**DLL и Дельфи**

Думаю, многие знают, что такое DLL (dynamic link library - динамические библиотеки). У библиотек есть немало преимуществ, достаточно веских, что бы их использовать. В этой статье мы научимся создавать и использовать динамические библиотеки в своих проектах.

Зачем они нужны

А зачем эти самые библиотеки мне нужны? - спросите вы. Ну я не знаю, может они вам вообще не нужны. А может и жизненно необходимы. Перечислю возможности и преимущества библиотек:

Универсальность. Любой программист, зная описания и имена функций, находящихся в библиотеке, может использовать их.

Удобность отладки. Вы можете разместить несколько важных функций в библиотеке и обьявить их в программе. При этом вы будете работать с библиотекой, отлаживая эти функций, не трогая основную программу.

Хранилища ресурсов.. В DLL можно хранить ресурсы, такие как рисунки, формы, иконки меню, и т.д.

Взгляд на будущее. С помощью библиотек можно легко создавать плагины, расширяющие стандартные возможности программы. Т.е. можно не выпускать различные версии программы, а выпускать плагины или модифицированные (например с исправленными ошибками) версии библиотек.

Совместное использование. Если библиотека загружена, то её могут использовать и другие приложения.

Экономия ресурсов. Библиотеку можно загрузить и выгрузить тогда, когда это действительно необходимо. Например, программа в нужный момент загрузила DLL, вызвала функцию, сделала работу и выгрузила библиотеку до следующего раза. Налицо экономия памяти

Вобщем DLL - зверь полезный и очень даже дружелюбный.

Структура динамической библиотеки

Что бы создать библиотеку в Delphi6 выберите File -> New -> Other и в появившемся окне выберите DLL Wizard. Дельфи сгенерирует шаблон для библиотеки:

library Project;

{ Important note about DLL memory management: ShareMem must be the

first unit in your library"s USES clause AND your project"s (select

Project-View Source) USES clause if your DLL exports any procedures or

functions that pass strings as parameters or function results. This

applies to all strings passed to and from your DLL--even those that

are nested in records and classes. ShareMem is the interface unit to

the BORLNDMM.DLL shared memory manager, which must be deployed along

with your DLL. To avoid using BORLNDMM.DLL, pass string information

using PChar or ShortString parameters. }

uses

SysUtils,

Classes;

{$R \*.res}

begin

end.

В комментарии указывается на необходимость вставить ссылку на модуль ShareMem, если библиотека экспортирует длинные строки в параметрах обращения к подпрограммам или как результат функций. Эта ссылка должна быть первой как в предложении uses библиотеки, так и в uses файла проекта программы, которая использует эту библиотеку. Если подпрограммы библиотеки экспортируют строки ShortString или PChar, ссылаются на ShareMem не обязательно. Что бы не возникало недоразумений в своих библиотеках я рекомендую вместо типа String пользоваться PChar, а по необходимости конвертируйте типы функциями PChar (конветирует из String в PChar) и StrPas (конвертирует из PChar в String).

Структура библиотеки похожа на структуру обычного модуля. Теперь создайте библиотеку с таким текстом:

library Project2;

uses

SysUtils,

Classes;

function MyFunc(num1, num2, Errcode : Integer; Operation : PChar) : Integer; stdcall;

begin

try

if Operation="plus" then

Result := num1+num2;

if Operation="minus" then

Result := num1-num2;

if Operation="multiply" then

Result := num1\*num2;

if Operation="div" then

Result := num1 div num2;

if Operation="mod" then

Result := num1 mod num2;

except Result := Errcode;

end;

end;

exports

MyFunc INDEX 1 NAME "MathFunc";

begin

end.

Сохраните это все куда нибудь и скомпилируйте (Ctrl+F9)

Это будет демонстрационная библиотека, на которой я буду показывать различные приемы работы с DLL. Но для начала давайте рассмотрим текст этой библиотеки.

function MyFunc(num1, num2, Errcode : Integer; Operation : PChar) : Integer; - это обычная функция, возвращающая целое число. Основываясь на параметре Operation функция решает, какую операцию сделать над операндами num1 и num2. В случае ошибки она возвращает переданный ей параметр Errcode. Т.е. в программе можно будет проанализировать, возникла ли ошибка во время исполнения функции.

stdcall указывает на то, что функция будет вызываться "обычным" способом, т.е. программы, написанные на других языках тоже смогут пользоваться библиотекой. Можно использовать - "register", предназначенным только для использования программами, написанными в среде дельфи, но тогда программы, написанные не в дельфи не смогут обращаться к этой функции.

exports

MyFunc INDEX 1 NAME "MathFunc";

Раздел Exports помогает компилятору и компоновщику создать специальный заголовок DLL-модуля, в котором перечисляются имена подпрограмм и адреса их точек входа. В DLL может быть несколько списков Exports, но перечисляемые в них подпрограммы должны быть описаны где-то выше по тексту библиотеки. Помимо имени подпрограммы в заголовок DLL помещается также ее порядковый

номер (INDEX), точнее, присвоенный ей целочисленный индекс. Это позволяет вызывающей программе ссылаться не на имя, а на индекс подпрограммы и тем самым уменьшить затраты времени на установление с ней связи. Индекс присваивается подпрограмме по порядку ее появления в списках Exports: первая подпрограмма в первом списке получает индекс 0, следующая 1 и т. д.

Программист может изменить умалчиваемую индексацию и явно указать индекс подпрограммы, добавив за ее именем в списке Exports слово index и целое число в диапазоне от 0 до 32767. Помимо индекса можно указать также и произвольное (NAME) имя функции.

Надеюсь, я понятно обьяснил ;) Вобщем наша демонстрационная библиотека готова. Теперь давайте научимся пользоваться библиотечными функциями

Использование библиотечных функций

Использовать функции из библиотеки можно двумя способами:

1. Привязка библиотеки к программе (статическая загрузка)

Недостатки:

- нет эффекта экономии ресурсов (библиотека загружается при запуске программы и выгружается при завершении программы)

- при отсутствии хотя бы одной из необходимых библиотек в папке с программой, либо в папке $windir$/system программа не запускается и выдает сообщение об ошибке

- при отсутствии хотя бы одной из необходимых функций в библиотеке при запуске программа выдает сообщение об ошибке и не запускается

Преимущества:

- легкость использования

У этого способа много недостатков. Но все же он будет полезен начинающим программистам. Для использования функций или процедур из библиотеки таким способом нужно всего лишь в разделе implementation указать имя функции или процедуры примерно так:

//если функция

function FunctionName(Par1: Par1Type; Par2: Par2Type; ParN : ParNType): ReturnType; stdcall; external "MyDLL.dll" name "FunctionName" index FunctionIndex;

//если процедура

procedure ProcedureName(Par1: Par1Type; Par2: Par2Type; ...); stdcall; external "MyDLL.dll" name "ProcedureName" index ProcIndex;

Рассмотрим обьявление функции.

function FunctionName(Par1: Par1Type; Par2: Par2Type; ParN : ParNType): ReturnType; - Это собственно обьявление функции

external "MyDLL.dll" эта директива указывает на имя библиотеки, из которой будет вызвана функция (в нашем случае это MyDLL.dll)

name "FunctionName" необьязательная директива, которая указывает на имя функции в библиотеке; используется для повышения скорости доступа к функциям (имя определяется внутри библиотеки)

index FunctionIndex тоже необьязательная директива, использующаяся для ускорения доступа к функциям; указывает на индекс функции (индекс обьявляется в самой библиотеке).

Рассматривать обьявление процедуры не имеет смысла, т.к. процедурв вызывается точно так же (за исключением того, что у процедура ничего не возвращает). Вот и все! Теперь можно пользоваться обьявленой функцие в пределах модуля, в котором она была обьявлена.

Рассмотрим пример на основе нашей демонстрационной библиотеки, которую мы скомпилировали выше.

Создайте новый проект Project1 и на его форму поместите четыре поля Edit. Присвойте им такие имена: Num1Edit, Num2Edit, OpEdit, ResultEdit. Так же поместите одну кнопку, имя которой значения не имеет. В разделе implementation обьявите функцию:

implementation

function MyFunc(num1, num2, Errcode : Integer; Operation : PChar) : Integer; stdcall; external "Project2.dll" name "MathFunc" index 1;

А обработчик единственной кнопки приведите к примерно такому виду:

procedure TForm1.DoItButtonClick(Sender: TObject);

const

Errcode : Integer=978987;//код ошибки - может быть абсолютно любым.

var

Num1, Num2, Result\_ : Integer;//для проверки чисел

Operation : String;//операция, для передачи параметра функции

begin

try //прежде чем передать числа

Num1 := StrToInt(Num1Edit.Text); //функции проверим их

Num2 := StrToInt(Num2Edit.Text);

except

Num1Edit.Text := "0";

Num2Edit.Text := "0";

ResultEdit.Text := "Введите целые ЧИСЛA";

EXIT;

end;

Operation := OpEdit.Text; //также проверим, введена ли правильная команда.

if (Operation<>"plus")and(Operation<>"minus")and(Operation<>"multiply")

and(Operation<>"div")and(Operation<>"mod") then

begin

ResultEdit.Text := "Введите корректную команду";

Exit;

end;

Result\_ := MyFunc(Num1, Num2, Errcode, PChar(Operation)); //использование библиотечной функции

if Result\_=Errcode then //если функция возвратила код ошибки то

begin //то сообщаем об этом.

ResultEdit.Text := "ОШИБКА";

EXIT;

end

else //а если результат отличный от кода ошибки

ResultEdit.Text := IntToStr(Result\_);//то выводим его

end;

Обратите внимание, что мы используем функцию из библиотеки так же, как и если она была бы написана в модуле. Ещё раз повторяю, что при привязке библиотеки к программе функцию можно использовать только в тех модулях, в которых она была обьявлена. Вот вам мини калькулятор, который работает на (хотел было сказать на батарейках) DLL.

2. Динамическая загрузка

Недостатки:

- громоздкость и сложность кода

- функции библиотеки доступны только тогда, когда библиотека загружена в память

Преимущества:

- начисто лишен всех недостатков первого способа + некоторые другие преимущества перед первым способом

Этот способ довольно сложен, особенно для новичков. Но преимуществ перед первым способом у него куда больше. Для работы с динамически загружаемыми библиотеками просто необходимо знать три WinAPI функции: LoadLibrary, GetProcAddress И FreeLibrary.

LoadLibrary(LibFileName: PChar) - загружает библиотеку LibFileName в память. Если библиотека загружена удачно, то функция возвращает дескриптор (THandle) DLL в памяти.

GetProcAddress(Module: THandle; ProcName: PChar) - находит точку входа в функцию ProcName. Внимание! Здесь нужно указать NAME функции, а не её название. Если функция найдена, то функция GetProcAddress возвращает дескриптор (TFarProc) функции в загруженной DLL.

FreeLibrary(LibModule: THandle) - выгружает библиотеку LibModule. При этом вся занятая этой библиотекой память освобождается. Следует заметить, что после вызова этой процедуры функции данной библиотеки больше недоступны и обращение к ним вызовет исключение.

Для того, что бы динамически загрузить функцию из библиотеки, то необходимо её обьявить в разделе var:

MyFunc: function(num1, num2, Errcode : Integer; Operation : PChar) : Integer; stdcall;

Также нужно обьявить переменную типа THandle. "На пальцах" не обьяснишь, поэтому давайте рассмотрим пример динамической загрузки DLL на основе нашей демонстрационной библиотеки.

Откройте предыдущий проект с демонстрацией статическо загрузки. В разделе var обьявите пару новых переменных:

LibHandle: THandle;

MyFunc: function(num1, num2, Errcode : Integer; Operation : PChar) : Integer; stdcall;

Обработчик кнопки приведите к такому виду:

procedure TForm1.DoItButtonClick(Sender: TObject);

const

Errcode : Integer=978987;//код ошибки - может быть абсолютно любым.

var

Num1, Num2, Result\_ : Integer;//для проверки чисел

Operation : String;//операция, для передачи параметра функции

begin

try //прежде чем передать числа

Num1 := StrToInt(Num1Edit.Text); //функции проверим их

Num2 := StrToInt(Num2Edit.Text);

except

Num1Edit.Text := "0";

Num2Edit.Text := "0";

ResultEdit.Text := "Введите ЧИСЛA";

EXIT;

end;

Operation := OpEdit.Text; //также проверим, введена ли правильная команда.

if (Operation<>"plus")and(Operation<>"minus")and(Operation<>"multiply")

and(Operation<>"div")and(Operation<>"mod") then

begin

ResultEdit.Text := "Введите корректную команду";

Exit;

end;

//до этого момента код остался без изменений.

@MyFunc := nil; //очищаем адрес функции

LibHandle := LoadLibrary("Project2.dll");//пытаемся загрузить библиотеку

if LibHandle >= 32 then

begin //если все прошло успешно то

@MyFunc := GetProcAddress(LibHandle, "MathFunc");//пытаемся найти адрес функции

if @MyFunc <> nil then //если адрес найден (функция существует в библиотеке)

Result\_ := MyFunc(Num1, Num2, Errcode, PChar(Operation)); //использование библиотечной функции

if Result\_=Errcode then //если функция возвратила код ошибки то

begin //то сообщаем об этом.

ResultEdit.Text := "ОШИБКА";

EXIT;

end

else //а если результат отличный от кода ошибки

ResultEdit.Text := IntToStr(Result\_);//то выводим его}

end;

end;

Заключение

В этой статье мы коснулись лишь основных аспектов программирования с применением динамически-подключаемых библиотек. А ведь в DLL можно хранить всякие картинки и даже формы! С помощью них удобно создавать всякие плагины. Но это уже совсем другая история