Math Series

C Programming

UG Sem-3 Major (Kalyani University)

Day-1

What is C?

C is a general-purpose programming language created by Denni Ritchie at the Bell Laboratories in 1972.

It is a very popular language, despite being old. The main reason for its popularity is because it is a fundamental language in the field of computer science.

C is strongly associated with UNIX, as it was developed to write the UNIX operating system.

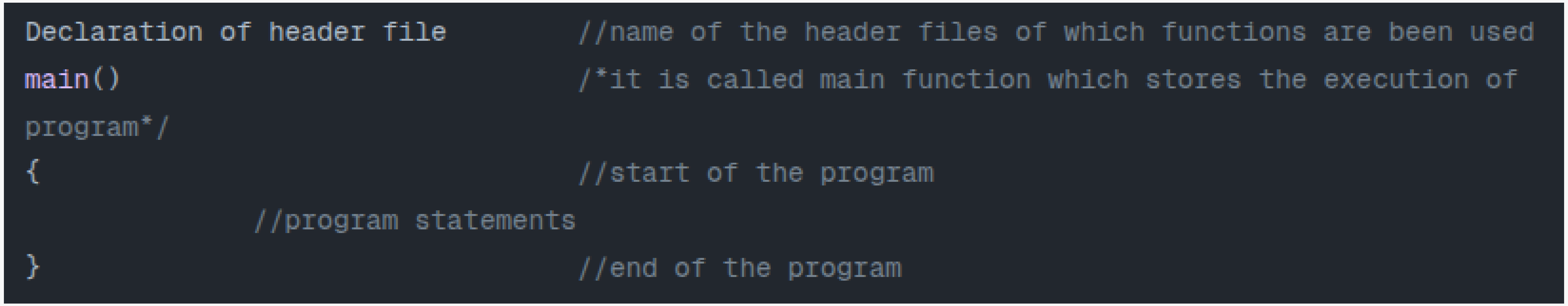
Uses of C?

C language is used to program a wide variety of systems. Some of the uses of C are as follows:

1. Major parts of Windows, Linux and other operating systems are written in C.
2. C is used to write driver programs for devices like tablets, printers etc.
3. C language is used to program embedded systems where programs need to run faster in limited memory (Microwave, Cameras etc.)
4. C is used to develop games, an area where latency is very important, i.e., the computer must react quickly to user input.

Syntax

An example below shows how a basic C program is written.

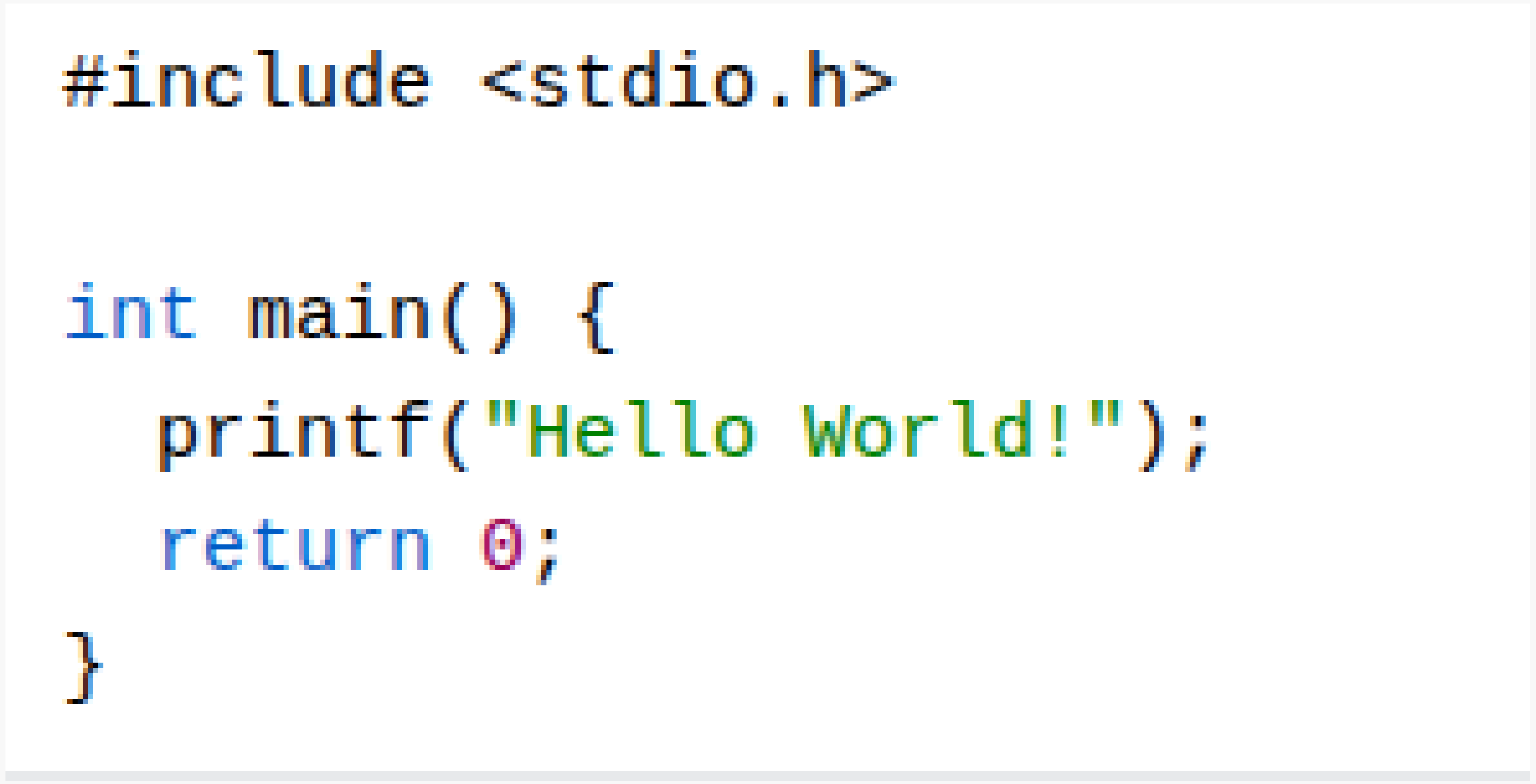


```
Declaration of header file        //name of the header files of which functions are been used
main()                            /*it is called main function which stores the execution of
program*/
{                                 //start of the program
              //program statements
}                                 //end of the program
```

- Here, the first line is a pre-processor command including a header file stdio.h.
- C ignores empty lines and spaces.
- There is a main() function then, which should always be there.

Syntax (Example)

Our First Hello World Program



```c
#include <stdio.h>

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

Syntax (Example Explained)

Line 1: #include <stdio.h> is a header file library that lets us work with input and output functions, such as printf() (used in line 4). Header files add functionality to C programs.

Line 2: A blank line. C ignores white space. But we use it to make the code more readable.

Line 3: Another thing that always appear in a C program is main(). This is called a function. Any code inside its curly brackets {} will be executed.

Syntax (Example Explained)

Line 4: printf() is a function used to output/print text to the screen. In our example, it will output "Hello World!".

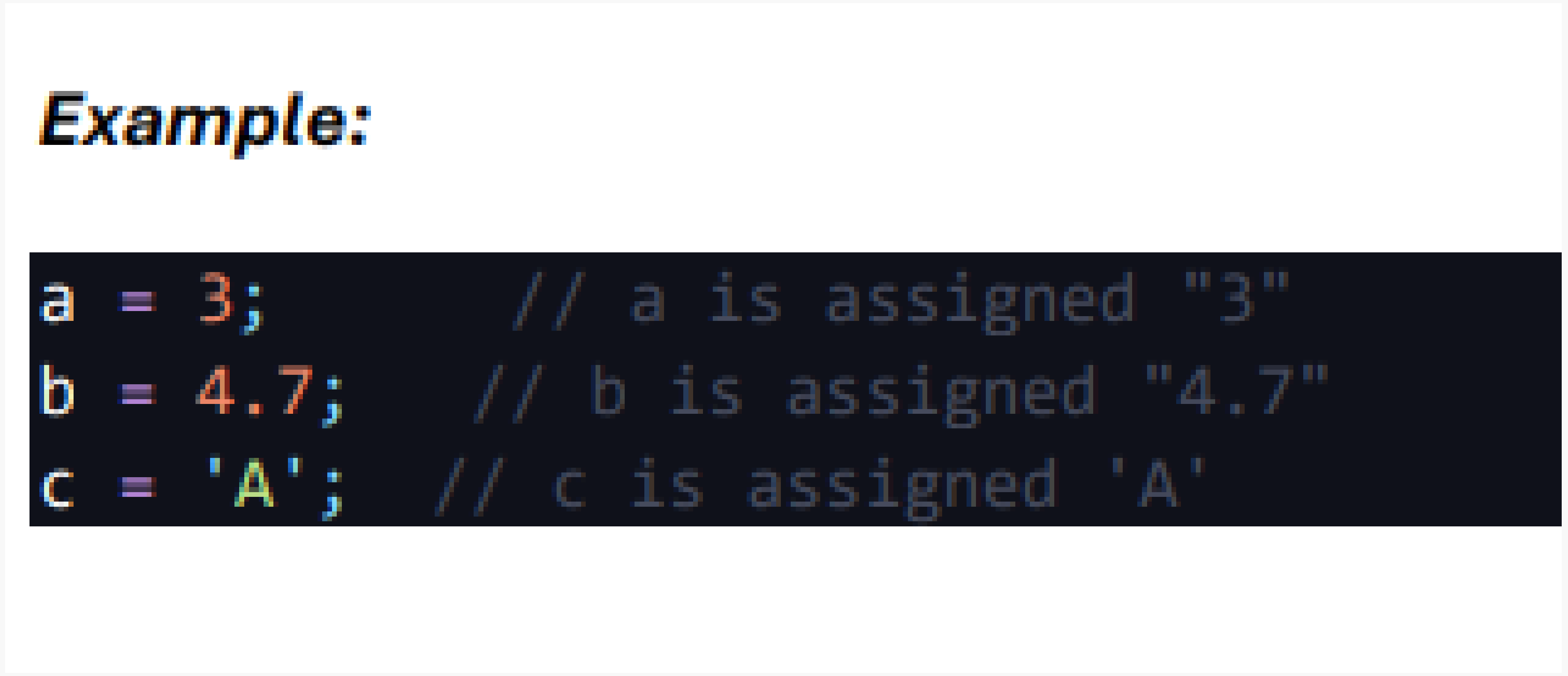
Line 5: return 0 ends the main() function.

Line 6: Do not forget to add the closing curly bracket } to actually end the main function.

Note that: Every C statement ends with a semicolon ;

Variables in C

A variable is a container which stores a ‘value’. In kitchen, we have containers storing Rice, Dal, Sugar etc. Similar to that, variables in C stores value of a constant.


Example:

```
a = 3;       // a is assigned "3"
b = 4.7;    // b is assigned "4.7"
c = 'A';   // c is assigned 'A'
```

Rules for naming variables in C

1. First character must be an alphabet or underscore (_)
2. No commas, blanks are allowed.
3. No special symbol other than (_) allowed.
4. Variable names are case sensitive.

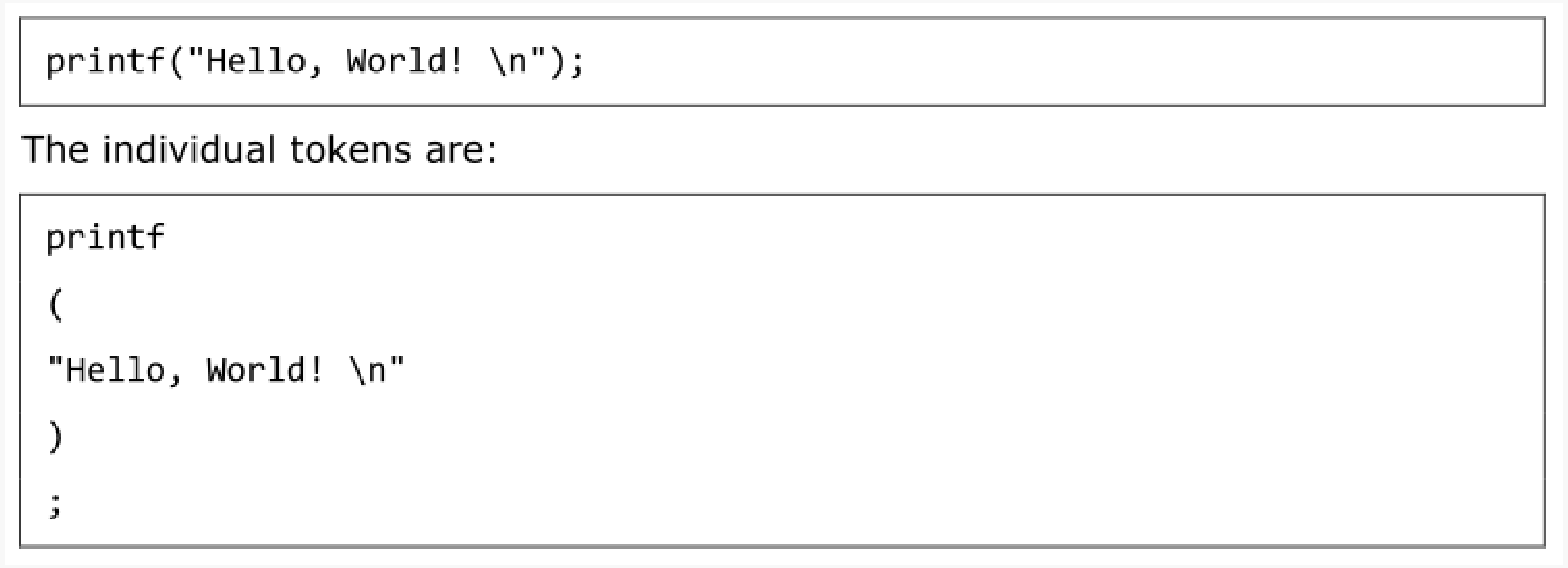
We must create meaningful variable names in our programs.
This enhances readability of our programs.

Tokens in C?

A C program is made up of different tokens combined. These tokens include :

- *Keywords*
- *Identifiers*
- *Constants*
- *String Literal*
- *Symbols*

For example, the following C statement consists of five tokens:



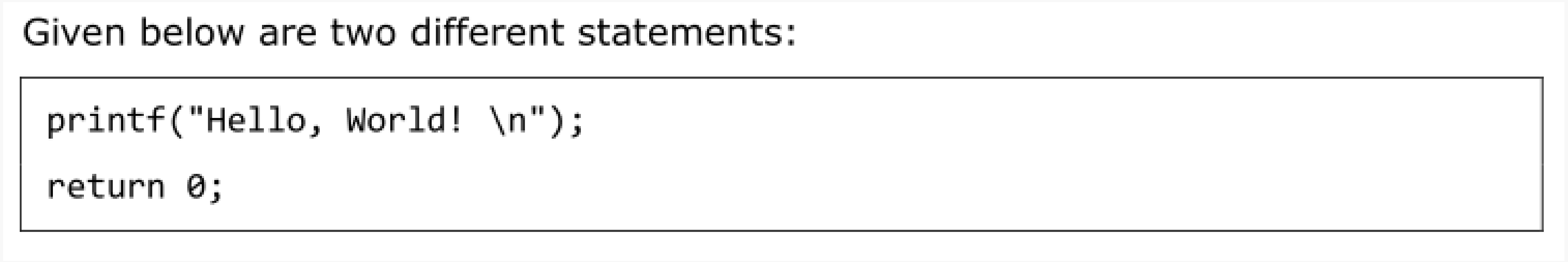
```
printf("Hello, World! \n");
```

The individual tokens are:

```
printf
(
"Hello, World! \n"
)
;
```

Semicolons

In a C program, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.


Given below are two different statements:

```
printf("Hello, World! \n");

return 0;
```

Comments in C

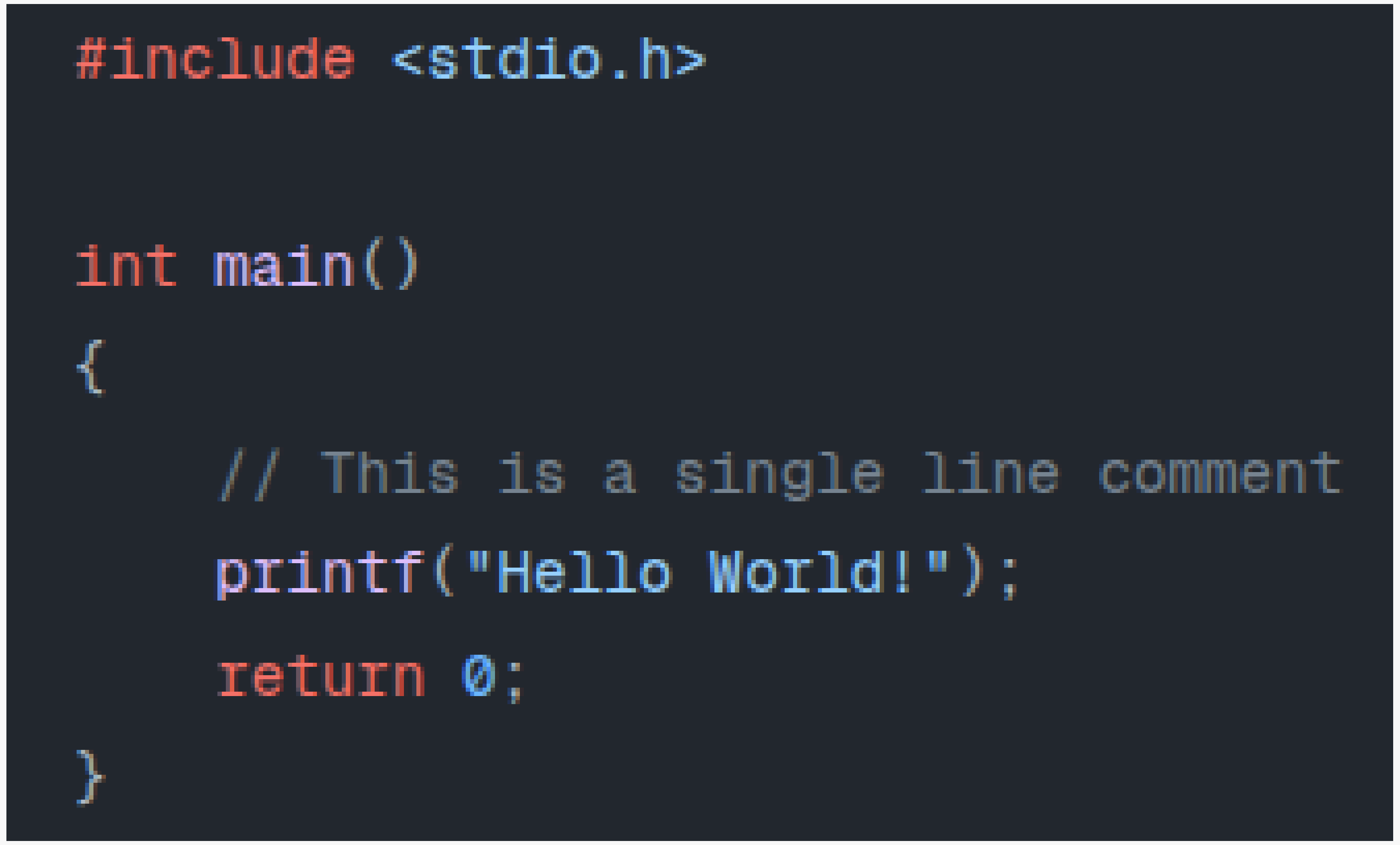
Comments can be used to insert any informative piece which a programmer does not wish to be executed. It could be either to explain a piece of code or to make it more readable. In addition, it can be used to prevent the execution of alternative code when the process of debugging is done.

Comments can be single-lined or multi-lined.

Single Line Comments in C

- Single-line comments start with two forward slashes (//).
- Any information after the slashes // lying on the same line would be ignored (will not be executed).

An example of how we use a single-line comment:



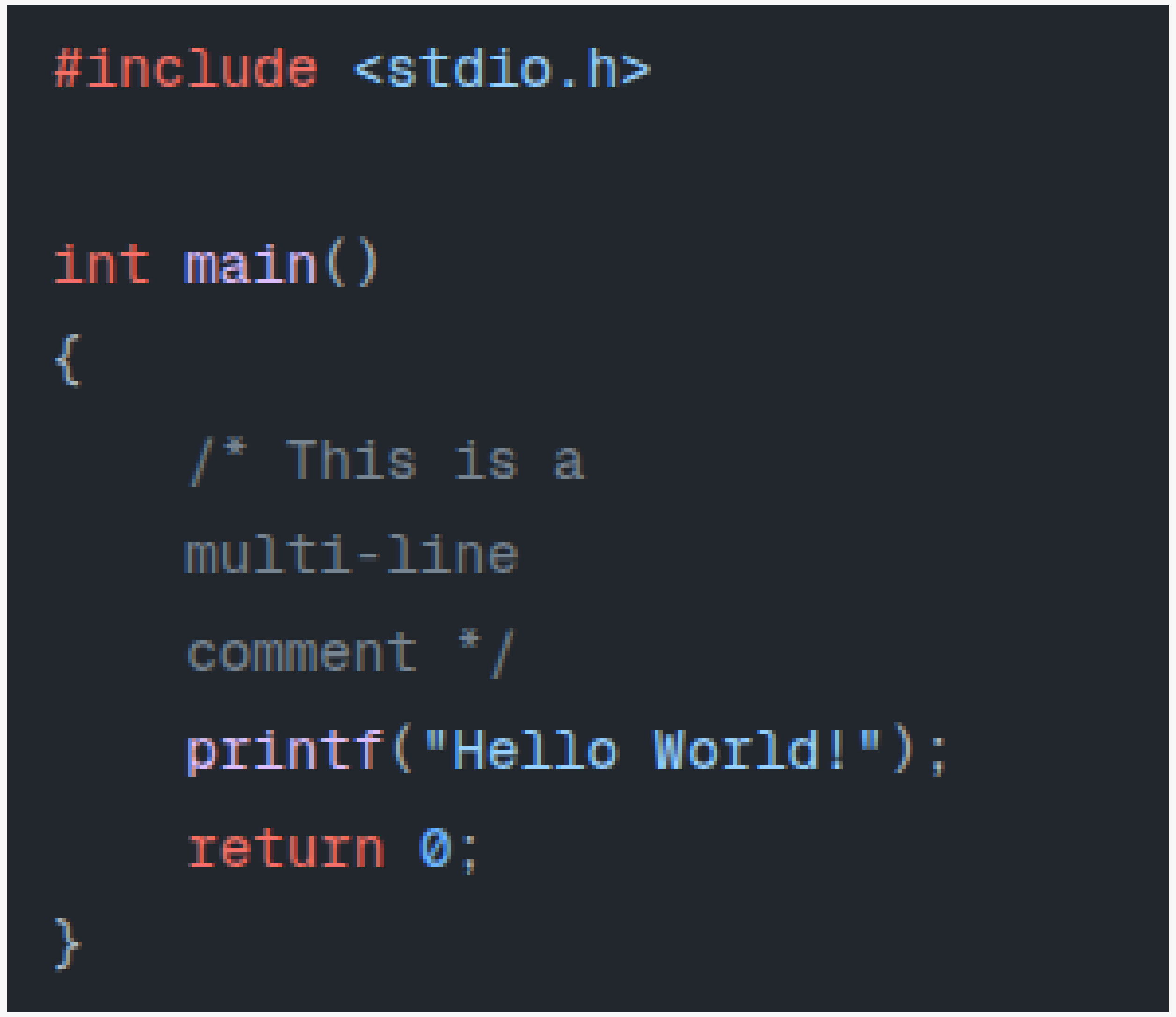
```
#include <stdio.h>

int main()
{
    // This is a single line comment
    printf("Hello World!");
    return 0;
}
```

Multi-line Comments in C

- A multi-line comment starts with /* and ends with */.
- Any information between /* and */ will be ignored by the compiler.

An example of how we use a multi-line comment:



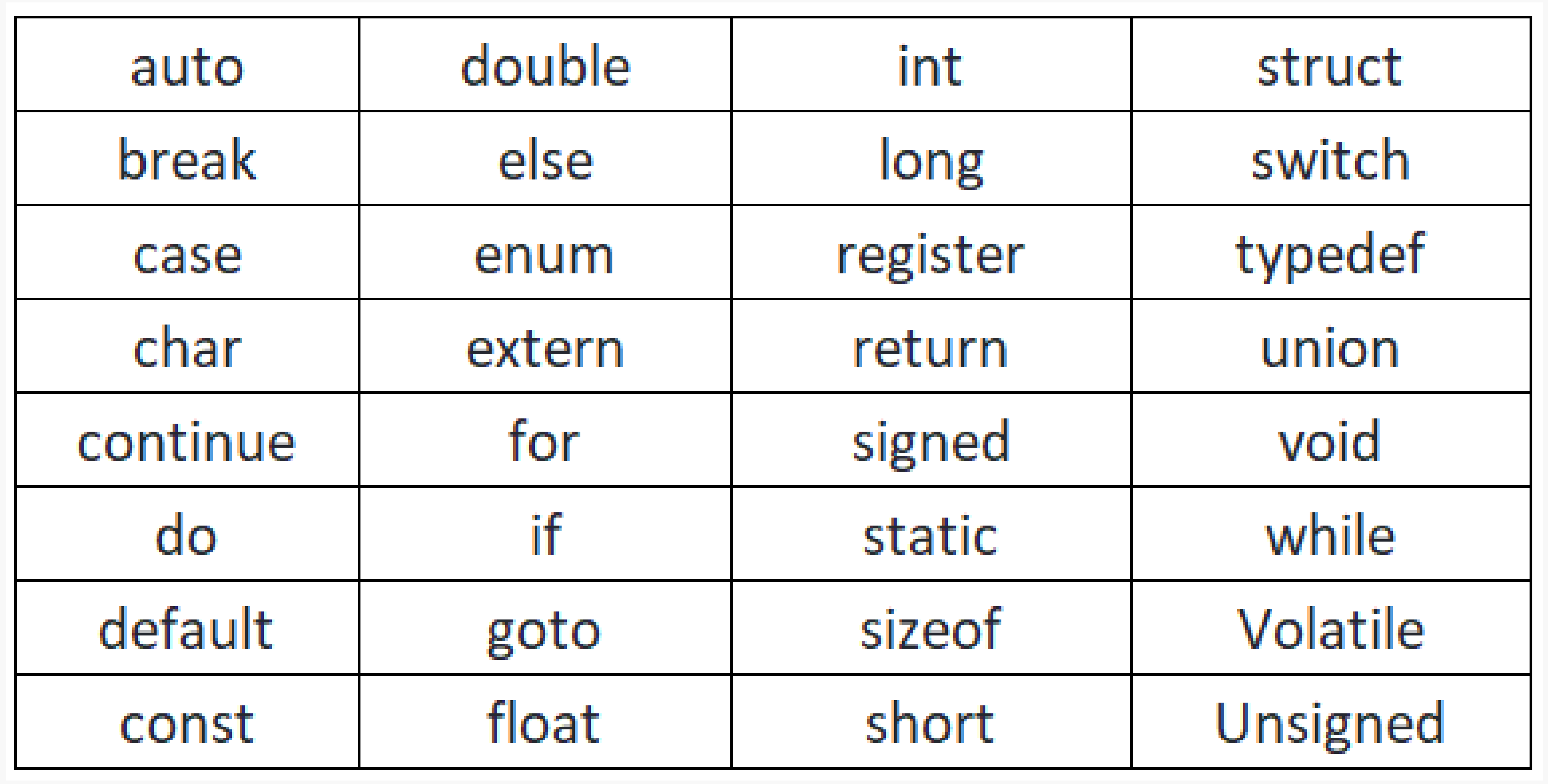
```c
#include <stdio.h>

int main()
{
    /* This is a
    multi-line
    comment */
    printf("Hello World!");
    return 0;
}
```

Keywords in C

Keywords are reserved words that cannot be used elsewhere in the program for naming a variable or a function. They have a specific function or task and they are solely used for that. Their functionalities are pre-defined.

32 Keywords in C



auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	Volatile
const	float	short	Unsigned

Identifiers in C

Identifiers are names given to variables or functions to differentiate them from one another. Their definitions are solely based on our choice but there are a few rules that we have to follow while naming identifiers. One such rule says that the name cannot contain special symbols such as @, -, *, <, etc.

C is a case-sensitive language so an identifier containing a capital letter and another one containing a small letter in the same place will be different. For example, the three words: Code, code, and cOde can be used as three different identifiers.

Rules for naming Identifiers in C

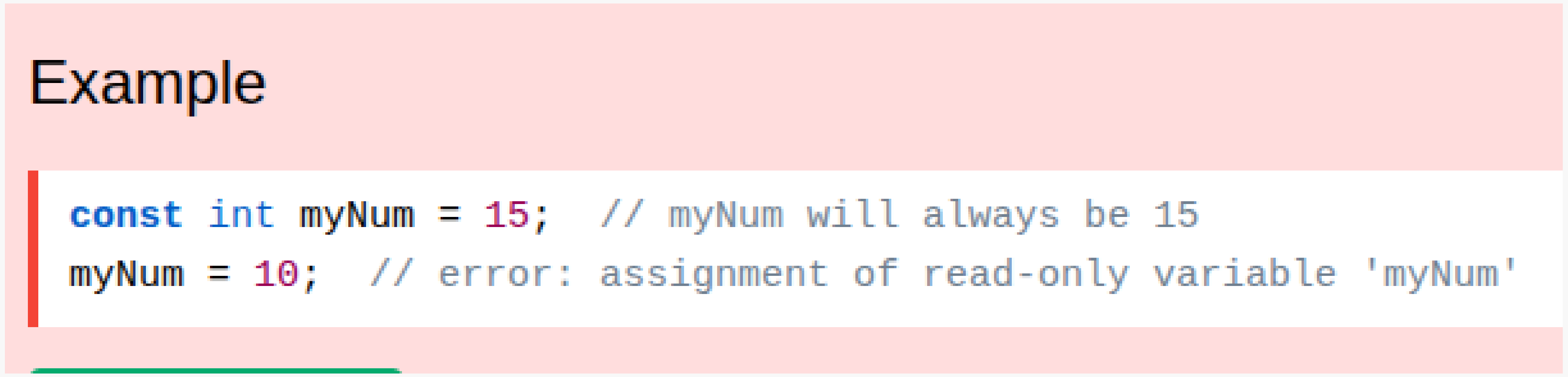
Rules for naming identifiers:

1. One should not name any identifier starting with a numeric value or symbol. It should start only with an underscore or alphabet.
2. They should not contain spaces.
3. Giving logical names is recommended as per our program.

Constants in C

Constants are very similar to a variable and they can also be of any data type. The only difference between a constant and a variable is that a constant’s value never changes.

This can be done with the **const** keyword, which makes a variable **unchangeable** and **read-only:**



Example

```
const int myNum = 15;  // myNum will always be 15
myNum = 10;  // error: assignment of read-only variable 'myNum'
```