ASSIGNMENT NUMBER: B - 01

TITLE : Dynamic Link Library

PROBLEM STATEMENT : To write a program to create Dynamic Link Library for Arithmetic Operation in VB.net

OBJECTIVES:

- To understand Dynamic Link Libraries Concepts
- To implement dynamic link library concepts
- To study about Visual Basic

OUTCOMES:

The students will be able to

- Understand the concept of Dynamic Link Library
- Understand the Programming language of Visual basic

THEORY:

Dynamic Link Library:

A dynamic link library (DLL) is a collection of small programs that can be loaded when needed by larger programs and used at the same time. The small program lets the larger program communicate with a specific device, such as a printer or scanner. It is often packaged as a DLL program, which is usually referred to as a DLL file. DLL files that support specific device operation are known as device drivers.

A DLL file is often given a ".dll" file name suffix. DLL files are dynamically linked with the program that uses them during program execution rather than being compiled into the main program.

The advantage of DLL files is space is saved in random access memory (RAM) because the files don't get loaded into RAM together with the main program. When a DLL file is needed, it is loaded and run. For example, as long as a user is editing a document in Microsoft Word, the printer DLL file does not need to be loaded into RAM. If the user decides to print the document, the Word application causes the printer DLL file to be loaded and run.

A program is separated into modules when using a DLL. With modularized components, a program can be sold by module, have faster load times and be updated without altering other parts of the program. DLLs help operating systems and programs run faster, use memory efficiently and take up less disk space.

DLLs are essentially the same as EXEs, the choice of which to produce as part of the linking

process is for clarity, since it is possible to export functions and data from either.
It is not possible to directly execute a DLL, since it requires an EXE for the operating system to load it through an entry point, hence the existence of utilities like RUNDLL.EXE or RUNDLL32.EXE which provide the entry point and minimal framework for DLLs that contain enough functionality to execute without much support.
DLLs provide a mechanism for shared code and data, allowing a developer of shared code/data to upgrade functionality without requiring applications to be re-linked or re-compiled. From the application development point of view Windows and OS/2 can be thought of as a collection of DLLs that are upgraded, allowing applications for one version of the OS to work in a later one, provided that the OS vendor has ensured that the interfaces and functionality are compatible.

DLLs execute in the memory space of the calling process and with the same access permissions which means there is little overhead in their use but also that there is no protection for the calling EXE if the DLL has any sort of bug.

Difference between the Application & DLL:

An application can have multiple instances of itself running in the system simultaneously, whereas a DLL can have only one instance.
An application can own things such as a stack, global memory, file handles, and a message queue, but a DLL cannot.

Executable file links to DLL:
An executable file links to (or loads) a DLL in one of two ways:

1. Implicit linking
2. Explicit linking

Implicit linking is sometimes referred to as static load or load-time dynamic linking. Explicit linking is sometimes referred to as dynamic load or run-time dynamic linking.
With implicit linking, the executable using the DLL links to an import library (.lib file) provided by the maker of the DLL. The operating system loads the DLL when the executable using it is loaded. The client executable calls the DLL's exported functions just as if the functions were contained within the executable.
With explicit linking, the executable using the DLL must make function calls to explicitly load and unload the DLL and to access the DLL's exported functions. The client executable must call the exported functions through a function pointer.

An executable can use the same DLL with either linking method. Furthermore, these mechanisms are not mutually exclusive, as one executable can implicitly link to a DLL and another can attach to it explicitly.

Calling DLL function from Visual Basic Application:
For Visual Basic applications (or applications in other languages such as Pascal or Fortran) to call functions in a C/C++ DLL, the functions must be exported using the correct calling convention without any name decoration done by the compiler.

__stdcall creates the correct calling convention for the function (the called function cleans up the stack and parameters are passed from right to left) but decorates the function name differently. So, when declspec(dllexport) is used on an exported function in a DLL, the decorated name is exported.

The stdcall name decoration prefixes the symbol name with an underscore (_) and appends the symbol with an at sign (@) character followed by the number of bytes in the argument list (the required stack space). As a result, the function when declared as:

int stdcall func (int a, double b)

is decorated as:

_func@12

The C calling convention (cdecl) decorates the name as _func.

To get the decorated name, use /MAP. Use of declspec(dllexport) does the following:

- If the function is exported with the C calling convention (_cdecl), it strips the leading underscore (_) when the name is exported.
- If the function being exported does not use the C calling convention (for example, stdcall), it exports the decorated name.

Because there is no way to override where the stack cleanup occurs, you must use stdcall. To undecorate names with stdcall, you must specify them by using aliases in the EXPORTS section of the .def file. This is shown as follows for the following function declaration:
int stdcall MyFunc (int a, double b); void stdcall InitCode (void);

```
In the .DEF file:
EXPORTS
MYFUNC=_MyFunc@12
INITCODE=_InitCode@0
```

For DLLs to be called by programs written in Visual Basic, the alias technique shown in this topic is needed in the .def file. If the alias is done in the Visual Basic program, use of aliasing in the .def file is not necessary. It can be done in the Visual Basic program by adding an alias clause to the Declare statement.

DLL's Advantages:

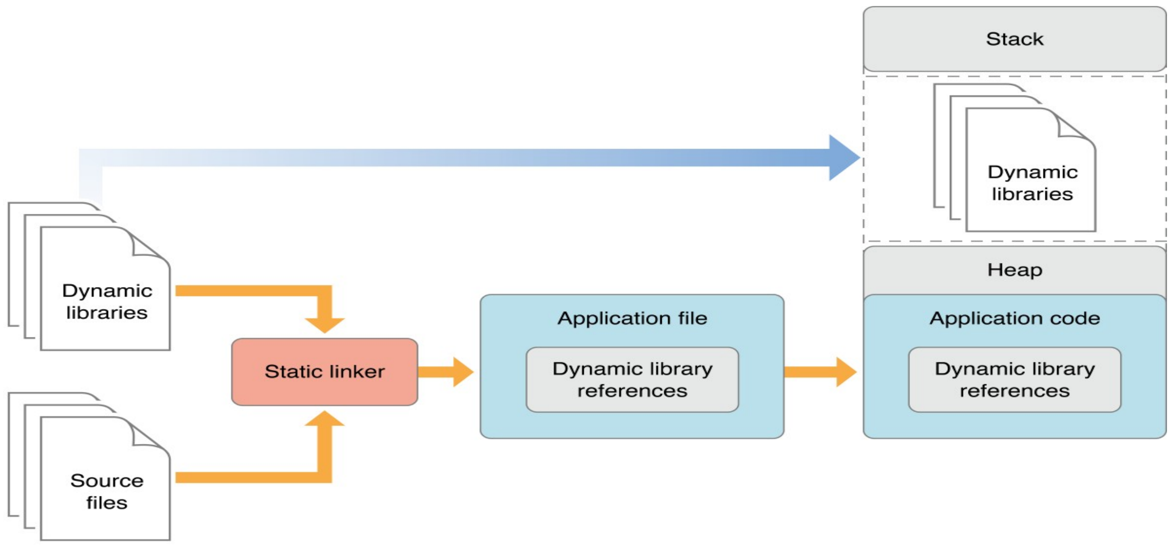
Saves memory and reduces swapping. Many processes can use a single DLL simultaneously, sharing a single copy of the DLL in memory. In contrast, Windows must load a copy of the library code into memory for each application that is built with a static link library.
Saves disk space. Many applications can share a single copy of the DLL on disk. In contrast, each application built with a static link library has the library code linked into its executable image as a separate copy.Upgrades to the DLL are easier. When the functions in a DLL change, the applications that use them do not need to be recompiled or relinked as long as the function arguments and return values do not change. In contrast, statically linked object code requires that the application be relinked when the functions changeProvides after-market support. For example, a display driver DLL can be modified to support a display that was not available when the application was shipped. Supports multi language programs. Programs written in different programming languages can call the same DLL function as long as the programs follow the function's calling convention. The programs and the DLL function must be compatible in the following ways: the order in which the function expects its arguments to be pushed onto the stack, whether the function or the application is responsible for cleaning up the stack, and whether any arguments are passed in registers.Provides a mechanism to extend the MFC library classes. You can derive classes from the existing MFC classes and place them in an MFC extension DLL for use by MFC applications.Eases the creation of international versions. By placing resources in a DLL, it is much easier to create international versions of an application. You can place the strings for each language version of your application in a separate resource DLL and have the different language versions load the appropriate resources.

Disadvantage:

A potential disadvantage to using DLLs is that the application is not self-contained; it depends on the existence of a separate DLL module.

Visual Basic:
Visual Basic is a third-generation event-driven programming language first released by Microsoft in 1991. It evolved from the earlier DOS version called BASIC. BASIC means Beginners' All-purpose Symbolic Instruction Code. Since then Microsoft has released many versions of Visual Basic, from Visual Basic 1.0 to the final version Visual Basic 6.0. Visual Basic is a user-friendly programming language designed for beginners, and it enables anyone to develop GUI window applications easily.
In 2002, Microsoft released Visual Basic.NET(VB.NET) to replace Visual Basic 6. Thereafter, Microsoft declared VB6 a legacy programming language in 2008. Fortunately, Microsoft still provides some form of support for VB6. VB.NET is a fully object-oriented programming language implemented in the .NET Framework. It was created to cater for the development of the web as well as mobile applications. However, many developers still favor Visual Basic 6.0 over its successor Visual Basic.NET.

Design Architecture:



Algorithm:

Note: you should write algorithm & procedure as per program/concepts

CONCLUSION : We have successfully learned and implemented basics of DLL.