Princes Nourah Bint Abdelrahman university
Electrical Engineering Department
ECE 201
Programming Applications for Engineers
1st Term 2022

Lab 1: Syntax and Variable

Name:	ID:	Section:	Grade:

1. Objective

- To Getting acquainted with major components of a C program.
- Typing and Executing a C Program.

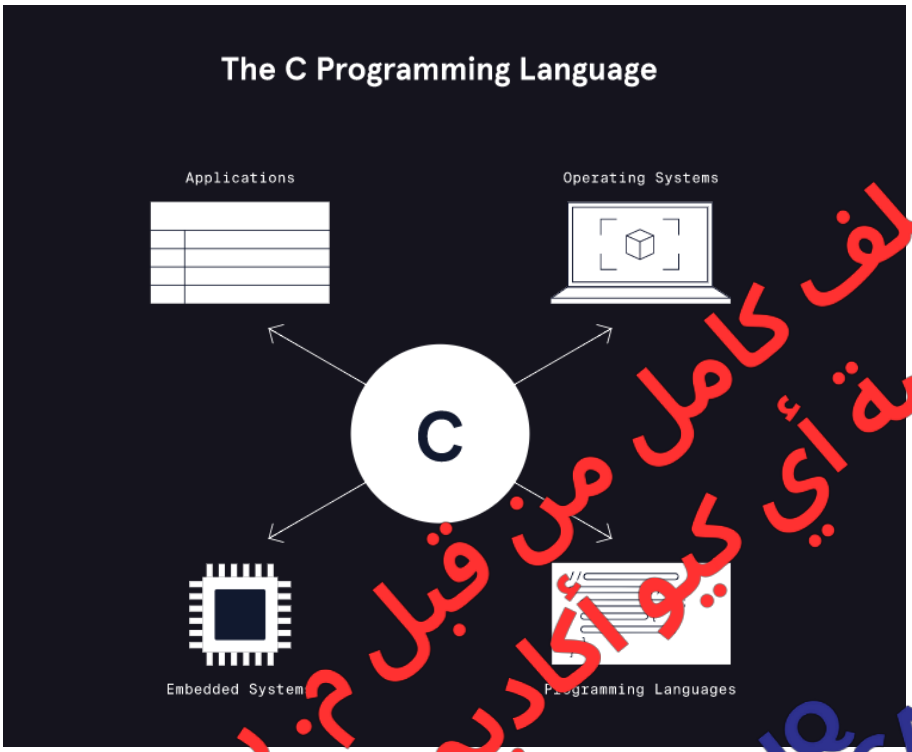
2. PREFFERED TOOL(S)

- Code Blocks

3. Introduction

Why Learn C?

C has been around for quite some time, and it is one of the foundational languages of computer science. Most operating systems today, including the Linux Kernel, are implemented with C code. The main version of the Python programming language is named CPython because it is implemented using C. The C programming language is everywhere, learning it will help you become a better programmer ready for the next challenge in any field of computer science!



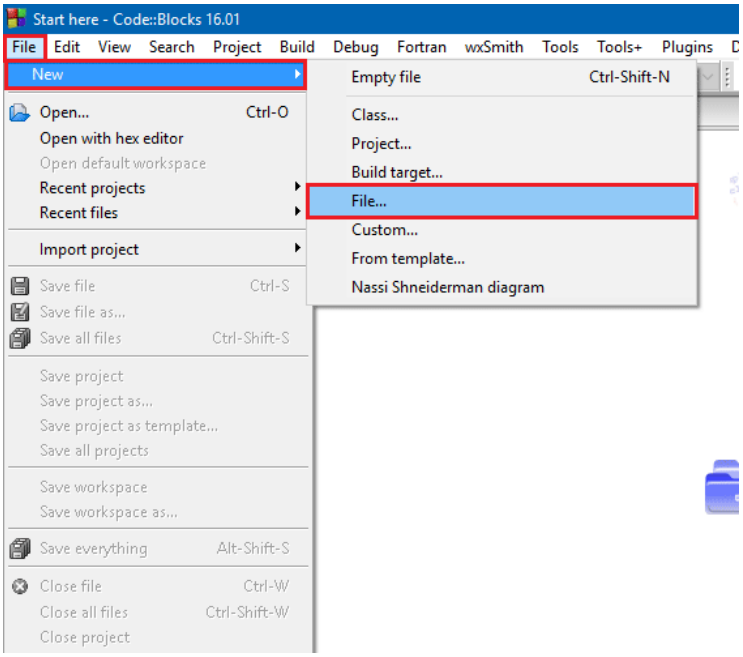
Princes Nourah Bint Abdelrahman university
Electrical Engineering Department
ECE 201
Programming Applications for Engineers
1st Term 2022

Lab 1: Syntax and Variable

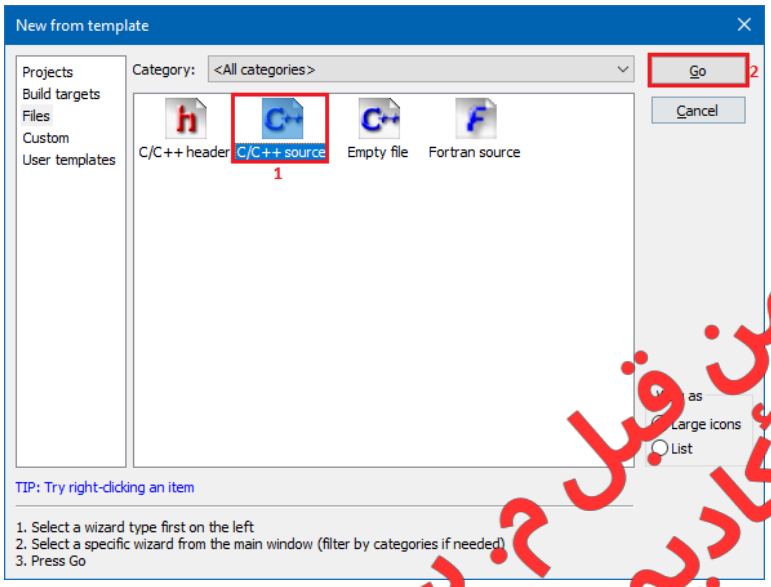
Part 1

Exercise 0: To create a program using CodeBlocks IDE:

1. Open CodeBlocks IDE and create a new file. Click on **`File`** → **`New`** → **`File`**



2. From the New form template window select **C/C++** source and click **Go** button.

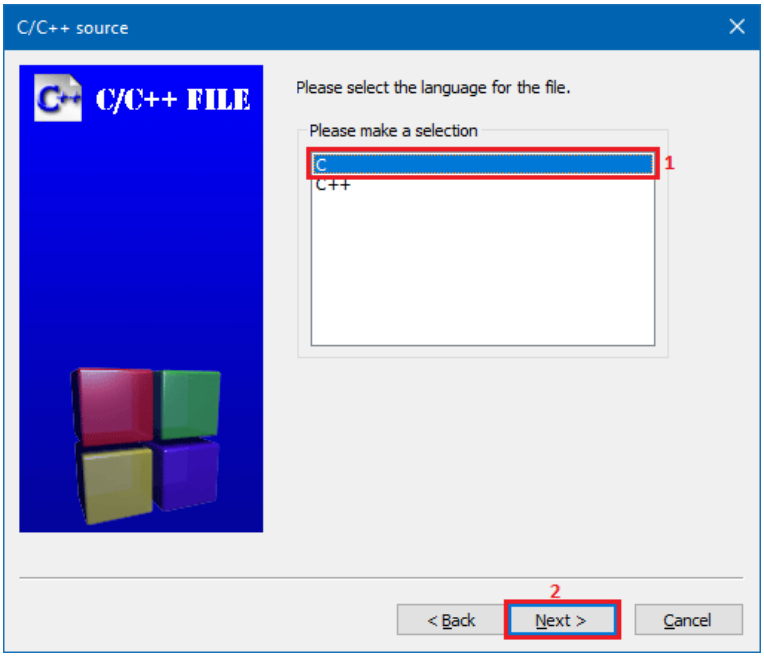


If you see a welcome message, click next to skip the welcome message. Make sure you have checked **Skip this next time** checkbox if you do not want to see this welcome message again.

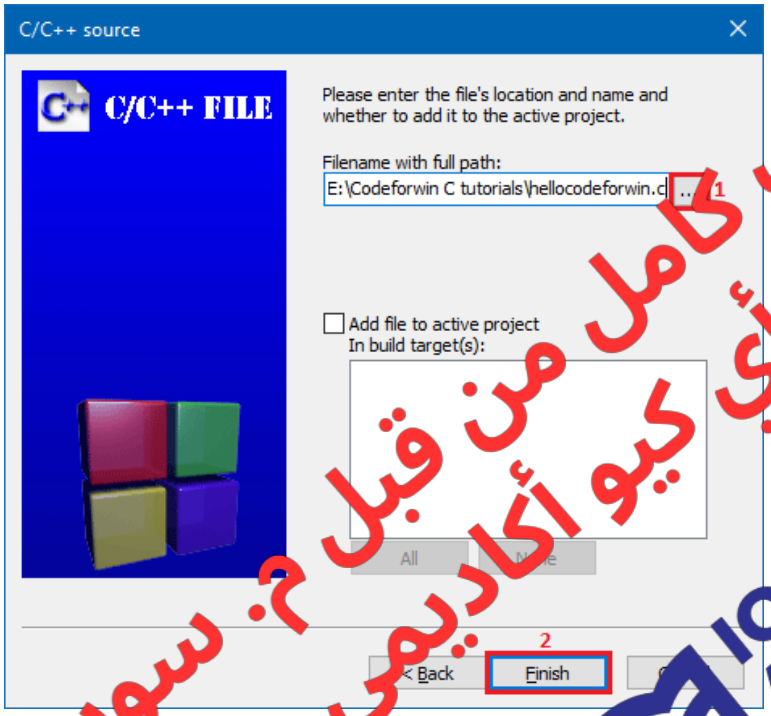
Princes Nourah Bint Abdelrahman university
Electrical Engineering Department
ECE 201
Programming Applications for Engineers
1st Term 2022

Lab 1: Syntax and Variable

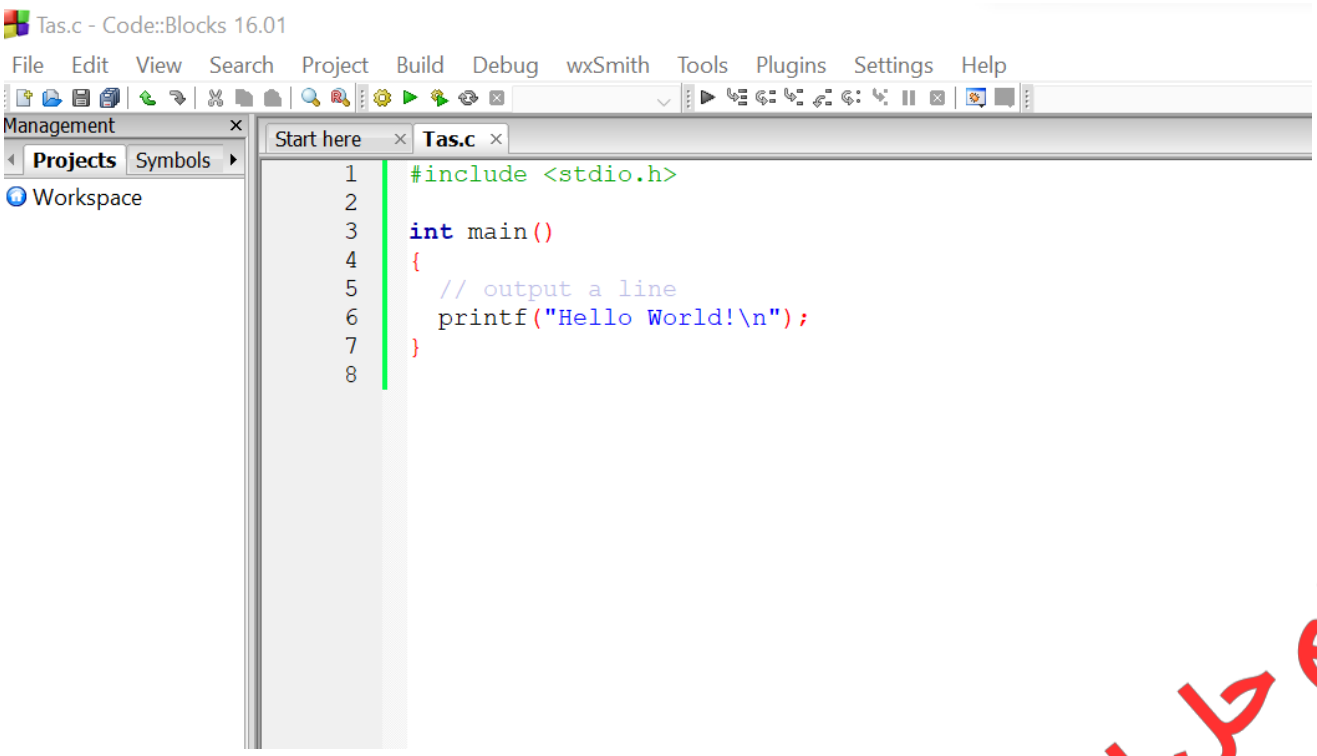
3. Next, select your language C and click **Next** button.



4. Give name to your file and specify the location.



5. To compile and run a C program, click **`Build`** → **`Build`** `and run` to compile and build your C program, alternatively use the shortcut key F9



So let's look at our first C program! Let's go through the code line by line to see what is happening. You don't need to understand everything right away, this is just a first look.

- `#include <stdio.h>`: This line is needed to run the line of code that starts with `printf`.
- `int main(){ }`: This is the starting point of the code. All the code inside the curly braces {} runs first.
- `// output a line`: This is a comment. It is not a line of code but a message we can add to code to tell ourselves or others what the code does. When the code is run this line will be ignored.
- `printf("Hello World!");`: This line of code prints, or outputs, the text "Hello World!" to the console. Printing text to the console is one way for a program to communicate with the user.

Try running the code and see the output yourself!

Lab 1: Syntax and Variable

Exercise 1: "Hello, world!"

Let's look at the Hello World code to examine common syntax error, correct the code and run the program.

#include <stdio.h>
 	
int Main()
{
  // output a line
  printf("Hello World!\n")
}
	Write the correct code here:
	
	#include <stdio.h>
	int main()
	{
	// output a line
	printf("Hello World!\n");
	}

Sample Output:
Hello World!

The **main goal** of our Hello World code example is to **output the text** **"Hello World!"** to the console.

Let's dive deeper into the 2 parts of this line are:

- **printf()** is known as a function and performs the action of printing text to the console.
- **"Hello World!\n"** is a string. A string is text in between a pair of double quotes.

Placing the string in between the parentheses of the **printf()** function prints the text (without the quotes) to the console.

What about the **\n** at the end of the string? Good question! This is called an escape sequence and is used to add a non-visual character within a string. In this case, **\n** adds a new line to the end of the string. Look what happens when you place it in between Hello and World! It's important to remember an escape sequence is a character and must be within the double-quotes.

Another escape sequence is **\t**. This is equivalent to the tab key and will insert spaces within a string. **\n** and **\t** are just two of many different escape sequences that can be put inside a string.

Princes Nourah Bint Abdelrahman university
Electrical Engineering Department
ECE 201
Programming Applications for Engineers
1st Term 2022

Lab 1: Syntax and Variable

Exercise 2: Escape sequence and printf function

```
#include <stdio.h>

int main()
{
  // Simple Recipe

  printf("2 Cups: All Purpose Flour");
  printf("1 Cups: Unsalted Butter(Room
Temperature)");

}
```

1st Question

Sample Output:
2 Cups: All Purpose Flour
1 Cups: Unsalted Butter(Room Temperature)

2nd Question

Sample Output:
2 Cups: All Purpose Flour
1 Cups: Unsalted Butter(Room Temperature)

3rd Question

Sample Output:
2 Cups: All Purpose Flour
1 Cups: Unsalted Butter(Room Temperature)
2/3 Cups: Granulated Sugar

Add a newline escape sequence to the end of the string.

```
#include <stdio.h>
int main()
{
// Simple Recipe
printf("2 Cups: All Purpose Flour\n");
printf("1 Cups: Unsalted Butter(Room Temperature)\n");
}
```

Add a tab escape sequence in between the words butter and (Room Temperature)

```
#include <stdio.h>
int main()
{
// Simple Recipe
printf("2 Cups: All Purpose Flour\n");
printf("1 Cups: Unsalted Butter\t(Room Temperature)\n");
}
```

Add a printf() statement that outputs the string: "\n2/3 Cups: Granulated Sugar"

```
#include <stdio.h>
int main()
{
// Simple Recipe
printf("2 Cups: All Purpose Flour\n");
printf("1 Cups: Unsalted Butter\t(Room Temperature)");
printf("\n2/3 Cups: Granulated Sugar");
}
```

Exercise 3: printf and Parameters

```
#include <stdio.h>

int main() {

 int ageLearnedToRide = 5;

 printf("I was years old when I learned to ride a
bike.\n");
 printf("I hope I still remember how to ride.");
}
```

Modify the printf() command in the code to add the variable to the output so that the output makes sense.

```
#include <stdio.h>
int main() {
int ageLearnedToRide = 5;
// Field Specification %d
printf("I was %d years old when I learned to ride a
bike.\n",ageLearnedToRide);
printf("I hope I still remember how to ride.");
}
```

Sample Output:

I was 5 years old when I learned to ride a bike.
I hope I still remember how to ride.

The basic format is `printf("string to display", [list of optional parameters])`.

The optional parameters let you add values dynamically into the string, such as values stored in the variables we have been learning about. For example, if we wanted to output `Hello World, today is the 3rd!`, you could do that like this:

```
#include <stdio.h>
int main()
{
int day = 3;
printf("Hello World, today is the %drd!", day);}
```

When the compiler runs the code, it will replace %d with the value in the variable list, taking them in the order found in the string matching the order they are listed; first in the string is matched with first in the list of parameters).

There are lots of optional formatting and parameter types that can be used, but for our purposes here are a few of the basic ones.

To indicate a variable type you want to replace:

symbol	type
%d or %i	int
%f	double or float
%c	char

Exercise 4: Keyboard input.

Write program to ask add two numbers a & b from user.

```
#include <stdio.h>
int main()
{
   int a;
   int b;

   printf("Enter 2 integer numbers:\n");
   scanf("%d %d",&a,&b);

   printf("The 2 numbers are %d and %d .\n",a,b);
   return 0;
}
```

Example

```
#include <stdio.h>
int main()
{
   int testInteger;
   printf("Enter an integer: ");
   scanf("%d", &testInteger);
   printf("Number = %d",testInteger);
   return 0;
}
```

Sample Output:

```
Enter an integer: 4
Number = 4
```

The **scanf() function**, which stands for "scan format", reads a formatted string from the standard input stream, which is usually the user keyboard.

Syntax

```
scanf("formatted string", &variable);
```

Here, the user can enter a value in the terminal, **press Enter**, and that number will get stored in the pinNumber variable.

```
scanf("%d", &pinNumber);
```

Note: There's an ampersand & before the variable names in the arguments. The program will crash if there's a missing & sign.

Appendix A

Comments

When we write code it is important to document the code's behavior. One way to do this is to add comments to our code.

Starting a line with a double forward slash, `//`, will create a comment and the entire line will be ignored when we run the code. once you use `//` the rest of the line is now a comment.

If you want to create a comment with a beginning and end, you can use `/*` to begin the comment and `*/` to end the comment. This is known as a block comment. As you can see from the above example a block comment can wrap multiple lines without the use of anything but the beginning notation `/*` and the ending notation `*/`.

Syntax and Errors

When writing in C, we need to follow a set of rules in order for the code to run properly. These rules are known as *syntax*.

Case Sensitivity

Most of the words in the code use all lowercase letters. This is known as case-sensitivity. Whether lowercase or uppercase, certain words in the code must follow the correct case in order for the code to run. The only lines of text that can change case are the comment and the text between quotes.

The Semicolon

All statements, like the `printf()` statement, need to end with a semicolon. This identifies the end of the statement and is needed for the code to run correctly.

Double Quotes

The text in between the double quotes `"` is known as a string (think of a string of characters). All strings must be surrounded by double-quotes. So what happens when we break the rules? The answer is *errors*.

Escape Sequences

In C, an escape sequence is a non-visual character used within a string. `\n` is an escape sequence that adds a newline to a string. `\t` is an escape sequence that adds a tab of spaces to a string.