РОЗДІЛ 4 АВТОМАТИКА, КОМП'ЮТЕРНІ ТА ТЕЛЕКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ

УДК 621.564

М.Г. Хмельнюк*✉*, Д.И. Важинский
Одесская национальная академия пищевых технологий, ул. Канатная, 112, Одесса, 65039, Украина
✉e-mail: hmel_m@ukr.net

СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ БИБЛИОТЕК ДИНАМИЧЕСКОЙ КОМПОНОВКИ В ИНЖЕНЕРНЫХ РАСЧЕТАХ

В статье описывается создание и использование библиотек динамической компоновки для инженерных расчетов. Рассмотрены варианты создания универсальных библиотек, также описано использование технологии OpenMaple, применимой только для математического пакета Maple. Показаны преимущества и недоставки, возникающие при использовании собственных библиотек динамической компоновки в инженерных расчетах.
Ключевые слова: *Динамическая библиотека; Maple; OpenMaple.*

М.Г. Хмельнюк, Д.І. Важинський
Одеська національна академія харчових технологій, вул. Канатна, 112, Одеса, 65039, Україна

СТВОРЕННЯ І ВИКОРИСТАННЯ БІБЛІОТЕК ДИНАМІЧНОГО КОМПОНУВАННЯ В ІНЖЕНЕРНИХ РОЗРАХУНКАХ

В статті наведено опис створення і використання бібліотек динамічного компонування для інженерних розрахунків. Розглянуто варіанти створення універсальних бібліотек, також є опис технології OpenMaple, сумісної тільки з математичним пакетом Maple. Показано переваги і недоліки використання власних бібліотек динамічної компоновки в інженерних розрахунках.
Ключові слова: *Динамічна бібліотека; Maple; OpenMaple.*

I. ВВЕДЕНИЕ

Использование современных математических пакетов в инженерных расчетах имеет повсеместный характер. Возможностей таких программ как *Mathcad, Maple, Wolfram Mathematica* с математической точки зрения более чем достаточно. Однако, наличие инженерных инструментов в данном программном обеспечении ограничено. Поэтому при однотипных расчетах приходится проводить много рутинной однообразной работы. Например, при расчетах теплообменников необходимы теплофизические свойства рабочих веществ, при расчетах систем вентиляции и кондиционирования не обойтись без свойств влажного воздуха. Отсутствие вышеизложенного приводит к необходимости использования дополнительного программного обеспечения, такого как *CoolPack, RefProp* и др. Также необходимые данные могут содержаться в различных таблицах и диаграммах. При этом в любой сложной инженерной задаче всегда имеется большое количество повторяющих действий, например, вариантные расчеты, при проведении которых приходится с постоянно обращаться к дополнительному программному обеспечению, что не позволяет окончательно автоматизировать процесс, и делает его значительным как по времени, так и по количеству символов необходимых для ввода. Появляется необходимость провести автоматизацию данных процессов. Одним из инструментов автоматизации расчетов является использование библиотек динамической компоновки (англ.. *Dynamic Link Library*), сокращенно DLL. Данную технологию поддерживают большинством математических пакетов. В данных библиотеках могут, хранятся наиболее часто используемые процедуры и функции, а также теплофизические и термодинамические свойства рабочих веществ. Корректно разработанная и задокументированная динамическая библиотека позволит сэкономить значительное количество времени при дальнейших расчетах, и сделает более читаемыми расчеты. Однако следует понимать, что создание собственной динамической библиотеки требует абсолютно ясного представления о протекающих процессах.

II. ОБЩИЕ СВЕДЕНИЯ О ДИНАМИЧЕСКИХ БИБЛИОТЕКАХ

Динамические библиотеки получили очень широкое распространение в программировании. Основным их преимуществом, является повторное использование кода, а также параллельное использование одной библиотеки двумя различными программами одновременно. При этом разработанная и скомпилированная на языке *C++* библиотека будет доступна и

для других языков программирования, например, Fortran или Java, что позволяет говорить об их универсальности. Разработанный файл будет работоспособным не только в контексте использования в математических пакетах, но и при разработке собственного приложения. При доработке кода нет необходимости в перекомпиляции всего проекта, работа в таких случаях производится только с самой библиотекой. Однако следует понимать, что использование данной технологии требует, как минимум, среднего уровня программирования. Динамическая библиотека не может быть разработана в математическом пакете, для ее создания необходима специальная среда разработки, такая как *Visual Studio* или *Borland C++Builder*. Следует отметить, что после компиляции на выходе получается файл с расширением dll, который затем необходимо корректно подключить к математическому пакету. Все расчетные зависимости будут скрыты от глаз, что в некоторых случаях может оказаться полезным в контексте авторского права.

III. ОСОБЕННОСТИ РАЗРАБОТКИ ПРОСТЕЙШИХ ДИНАМИЧЕСКИХ БИБЛИОТЕК ДЛЯ МАТЕМАТИЧЕСКОГО ПАКЕТА MAPLE.

Использованию динамических библиотек в пакете Maple уделяется довольно значительное внимание. Официально поддерживаемыми языками программирования являются *C, Java, Fortran*. Для данных языков в Таблице 1 приводится соответствие типов.

Таблица 1 – Соответствие типов языков программирования

Maple	C	Fortran	Java
integer[1]	char	BYTE	byte
integer[2]	short	INTEGER*2	short
integer[4]	int, long	INTEGER	int
integer[8]	__int64	INTEGER*8	long
float[4]	float	REAL	float
float[8]	double	DOUBLE PRECISION	double
char[1]	char	CHARACTER	char
boolean[1]	char	LOGICAL*1	boolean
boolean[4]	int	LOGICAL	int

Для структурированных типов данных также имеется таблица, которая представлена в справке программы [1]. Можно с уверенностью сказать, что и другие языки вполне могут быть использованы, в данном случае будет необходимо более детально проработать использование типов данных, так для других языков не представлены таблицы соответствия типов данных. Библиотека также может быть разработана непосредственно на языке программирования, который используется в самом Maple. Однако, возникает логичный вопрос: стоит ли вникать в еще один язык программирования, если можно воспользоваться языком *Pascal* или *C*. Динамические библиотеки, содержащиее математические функции, написанные на других языках программирования, могут быть подключены непосредственно к Maple. Основным неудобством, возникающим при подключении, является согласование типов данных. Например, если переменная в библиотеке описывается как целая, а в Maple её же описать как вещественное число, возникнет исключительная ситуация. Простейшая программа, написанная на языке *C*, будет иметь следующий вид:

```
int __stdcall __declspec(dllexport) F45(int b)
{
return b+b;
}
```

Как видно, данная программа производит удваивание аргумента. При создании подпрограмм допустимо использование только соглашения **__stdcall**. Использование таких соглашений как **__cdecl** и **__fastcall** приведет к ошибке. Ключевое слово **__declspec(dllexport)** обязательно к использованию, и гарантирует экспорт данных и функций. Удачно скомпилированная, данная функция затем может быть вызвана из Maple:

```
Simple:=define_external('F45',x::integer[4],
RETURN::integer[4],LIB='Project1.dll')
```

В первой позиции указано название функции – то самое, которое указывается в коде в качестве названия функции. Далее следует тип данных, которые будет обрабатывать функция, в данном случае целый. Следующий аргумент – это возвращаемый функцией результат. Последний аргумент представляет собой адрес самой библиотеки. В данном случае, библиотека имеет тот же адрес что и исполняемый файл Maple. Использование функции из математического пакета будет иметь следующий вид:

```
Simple(30)
```

Основное неудобство при создании DLL традиционным способом является трудность работы с массивами, и ограниченные возможности при работе со сложными типами данных. При необходимости создания функции, которая будет возвращать одно значение, такой подход вполне обоснован. Однако, в случае более сложных технических задач, где необходима серьезная работа с массивами различных данных, а также требуется работа со строками, в Maple предусмотрен богатый набор инструментов, позволяющий разработать достаточно сложную библиотеку. Вполне понятно, что в данном случае произойдет потеря универсальности, однако открывающиеся возможности вполне перекрывают недостатки. Для создания динамических библиотек непосредственно под математический пакет Maple разработана технология **OpenMaple** [1, 2].

IV. РАЗРАБОТКА ДИНАМИЧЕСКИХ БИБЛИОТЕК С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ OPENMAPLE

OpenMaple это набор функций, позволяющий получить доступ к типам данных, которые используются непосредственно в Maple и различным функциям для работы с данными. Учитывая разнообразие типов данных в Maple, это открывает большие возможности. Для использования вышесказанного, используется библиотека **maplec.dll.** Все необходимое для ее подключения поставляется вместе с Maple, и может быть найдено в папке с программой. Первые трудности, возникающие при подключении данной библиотеки, появляются при выборе среды разработки. При работе в **Visual Studio** проблем не возникает, однако, если вы используете среду разработки от **Borland**, библиотека корректно не подключится. Данная проблема вызвана принципиально различными подходами к компиляции библиотек в данных средах. Для решения данной проблемы необходимо воспользоваться консольной утилитой **coff2omf.exe**. Она поставляется вместе с **Borland C++Builder**. Данная утилита преобразовывает .lib-файл, созданный для работы в Visual Studio, в .lib-файл, совместимый с C++Builder. Для этого необходимо зайти в меню **Пуск** -> **Выполнить**. Запускается консоль. Для удобства дальнейшей работы фиксируется папка проекта, для этого:

cd полный путь к папке

Для непосредственной конвертации .lib-файла выполняется следующая команда: ***coff2omf maplec.lib maplec-bcb.lib***

Полученный .lib-файл будет совместим с **Borland C++ Builder**. Однако для успешного использования библиотеки **maple.dll** этого недостаточно. После ее подключения, не все функции будут доступны. При анализе таблицы экспорта данной библиотеки, выясняется, что не все функции были скомпилированы одинаковым образом. Используются различные соглашения об наименовании. Для решения данной проблемы необходимо отключить реакцию на нижнее подчеркивание перед функцией в таблице экспорта. Для этого необходимо перейти в **Project** -> **Options**-> **C++compiler**-> **Output**. Значение **Generate underscores on symbol names** должно быть отключено.

V. ПРОСТЕШАЯ БИБЛИОТЕКА С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ OPENMAPLE

Сами по себе возможности задокументированные в справе. Корпорация MapleSoft распространяет файл с примерами в папке с установленной программой. Его можно найти по следующему адресу ***Maple 18\samples\ExternalCall*** под названием **HelpExamples.c.** Данный файл позволяет получить доступ ко всем примерам по данной теме, расположенным в обширной справочной системе **Maple**. Данные примеры дадут возможность довольно быстро создавать свои собственные библиотеки, так как содержат основные блоки, необходимые для построения собственных программ. Стоит обратить внимание, что использование технологии **OpenMaple** позволяет использовать массивы с различными типами данных, что значительно улучшает читаемость полученных расчетов и уменьшает их общий размер. Важным нюансом является возможность использовать типа данных **anything** применительно к массивам [3].

```
ALGEB__declspec(dllexport) M_DECL Simple1(
MKernelVector kv, ALGEB *args )
    {
    M_INT argc;
    FLOAT64 a, b, r;
    argc = MapleNumArgs(kv,(ALGEB)args);
    if( argc != 2 ) {
    MapleRaiseError(kv,"Two arguments
expected");
    return( NULL );
    }
    MaplePrintf(kv,"Summation a + b");
    a = MapleToFloat64(kv,args[1]);
    b = MapleToFloat64(kv,args[2]);
    r=a+b;
    return( ToMapleFloat(kv,r) );
    }
```

Анализируя заголовок, стоит отметить, что тип **ALGEB**, по сути, представляет собой любой тип данных поддерживаемый OpenMaple. Это, может быть как массив, так и строка. Просто при возвращении результат в Maple необходимо привести соответствующую конвертацию необходимой функцией. К объявлению обязательно использовать ключевое **M_DECL**. В качестве аргументов функции выступает дескриптор ядра Maple, а также список данных которые вводит пользователь, при этом дескриптор окна сообщается автоматически, пользователю просто необходимо ввести через запятую ввести ряд параметров. Вышеизложенная программа производит сложение двух чисел. Отметим, что у рассматриваемой функции есть своя специфическая структура. После объявления переменных следует блок проверки количества аргументов, если количество введённых данных отлично от установленного будет выведен текст об ошибке, в данном случае:"**Two arguments expected**". Далее имеется возможность выводить текстовое сообщение, можно выводить любую информацию, в любой последовательности, в том числе и после вычислений. Далее идет присваивание переменным определенных аргументов:

```
a = MapleToFloat64(kv, args[1]);
```

Выражение имеет следующий смысл: переменной a присваивается значение первого аргумента из списка введенных. Подразумевается, что данное число будет рациональным, о чем свидетельствует процедура **MapleToFloat64.** В случае если будет введен какое-либо другой тип данных, программа может отреагировать самым различным образом, поэтому рекомендуется по возможности внедрять блоки контроля типов данных. После блока расчетов, проводится вывод результатов.

```
return ( ToMapleFloat(kv,r))
```

ToMapleFloat предполагает вывод на экран значения переменной r, при условии, что она объявлена как **FLOAT64.** В случае другого формата понадобится другая функция.

Использование данной функции из Maple будет выглядеть следующим образом:

```
With(ExternalCalling)
dll:=ExternalLibraryName(‘’Project1’’):
Simple:=DefineExternal(‘Simple1’,dll);
Simple(4,5)
Summation a + b
```

VI. ВЫВОДЫ

Использование в современных инженерных и научных расчетах математических пакетов способствует экономии времени и улучшает читаемость расчетов. Применение динамических библиотек собственной разработки способствует еще большей структурированности расчетов, намного ускоряет вариантные расчеты, однако, разработка такой библиотеки требует значительных знаний программирования и алгоритмизации, а также большого количества времени. Разработка собственной динамической библиотеки однозначно может быть оправдана при многократном использовании функций заложенных в ней. Если же расчет проводится всего один раз, и не предвидится использование использованной методики в дальнейшем, разрабатывать библиотеку не имеет смысла. Основным недостатком в языке С является то, что уравнение не выступает как объект, это создает непреодолимые трудности, если необходимо запрограммировать действительно сложное уравнение, так как придётся перебрать все возможные комбинации [4]. Для более сложных инженерных задач стоит обратится к модельно ориентированным языкам, таким как Modelica, в котором уравнение выступает как объект. Стоит учесть, что при желании сохранить детали расчетов от чужих глаз, использование данной технологии решает данную проблему, так как пользователь будет видеть только результат функции, но никак не подробности расчетов. Очень важным является правильное документирование библиотеки, без соответствующего файла справки, в котором будет достаточно подробно изложена методика и описаны подробности работы с функциями использование библиотеки другими людьми значительно затруднено, если вообще возможно.

ЛИТЕРАТУРА

1. Maplesoft. [Электронный ресурс] http://www.maplesoft.com (Дата обращения 02.02.2016).
2. Mapleprimes. [Электронный ресурс] http://www.mapleprimes.com (Дата обращения 02.02.2016).
3. **Аладьев В.З.** Программирование в пакетах Maple и Mathematica: Сравнительный аспект / В.З. Аладьев, В.К. Бойко, Е.А. Ровба // – Гродно: ГрГУ. – 2011. – 517 с. ISBN 978-985-515-481-6.
4. **Керниган Б.** Язык программирования Си. / Б. Керниган, Д. Ритчи // – Санкт-Петербург: Невский диалект. – 2000. – 352 с. ISBN 5-7940-0045-7.

Отримана в редакції 08.02.2016, прийнята до друку 03.03.2016

M.G. Khmelniuk ✉, ***D.I. Vazhinsky***
Odessa National Academy of Food Technologies, 112 Kanatnaya str., Odessa, 65039, Ukraine
✉e-mail: hmel_m@ukr.net

CREATING AND USING DYNAMIC LINK LIBRARIES IN ENGINEERING CALCULATIONS

The article describes creation and using of dynamic link libraries in technical calculation. Description of creation universal libraries is given, also it is spoken in detail about OpenMaple technology which can be used only with Maple package. In conclusion the authors write about advantages and disadvantages of calculation with own dynamic link library. Today there are many mathematical packages, such as Maple or Mathcad, which give many different tools for calculations. The main problem that these packages do not solve most engineering tasks. There are no good thermodynamic or thermophysical libraries which could give possibility do not use any other programs, such as Cool Pack or REFPROP. Actually, if there is necessity to design apparatus or to calculate thermodynamic cycle, it's necessary many time. The main problems are unwieldy and long calculations which cannot be automatized completely. One of the solutions that can help is creation of your own dynamic link library (dll). Dynamic libraries can be either universal or designed for a specific mathematical package. The OpenMaple technology allows to create own libraries for Maple. If you want to make library you should know C, Java or FORTRAN languages. This library could not be created directly in Maple. Traditionally, the basic development environments are Embarcadero C++ or Visual Studio. This technology still has both advantages and disadvantages.

Keywords: *Dynamic link library; Maple; OpenMaple.*

REFERENCES

1. Maplesoft. [Electronic source] Available at: http://www.maplesoft.com. (Date of access: 02.02.2016)
2. Mapleprimes. [Electronic source] Available at: http://www.mapleprimes.com (Date of access: 02.02.2016)
3. **Aladev, V. Z., Boiko V. K., Rovba E. A.** 2011. Programmirovanie v paketakh Maple i Mathematica: Sravnitelnyi aspect. *Grodno: GrGU*, 517 p. (in Russian). ISBN 978-985-515-481-6.
4. **Brajan Kernigan, Dennis Ritchi.** 2000. Yazyk programmirovania Si. [The C Programming Language] *Sankt-Peterburg: Nevskij dialekt*, 352 p. (in Russian). ISBN 5-7940-0045-7.

Received 08 February 2016
Approved 03 March 2016
Available in Internet 29 April 2016