LECTURE-1

WINDOW PROGRAMMING

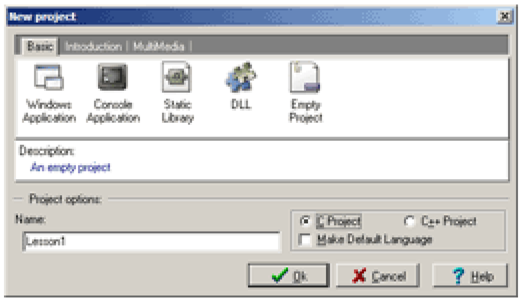
The Windows Environment

Windows is a graphics-based multitasking system. This tutorial is intended to present to you the basics (and common extras) of writing programs using the **Win32 API**. API (Application Programming Interface) is a set of commands, which interfaces the programs with the processors. The language used is C(procedure oriented approach), most C++ compilers will compile it as well. As a matter of fact, most of the information is applicable to any language that can access the API, including Java, Assembly and Visual Basic

1.1 First windows program

In this lesson, we will create our first windows application which will display a "Hello World" message box.

1. If you haven't started Dev-C++ already, do so now.
2. Select File->New Project. In the dialog box that appears, choose the project type as an Empty Project and name the project Lesson1. Also choose Win32 Application.

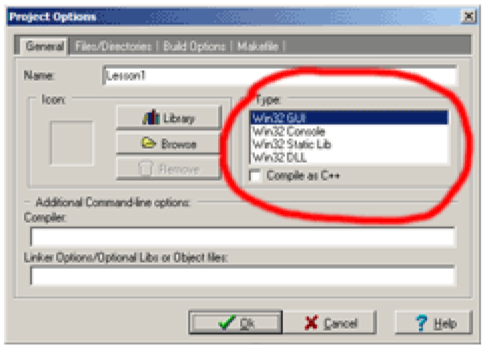


Description:
An empty project

Project options:
Name: Lesson1

[✓ C Project] C++ Project
Make Default Language

3. Press finish and then choose empty project and press OK. You will come back to IDE.


Project Options

General | Files/Directories | Build Options | Makefile

Name: Lesson1

Type: [✓ Win32 GUI] Win32 Console Win32 Static Lib Win32 DLL

Compile as C++

Additional Command-line options:

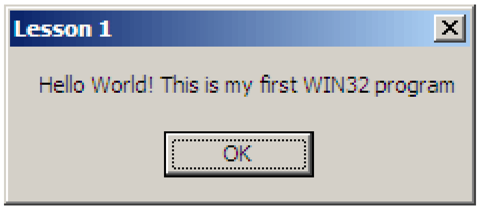
Compiler:

Linker Options/Optional Libs or Object files:

4. Choose Project->Add to project->new. A window opens and asks for filename. Enter Lesson1.c and press OK. A new blank window opens.
5. Now enter the code as given below:

```
6.   /* Windows Programming Tutorial Series
7.    * Lesson 1 - Your first windows program
8.    * 3CX
9.   **/
10.  #include <windows.h>
11.
12.  int WINAPI
13.  WinMain(HINSTANCE hInst,
14.          HINSTANCE hPrevInstance,
15.          LPSTR lpCmdLine,
16.           int nCmdShow)
17.  {
18.          MessageBox (NULL, "Hello World! This is my first WIN32
   program", "Lesson 1", MB_OK);
19.          return 0;
20.  }
```

21. Press F9 to compile and run the project. You should see the following,


Lesson 1

Hello World! This is my first WIN32 program

This is the output of your first windows application.

1.2 ANALYSIS

Lets break down the code to understand the program.

1. **#include <windows.h>**

   All Windows programs must include the header file windows.h. This file has the definitions of Windows system calls or the WinAPI (Window Application Programming Interface). The WinAPI has everything necessary for programming under windows.

2. **WinMain(..)**

   This is the entry point of a windows application. This is like the main() of a console based application. WinMain() is declared as,

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
  LPSTR lpCmdLine, int nCmdShow)
```

WinMain() is windows equivalent of main() from DOS or UNIX. This is where your program starts execution. Return type of WinMain() is int type. The parameters are as follows:

HINSTANCE hInstance

'hInstance' the first argument is of type HINSTANCE which is a handle to an instance (copy or object). Here running programs i.e. executable module (the .exe file in memory) is referred as an instance.A handle is a 32 bit integer value.

HINSTANCE hPrevInstance

Always NULL for Win32 programs.

LPSTR lpCmdLine

'LPSTR' is also a typedef and means long pointer to string. The 'lpCmdLine' is a third argument to WinMain() function and is of thetype LPSTR.

int nCmdShow is an argument. An integer value which may be passed to ShowWindow(). We'll get to this later.

hInstance is used for things like loading resources and any other task which is performed on a per-module basis. A module is either the EXE or a DLL loaded into your program.

hPrevInstance used to be the handle to the previously run instance of your program (if any) in Win16. This no longer applies. In Win32 you ignore this parameter.

3. **MessageBox(..)**

   This is a windows function which displays a messagebox. The MessageBox function is declared as,

```
 int MessageBox(
         HWND hWnd,        /* Handle of owner window */
         LPCTSTR lpText,   /* Address of text in message box
LPCTSTR stands for const* string*/
         LPCTSTR lpCaption,/* Address of title of message box */
         UINT uType);      /* Style of message box */
```

9. **return 0**

   This is the return value to the system.

Win32 Data Types

You will find that many of the normal keywords or types have windows specific definitions, UINT for unsigned int, LPSTR for char* etc... Which you choose is really up to you. If you are more comfortable using char* instead of LPSTR, feel free to do so. Just make sure that you know what a type is before you substitute something else.

Just remember a few things and they will be easy to interpret. An LP prefix stands for *Long Pointer*. In Win32 the *Long* part is obsolete so don't worry about it.

Next thing is a C following a LP indicates a const pointer. LPCSTR indicates a pointer to a const string, one that can not or will not be modified. LPSTR on the other hand is not const and may be changed.

WINDOWS DATA TYPE

- All data types used by window have been 'typedef' within the **windows.h** and its related files.
- Some command data types are:
- HANDLE is a 32 bit integer, which is simply a value that identifies some resources.
- BYTE is a 8-bit unsigned integer.
- WORD is a 16-bit unsigned integer.
- DWORD is a unsigned longinteger.
- UINT is an unsigned 32-bit integer.
- BOOL is an integer (does not resemble T/F0.
- LONG represents long.
- LPSTR is a pointer to string.

- LPCSTR is a constant pointer to string.