REVISTA DIGITAL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES
TELEM@TICA
Revista Telem@tica. Vol. 11. No. 2, mayo-agosto, 2012, p. 36-45 ISSN 1729-3804

UTILIZANDO BIBLIOTECAS DE CÓDIGO *C* EN APLICACIONES *C#*

Ing. Wifredo Massó Gómez[1]

1) UEB Aplicaciones de Redes, Sancti Spíritus. wifredo@elecssp.une.cu Calle Oeste, Final.OBE Provincial, Sancti Spíritus.

Resumen

Actualmente es muy frecuente emplear dos lenguajes de programación para desarrollar un sistema informático. Casi siempre con el objetivo de mejorar la calidad de los mismos, en una industria cada vez más competitiva y exigente. Esta técnica permite lograr que un software complejo pueda ser rápido y a la vez vistoso y cómodo. El uso creciente, el reconocimiento internacional y la calidad probada del lenguaje de programación C y uno de sus derivados, C#, los convierten en puntos de referencia a la hora de escoger, cuando se decide realizar un proyecto de gran envergadura. Además, para un desarrollador resulta bastante fácil adaptarse a uno de ellos teniendo conocimiento del otro. Una forma eficiente de emplear código escrito en C, en aplicaciones desarrolladas en C#, resulta la creación de una biblioteca de vínculos dinámicos o DLL. La cual puede ser fácilmente portada e invocada, y su uso está muy extendido, sobre todo en el funcionamiento de los sistemas operativos. Este método de implementación de sistemas cobra fuerza desde hace ya un tiempo, sobre todo cuando se desarrolla software que necesita cálculos matemáticos complejos, procesamiento de intenso de imágenes o manejo de grandes volúmenes de datos.

Palabras Clave: lenguaje C, lenguaje C#, DLL, biblioteca de código, MS Visual Studio.

USING C CODE LIBRARIES IN C# APPLICATIONS

Abstract

At present is very common to use two different programming languages to develop software; its main objective is improving the quality of the informatics systems. The technique allows to achieve more velocity, performance and comfort, all in one. Programming languages like C and C# have international recognition, because of their quality and standardization, for that reason they are very used in complex projects. Also, for a developer it turns out to be quite easy to adapt itself to one of them, having knowledge of other. To make a Dynamic Link Library (DLL) is an efficient way of using code written in C, in applications developed in C#. DLLs are easily loaded and its use is most extended than other kinds of library, operating systems are based mostly in DDLs. The code libraries are very useful in the creation of software that needs complex mathematical calculation, intense image processing or voluminous data handling.

Keywords: language C, language C#, DLL, code library, MS Visual Studio.

INTRODUCCIÓN

La implementación de cálculos matemáticos complejos o el manejo de grandes volúmenes de información, necesitan un tiempo de respuesta adecuado; por lo que generalmente se emplean lenguajes de programación al más bajo nivel posible. Por otro lado, los usuarios de los sistemas son cada vez más exigentes con la calidad visual y la comodidad para operar el producto; las herramientas de implementación de sistemas más avanzadas son puestas en función de obtener mejores resultados en este aspecto. Es muy eficiente, para la creación de software que necesite ambas características, combinar dos lenguajes de programación. Por supuesto, mientras más semejanzas existan entre ellos, más fácil será establecer la conexión e intercambiar información; de igual forma será mucho más cómodo para los desarrolladores.

El lenguaje C es muy empleado a nivel mundial para la creación de repositorios o bibliotecas de código. Es muy común que estos "almacenes" de funciones se hagan en forma de fichero DLL, debido a que los sistemas operativos actuales basan su funcionamiento en archivos de este tipo. Además la plataforma .NET y el lenguaje de programación C# permiten crear sistemas con una gestión de recursos al más alto nivel y se integran perfectamente a un conjunto de herramientas que son, además de muy atractivas visualmente, notablemente eficientes para la creación de sistemas complejos.

Existe la tendencia de dividir los grandes proyectos de software en partes. Las partes críticas por su necesidad de rapidez de cálculo son realizadas en lenguajes de tipo compilado, en tanto las que realizan una mayor gestión de los recursos del sistema se realizan en lenguajes de tipo dinámico o script. La parte crítica debe rebasar un determinado por ciento del total (aproximadamente un 10 %), de otro modo se emplean solamente lenguajes dinámicos.

Se propone la creación de un repositorio de datos, programado en C, con el código necesario para el funcionamiento de una aplicación; cuyos componentes principales han sido desarrollados en C#. Teniendo en cuenta las ventajas de la creación y utilización de DLLs en la implementación de sistemas de gran envergadura y lo sencillo que resulta para un programador con conocimiento de estos lenguajes integrarlos en una misma aplicación, se sugiere que dicho repositorio esté implementado en un fichero de este tipo.

Lenguaje C

El lenguaje de programación C, es uno de los más conocidos y empleados en la actualidad; es muy apreciado por la eficiencia del código que produce. Fue desarrollado entre 1971 y 1973, por Dennis Ritchie en los Laboratorios Bell de la AT&T (**American Telephone and Telegraph**), basado en un lenguaje anterior llamado B. Su crecimiento fue de la mano con el desarrollo temprano de los sistemas Unix; incluso inicialmente estaba asociado exclusivamente a éstos. A mediados de los años 80 fue estandarizado por primera vez, con el estándar X3.159 de la ANSI (**American National Standards Institute**) [1]. Posteriormente en 1990 fue ratificado como estándar ISO (ISO/IEC 9899), lo que garantiza la portabilidad del código entre plataformas y arquitecturas.
Ha sido influido por Fortran, y a su vez pueden encontrarse algunas de sus características en la mayor parte de los lenguajes existentes en la industria informática (ej. Java, JavaScript, Modula 3, Eiffel,…). Algunas de sus evoluciones más conocidas son Objective-C, C++ y C#; también muy populares. Inicialmente desarrollado en una pequeña computadora personal se ha convertido en uno de los lenguajes dominantes en la actualidad [2]. Como se mencionó, fue concebido originalmente para la

fluidez de la programación en los sistemas Unix, sin embargo con el tiempo se ha empleado también en el desarrollo de otras plataformas y sistemas operativos, como Windows.

Está débilmente tipificado como un lenguaje de medio nivel. Dispone de las estructuras típicas de alto nivel y a su vez de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones a este lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria y dispositivos periféricos. Estas características propician la realización de implementaciones óptimas y a la vez no es necesario un soporte intenso en tiempo de ejecución. Es uno de los lenguajes más portados en existencia, se conocen compiladores para casi todos los sistemas operativos modernos; proporciona muchas facilidades para realizar programas modulares y emplear código existente. Es muy flexible y posibilita programar con varios estilos, uno de los más empleados es el estructurado. Tiene entre sus características: un conjunto reducido de palabras clave, acceso a memoria de bajo nivel mediante punteros, sistema de tipos que impide operaciones sin sentido e interrupciones al procesador mediante uniones.

Sin embargo, carece de algunas ventajas que proporcionan otros lenguajes, como son: recolección de basura, encapsulamiento, soporte para programación orientada a objetos, funciones anidadas, polimorfismo y solo dispone de un soporte rudimentario para la programación genérica. Además, comparado con otros lenguajes, puede resultar un poco lento desarrollar en C y el mantenimiento puede ser tan costoso como inexperto sea el programador, ya que depende casi exclusivamente de su habilidad [2].

Existe un gran número de bibliotecas creadas en código C. Algunas incluyen la implementación de gran cantidad de fórmulas y funciones; por ejemplo la GNU Scientific Library (GSL), incluye más de mil funciones estadísticas y matemáticas, que pueden ser invocadas de forma sencilla desde cualquier proyecto [3]. En la Tabla 1 podemos encontrar tres ejemplos de repositorios de código C; los tres son muy empleados a nivel mundial en sus respectivos campos y son muestra de la fuerza adquirida por este lenguaje en la creación de bibliotecas [3, 4, 5].

Tabla 1. Ejemplo de bibliotecas de código implementadas en lenguaje C.

Biblioteca	Creado en	Desarrollado por	Implementa
GNU Scientific Library (GSL)	2009 (Versión 1.13)	GNU National Laboratory, Los Álamos CA, EUA	Rutinas Matemáticas y Estadísticas
SGI Open GL	2004 (Versión 2.0)	Silicon Graphics Inc., EUA	Generación de Gráficos y Procesado de Imágenes
SVM Light	2008 (Versión 6.2)	Thorsten Joachims, Dotrmund University, Alemania	Máquinas de Soporte Vectorial

Lenguaje C #

En la actualidad se han desarrollado varias herramientas y técnicas para lograr una rápida implementación y puesta a punto de aplicaciones empresariales. Existen lenguajes y entornos de programación que permiten un rápido desarrollo, que unido con la flexibilidad del lenguaje de programación, hacen que los resultados obtenidos sean muy eficientes. Uno de ellos es C# (se pronuncia

Sharp), creado por Anders Hejlsberg en 1999; es un lenguaje de programación orientado a objetos, simple y moderno [6].

El primer comité técnico para su aprobación como un estándar, promovido por Hewlett- Packard, Intel y Microsoft, se reunió inicialmente en septiembre de 2000 y en diciembre de 2001 se hizo público el borrador de la especificación oficial del estándar por parte de la ECMA (siglas en inglés de la **Asociación Europea de Fabricantes de Computadores**, con sede en Ginebra, Suiza) [7]. Algunas de las características del lenguaje que influyeron en su desarrollo y rápida aceptación son: la inclusión de principios de ingeniería del software, la capacidad para desarrollar componentes de software que pueden ser empleados en ambientes distribuidos, la portabilidad del código fuente y la fácil migración del programador al nuevo lenguaje, especialmente los familiarizados con C/C++. Posteriormente sería adoptado, también, como un estándar ISO.

Su desarrollo se debe a Microsoft, como parte de su plataforma .NET; sin embargo, este es un lenguaje independiente. Su sintaxis básica se deriva de C y C++, lo que permite un fácil ajuste de cada una de las partes del sistema en desarrollo, en tanto su sencillez y alto nivel de productividad son equiparables a Visual Basic. El modelo de objetos de la plataforma .NET es similar a Java, pero incluye mejoras derivadas de otros lenguajes como Delphi, Modula 2 y Smalltalk [6]. Existen otros compiladores para el lenguaje, destacan: Borland Developer Studio 2006 y el que provee el framework DotGNU-Mono.

Su principal desventaja radica en que aunque sus aplicaciones están orientadas a ser económicas en cuanto a requisitos de memoria y proceso, no puede competir directamente en velocidad y tamaño con lenguajes como C.

La plataforma .NET constituye un conjunto de tecnologías para desarrollar y utilizar componentes que nos permitan crear páginas web dinámicas, servicios web XML, componentes de acceso a datos, aplicaciones de escritorio clásicas de Windows o incluso aplicaciones de cliente inteligente. Aunque es posible escribir código para esta plataforma en otros lenguajes, el único que ha sido desarrollado específicamente para el trabajo en ella es C#. Trabajar con este lenguaje es mucho más sencillo e intuitivo, pues carece de elementos heredados innecesarios en esta plataforma. Permite la implementación de software robusto y duradero siendo reconocido por el uso de excepciones, recolección de basura, comprobación de tipos, ausencia de variables sin inicializar, gestión de versiones y eliminación de errores comunes [8].

Biblioteca de Vínculos Dinámicos

Una biblioteca de vínculos dinámicos o DLL (acrónimo de **Dynamic Link Library**) como cualquier otra biblioteca, contiene subrutinas ejecutables utilizadas en sistemas operativos o programas específicos, que proporcionan funciones para los más disímiles fines. Se crean como archivos independientes con extensión DLL y sólo se cargan cuando el programa principal las llama. Mientras no son llamadas por un programa no ocupan memoria y pueden ser utilizadas desde varios sitios [7]. Su diferencia principal con las bibliotecas estándar, radica en que una vez que ya no son necesarias sus funciones, el sistema operativo las puede descargar de la memoria. Esta extensión es empleada en sistemas operativos Windows, aunque los ficheros con este concepto existen prácticamente en todos los sistemas operativos modernos.

Su uso provee varias ventajas, como pudieran ser: la reducción de los archivos ejecutables, debido a que gran parte del código estaría almacenado fuera de ellos o el hecho de poder emplearse por varias

aplicaciones (siempre que el código sea suficientemente genérico). Además facilitan la gestión y aprovechamiento de la memoria del sistema y en muchos casos propician una mayor flexibilidad frente a cambios, puesto que en ocasiones una mejora o corrección beneficiará a varias aplicaciones. Pero esta técnica no está exenta de problemas, el más común ocurre cuando una nueva versión de un programa borra o cambia considerablemente un fichero DLL sin tener en cuenta que este es indispensable para el correcto funcionamiento de otro programa.

Una DLL muy conocida es la MFC (**Microsoft Foundation Classes**), la que provee un acceso sencillo a las API (**Interfaz de Programación de Aplicaciones**, según sus siglas en inglés) de Windows. Desde 1992, Microsof ha venido desarrollando nuevas versiones con actualizaciones del entorno de programación Visual C++ [7].

Para la creación de una DLL con código en lenguaje C, pueden emplearse varias herramientas; Microsoft Visual Studio (en cualquiera de sus versiones) pudiera ser la elección, teniendo en cuenta que está concebido para el trabajo en el entorno .NET y la aplicación se desarrollará en C#. Visual Studio permite explotar, como ningún otro IDE, todas las ventajas de la plataforma .NET y posee extensiones para el desarrollo en varios lenguajes de programación (C++, C#, J#, Visual Basic.NET, ASP.NET,…) [9].

Creando una DLL

A continuación se propone un ejemplo de cómo puede crearse una DLL, programada en C. Para ello puede emplearse la herramienta MS Visual Studio 2008, previamente introducida. Además el ejemplo incluye la forma de emplear esta biblioteca desde una aplicación desarrollada en C#.

Para comenzar debe crearse un nuevo proyecto en C, en Visual Studio se debe seleccionar un nuevo proyecto de consola C++, justo como se indica en la Fig.1; una vez creado el proyecto téngase en cuenta que las opciones que se seleccionen se ajusten a las características del proyecto (véase la Fig.2).

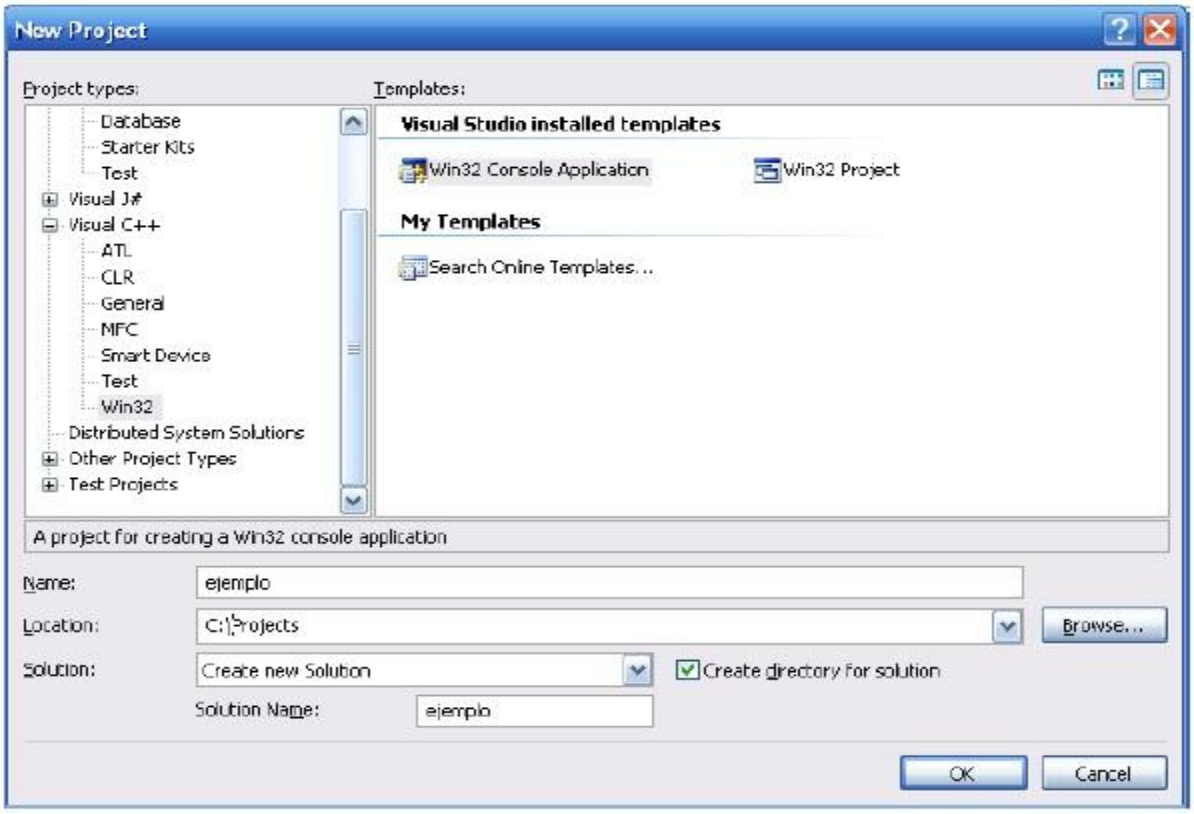

Figura 1. Creando un nuevo proyecto en C.

Figura 2. Opciones de la Aplicación

Con el proyecto y los archivos de código (*.h, *.c) ya creados se debe especificar (en las propiedades generales) que se empleará un compilador de Ansi C; si en la versión de Visual Studio utilizada no existe un compilador de este tipo pudiera instalarse uno (ej. Intel C++ Compiler). Se puede compilar la DLL tanto en Debug como en Release.

Los métodos y estructuras deben hacerse preferentemente, no es una regla, en una misma Unit (*.c), en tanto las declaraciones en otra (*.h). Las declaraciones de las funciones o tipos de datos definidos por el programador, que se quieran emplear desde fuera de la DLL, deben estar precedidas por la sentencia:

```
__declspec (dllexport) (Declaración de la función);
```

para organizar mejor el trabajo se sugiere definir una variable de la siguiente forma:

```
#define DLLEXPORT __declspec (dllexport),
```

así cada función estará antecedida por la sentencia DLLEXPORT. Una vez compilado el proyecto, la herramienta genera algunos archivos con extensiones distintas (uno de ellos *.dll), pero todos con el mismo nombre.

Una vez creado un proyecto C#, los archivos generados anteriormente se deben ubicar dentro de la carpeta (Nombre del Proyecto) \bin\Debug. Dentro del proyecto sugerimos crear una unit (*.cs), para importar las funciones. Esta página necesita en el encabezamiento la siguiente sentencia:

```
using System.Runtime.InteropServices;
```

y la función puede ser usada de la siguiente forma:

```
[DllImport("(Nombre de la DLL).dll", CharSet = CharSet.Ansi)]
public static extern (Declaración de la función);
```

Debe recordarse que muchos tipos de datos no están declarados de la misma forma en C y C#. Por tanto una cadena de caracteres en la declaración de la función en C, aparecería como char*; y en C# sería StringBuilder, por solo citar un ejemplo. Para una mejor comprensión de la explicación se muestran a continuación algunos ejemplos de código.

Suponiendo que se ha creado una DLL en lenguaje C con el nombre “ejemplo”, en ejemplo.h pudiera existir el siguiente código:

```
#define DLLEXPORT __declspec (dllexport)
DLLEXPORT void Cargar_Fichero (char* nombre);
DLLEXPORT int Suma (int a, int b);
```

Aquí se declaran dos funciones que estarán implementadas en ejemplo.c; la primera carga los datos de un fichero en memoria, conociendo el nombre y la otra realiza una suma de enteros.

Con un proyecto ya creado en C#, se puede llamar las funciones de la DLL de la siguiente forma:

```
using System;
```

```
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;
using System.IO;

namespace Mi_Proyecto
{
    public class Wrapper
    {
        [DllImport("ejemplo.dll", CharSet = CharSet.Ansi)]
        public static extern int Cargar_Fichero(StringBuilder nombre);
        public static extern int Suma(int a, int b,);
    }
}
```

Este código se recomienda ubicarlo por separado, empleando para ello una clase y un archivo *.cs específicos. Las funciones importadas desde el archivo DLL pueden utilizarse dentro del proyecto de la misma forma que las funciones implementadas en la aplicación.

CONCLUSIONES

Las ventajas de esta técnica son de gran utilidad en la elaboración de proyectos informáticos de gran envergadura, complejos por la cantidad y tipo de datos que necesitan. Los sistemas de tratamiento de imágenes son algunos de los que la emplean, debido a la necesidad del cálculo constante mediante fórmulas complejas. En la minería de datos, teniendo en cuenta el volumen elevado de información que se maneja y los complejos métodos de predicción y clasificación que se utilizan, también se hace bastante uso de la combinación de lenguajes de programación en las implementaciones. Podemos encontrar otros ejemplos en distintos campos de la inteligencia artificial, así como en la creación de sistemas para controles automáticos.

REFERENCIAS

1. Ritchie, Dennis M.; The Development of the C Language. Bell Labs/Lucent Technologies Inc. Murray Hill, New Jersey, USA, 1993.

2. Kernighan, Brian; y Ritchie, Dennis M.; The ANSI C Programming Language - Second Edition. Prentice Hall Software Series. 2008.

3. Galassi, M.; Theiler J.; y Jungman G.; GNU Scientific Library Reference Manual - Third Edition. Free Software Foundation, Boston, MA. Enero, 2009.

4. SGI OpenGL, www.sgi.com. 15/05/2010

5. SVM Light. Support Vector Machine. www.svmlight.com. 10/5/2010.

6. Microsoft Corporation. www.microsoft.com. 15/05/2010.

7. Microsoft Corporation. Biblioteca de vínculos dinámicos. Microsoft Encarta, 2009.

8. González, José A.; El Lenguaje de Programación C#. Wrox Press, 2002.

9. Microsoft Corporation. The Visual Studio Combined Help Collection. MS Visual Studio, 2005.