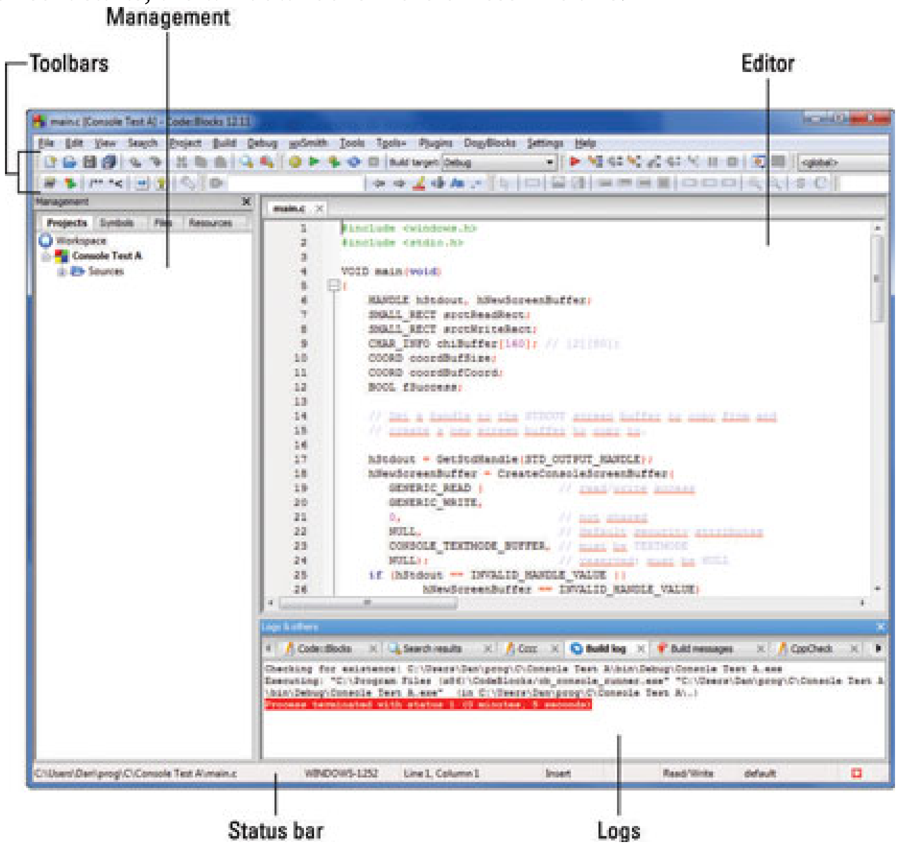
CSE115L – Programming Language I Lab
Lab 01
Introduction to Programming

In this lab, we will learn how to use the Code::Blocks IDE (Integrated Development Environment) to write a simple program, compile it and then finally execute it.

Write and execute your first C program: Follow the steps to write your first program.

Start Code::Blocks. Once it starts, the window looks more or less like this.



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

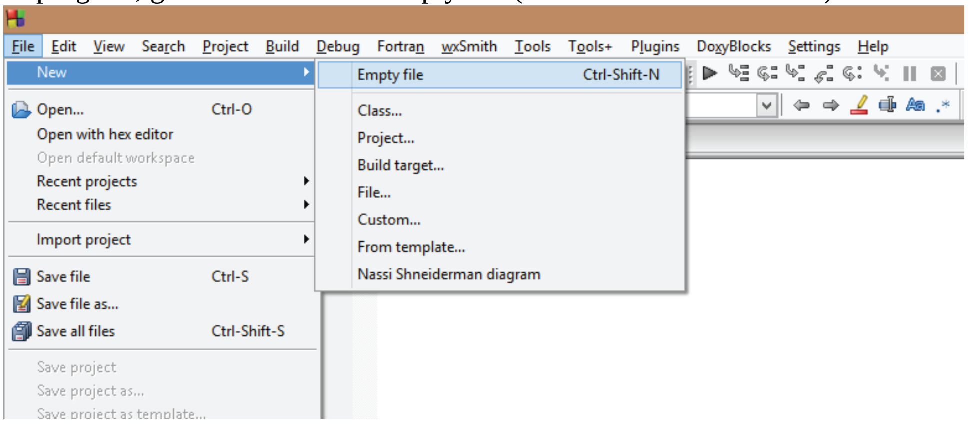
```
#include <windows.h>
#include <stdio.h>

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)
```

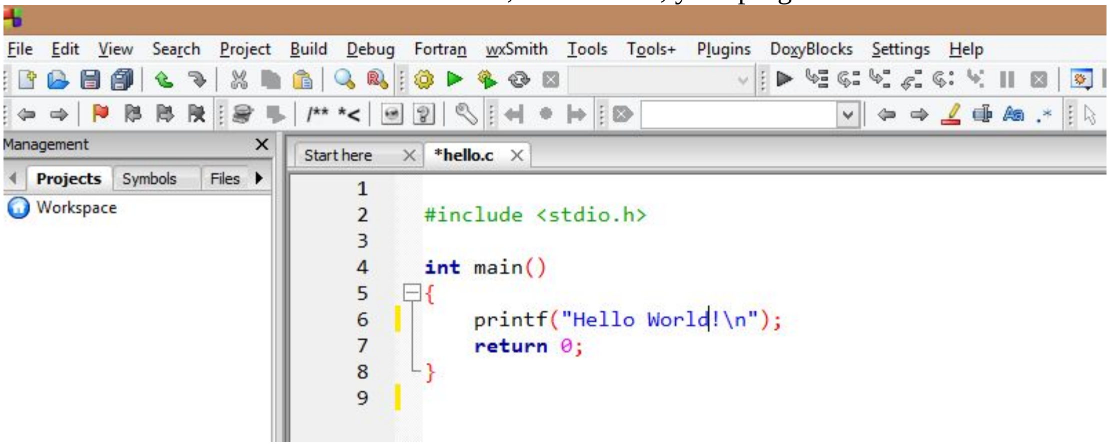
The main areas in the workspace are:

1. 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. Don't mess with them until you get comfortable with the interface.
2. Editor: The big window in the center-right area of the screen is where you type code.
3. 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. 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.
5. 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.

To write your first C program, go to File > New > Empty file (Shortcut: Ctrl + Shift + N).



Write the C code and save the file with .c extension. To save the file, go to File > Save (Shortcut: Ctrl + S).
Important: The filename should end with a .c extension, like: hello.c, your-program-name.c etc.


File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
Start here
*hello.c

```c

#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}

```

To run the program, go to Build > Build and Run (Shortcut: F9). This will build the executable file and run it.

How the “Hello, World!” program works.

1. Include `stdio.h` header file in your program: C programming language is quite small and cannot do much by itself. You need to use libraries that are necessary to run the program. The `stdio.h` is a header file and C compiler knows the location of that file. To use the file, you need to include it in your program using #include preprocessor. In this program, we have used the `printf()` function which displays the text inside the quotation mark. Since `printf()` is defined in the library file `stdio.h`, you need to include `stdio.h`.

2. The `main()` function: In C programming, the code execution begins from the start of `main()` function (doesn’t matter if `main()` isn’t located at the beginning). The code inside the curly braces `{ }` is the body of `main()` function. The `main()` function is mandatory in every C program.

   ```c
   int main() {
   }
   ```

   This program doesn’t do anything but, it’s a valid C program.

3. The `printf()` function: The `printf()` is a library function that sends formatted output to the screen (displays the string inside the quotation mark). Notice the semicolon at the end of the statement. Every statement must end with a semicolon. In our program, it displays *Hello, World!* on the screen. Remember, you need to include `stdio.h` file in your program for this to work.

4. The `return` statement: The statement `return 0` inside the `main()` function ends the program. This statement isn’t mandatory. However, it’s considered good programming practice to use it.

Experiment: Replace the `printf`() with `Printf()` and rebuild the program. You will find that the program fails to build and the build log shows an error message describing where the error is. Try to understand what the error message means.

Things to remember:

1. All C program starts from the `main()` function and it’s mandatory.
2. C is case-sensitive; the use of uppercase letter and lowercase letter have different meanings.
3. The C program ends when the `return` statement inside the `main()` function is executed.
4. The statements in a C program end with semicolons.