EE475 Lab #1 Fall 2004

Getting Started With Simple C Programs

The object of this lab is to use the Microsoft Visual C compiler to create, build, and run several simple C programs. We will be using other code development environments specifically designed for embedded programming, but it is useful to have access to a conventional C compiler for testing and prototyping.

Logging On:

1. Wake up the computer (or turn on the power if it is off).
2. Begin by pressing Ctrl+Alt+Delete (Windows XP).
3. Enter your username and password, and select the proper domain.

Launching :

Start the development application: > All Programs | Microsoft Visual Studio 6.0 | Microsoft Visual C++ 6.0

Setting up a Visual C software project:

1. From the Visual C File menu, create a "new project." Make it a Win32 Console Application, give it the name "`Lab1`", and save it in a directory you have read/write permission: `c:\eeclasses\ee475` is a good choice.
2. Again from the File menu, create a new text file. Use the Save As… menu item to save the file with the name `Lab1-1.c` into the `Lab1` subdirectory that was created for the project in step 1.
3. In the left window pane, select the "FileView" tab and open up the browse selections. Do a right-click on the Source Files folder and add the `Lab1-1.c` file to the project. You are now set to write some C code!

Running a simple program:

Locate the `Lab1-1.c` window in the workspace and type in the following lines. NOTE that the editor recognizes that this is a C program because of the `.c` file extension, and therefore it highlights various keywords, automatically indents, and so forth.

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
     printf("Hello, World!\n");
     exit(0);
}
```

Save the `Lab1-1.c` file, then select "compile" from the Build menu (or press the compile icon on the task bar).

Any compiler errors will appear in the window at the bottom of the display.

If the compile is successful, select the "build" option, which will link the program and create an executable. If the compile reported any errors or warnings, go back to the edit window and fix the problems.

Finally, select the "run" option (or press the exclamation point symbol) and see what happens: a console window should pop up and show the results of your program.

Ask for help if any of the preliminary steps did not work properly.

→ Next, add a few lines of code to your program so that it requests the user to type a line of input text and echoes it (prints it out). One possibility is something like:

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
     char buffer[50];

     printf("Hello, World!\n");

     printf("Enter a line of text: ");
     gets(buffer);
     printf(buffer);
     printf("\n\n");

     exit(0);
}
```

Save, build, and run. Did it work?

→ What would happen if the user entered more characters than would fit into `buffer`? Try it!

→ Also, deliberately make an error in the file, like leaving a semicolon off a line or misspelling a word, and see what the compiler error(s) look like.

Problems to do in the lab:

For the following lab problems you need to demonstrate a working program to get credit for it. Have the instructor sign the verification sheet for each working assignment that is completed. Each problem is worth 10 points.

Problem #1: Modify the `Lab1-1.c` program so that in addition to echoing the input string it will also return each letter of the input string on a separate line.

For example, your program should take the input:

```
Hello
```

And return:

```
H
e
l
l
o
```

→ How do you know how long the input string is? In other words, how does C indicate the end of the input string?

Problem #2: Modify the `Lab1-1.c` program so that the input string is also echoed in *reversed* order. That is, an entry of

```
Hello
```

results in an output of

```
olleH
```

Problem #3: Find the size in bytes of each of the following data types, using the `sizeof()` function. Consult a C reference to see what each type means.

```
char
unsigned char
signed char
short int
int
long int
float
double
```

Problem #4: The term "little endian" means that the least significant byte is stored in the first (lowest) address location followed by the more significant bytes in order. The term "big endian" means that the most significant byte is stored in the first (lowest) address location. Some microprocessors use little endian storage and others use big endian.

(a) To find out how the PC stores its data, write a new C program called `Lab1-2.c` that:

1. Assigns the integer 255 to a variable of type `int`.
2. Declares a pointer of type `unsigned char *`, then forces the pointer to point to the integer variable.
3. Prints out each of the bytes of the integer using the pointer reference.

For example:

```
int i, val;
unsigned char *ptr;

val=255;

/* Convert the address of val into a pointer of type
unsigned char */
ptr = (unsigned char *) &val;

printf("val is:  %d\n",val);
for(i=0;i<sizeof(int);i++)
  printf("byte %d:  %d\n",i,*(ptr+i));
```

→ How many bytes are used by the compiler to store an `int` variable on this computer?
→ Does the processor appear to be little endian or big endian?
→ Can you have the bytes be displayed as hex digits instead of decimal digits? What about displaying in binary?

(b) Now modify your program to store the negative number -42563 in the integer variable. Observe the output bytes and explain the result.

(c) Finally, modify your program again to store the negative number -42563 into a `float` variable and print out all of the bytes in order.

→ Locate a floating point format reference book (print or on the web), find out how the bits are deployed, and explain the resulting value in each byte.

BE SURE TO KEEP COPIES OF YOUR CODE AND INPUT/OUTPUT EXAMPLES. Even though you may work in pairs in the lab, each student needs to write his or her own individual report.

Lab Report: Due at THE START OF THE LAB PERIOD NEXT WEEK.

The lab report is to be written up in the Memo format. Each student should submit a separate lab report. For each problem, write a short description of what you did to solve the problem. Include **commented** C code *excerpts* for each problem and include them within the memo.

Note: You will need to include the instructor verification sheet to get any credit for the lab.

Instructor Verification Sheet
EE475 Lab #1
Fall 2004

Student Name:

	Instructor Signature	**Date**
Problem #1 runs correctly		
Problem #2 runs correctly		
Problem #3 results		
Problem #4 runs correctly		

Note: This verification sheet must be signed by the instructor and submitted with the lab report to get any credit for the lab.