

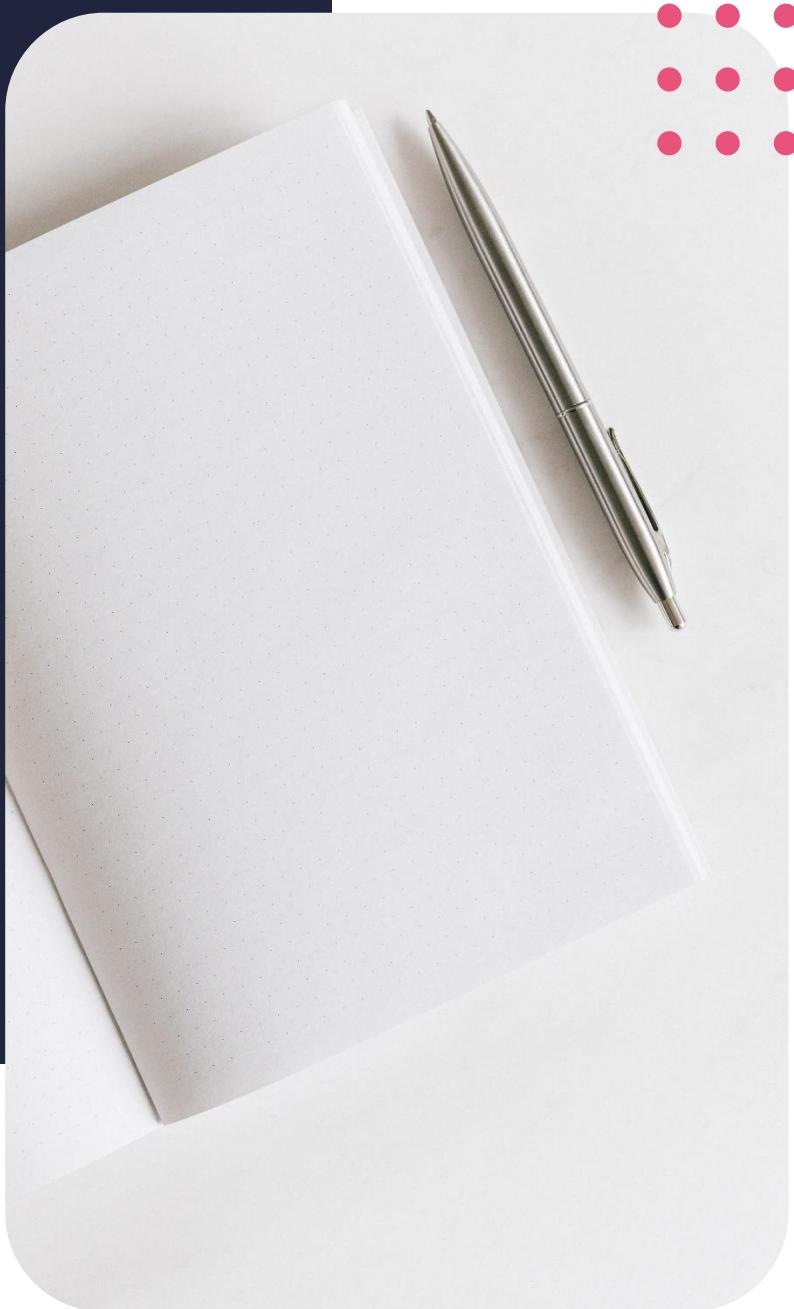
Diploma in Computer Science

Basic input and output



Contents

Format specifiers	3
Specifier chart	3
Ground rules	4
Formatting	4
Input	5
Input syntax	5
Scanf vs gets	5
Output	5
Output syntax	5
Examples	6
Conclusion	6



Lesson outcomes

By the end of this lesson you should be able to:

Discover the ways in which input and output is carried out

Understand the syntax involved in using input and output functions

Develop skills in picking the right specifiers for a data type

Understand how to avoid errors when dealing with output

Welcome to our 7th lesson in our series. We are almost done with our second module, and pretty soon we will be ready to dive even deeper into more advanced programming concepts. In our previous lesson, we learnt about variables and constants, and how they're the most important components of programming, as this is what allows us to manipulate data. As promised, this lesson is going to be a little different, as there is going to be a lot of IDE time. In this lesson, we are going to learn about a new concept that we picked up from our last lesson, accepting data from the keyboard.

Format specifiers

Just like variables, when inputting data or outputting data, we need a way in which we can tell the computer which format the data is in. This allows the computer to process the data correctly. Surely, it would be really nice to be able to throw any type of data at the computer and it would automatically know which type of data it is, but hey, **SOMEBODY** has to program the computer so it can know how to do that to begin with, and that somebody is you, the programmer!

Specifier chart

To makes things simple, we will put everything we need in a handy table that you can use as a reference. Of course, with time, you will know all the specifiers by head, but for a bit of time you need to refer to the table to avoid syntax errors.

There are several input and output functions, namely getchar, putchar, puts, scanf and printf.

Getchar and putchar are used to transfer single characters.

Puts is used to display strings, and scanf and printf are the more versatile ones, can be used for the transfer of single characters, numerical values and strings.

You can call an input/ output function from anywhere in the program, by following the syntax that we have been using in previous lessons. Just to jog your memory, the syntax is as follows, printf(specifier, "string/variable") and for input, scanf(specifier, "string/variable")

The names of functions that return data types can be placed within expressions. Some other functions, like putchar, do not return data types.

The functionality of input and output in C is not a standard part of the programming language, rather, needs to be included at the top of the program as a header file. You recall from our previous lesson on C header files, that the stdio.h and conio.h files are used for input and output. Essentially, the input output functions are called from within your program.

This table shows the specifiers that you can use in c and what they can do

Format Specifier	Type
%c	Character
%d	Signed integer
%e or %E	Scientific notation of floats
%f	Float values
%g or %G	Works the same as %e or %E
%hi	Signed integer (short)
%hu	Unsigned Integer (short)
%i	Integer
%l or %ld or %li	Long
%lf	Double
%Lf	Long double
%lu	Unsigned int or unsigned long
%lli or %lld	Long long
%llu	Unsigned long long
%o	Octal representation
%p	Pointer
%s	String
%u	Unsigned int
%x or %X	Hexadecimal representation
%n	Does not print anything
%%	Prints % character

Ground rules

As with all other aspects of c programming, there are rules that you need to adhere to so that your program correctly runs every time.

More in the demo

Formatting

A lot of times, displaying text in a straight line is not always optimal, or just doesn't look good. This is where we would include formatting symbols. The formal name for those is escape sequences these allow you to display text in different ways. Below is a table that summarises this.

Escape sequence	Functions
\a	Text along with an alarm or beep to alert the user
\b	Inserts a backspace
\e	Escape character
\f	Form feed page break
\n	Inserts a new line
\r	Carriage return. Used to position the cursor to the beginning of the current line.
\t	Inserts a horizontal tab to the right displayed text
\v	Inserts a vertical tab on the displayed text
\\\	This is how you display a backslash
\'	This is how you display a single quote
\\"	This is how you display double quotes
\?	This is how you display a question mark
\ooo	Octal number
\xhh	Hexadecimal number
\0	Null

Input

C allows the programmer to use input from a wealth of sources ranging from the keyboard, file and devices among others. As previously mentioned, the scanf function is used to receive input from the keyboard.

Input syntax

Scanf vs gets

Illustrated in table and covered in demo

Output

As we discussed earlier output is handled by the printf function.

Output syntax

The symbol “%” denotes the start of a format mark. The mark “%d” is replaced by the value of variable counter and the string that follows is printed. If you do not add any of the special formatting that we discussed earlier, your output will be in the default format, which is aligned to the right., that is, is placed on the right edge of the field and, by default, the width of the field is the same width as the string length.

Examples

Now for the part we have all been waiting for. To get an understanding of what the various formatting symbols do, we will start off with a simple hello world, written in different ways

We will start off with an example. Like last time, we will state the problem, just like we learnt in module 1. We will then create a flowchart and write pseudocode, then write code. It might seem like a lot of work, especially with these short programs, but you are going to quickly realise that for large programs, it is very difficult to just sit on an IDE and start writing code. Enough with the jibber-jabber, let's get coding!

We want to create a program that will read input from the keyboard in various forms, then displays it to the user using the escape sequences that we learnt about today.

Problem: a school asks you to create a module in a program that asks the user for the name and surname of a student, and marks for Maths, English, and Science. The program will calculate the average mark for the student then display the student details and average mark vertically

Conclusion

In this lesson, we have covered the basics of input and output. This is one of the crucial components in programming, after all, a program wouldn't be of much use if it cannot accept input or produce output, right? We learnt how to display output in fun new ways, not the old boring one line text that we were used to. In our next lesson, we will wrap up module 2 with operators, then we dive into even deeper water as we continue our journey to become programming ninjas! That's all from me today, happy programming and remember, keep practising!

References

DataFlair. (2019). 15 Types of Escape Sequence in C that Make your Coding Better - DataFlair. [online] Available at: <https://data-flair.training/blogs/escape-sequence-in-c/>

GeeksforGeeks. (2018). Format specifiers in C - GeeksforGeeks. [online] Available at: <https://www.geeksforgeeks.org/format-specifiers-in-c/>

Mit.edu. (2020). Tips on printf. [online] Available at: http://web.mit.edu/10.001/Web/Course_Notes/c_Notes/tips_printf.html

Study.com. (2020). Basic Input & Output in C Programming | Study.com. [online] Available at: <https://study.com/academy/lesson/basic-input-output-in-c-programming.html>

Surabhi Saxena (2012). Data Input and Output in C, Part 1 - SitePoint. [online] Sitepoint.com. Available at: <https://www.sitepoint.com/data-input-and-output-in-c-part-1/>

Surabhi Saxena (2012). Data Input and Output in C, Part 1 - SitePoint. [online] Sitepoint.com. Available at: <https://www.sitepoint.com/data-input-and-output-in-c-part-1/>

Tutorialspoint.com. (2019). Format specifiers in C. [online] Available at: <https://www.tutorialspoint.com/format-specifiers-in-c#:~:text=The%20format%20specifiers%20are%20used,a%20list%20of%20format%20specifiers.>