Diploma in

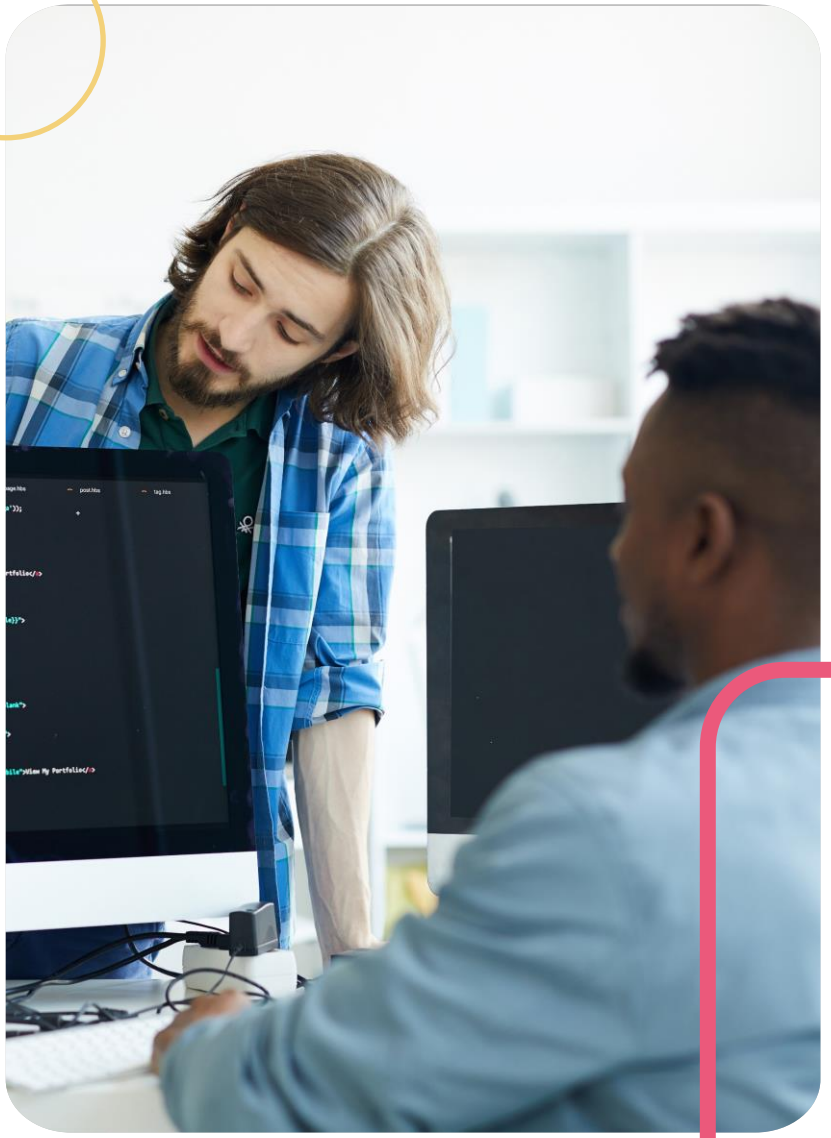
Computer Science

C Syntax

Lesson Challenge Feedback

Analyse the demo from our last lesson – the “Hello world’ demo – and try to see which sections match the basic structure of a C program.

Link section → #include<stdlib.h>
Main function → main()
→ return



Objectives

- Explore the syntax of the C programming language
- Discuss the uses of various features of the language
- Identify the keywords in C
- Explore the correct way of including comments in code

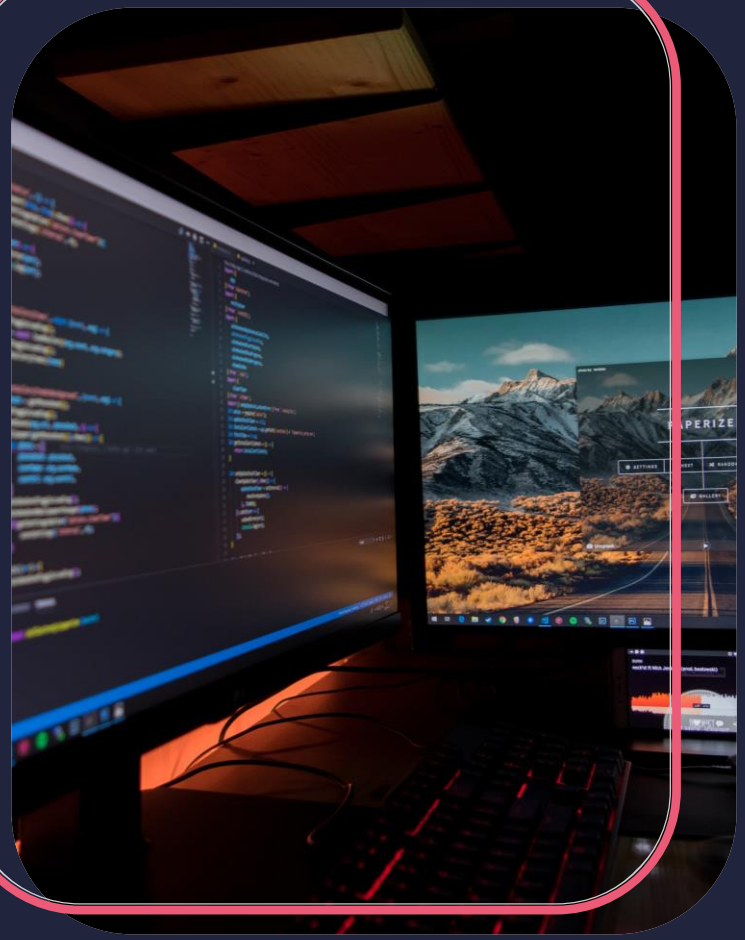
Tokens

The C character set

Letters	
	Uppercase characters (A-Z)
	Lowercase characters (a-z)
Numbers	
	Digits from 0 to 9
	White space
	New line
	Carriage return
	Horizontal tab

Special characters	
,	(comma)
{	(opening curly bracket)
.	(period)
}	(closing curly bracket)
;	(semi-colon)
[	(left bracket)
:	(colon)
]	(right bracket)
?	(question mark)
(	(opening left parenthesis)
'	(apostrophe)
)	(closing right parenthesis)
"	(double quotation mark)
&	(ampersand)
!	(exclamation mark)

Special characters	
^	(caret)
|	(vertical bar)
+	(addition)
/	(forward slash)
-	(subtraction)
\	(backward slash)
*	(multiplication)
~	(tilde)
/	(division)
_	(underscore)
>	(greater than or closing angle bracket)
$	(dollar sign)
<	(less than or opening angle bracket)
%	(percentage sign)
#	(hash sign)



Keywords

- Words that have a specific, fixed meaning in a programming language
- Cannot under any circumstances be used to refer to anything else within the programming language
- 32 keywords in C, all of which are written in lower case

auto	Declares automatic variables. Variables declared within function bodies are automatic by default. They are recreated each time a function is executed.
break	Terminates the innermost loop immediately when it's encountered. It's also used to terminate the switch statement.
case	Used when a block of statements has to be executed among many blocks. This is used in conjunction with switch and default.
char	Declares a character variable.
const	Used to declare constants.
continue	Generally used with for, while and dowhile loops, when the compiler encounters this statement it performs the next iteration of the loop, skipping rest of the statements of current iteration.

default	Used when a block of statements has to be executed among many blocks. This is used in conjunction with case and switch. The statement under default is executed if all other conditions are not met.
do	Used in the while loop for conditional execution.
double	Used for declaring floating type variables.
else	Used in the if statement for conditional execution.
enum	Used to declare enumerated variables.
extern	Declares that a variable or a function has external linkage outside of the file it is declared.
short	Used in conjunction with int for small integers from -32768 to 32767.
for	Used for looping code.
goto	Used to transfer control of the program to the specified label.

if	Used for conditional execution.
int	Used to declare integer type variables.
long	Used in conjunction with int for long integers, from -2147483648 to 214743648.
register	Creates register variables which are much faster than normal variables.
return	Terminates the function and returns the value.
float	Used for declaring floating type variables.
signed	Used in conjunction with int for normal integers, from -32768 to 32767.
sizeof	Evaluates the size of data in a variable or a constant.
static	Creates a static variable. The value of the static variables persists until the end of the program.

struct	Used for declaring a structure. A structure can hold variables of different types under a single name.
switch	Used when a block of statements has to be executed among many blocks. This is used in conjunction with case and default.
typedef	Used to explicitly associate a type with an identifier.
union	Used for grouping different types of variables under a single name.
unsigned	Used in conjunction with int for positive integers, from 0 to 65535.
void	Means no value is passed or evaluated.
volatile	Used for creating volatile objects. A volatile object can be modified in an unspecified way by the hardware.
while	Used to execute a block of code repeatedly.

Basic usage of keywords

if, else, switch, case, default, for, while, do, break: used for conditional execution

int, float, char, double, long: data types used during variable declaration

void – a return type

goto – redirects flow of execution

auto, signed, const, extern, register, unsigned – used for returning a variable

return – used for returning a value

continue – used to move to the next iteration

enum – set of constants

sizeof – used to evaluate the size of a variable or constant

struct, typedef – used to define data structures

union – collection of variables

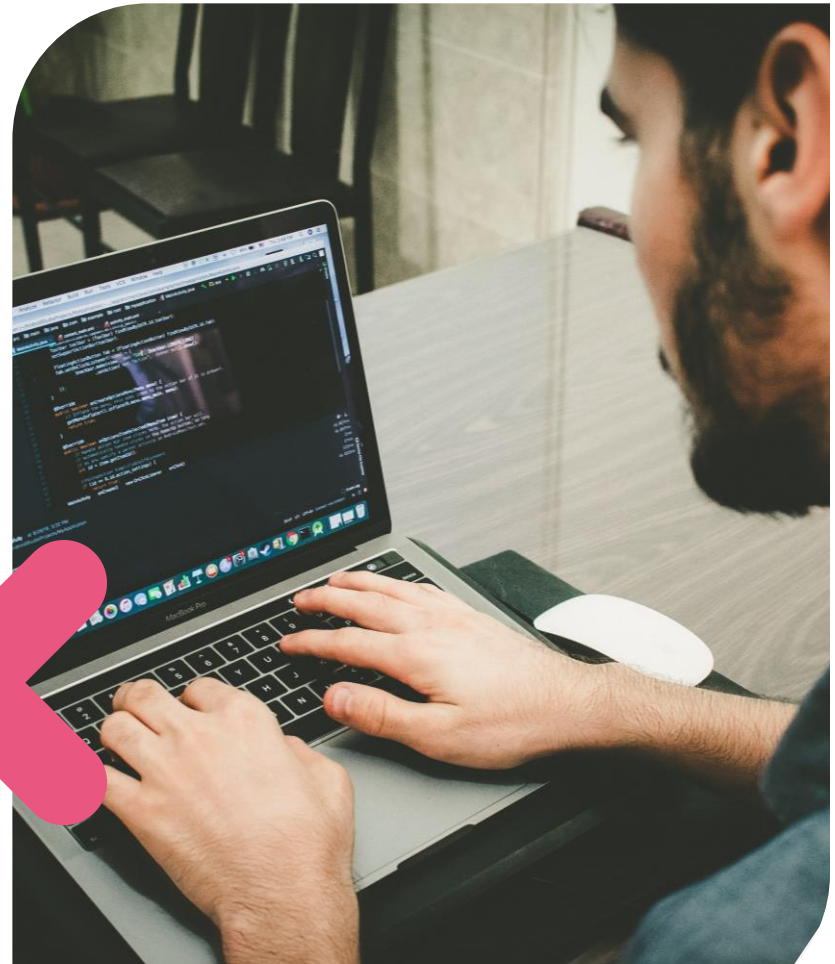
volatile – used for creating volatile objects

C is case sensitive!

Identifiers - assigned to variables, functions, array and structures

Characteristics of identifiers

- First character: either a letter of the alphabet or an underscore
- No special characters, keywords or white space
- Word limit of 31 characters
- Case sensitive



Variables

- Symbols which functions as a placeholder for varying expression or quantities
- Represent an actual block of memory
- Hold the information you will be processing
- Can update them as program that runs

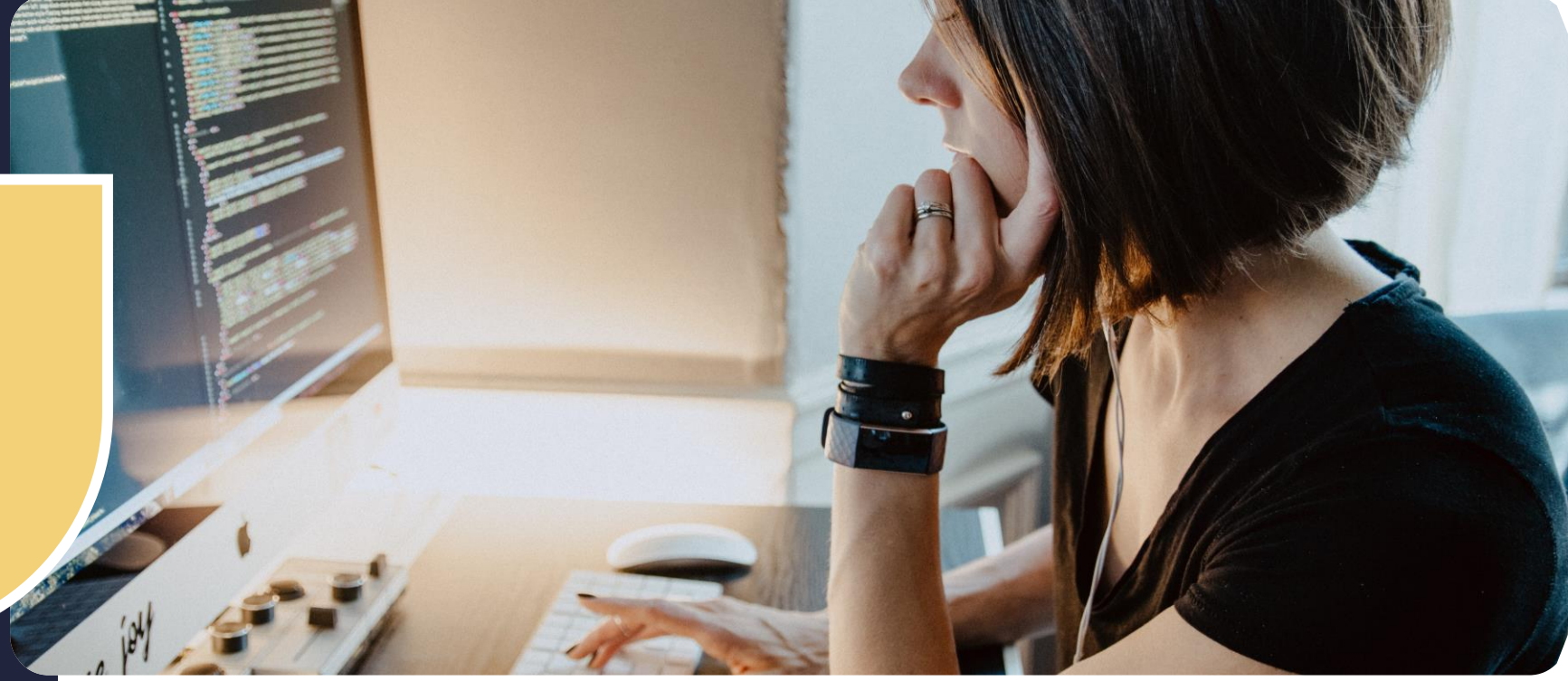
Constants

Do not change once defined

You assign a block of memory to a constant and give it a value

For example: 22/7

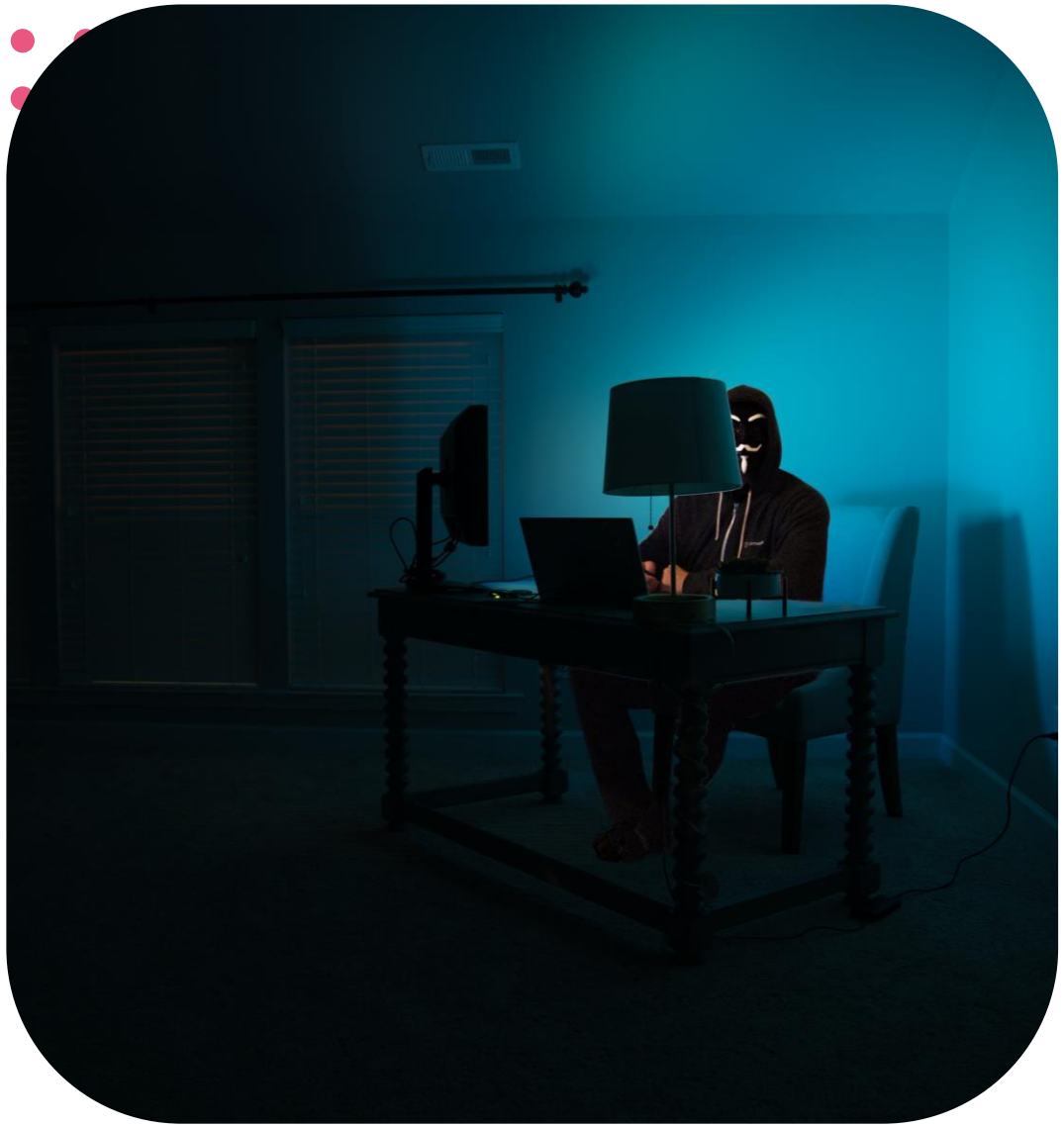
Did you know?



About 95% of world's developers started their career from C programming.

Punctuators are symbols that have syntactic and semantic meaning to the compiler.

Punctuator	Usage	Example
< >	Header name	#include <stdio.h>
[ ]	Array delimiter	char a[4];
{ }	Initialiser list, function body, or compound statement delimiter	char a[4] = {'S', 'h', 'a', 'w' };
( )	Function parameter list delimiter; also used in expression grouping	int a (x,y)
*	Pointer declaration	int *a;
,	Argument list separator	char a[4] = { 'S', 'h', 'a', 'w'};
:	Statement label	Label a: if (x == 0) x += 1;
=	Declaration initialiser	char a[4] = { "Shaw" };
;	Statement end	x += 1;
...	Variable- length argument list	int a (int y, ...)
#	Preprocessor directive	#include "shapes.h"
' '	Character constant	char a = 'a';
" "	String literal or header name	char a[] = "Shaw";



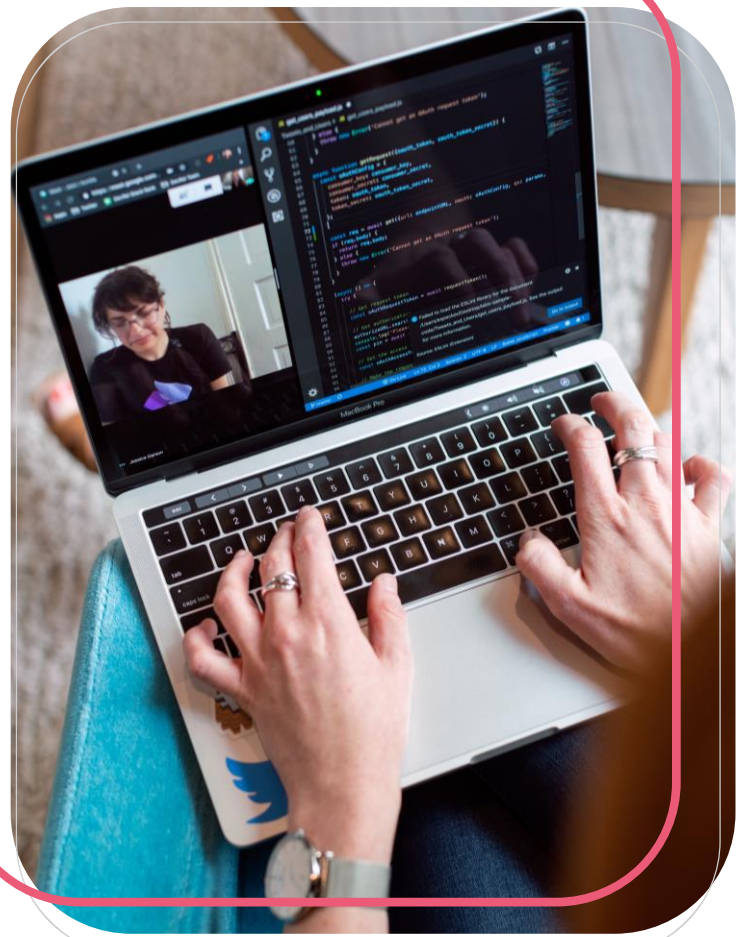
Semicolon ;

- Signifies end of a statement
- Referred to as a ‘terminator’

Operators

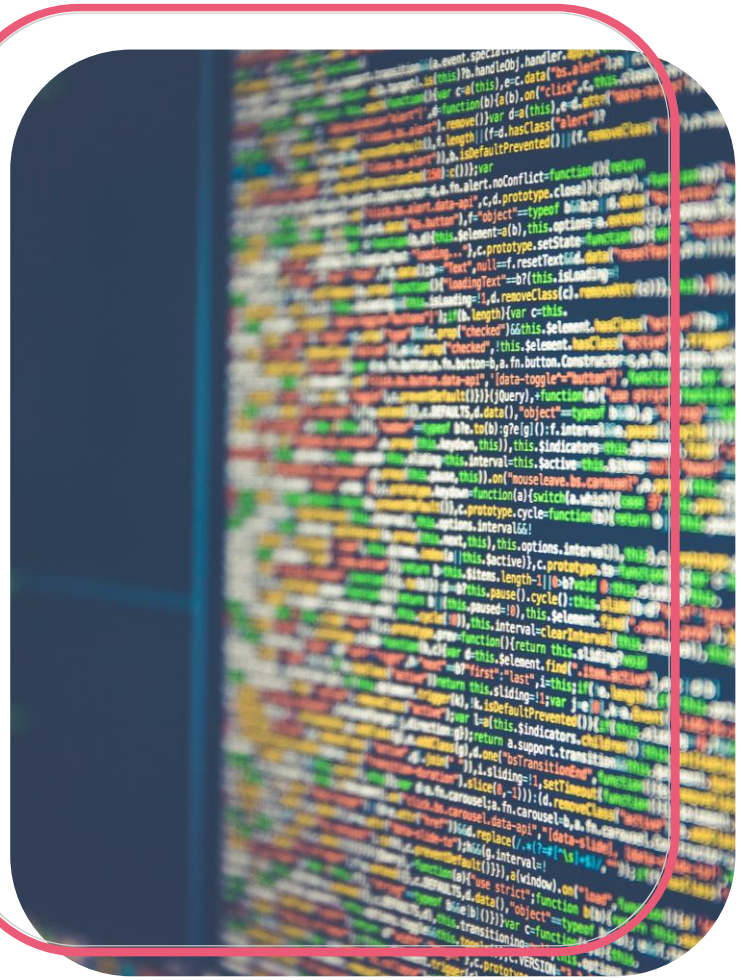
Symbols that tell the computer to perform a specific operation

- Arithmetic operators
- Relational operators
- Logical operators
- Bitwise operators
- Assignment operators



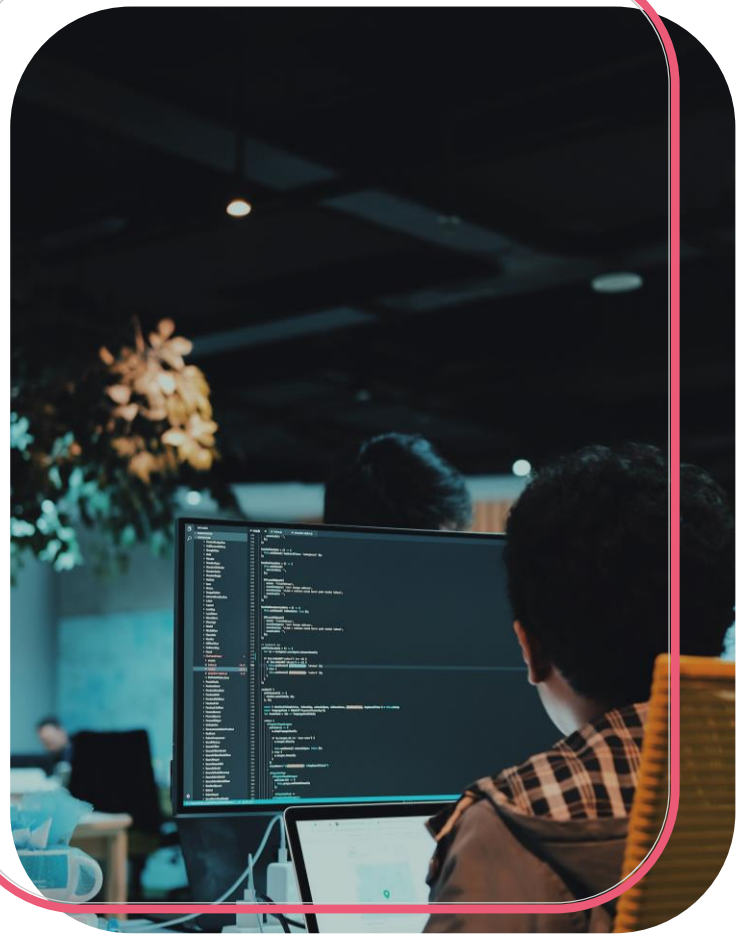
Arithmetic operators

Operator	Description
+	Adds two operands
-	Subtracts second operand from the first
*	Multiplies both operands
/	Divides numerator by de-numerator
%	Modulus operator and remainder or after an integer division
++	Increments operator increases integer value by one
--	Decrements operator decreases integer value by one



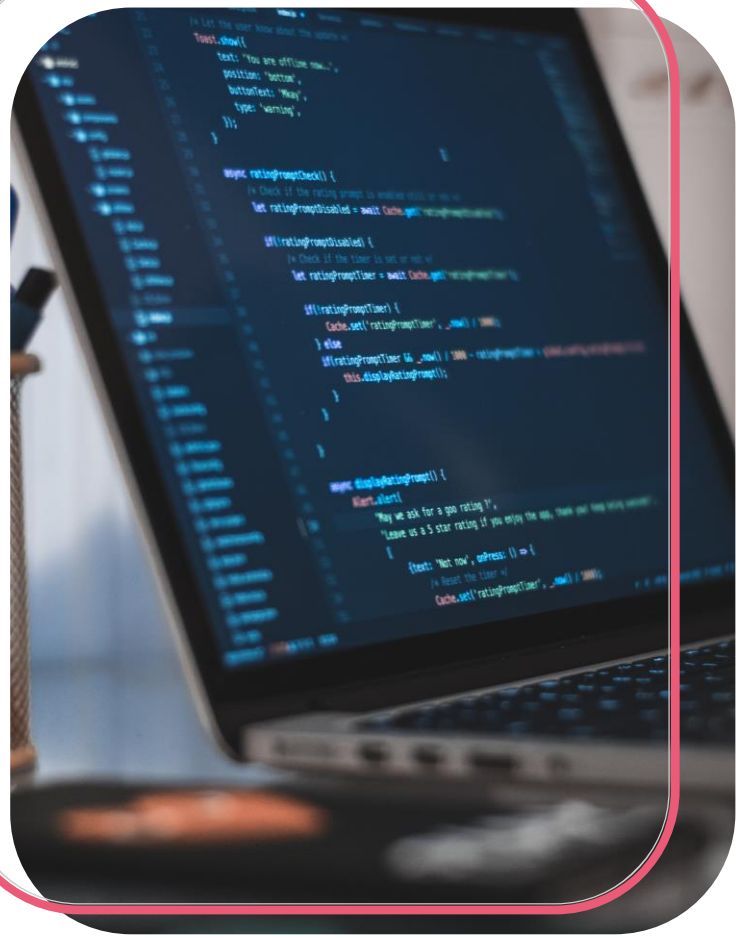
Relational operators

Operator	Usage
==	Used to check if two operands are equal
!=	Used to check if two operands are not equal
>	Used to check if the operand on the left is greater than the operand on the right
<	Used to check if the operand on the left is smaller than the operand on the right
>=	Used to check if the operand on the left is greater than or equal to the operand on the right
<=	Used to check if the operand on the left is smaller than or equal to the operand on the right



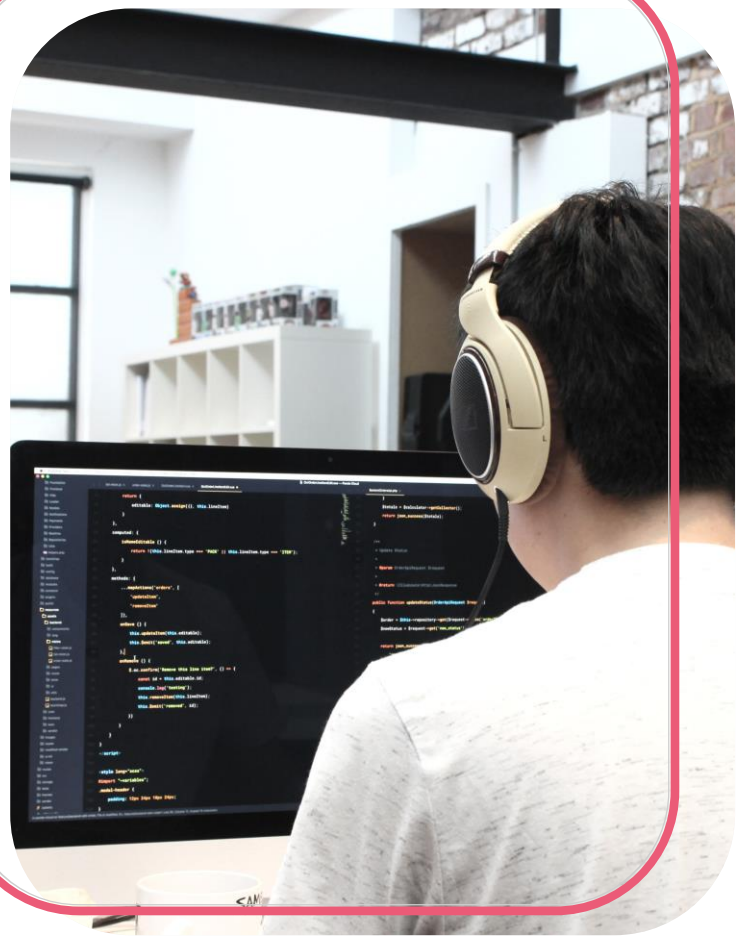
Logical operators

Operator	Usage
&&	Logical AND
||	Logical OR
!	Logical NOT



Bitwise operators

Operator	Usage
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	left shift
>>	right shift



Assignment operators

Operator	Description
=	Whatever is on the right side is copied into the operand on the left
+=	Whatever is on the right side is added into the operand on the left and the result is stored in the operand on the left
-=	Whatever is on the right side is subtracted the operand on the left and the result is stored in the operand on the left
*=	Whatever is on the right side is multiplied by the operand on the left and the result is stored in the operand on the left
/=	Whatever is on the right side is divided by the operand on the left and the result is stored in the operand on the left
%=	Calculates the modulus using two operands and assigns the result to left operand

Other operators

Operator	Description
sizeof	Returns the size of a variable
&	Returns the address of a variable
*	Pointer to a variable

Syntax categorisation

Functions

- Set of statements that take inputs, do specific computation and produce output
- Comprises a return type, a function name, arguments and code

Functions

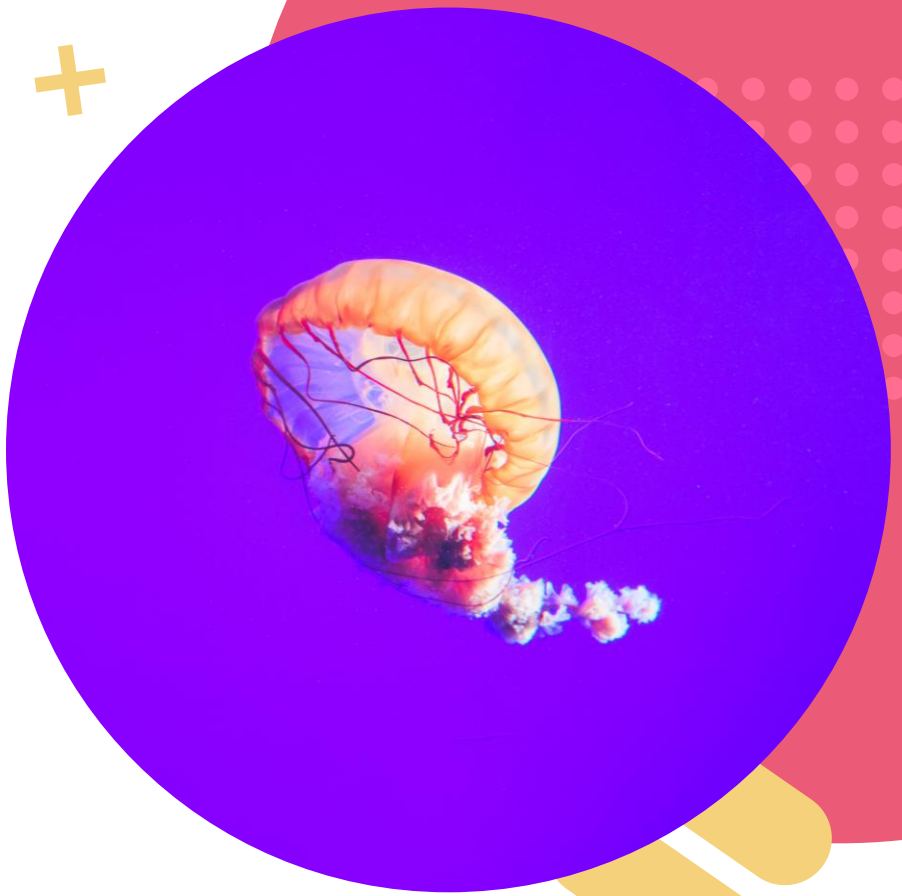
return_type function_name([arg1_type arg1_name, ...]) { code }

int main(void){}

Flow control

Use control structures to execute a block of code only when certain conditions

- if structure
- case structure
- while structure
- for structure



if structure

```
If(condition){
code}
else{
code};
```

case statement

```
switch(expression) {

  case constant-expression  :
    statement(s);
    break;

  case constant-expression  :
    statement(s);
    break;
   default :
  statement(s);
}
```

while structure

```
while(condition) {
  statement(s);
}
```

for statement

```
for ( initialCondition; FinalCondition; increment ) {
  statement(s);
}
```

Data types

Type	Explanation	Minimum size (bits)	Format specifier
char	Smallest addressable unit of the machine that can contain a basic character set. It is an integer type	8	%c
signed char	Of the same size as char, but guaranteed to be signed	8	%c (or %hhi for numerical output)
unsigned char	Of the same size as char, but guaranteed to be unsigned	8	%c (or %hhu for numerical output)
short short int signed short signed short int	Short signed integer type	16	%hi or %hd
unsigned short unsigned short int	Short unsigned integer type	16	%hu
int signed signed int	Basic signed integer type	16	%i or %d

Type	Explanation	Minimum size (bits)	Format specifier
unsigned unsigned int	Basic unsigned integer type	16	%u
long long int signed long signed long int	Long signed integer type	32	%li or %ld
unsigned long unsigned long int	Long unsigned integer type	32	%lu
long long long long int signed long long signed long long int	Long long signed integer type	64	%lli or %lld
unsigned long long unsigned long long int	Long long unsigned integer type	64	%llu

Type	Explanation	Minimum size (bits)	Format specifier
float	Real floating-point type, usually referred to as a single-precision floating-point type		Converting from text: %f %F %g %G %e %E %a %A
double	Real floating-point type		%lf %lF %lg %lG %%la %lA
long double	Real floating-point type, usually mapped to an extended precision floating-point number format		%Lf %LF %Lg %LG %Le %LE %La %LA

Extras

Comments make code more readable

multiline comments

Use a forward slash followed by an asterisk (/*) then close off the comment with an asterisk and a forward slash (*/)

In-line comments

- Indicated by a double forward slash//
- Do not need to be closed

You can include code in comments.

TODO comment:

- Implemented but does not yet work as expected

  OR

- Was left out so that it would be attended to at a later time

Case sensitivity

- Don’t name your variables too similarly
- Code must be readable

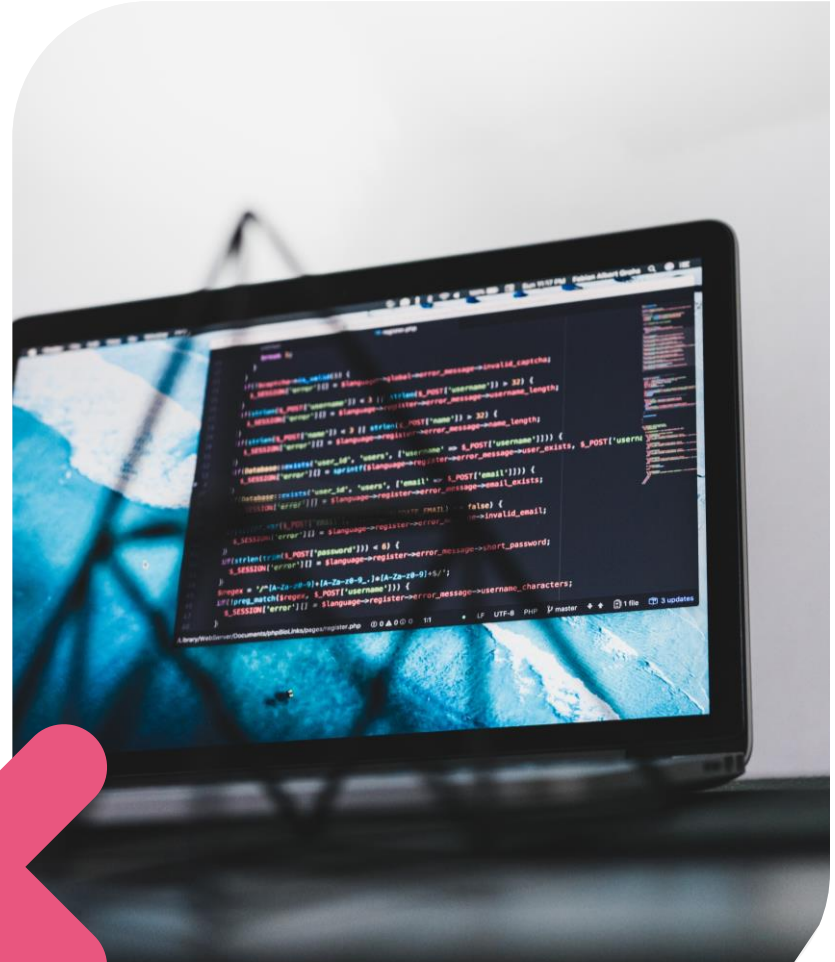
White space = blank line

Makes code readable for humans

Refers to blanks, tabs, newline characters and comments

Enables compiler to identify where one element in a statement ends and next begins

```
int num1
```



A few side notes...

Learning a language

Learning code is like learning to speak a new language.

The more you write , the more comfortable you will become with various syntax and semantics.

Quality code: creating faster and better instructions for a computer.

Practice makes perfect

- Good writers interact with language intricately.
- Programming can be frustrating.
- Every single thing you type has an effect.

‘Code and cats’

- There are many algorithms that are endorsed as the best way to solve a problem.
- Learning the syntax of programming language is just one tool in Computer Science.
- There are other tools to solve CS problems.