United International University

Dept. of Electrical and Electronic Engineering (EEE)

Course No. : **EEE 122**
Course Title: **Structured Programming Language**

Lab Sheet 1
Introduction to C

Outcomes

After finishing this lab students should be able to

1. get familiarized about computer programming.
2. get familiarized about the C programming environment.
3. Compile and run sample programs
4. write simple programs with formatted input and output.

Contents

1 A theoretical overview **2**
1.1 Computer Programming . . . 2
1.2 Levels of programming language: . . . 2
1.3 The Translator . . . 2

2 C compilers and IDE **3**
2.1 C Compilers . . . 3
2.2 Integrated development environment (IDE) . . . 3

3 A short tutorial on Code::Blocks **4**

4 Creating a new project: **5**

5 Writing first program **6**

6 Building and running the project **6**

7 Saving and closing **7**

8 Programming Examples **7**

9 Practice session **8**

10 Lab Assignments **9**

1 A theoretical overview

1.1 Computer Programming

Computer programming (often shortened to **programming**) is a process that leads from an original formulation of a computing problem to executable computer programs.
Source code is written in one or more programming languages.

1.2 Levels of programming language:

Programming languages can be broadly classified into three categories:

1. **MACHINE LANGUAGES**: Computers are made to understand only two states pulse and no pulse (or 1 and 0) conditions. Computers do not understand English, Bangla or any other common language. They respond only to 1 and 0.A language composed of only 1 and 0 is called the machine language. Since computers are not identical in architecture, each computer has its own machine language.
   Thus the programmer himself should convert his code to language composed of 1 and 0(i.e. to machine language). For example if one wants to make understand ADD 0184 to an early IBM machine, it would have been written :

   000100000000000000000000000010111000

   Again, every computer has its own machine language, the programmers cannot communicate with other computers, if he does not know his machine language.

2. **ASSEMBLY LANGUAGES**:

   In assembly language instructions are given in English like words, such as MOV, ADD, SUB etc. Hence it is easy to write and understand assembly programs. As you know computer can understand only machine level language. Hence assembly language program must be translated into machine language. The translator which is used for translating is called assembler.

   In assembly language data are stored in the computer registers. Every computer has different set of registers. Hence the assembly language program is also not portable.

   Since the low level language is related with hardware, the execution time of low level program is faster.

3. **HIGH LEVEL LANGUAGES**:High level language is designed keeping in mind the features of portability, means these languages are machine independent. These are the English like language, so it is easy to write and understand the programs of high level language. For translating high level language program into machine language, compiler or interpreter is used. Every language has its own compiler or interpreter. The language in this category is FORTRAN, COBOL, BASIC, PASCAL etc.

   C is the middle level language which is in between these two categories. In C program can be written same as other high level language and it may be also possible to interact this with low level language. For this reason, some call C the high level language.

1.3 The Translator

As we know computer can understand only machine level language which is in binary 1 or 0. It is difficult to write and maintain the program of machine level language. So the need arises for converting the code of high level and low level language into machine level language. So the translator are used to achieve this process. These are:

1. **Compiler**: Compilers are used to convert high level languages (like C, C++) into machine code (i.e. gcc , Microsoft Visual Studio)
2. **Assembers**: Assembler are used to convert assembly language code into machine code.
3. **Interpreter**:An interpreter is a computer program which executes a statement directly (at run-time). (i.e.python , LISP, Ocamle)

2 C compilers and IDE

2.1 C Compilers

C programming language uses compiler to process C source code. There are many compilers used for C language. Popular C compilers Include:

Name	Platform	License	Details
Microsoft Visual Studio Express	Windows	Free Version	Powerful and student-friendly version of an industry standard compiler.
Tiny C Compiler (TCC)	GNU/Linux, Windows	LGPL	Small, fast and simple compiler.
Clang	GNU/Linux, Windows, Unix, OS X	University of Illinois/NCSA License	A front-end which compiles (Objective) C/C++ using a LLVM backend.
GNU C Compiler	GNU/Linux, MinGW (Windows), Unix, OS X.	GPL	The De facto standard. Ships with most Unix systems.

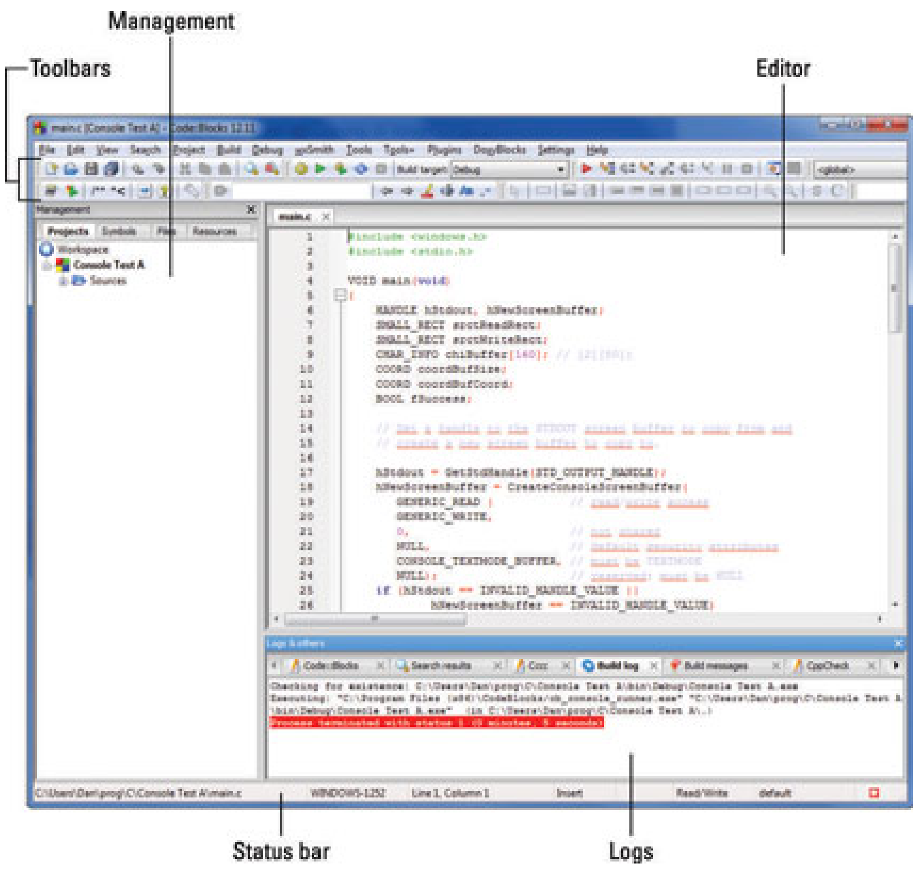
2.2 Integrated development environment (IDE)

Though not absolutely needed, many programmers prefer and recommend using an Integrated development environment (IDE) instead of a text editor. An IDE is a suite of programs that developers need, combined into one convenient package, usually with a graphical user interface. These programs include a text editor, linker, project management and sometimes bundled with a compiler. Popular IDEs Include:

Name	Platform	License	Details
Eclipse CDT	Windows, Mac OS X, Linux	Open source	EclipseIDE for C/C++ developement, a popular open source IDE.
Netbeans	Cross-platform	CDDLandGPL 2.0	A Good comparable matured IDE to Eclipse.
Anjuta	Linux	GPL	A GTK+2 IDE for theGNOMEdesktop environment.
Geany	Cross-platform	GPL	A lightweight cross-platform GTK+ notepad based on Scintilla, with basic IDE features.
Little C Compiler (LCC)	Windows	Free for non-commercial use	Small open source compiler.
Xcode	Mac OS X	Free	Available for free atMac App Store.
Pelles C	Windows, Pocket PC	Free	A complete C development kit for Windows.
Dev C++	Windows	GPL	Updated version of the formerly popular Bloodshed Dev-C++.
Microsoft Visual Studio Express	Windows	Free	A powerful, user friendly version of an industry standard compiler.
CodeLite	Cross-platform	GPL2	Free IDE for C/C++ development.
Code::Blocks	Cross-platform	GPL3.0	Built to meet users' most demanding needs. Very extensible and fully configurable.

3 A short tutorial on Code::Blocks

1. Obtain Code::Blocks from the Internet at this website: www.codeblocks.org and install it. We will use version 13.12 of this software
2. It starts just like any other program does: Locate its icon on the Start button menu, or you may also find the Code::Blocks shortcut icon on the desktop, which is the easiest way to start the IDE in Windows 7/8/10.


Management
Toolbars
Editor
main.c [Console Test A] - Code::Blocks 12.11
File Edit View Search Project Build Debug wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Build target: Debug
<global>
Management
Projects Symbols Files Resources
Workspace
Console Test A
Sources
main.c
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck
C:\Users\Dan\prog\C\Console Test A\main.c
WINDOWS-1252
Line 1, Column 1
Insert
Read/Write
default
Status bar
Logs

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

VOID main(void)
{
    HANDLE hStdout, hNewScreenBuffer;
    SMALL_RECT srctReadRect;
    SMALL_RECT srctWriteRect;
    CHAR_INFO chiBuffer[160]; // [2][80];
    COORD coordBufSize;
    COORD coordBufCoord;
    BOOL fSuccess;

    // Get a handle to the STDOUT screen buffer to copy from and
    // create a new screen buffer to copy to.

    hStdout = GetStdHandle(STD_OUTPUT_HANDLE);
    hNewScreenBuffer = CreateConsoleScreenBuffer(
       GENERIC_READ |           // read/write access
       GENERIC_WRITE,
       0,                       // not shared
       NULL,                    // default security attributes
       CONSOLE_TEXTMODE_BUFFER, // must be TEXTMODE
       NULL);                   // reserved; must be NULL
    if (hStdout == INVALID_HANDLE_VALUE ||
            hNewScreenBuffer == INVALID_HANDLE_VALUE)
```

```
Checking for existence: C:\Users\Dan\prog\C\Console Test A\bin\Debug\Console Test A.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\Dan\prog\C\Console Test A
\bin\Debug\Console Test A.exe"  (in C:\Users\Dan\prog\C\Console Test A\.)
Process terminated with status 1 (0 minutes, 5 seconds)
```

The main areas in the workspace are:

- **Toolbars**: These messy strips, adorned with various command buttons, cling to the top of the Code::Blocks window. There are eight toolbars, which you can rearrange, show, or hide. Dont mess with them until you get comfy with the interface.
- **Management**: The window on the left side of the workspace features four tabs, though you may not see all four at one time. The window provides a handy oversight of your programming endeavors.
- **Editor**: The big window in the center-right area of the screen is where you type code.
- **Status bar**: At the bottom of the screen, you see information about the project and editor and about other activities that take place in Code::Blocks.
- **Logs**: The bottom of the screen features a window with many, many tabs. Each tab displays information about your programming projects. The tab you use most often is named Build Log.

4 Creating a new project:

The examples presented in this lab are all console applications, which means that they run in Text mode in a terminal window.Thats the best way to teach basic programming concepts without overwhelming you with a large, complex, graphical beast of a program. So even though an IDE is capable of more, you use it in this book to create simple, console-based programs.

1. Start Code::Blocks. You see the Start Here screen, which displays the Code::Blocks logo and a few links. If you dont see the Start Here screen, choose File ->Close Workspace.
2. Click the "**Create a New Project**" link.
3. Choose **Console Application** and then click the **Go** button. The Console Application Wizard appears. You can place a check mark by the item Skip This Page Next Time to skip over the wizards first screen.

4. Click the **Next** button.
5. Choose **C** as the language you want to use, and then click the **Next** button.
   C is quite different from C++, you can do things in one language that arent allowed in the other.
6. Type **eee12201** as the project title.
   All the code in this lab follows this same project title convention. When you set the project title, the projects filename is automatically filled in.
7. Click the **Browse** button to the right of the text box titled Folder to Create Project In.
8. Use the **Make New Folder** button in the Browse for Folder dialog box to create a project folder.
9. Click the **OK** button to select the folder and close the dialog box.
10. Click the **Next** button.
11. Remove the check mark by Create Debug Configuration.
12. Click the **Finish** button.

5 Writing first program

Into the **Management** box navigate to **workspace** –>**eee12201** –>**Sources** –>**main.c**. Now double click on `main.c`. Now our main source code will be open into our Editor Box. Make sure that your code is look like as follows:

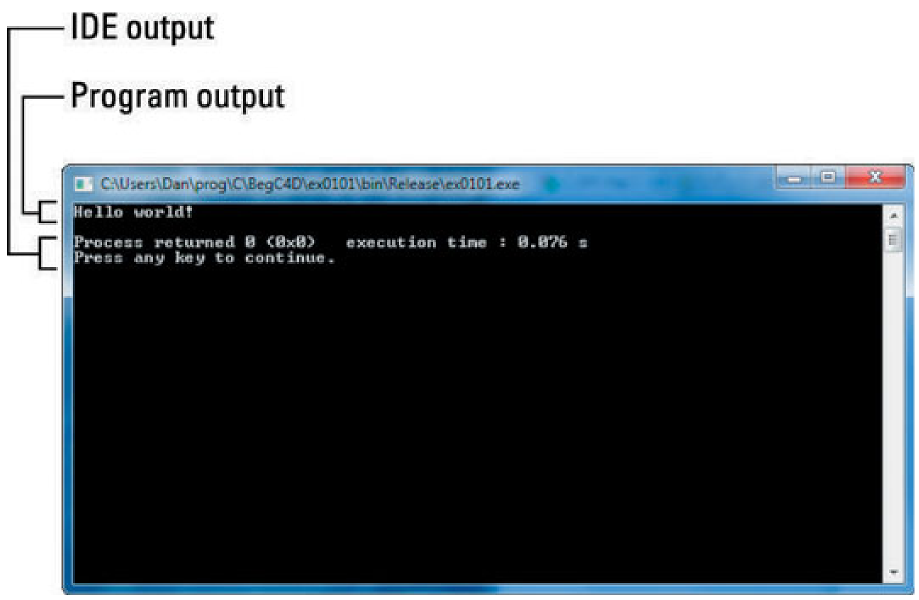
```
#include <stdio.h>

int main(void){
    printf("Hello world!\n");
    return 0;
}
```

6 Building and running the project

To create a program in the Code::Blocks C integrated development environment, you must build the project. This single step does several things. If youve already started your first project, **eee12201**, and its open and displayed in Code::Blocks, youre ready to build. Heed these steps:

1. **Ensure that the project you want to build is activated in the Management window.**
   Activated projects appear in bold text. If you have more than one project shown in the Projects window, activate the one you want to work with by right-clicking the project name (by the Code::Blocks icon) and choosing the Activate Project command.
2. **Choose Build –>Build from the menu.**
   The Build Log tab in the Logs part of the window displays the results of building the project. You see a few lines of text.
3. **Choose Build–>run from the menu.** You see the terminal window appear, listing the programs output, plus some superfluous text as shown in following figure -



```
Hello world!

Process returned 0 (0x0)   execution time : 0.076 s
Press any key to continue.
```

Congratulation !!! You have completed your first program, and you became a programmer. **"Welcome to Programming world!!!"**

7 Saving and closing

When youre done with a project, or even after youve changed a minor thing,you should save. In an IDE, however, you have several things to save, such as the source code file, the workspace, and the project. Commands are found on the File menu to save each of those items individually.

A handy way to save everything all at once is to use the Save Everything command. This command is found on the File menu, and its handy keyboard shortcut is **Alt+Shift+S**. If you havent yet saved the project, do so now.

You can also close the current project by choosing **File–>Close Project**.

8 Programming Examples

In this first lab, be familiar with the environment of C editor, create and run some elementary C programs as the instructor suggests.

Example: 1	
Description: Write a C program that asks the user to enter two integers and find their sum.	
Source Code	**Output**

`#include <stdio.h>`
`int main(void){`
`    int a, b, sum;`
`    printf("Enter two integers: ");`
`    scanf("%d %d",&a,&b);`
`    sum = a + b;`
`    printf("%d + %d = %d", a, b, sum);`
`    return 0;`
`}`
	Enter two integers: 10 20
	10 + 20 = 30

Example: 2

Write a C program that converts a temperature in Centigrade to its Fahrenheit equivalent. The relation between Centigrade and Fahrenheit scale is: $\frac{C}{5} = \frac{F-32}{9}$

Source Code

```c
#include<stdio.h>
int main(void){
    float c,f;
    printf("\nEnter a temperature in centigrade:");
    scanf("%f",&c);
    f=1.8*c+32;
    printf("\nThe equivalent temperature in Fahrenheit is %.2f.\n",f);
    return 0;
}
```

Output

```
Enter a temperature in centigrade:
36.9
The equivalent temperature in
Fahrenheit is 98.42.
```

9 Practice session

Students will type the following codes in individual file and analyze the output. They will help students understand beginner level syntax of C programs.

Sl — **Source Code**

Practice 1

```c
#include<stdio.h>
int main(void){
    float c,f;
    printf("\nEnter a temperature in centigrade:");
    scanf("%f",&c);
    f=1.8*c+32;
    printf("\nThe equivalent temperature in Fahrenheit is %.2f.\n",f);
    return 0;
}
```

Practice 2

```c
#include<stdio.h>
int main (void){
    printf("Welcome\tto\tUIU\n");
    printf("Department\tof\tEEE\n") ;
    return 0;
}
```

Practice 3

```c
#include<stdio.h>
int main (void){
    char ch;
    printf("Enter a character:");
    scanf("%c",&ch);
    printf("%d\n",ch) ;
    return 0;
}
```

Practice 4

```c
    #include<stdio.h>
    int main (void){
    float b = 123.1265;
    printf("%f\n",b) ;
    printf("%.2f\n",b) ;
    printf("%.3f\n",b) ;
    return 0;
    }
```

Practice 5

```
#include<stdio.h>
int main (void){
    float a1,b1,a2,b2,a3,b3; a1=2;
    b1=6.8;
    a2=4.2;
    b2=3.57;
    a3=9.82;
    b3=85.673;
    printf("%3.1f,%4.2f\n",a1,b1) ;
    printf("%5.1f,%6.2f\n",a2,b2) ;
    printf("%7.1f,%8.2f\n",a3,b3);
    return 0;
}
```

10 Lab Assignments

1. Write a C program which prints a line of text ”Welcome to EEE”.
   (a) In one line
   (b) In three lines each containing one word
   (c) Create errors omitting semicolons, brackets or inverted comma and see what error messages are shown. Correct them and run again
   (d) Replace **`main`** with **`Main,`** **`printf()`** with **`print()`**.What happens?
2. The base and height of a triangle are given. Find it’s area.
3. Accept the radius of a circle and calculate the area and perimeter of the circle.
4. Write a C program which will swap two integers. Swapping means exchange values. For example, If a=10 and b=20; after swapping, a=20 and b=10.
5. Repeat number 4 (Swapping) without using a third variable.

Acknowledgment

First OBE version, prepared by:
B.K.M. Mizanur Rahman,
Assistant Professor,
Department of EEE, UIU

Second Update , prepared by:
Nazmul Alam,
Part Time Faculty,
Department of EEE, UIU