C Code Structure

Lesson 4

Learning Objectives

01 Identify the parts of a basic C Program

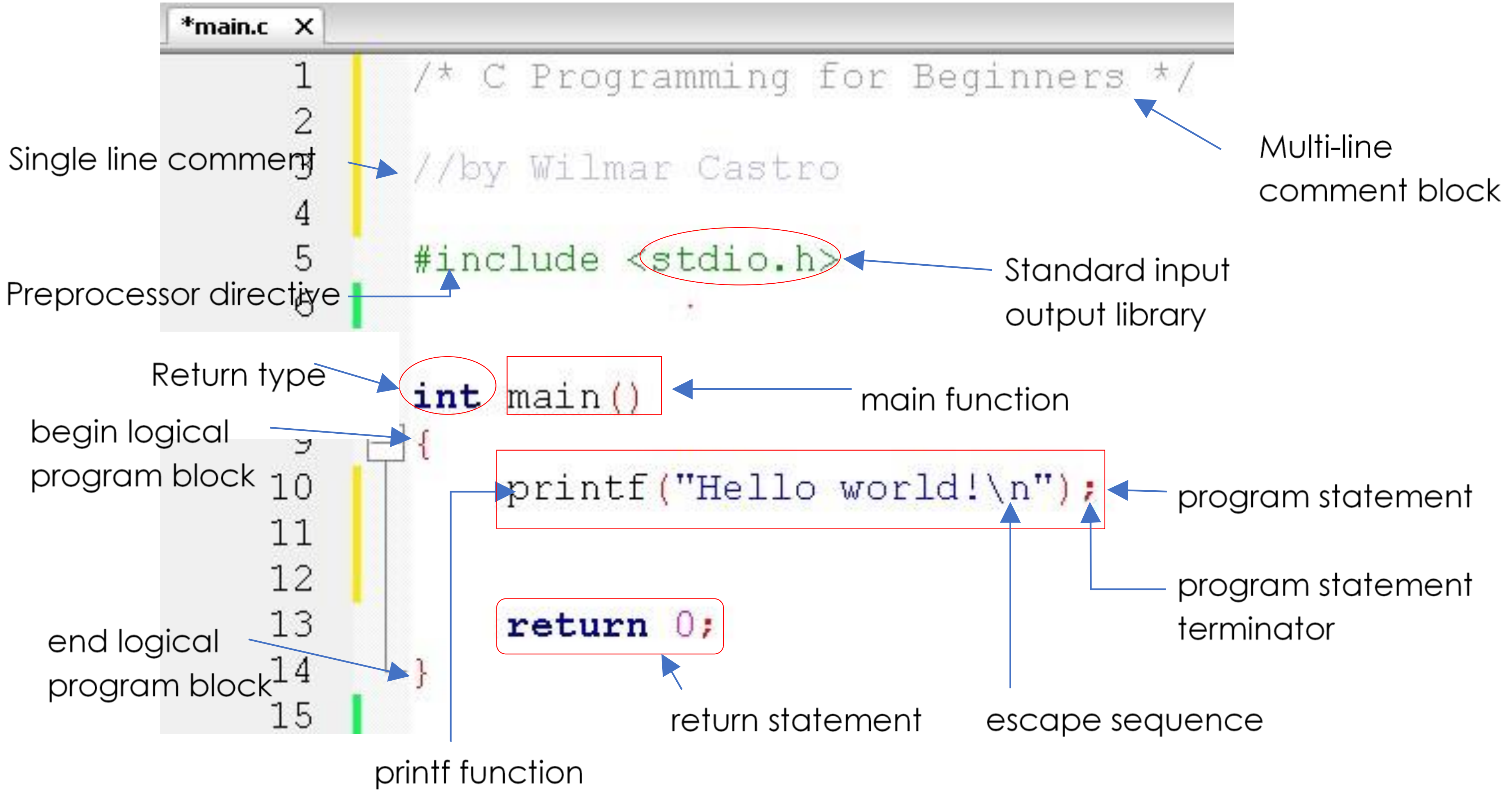
02 Familiarize with the different C Language Keywords

03 Create C Program statements

04 Familiarize the different common escape sequences

05 Identify the different common program errors in C.

Basic C Program



```c
/* C Programming for Beginners */

//by Wilmar Castro

#include <stdio.h>

int main()
{
printf("Hello world!\n");

return 0;

}
```

Main Function

- In C programming, every program starts with the main() function.
- A function allows you to group logical series of activities or program statements under one name.
- Functions are not static, meaning, they can accept inputs (data), process those inputs, and give back output (information).
- What makes the main() function different from other typical functions is that the values it returns are returned back to the operating system.
- Also called as the “Mother of all functions” or driver function, because the program execution will start at this function.

Main Function

- We will use main() function that is void of parameters (functions that do not accept parameters) and return value with type of int (integer).

```
int main()
{
        //program statements here
}
```

Note….

- *Bear in mind that a C program can only have one main() function for the program to work properly.*
- *C language is a case-sensitive programming language. For example, the function names **main()**, **Main()**, and **MAIN()** are not the same.*

Comments

- A comment is a programmer-readable explanation or annotation in the source code of a computer program.
- Comments must be useful in documenting your program code.
- *Two types of comments:*
    - *Single-line comment*
    - *Multi-line comment*

Single-line comment

- Double forward slash (//) is used to create a single-line comment
- Any characters placed after the character set // are ignored by the compiler for that line only.

```
//C Programming for Absolute Beginners

//Module 1 – Elementary Programming Concepts

//Prepared by Mr. Wilmar Castro
```

Multi-line comment or Block comment

- The character set /* indicates the beginning of a block comment; the character set */ indicates the end of a block comment.
- The character sets are not required within the same line and can be used to create both single-line comments and multi-line comments.

```
/* C Programming for Absolute Beginners

   Module 1 – Elementary Programming Concepts

   Prepared by: Mr. Wilmar Castro

*/
```

Keywords

- There are 32 words defined in the Standard ANSI C programming language.
- These keywords have predefined uses and cannot be used for any other purpose in a C program.
- Remember that these keywords must be written in lowercase.

C Keywords

Table 1.1 C Language Keywords

Keyword	**Description**
auto	defines a local variable as having a local lifetime
break	passes control out of the programming construct
case	branch control
char	basic data type
const	unmodifiable value
continue	passes control to loop's beginning
default	branch control

C Keywords

do	Do While loop
double	floating-point data type
else	conditional statement
enum	defines a group of constants of type *int*
extern	indicates an identifier as defined elsewhere
float	floating-point data type
for	For loop
goto	transfers program control unconditionally
if	conditional statement
int	basic data type

C Keywords

long	type modifier
register	stores the declared variable in a CPU register
return	exits the function
short	type modifier
signed	type modifier
sizeof	returns expression or type size
static	preserves variable value after its scope ends

C Keywords

struct	groups variables into a single record
switch	branch control
typedef	creates a new type
union	groups variables that occupy the same storage space
unsigned	type modifier
void	empty data type
volatile	allows a variable to be changed by a background routine
while	repeats program execution while the condition is true

Program Statements

- Many lines in a C program are considered program statements, which serve to control program execution and functionality.
- A program statement must end with a statement terminator.
- Statement terminator is basically a semicolon (;).

Program Statements Example

```
printf(“Hello World \n”);

int sum = firstNum +secondNum;

int finalResult = sum;
```

Program Statements

- Here are some common program statements that do not require a statement terminator:
  - Comments
  - Preprocessor directives (e.g. #include and #define)
  - Begin and end program block identifiers
  - Function definition beginnings (for example, main())

Program Statements

- The preceding program statements do not require a statement terminator because they are not executable C statements or function calls.
- Only C statements that perform work during program execution require semicolons.

Escape Sequences

- When you start coding, there will be times when you need to format the result that you will display on the computer screen.
- For the most part, the characters or text that you want to display on-screen are put inside quotation marks, with the exception of escape characters or escape sequences.
- The **backslash character** (\) is the escape character.

Escape Sequences

- When the printf() function is executed, the program looks forwar d to the next character that follows the escape character.
- In this case, the next character is the character n. Together, the backslash (\) and n character form an escape sequence.
- This particular escape sequence (\n) notifies the program to add a new line.

```
printf(“Hello World\n”);
```

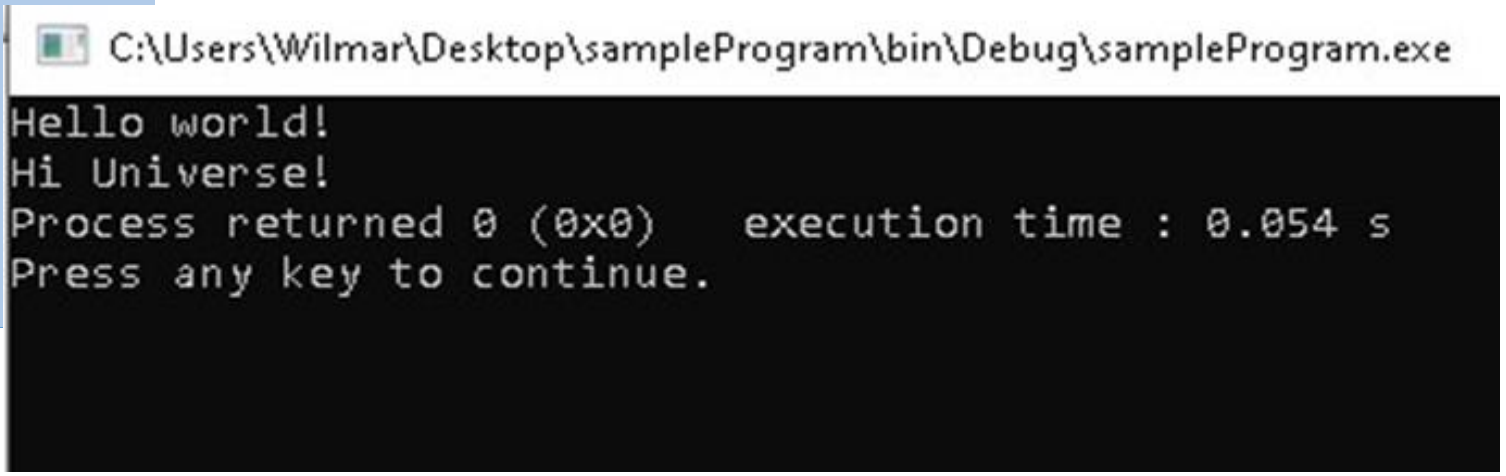
Common Escape Sequences

Table 1.2 Common Escape Sequences

Escape sequence	Purpose
\n	Creates a new line
\t	Moves the cursor to the next tab
\\	Inserts a backslash
\”	Inserts a double quote
\’	Inserts a single quote

Escape Sequence \n

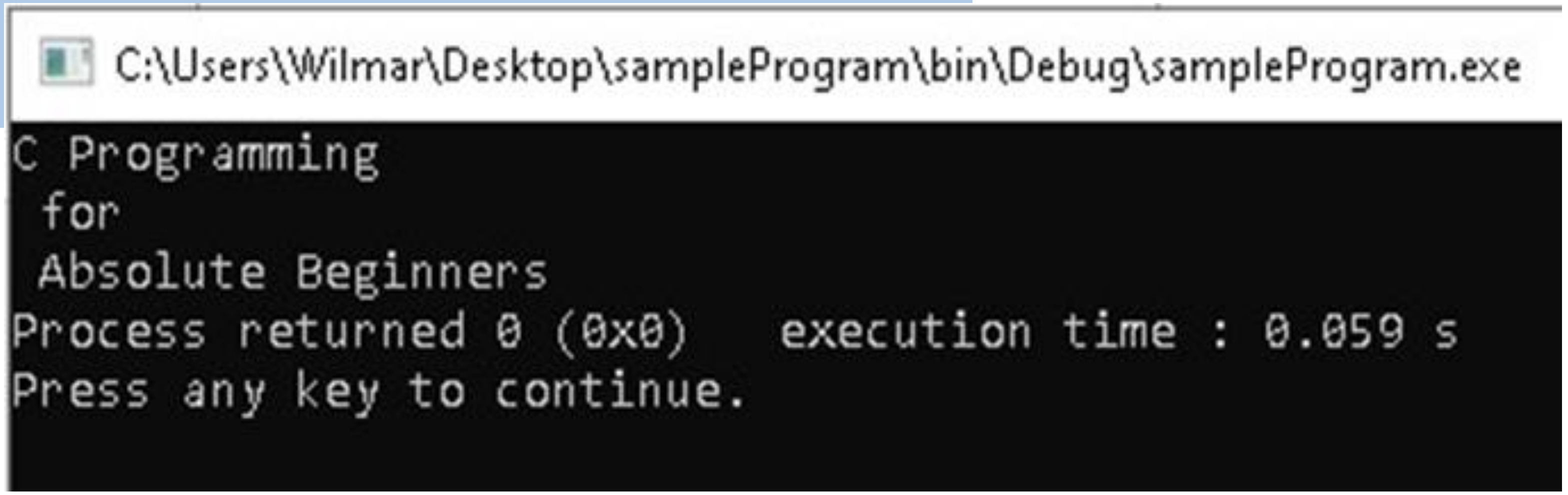
```
#include <stdio.h>

int main()

{

    printf("Hello world!\n");

    printf("Hi Universe!");

    return 0;

}
```



```
C:\Users\Wilmar\Desktop\sampleProgram\bin\Debug\sampleProgram.exe
Hello world!
Hi Universe!
Process returned 0 (0x0)   execution time : 0.054 s
Press any key to continue.
```

Escape Sequence \n

```
#include <stdio.h>

int main()

{

    printf("C Programming\n for\n Absolute Beginners");

    return 0;

}
```

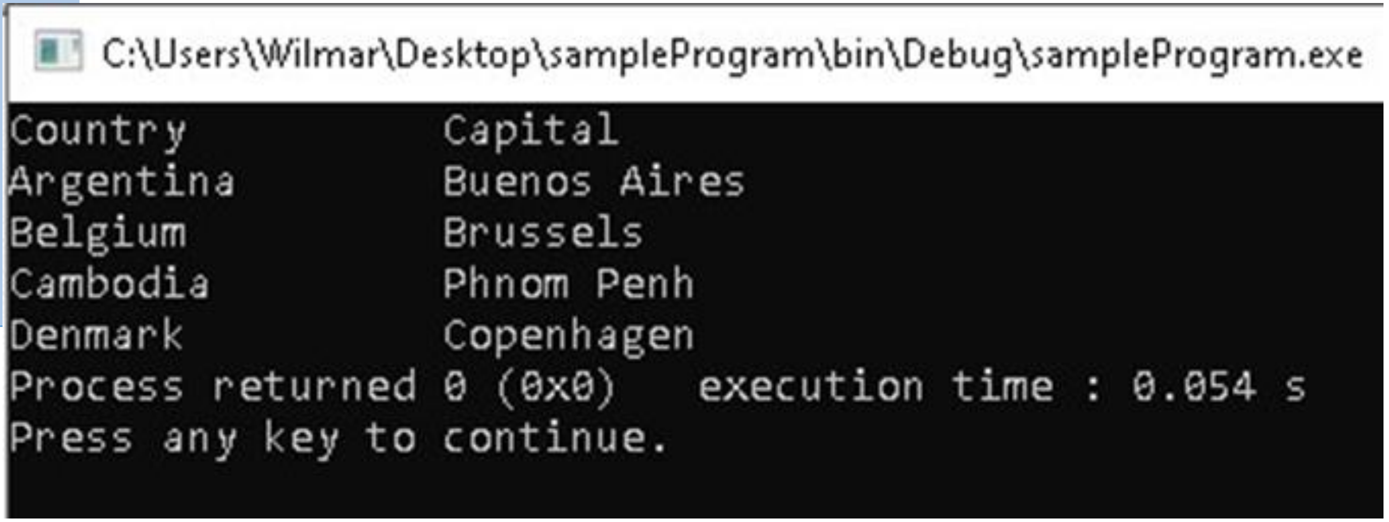


```
C:\Users\Wilmar\Desktop\sampleProgram\bin\Debug\sampleProgram.exe
```

```
C Programming
 for
 Absolute Beginners
Process returned 0 (0x0)   execution time : 0.059 s
Press any key to continue.
```

Escape Sequence \t

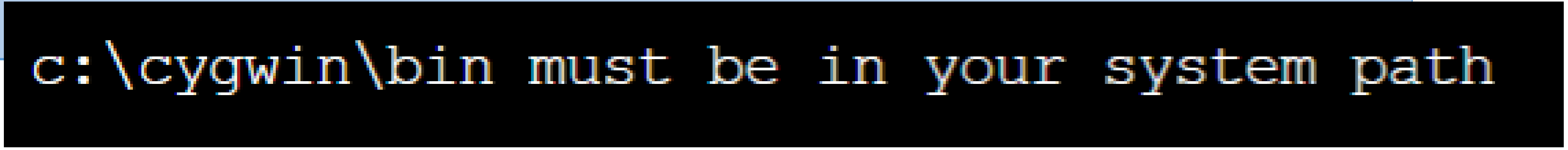
```
#include <stdio.h>

int main()

{

    printf("Country \t Capital \n");

    printf("Argentina \t Buenos Aires \n");

    printf("Belgium \t Brussels \n");

    printf("Cambodia \t Phnom Penh \n");

    printf("Denmark \t Copenhagen");

    return 0;

}
```



```
C:\Users\Wilmar\Desktop\sampleProgram\bin\Debug\sampleProgram.exe
Country                 Capital
Argentina               Buenos Aires
Belgium                 Brussels
Cambodia                Phnom Penh
Denmark                 Copenhagen
Process returned 0 (0x0)   execution time : 0.054 s
Press any key to continue.
```

Escape Sequence \\

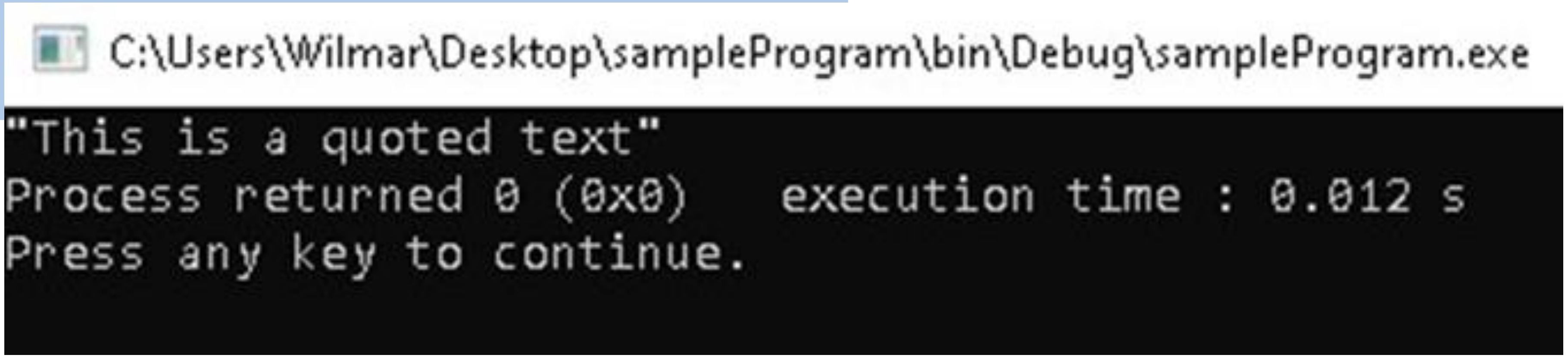
```
#include <stdio.h>

int main()

{

    printf("\n c:\\cygwin\\bin must be in your system path\n");

   return 0;

}
```



```
c:\cygwin\bin must be in your system path
```

Escape Sequence \”

```
#include <stdio.h>

int main()

{

    printf("\"This is a quoted text\"");

    return 0;

}
```

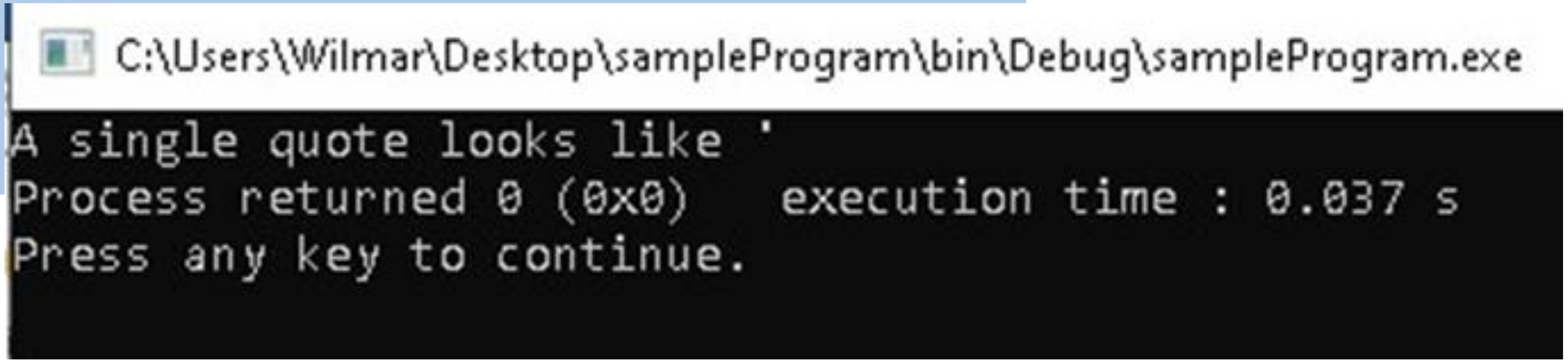


C:\Users\Wilmar\Desktop\sampleProgram\bin\Debug\sampleProgram.exe

```
"This is a quoted text"
Process returned 0 (0x0)   execution time : 0.012 s
Press any key to continue.
```

Escape Sequence \’

```
#include <stdio.h>

int main()

{

   printf("A single quote looks like \'");

   return 0;

}
```



C:\Users\Wilmar\Desktop\sampleProgram\bin\Debug\sampleProgram.exe

```
A single quote looks like '
Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.
```

Directives

- Notice the program statement that starts with the pound sign (#)

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

- When the C preprocessor encounters the pound sign, it performs certain actions depending on the directive that occurs prior to compiling.

Directives

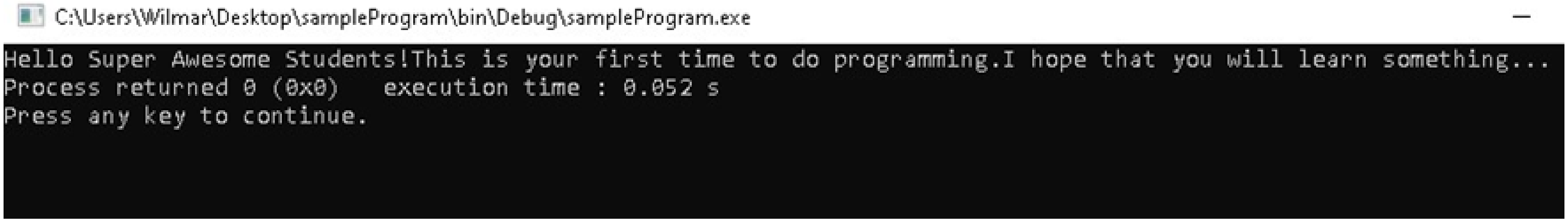
- Omitting this preprocessor directive will not cause any undesirable effect when compiling or running your program. However, including the header file allows the compiler to determine error locations.

How to Debug C Programs

- Debugging is an art that is important in Computer Science. The more you practice programming, the easier you will find and correct bugs.
- Majority of times your program will compile and run smoothly, but with results that did not go as planned.

How to Debug C Programs

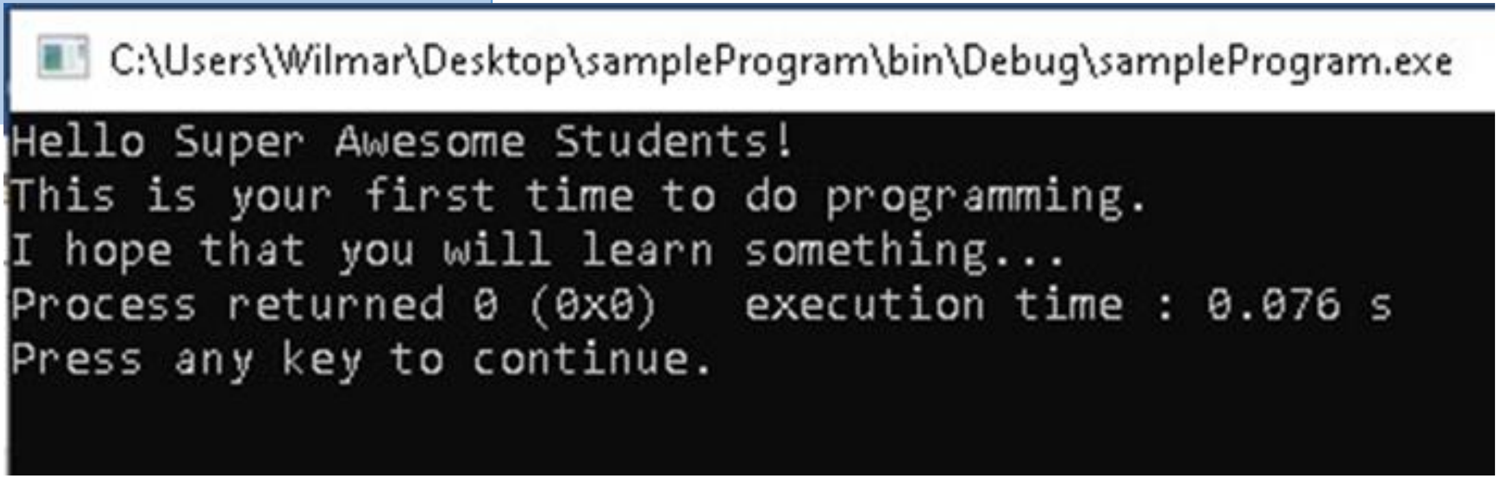
```
#include <stdio.h>

int main()

{

    printf("Hello Super Awesome Students!");

    printf("This is your first time to do programming.");

    printf("I hope that you will learn something...");

    return 0;

}
```



```
C:\Users\Wilmar\Desktop\sampleProgram\bin\Debug\sampleProgram.exe

Hello Super Awesome Students!This is your first time to do programming.I hope that you will learn something...
Process returned 0 (0x0)   execution time : 0.052 s
Press any key to continue.
```

How to Debug C Programs

```c
#include <stdio.h>

int main()

{

    printf("Hello Super Awesome Students! \n");

    printf("This is your first time to do programming.\n");

    printf("I hope that you will learn something...");

    return 0;

}
```



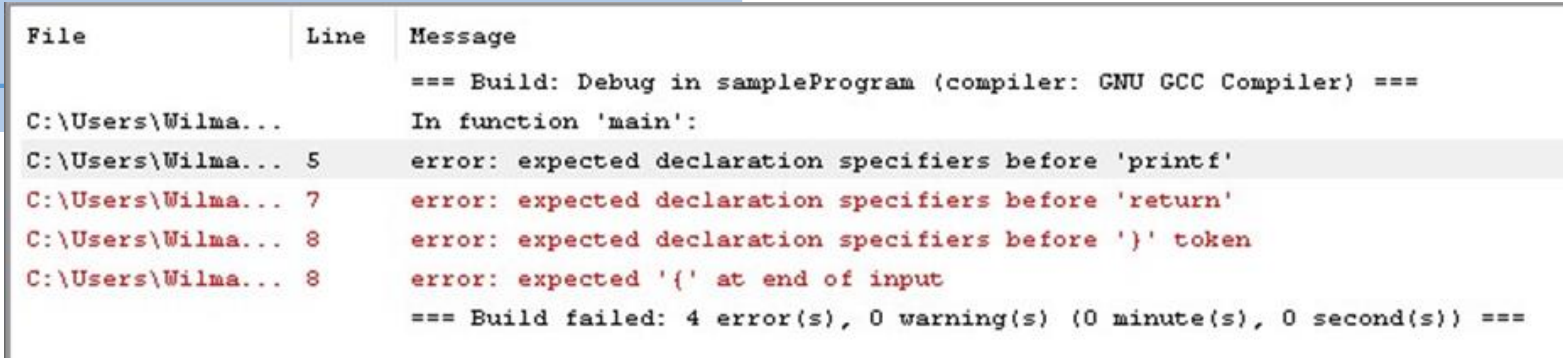
C:\Users\Wilmar\Desktop\sampleProgram\bin\Debug\sampleProgram.exe

```
Hello Super Awesome Students!
This is your first time to do programming.
I hope that you will learn something...
Process returned 0 (0x0)   execution time : 0.076 s
Press any key to continue.
```

Common Error #1: Missing Program Block Identifiers

- *If you forget to insert a beginning or a corresponding end program block identifier ({or}).*

```
#include <stdio.h>

int main()

   printf("Hello Super Awesome Students! \n");

      return 0;

}
```

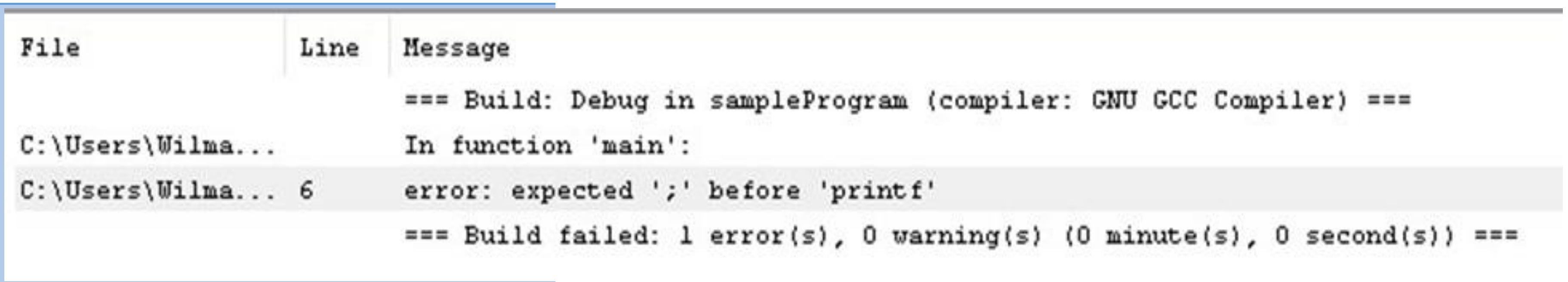


```
File                Line   Message
                           === Build: Debug in sampleProgram (compiler: GNU GCC Compiler) ===
C:\Users\Wilma...          In function 'main':
C:\Users\Wilma...   5      error: expected declaration specifiers before 'printf'
C:\Users\Wilma...   7      error: expected declaration specifiers before 'return'
C:\Users\Wilma...   8      error: expected declaration specifiers before '}' token
C:\Users\Wilma...   8      error: expected '{' at end of input
                           === Build failed: 4 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

Common Error #2: Missing Statement Terminators

- *Terminator is a very important character that ends a program statement. If you neglected the terminator a parse error might occur like the sample code below.*

```
#include <stdio.h>

int main()

{

    printf("Hello Super Awesome Students! \n")

    printf("Welcome to C Programming");

    return 0;

}
```

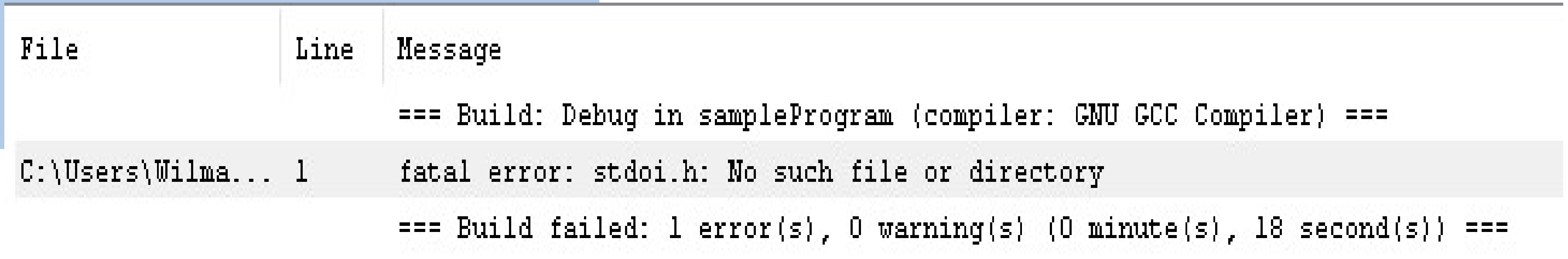


```
File                Line   Message
                           === Build: Debug in sampleProgram (compiler: GNU GCC Compiler) ===
C:\Users\Wilma...          In function 'main':
C:\Users\Wilma...   6      error: expected ';' before 'printf'
                           === Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

Common Error #3: Invalid Preprocessor Directive

- *Non-existing library or misspelled library names.*

```
#include <stdoi.h>

int main()

{

   printf("Hello Super Awesome Students! \n");

   printf("Welcome   to   C   Programming");

   return 0;

}
```



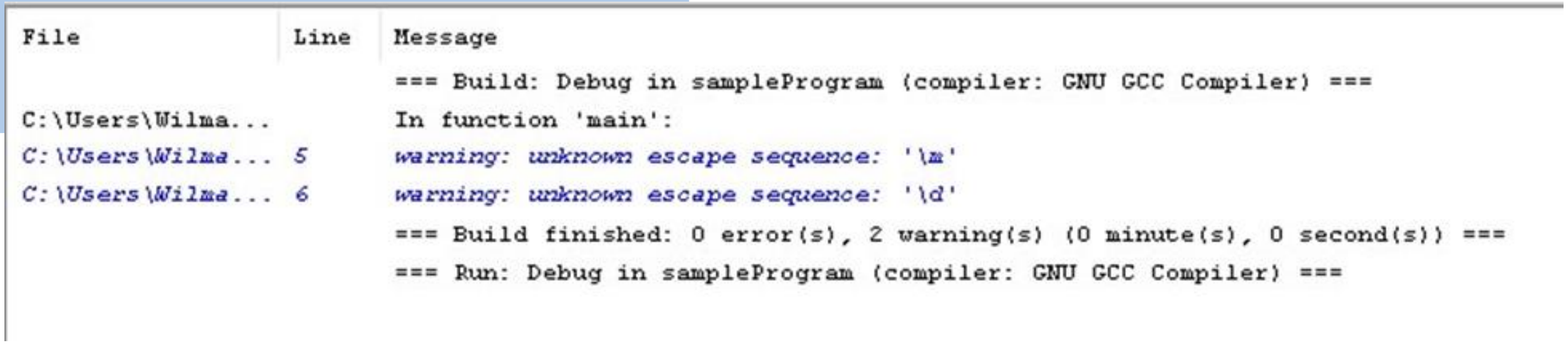
```
File              Line  Message
                        === Build: Debug in sampleProgram (compiler: GNU GCC Compiler) ===
C:\Users\Wilma...  1    fatal error: stdoi.h: No such file or directory
                        === Build failed: 1 error(s), 0 warning(s) (0 minute(s), 18 second(s)) ===
```

Figure 1.14 Misspelling library names

Common Error #4: Invalid Escape Sequences

- *It is common to use invalid characters or invalid sequences.*

```
#include <stdio.h>

int main()

{

    printf("Hello Super Awesome Students! \m");

    printf("  \d  Welcome  to  C  Programming");

    return 0;

}
```

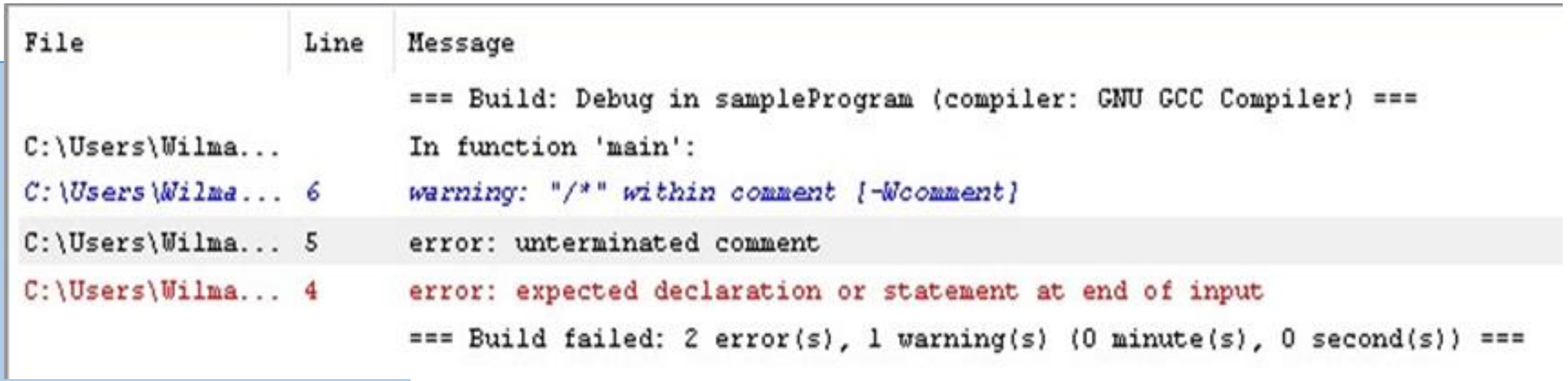


```
File                 Line   Message
                            === Build: Debug in sampleProgram (compiler: GNU GCC Compiler) ===
C:\Users\Wilma...           In function 'main':
C:\Users\Wilma...    5      warning: unknown escape sequence: '\m'
C:\Users\Wilma...    6      warning: unknown escape sequence: '\d'
                            === Build finished: 0 error(s), 2 warning(s) (0 minute(s), 0 second(s)) ===
                            === Run: Debug in sampleProgram (compiler: GNU GCC Compiler) ===
```

Common Error #5: Invalid Comments

- *Interchanging the sequence of characters of creating comments.*

```
#include <stdio.h>

int main()

{

        /*This demonstrates a comment block

    which I think is very awesome! /*

    printf("Hello Super Awesome Students! \n");

    printf("Welcome to C Programming");

    return 0;

}
```



```
File                Line   Message
                           === Build: Debug in sampleProgram (compiler: GNU GCC Compiler) ===
C:\Users\Wilma...          In function 'main':
C:\Users\Wilma...   6      warning: "/*" within comment [-Wcomment]
C:\Users\Wilma...   5      error: unterminated comment
C:\Users\Wilma...   4      error: expected declaration or statement at end of input
                           === Build failed: 2 error(s), 1 warning(s) (0 minute(s), 0 second(s)) ===
```

Any questions or clarifications? ☺

THANK YOU