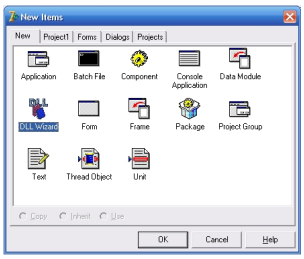
Использование **DLL** (динамически подключаемая библиотека) широко распространено в программировании Windows. **DLL** на самом деле часть кода исполняемого файла с расширением
DLL
. Любая программа может вызывать
DLL
.



Преимущество **DLL** заключается в следующем:

- Повторное использование кода.
- Совместное использование кода приложениями.
- Разделение кода.
- Улучшение потребления ресурсов в Windows.

Создание DLL

В меню **File** выберите пункт New -> Other. В диалоговом окне на вкладке **New** выберите
DLL Wizard
. Автоматически будет создан модуль – пустой шаблон будущей
DLL
.

Синтаксис DLL

library Example_dll; uses SysUtils, Classes; var { Объявляем переменные } k1: integer; k2: integer; { Объявляем функцию } function SummaTotal(factor: integer): integer; register; begin factor:= factor * k1 + factor; factor:= factor * k2 + factor; result:= factor; end; { Экспортируем функцию для дальнейшего использования программой } exports SummaTotal; { Инициализация переменных } begin k1:= 2; k2:= 5; end.

Для того, чтобы построить **DLL**, выберите **Project -> Build *Имя_проекта***.

Видимость функций и процедур

Функции и процедуры могут быть локальными и экспортируемыми из **DLL**.

Локальные

Локальные функции и процедуры могут быть использованы внутри **DLL**. Они видны только внутри библиотеки и ни одна программа не может их вызывать извне.

Экспортируемые

Экспортируемые функции и процедуры могут быть использованы за пределами **DLL**. Другие программы могут вызывать такие функции и процедуры.

Исходный код выше использует экспортируемую функцию. Имя функции следует за зарезервированным словом **Exports**.

Загрузка DLL

В Delphi есть два вида загрузки **DLL**:

- Статическая загрузка
- Динамическая загрузка

Статическая загрузка

При запуске приложения загружается автоматически. Она остается в памяти на протяжении выполнения программы. Очень просто использовать. Просто добавьте слово **external** после объявления функции или процедуры.

```
function SummaTotal(factor: integer): integer; register;      external 'Example.dll';
```

Если **DLL** не будет найден, программа будет продолжать работать.

Динамическая загрузка

DLL загружается в память по мере необходимости. Ее реализация более сложная,

потому что Вы сами должны загружать и выгружать ее из памяти. Память используется более экономно, поэтому приложение работает быстрее. Программист сам должен следить, чтобы все работало правильно. Для этого нужно:

- Объявить тип описываемой функции или процедуры.
- Загрузить библиотеку в память.
- Получить адрес функции или процедуры в памяти.
- Вызвать функцию или процедуру.
- Выгрузить библиотеку из памяти.

Объявление типа, описывающего функцию

```
type    TSumaTotal = function(factor: integer): integer;
```

Загрузка библиотеки

```
dll_instance:= LoadLibrary('Example_dll.dll');
```

Получаем указатель на функцию

```
@SummaTotal:= GetProcAddress(dll_instance, 'SummaTotal');
```

Вызов функции

```
Label1.Caption:= IntToStr(SummaTotal(5));
```

Выгрузка библиотеки из памяти

```
FreeLibrary(dll_instance);
```

Динамический вызов **DLL** требует больше работы, но легче управлять ресурсами в памяти. Если Вы должны использовать **DLL** в пределах программы, тогда предпочтительнее статическая загрузка. Не забывайте использовать блок **try…except** и **try…finally**, чтобы избежать ошибок во время выполнения программы.

Экспорт строк

Созданная **DLL** с использованием Delphi, может быть использована и в программах, написанных на других языках программирования. По этой причине мы не можем использовать любой тип данных. Типы, которые существуют в Delphi могут отсутствовать в других языках. Желательно использовать собственных типы данных из Linux или Windows. Наша **DLL** может быть использована другим программистом, который использует другой язык программирования.

Можно использовать *строки* и *динамические массивы* в **DLL**, написанной в Delphi, но для этого нужно подключить модуль **ShareMem** в раздел **uses** в **DLL**

и программе, которая будет ее использовать. Кроме того, это объявление должно быть первым в разделе
uses
каждого файла проекта.

Типов **string**, как мы знаем, C, C++ и других языках не существует, поэтому желательно использовать вместо них **PChar**.

Пример DLL

```
library Example_dll; uses SysUtils, Classes; var { Объявляем переменные } k1: integer; k2: integer; { Объявляем функцию } function SummaTotal(factor: integer): integer; register; begin factor:= factor * k1 + factor; factor:= factor * k2 + factor; result:= factor; end; { Экспортируем функцию для дальнейшего использования программой } exports SummaTotal; { Инициализация переменных } begin k1:= 2; k2:= 5; end.
```

Пример вызова функции из DLL

```
unit Unit1; interface uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls; type TForm1 = class(TForm) Button1: TButton; procedure Button1Click(Sender: TObject); private { Private declarations } public { Public declarations } end; type TSummaTotal = function(factor: integer): integer; var Form1: TForm1; implementation {$R *.dfm} procedure TForm1.Button1Click(Sender: TObject); var dll_instance: Thandle; SummaTotal: TSummaTotal; begin dll_instance:= LoadLibrary('Example_dll.dll'); @SummaTotal:= GetProcAddress(dll_instance, 'SummaTotal'); Label1.Caption:= IntToStr(SummaTotal(5)); FreeLibrary(dll_instance); end; end.
```