguaje de programación. Principios y Prácticas. Segunda edición.

[2] G.Sanchez Dueñas, J.A Valverde Andreu. Compiladores e intérpretes: un enfoque pragmático. 2ª edición corregida y ampliada.

[3] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman. Compiladores: principios, técnicas y herramientas. Primera edición, 1990. Primera reimpresión en México, 1998

[4] De Castro Rodrigo. Teoría de la computación, Lenguajes, autómatas, gramáticas. Primera edición, 2004.Pag.82

[5] Pérez Iván. Lenguaje y Compiladores. Primera edición 2005.Caracas, Venezuela.Pag.38.

[6] De Pablos Carmen, López-Hermoso José Joaquín, Martin-Romo Santiago, Medina Sonia. Informática y comunicaciones en la empresa. Editorial ESIC. Pág. 110

[7] Ceballos Francisco Javier. C/C++ curso de programación 2ª edición. Madrid España. Editorial Alfaomega Ra-ma.663 p.

[8] Programatium, visual basic.net, consultado el 25 de octubre del 2009, disponible en la url: http://www.programatium.com/vbnet.html