Kasetsart University, Sriracha Campus

03603111: Programming Fundamentals I

Introduction to C

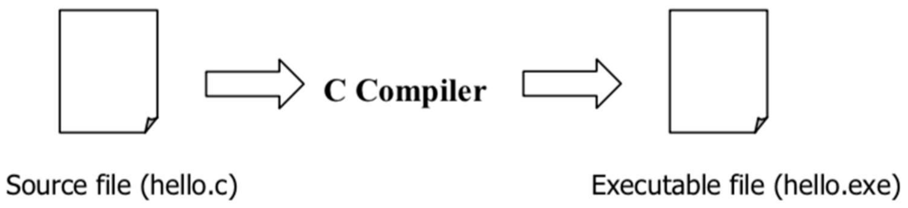
Lab
1

Objective:

- Introduce how to use Code::Blocks, an open source integrated development environment
- Write and execute simple C programs
- Understand and use *print*
- Understand how to format output

---

1. Introduction



Basically, to develop a C program, we use an editor or a word-processor program to write C program code and save it as a source file (with an extension .c). We then use a C compiler to process the source file and translate the program code into machine code generated as an executable file (.exe). The generated executable file (or program) can be executed or run to perform the task it was programmed for.

2. A Simple C Program

Here is your first simple C program to start with – a HelloWorld program.

Example 1 A HelloWorld program

```
#include <stdio.h> /*(1)*/
int main()         /*(2)*/
{
   printf("Hello, World\n");  //(3) prints "Hello, World" to screen
   return 0;
}
```

(1) Preprocessor directive

We usually see some preprocessor directives at the beginning of C source files, for example, **#include** or **#define**. A preprocessor directive is just a command telling what to be done before a compilation begins. In this case, **#include<stdio.h>** means the standard input and output library should be included in the program so that we can use printf() function to print a text onto the screen.

(2) main() function
main() function is where every C program starts its execution. The body of the function is enclosed between curly braces **{ }** and contains statements or commands that the program must do. In this example, the main() function contains the statement that prints the "Hello, World" message to the screen. The printf() function takes a text string and prints it out to the screen. Notice that a string in C is enclosed within the double quotes **""** and **\n** is an escape sequence which represents a new-line character.

(3) Comments
In C, a program comment is enclosed within a pair of /* and */. A comment is simply a text that you write to explain the design or how you program works, and is ignored by compiler. Comments are important and useful in programming as it is the key mechanism in documenting your program code. Always write comments in your programs.

3. **Good Programming Practices**
The following code snippets show the differences between a good programming style and a bad one. Ask yourself a question, which one do you prefer to read?

```
#include <stdio.h>
int main(){ printf("Hello, World\n");
return 0;}
```

```
/* Name: Mr. C Programmer */
/*   ID: 0123456789 */
/* Desc: A Hello World program */

#include <stdio.h>
int main()
{
   printf("Hello, World\n"); /* prints "Hello, World" to screen */
   return 0;
}
```

As a general practice, you should

- Use spaces and indentation to make your program code easy to read.
- Always write comments to let others (and yourself at later time) understand your program code and its design.

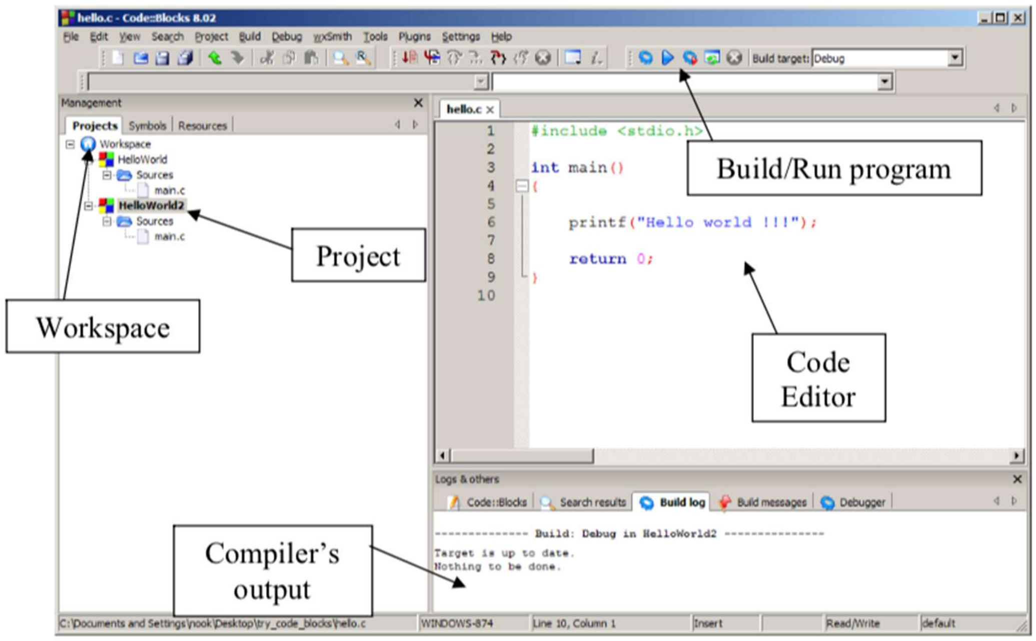
4. **Code::Blocks**

In our C programming laboratories, we will use an IDE (Integrated Development Environment) tool named Code::Blocks to develop, debug, and run C programs. Basically, an IDE tool integrates different development tools required by programmers, i.e., editor, compiler, and debugger into one single program to make it easy and convenient for programmers to develop programs. Code::Blocks is an open source program and is freely available at http://www.codeblocks.org/ . Note that when you download Code::Blocks for Windows, make sure that you select the binary release with

"mingw- setup" in the file name of the package you download. Otherwise, you may **not** be able to compile your source code.

Figure 1 shows different areas in Code::Blocks program. The workspace on the left shows projects and its associated files. The code editor (on the right) allows you to compose/edit your program code. The output pane (at the bottom) shows output messages from a compilation such as errors (if any).

Basically, Code::Blocks manages programs as projects within a workspace. A workspace is a collection of projects whereas a project can have many different files (e.g., source files and header files). Code::Blocks keeps a project file (a .cbp file) for each project, which holds the list of files associated with the project. In our programming labs, we will **not** create projects in Code::Blocks. Since most of the programs we write will be small, we will simply create a new empty .c file, write some code, compile it, and run it on the fly.


hello.c - Code::Blocks 8.02
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target: Debug
Management
Projects Symbols Resources
Workspace
HelloWorld
Sources
main.c
HelloWorld2
Sources
main.c
hello.c
Logs & others
Code::Blocks Search results Build log Build messages Debugger
Build/Run program
Project
Workspace
Code Editor
Compiler's output
C:\Documents and Settings\nook\Desktop\try_code_blocks\hello.c
WINDOWS-874
Line 10, Column 1
Insert
Read/Write
default

```
#include <stdio.h>

int main()
{

    printf("Hello world !!!");

    return 0;
}
```

```
-------------- Build: Debug in HelloWorld2 ---------------
Target is up to date.
Nothing to be done.
```

Figure 1: The main window of Code::Blocks IDE.

5. Creating and Running Hello World Program

Now let's try to create and run the HelloWorld program in Example 1 with Code::Blocks.

Firstly, a new empty source file can be created by clicking the menu **File News Empty** file or simply use the shortcut key **Ctrl + Shift + n**. A new empty source editing panel will be created as shown in Figure 2.

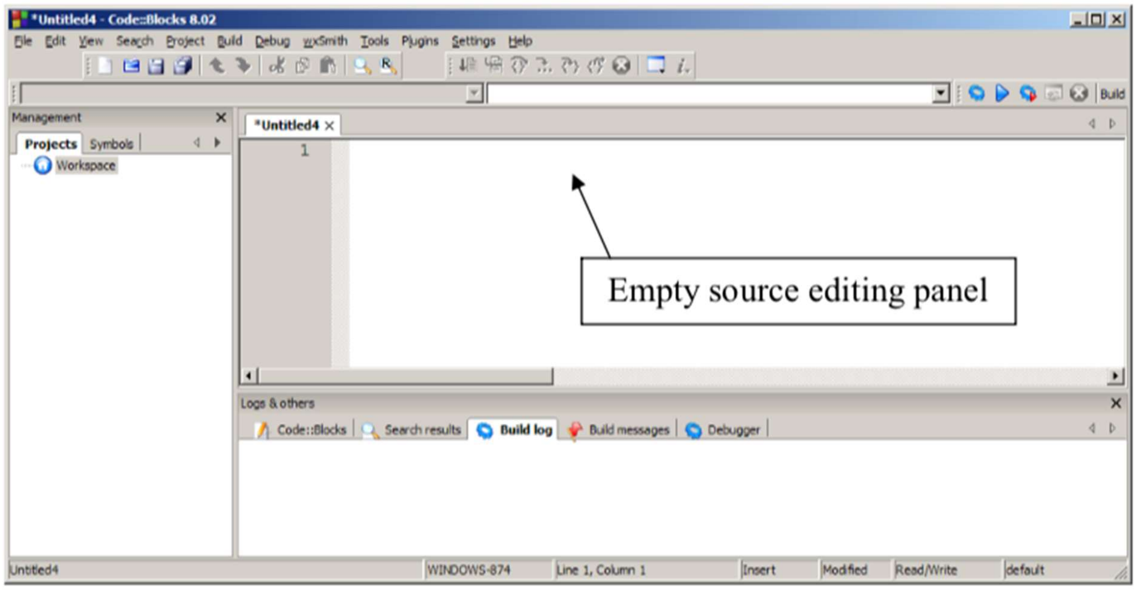

*Untitled4 - Code::Blocks 8.02
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Management
Projects Symbols
Workspace
*Untitled4
Empty source editing panel
Logs & others
Code::Blocks Search results Build log Build messages Debugger
Untitled4 WINDOWS-874 Line 1, Column 1 Insert Modified Read/Write default

Figure 2: Code::Blocks window after creating a new empty file.

The next step is to enter the code in Example 1 using the editor. Once you are done with the code, save it by clicking the menu **File Save** or use **Ctrl + s**. You can save your code anywhere you want. In this case, we will save on the Desktop with the name hello.c.

Lastly, compile and run the program. To compile the program, press Build button on the toolbar (Figure 3). To run the program press the Run button on the toolbar (Code::Blocks will automatically create an executable file if there is none). Alternatively, you can click the third button Compile and run to do both steps. If the compilation step goes well, you should see an object file named hello.o and an executable file named hello.exe which are automatically created on the same folder that you save your C code i.e., Desktop.

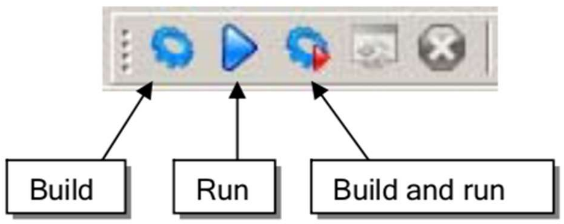
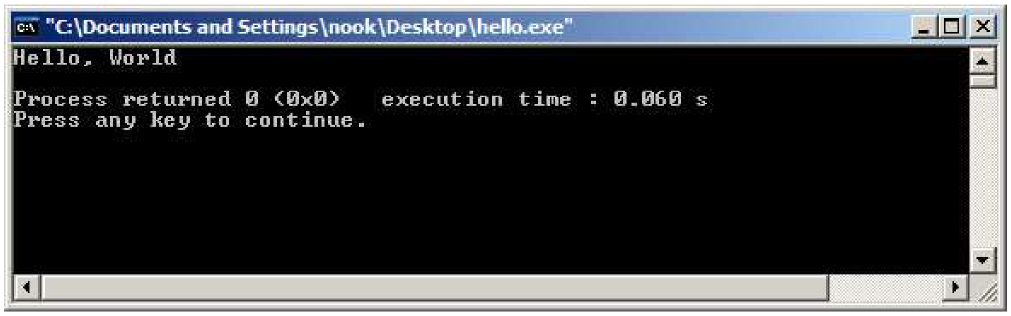


Figure 3: Build and run toolbar in Code::Blocks

After clicking Run, a new window will pop up to show the result of running your program. In this example, if you have done everything correctly, you should see the phrase “Hello, World” on the screen like in Figure 4. In addition to the output from your program, the program also reports the time it takes to run your program (execution time).



```
Hello, World

Process returned 0 (0x0)   execution time : 0.060 s
Press any key to continue.
```

Figure 4: The output of the HelloWorld program.

6. Do it yourself

Exercise 1: Write a program to show the following text to the screen:

```
Good morning teacher.
How are you today?
I'm Fine, I ok, better than our yesterday.
```

Exercise 2: The following program has an error. Find and correct the error.

```
#include <stdio.h>
int main()
{
      printf("What is a bug?\n");
      printf("It is a very small insect.\n")
      printf("No, It is a mistake in program.\n");
}
```

Exercise 3: The following program has some errors. Find and correct the error.

```
#include <stdio.h>
int main()
{
      printf("I am programming big\n");
      printf("Do you see me? "\n");
      print("Find me, if you can.\n");
}
```

Exercise 4: Identify the errors and correct the program.

```
#include <stdio.h>
int main()
{
      printf("I’m a barbie girl, \n");
      printf("in the barbie world\n")
      printf('Life in plastic, \n');
      printf("it's fantastic!\n");
}
```

Exercise 5: Write program to show the following picture to the screen.

```
    *
   **
   ***
  ****
  *****
   **
   **
```

Exercise 6: Write program to show the following picture to the screen.

```
(\_/)
(o.o)
(___)0
```