

C programming notes quick revision

70-day challenge

1. Character Set

- A character set is a collection of symbols that are used to represent text. In the context of programming languages, it refers to the set of characters that the language can recognize. For example, C uses the ASCII (American Standard Code for Information Interchange) character set.
- ASCII Table: ASCII is a 7-bit character encoding standard that represents text in computers. The ASCII table includes 128 characters (0-127) such as digits, letters, punctuation, and control characters.

2. ASCII TABLE 😊

Decimal	Hex	Character	Description
0	0	NUL	Null character
1	1	SOH	Start of Header
2	2	STX	Start of Text
3	3	ETX	End of Text
4	4	EOT	End of Transmission
5	5	ENQ	Enquiry
6	6	ACK	Acknowledge
7	7	BEL	Bell
8	8	BS	Backspace
9	9	TAB	Horizontal Tab
10	0A	LF	Line Feed
11	0B	VT	Vertical Tab
12	0C	FF	Form Feed
13	0D	CR	Carriage Return
14	0E	SO	Shift Out
15	0F	SI	Shift In
16	10	DLE	Data Link Escape
17	11	DC1	Device Control 1
18	12	DC2	Device Control 2
19	13	DC3	Device Control 3
20	14	DC4	Device Control 4
21	15	NAK	Negative Acknowledge
22	16	SYN	Synchronous Idle
23	17	ETB	End of Block

C programming notes quick revision

70-day challenge

24	18	CAN	Cancel
25	19	EM	End of Medium
26	1A	SUB	Substitute
27	1B	ESC	Escape
28	1C	FS	File Separator
29	1D	GS	Group Separator
30	1E	RS	Record Separator
31	1F	US	Unit Separator
32	20	(Space)	Space
33	21	!	Exclamation mark
34	22	"	Double quote
35	23	#	Hash symbol
36	24	\$	Dollar sign
37	25	%	Percent
38	26	&	Ampersand
39	27		Single quote
40	28	(Left parenthesis
41	29)	Right parenthesis
42	2A	*	Asterisk
43	2B	#ERROR!	Plus sign
44	2C	,	Comma
45	2D	-	Hyphen
46	2E	.	Period
47	2F	/	Slash
48	30	0	Zero
49	31	1	One
50	32	2	Two
51	33	3	Three
52	34	4	Four
53	35	5	Five
54	36	6	Six
55	37	7	Seven
56	38	8	Eight
57	39	9	Nine
58	3A	:	Colon
59	3B	;	Semicolon

C programming notes quick revision

70-day challenge

60	3C	<	Less-than symbol
61	3D	#ERROR!	Equal sign
62	3E	>	Greater-than symbol
63	3F	?	Question mark
64	40	@	At symbol
65	41	A	Uppercase A
66	42	B	Uppercase B
67	43	C	Uppercase C
68	44	D	Uppercase D
69	45	E	Uppercase E
70	46	F	Uppercase F
71	47	G	Uppercase G
72	48	H	Uppercase H
73	49	I	Uppercase I
74	4A	J	Uppercase J
75	4B	K	Uppercase K
76	4C	L	Uppercase L
77	4D	M	Uppercase M
78	4E	N	Uppercase N
79	4F	O	Uppercase O
80	50	P	Uppercase P
81	51	Q	Uppercase Q
82	52	R	Uppercase R
83	53	S	Uppercase S
84	54	T	Uppercase T
85	55	U	Uppercase U
86	56	V	Uppercase V
87	57	W	Uppercase W
88	58	X	Uppercase X
89	59	Y	Uppercase Y
90	5A	Z	Uppercase Z
91	5B	[Left square bracket
92	5C	\	Backslash
93	5D]	Right square bracket
94	5E	^	Caret (circumflex)
95	5F	_	Underscore

C programming notes quick revision

70-day challenge

96	60	`	Grave accent
97	61	a	Lowercase a
98	62	b	Lowercase b
99	63	c	Lowercase c
100	64	d	Lowercase d
101	65	e	Lowercase e
102	66	f	Lowercase f
103	67	g	Lowercase g
104	68	h	Lowercase h
105	69	i	Lowercase i
106	6A	j	Lowercase j
107	6B	k	Lowercase k
108	6C	l	Lowercase l
109	6D	m	Lowercase m
110	6E	n	Lowercase n
111	6F	o	Lowercase o
112	70	p	Lowercase p
113	71	q	Lowercase q
114	72	r	Lowercase r
115	73	s	Lowercase s
116	74	t	Lowercase t
117	75	u	Lowercase u
118	76	v	Lowercase v
119	77	w	Lowercase w
120	78	x	Lowercase x
121	79	y	Lowercase y
122	7A	z	Lowercase z
123	7B	{	Left curly brace
124	7C		Vertical bar (pipe)
125	7D	}	Right curly brace
126	7E	~	Tilde
127	7F	DEL	Delete (Non-printable)

Key Points:

C programming notes quick revision

70-day challenge

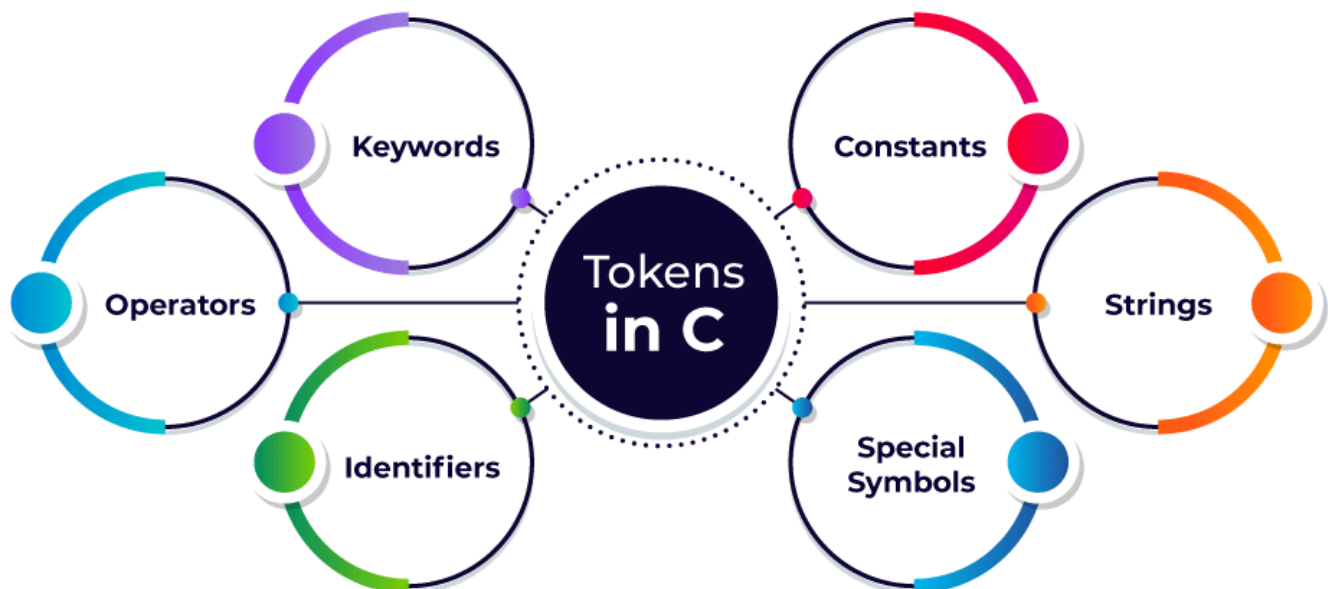
The first 32 characters (0-31) are control characters (non-printable), often used for text formatting and communication protocols.

Characters from 32-126 include printable characters, like numbers, letters, punctuation, and symbols.

The last character (127) is DEL, which is used to delete a character in text processing.

3. Tokens

- In programming, a token is a basic unit of a program, such as a keyword, identifier, constant, or operator. The process of breaking a program into tokens is known as lexical analysis.
- Tokens can include:
 - Keywords: Reserved words that have special meaning (e.g., **if**, **else**, **while**).
 - Identifiers: Names used to identify variables, functions, etc.
 - Constants: Values that do not change.
 - Operators: Symbols like **+**, **-**, *****, etc.
 - Punctuation: Symbols like **;**, **{**, **}**, etc.



C programming notes quick revision

70-day challenge

4. Keywords

- **Keywords are reserved words in a programming language that cannot be used as identifiers (names for variables, functions, etc.). They are predefined by the language and have special meaning.**

Keyword	Description	
auto	Used to define automatic variables (local variables by default).	
break	Exits from a loop or a switch statement.	
case	Marks a branch in a switch statement for specific values.	
char	Used to declare a variable of type char, which stores a single character.	
const	Specifies that the value of a variable cannot be changed.	
continue	Skips the current iteration of a loop and proceeds to the next iteration.	
default	Specifies the default case in a switch statement when no case matches.	
do	Used to start a do-while loop, which executes the loop body first and then checks the condition.	
double	Declares a variable of type double, used for floating-point numbers with double precision.	
else	Specifies the block of code to execute if the if condition is false.	
enum	Defines an enumerated data type, which represents a set of named integer constants.	
extern	Declares a variable or function that is defined outside the current file (external linkage).	
float	Declares a variable of type float, used for single-precision floating-point numbers.	
for	Used to start a for loop, which allows repeated execution of a block of code.	
goto	Transfers control to a labeled statement in the program.	
if	Used to start a conditional statement that executes a block of code if a condition is true.	
inline	Suggests to the compiler to insert the function's code at the point of the function call.	
int	Declares a variable of type int, used for integer values.	
long	Declares a variable of type long, used for long integers (larger range).	
register	Suggests that the variable be stored in a CPU register for faster access (though modern compilers often ignore this).	
return	Exits from a function and optionally returns a value to the caller.	
short	Declares a variable of type short, used for short integer values.	
signed	Specifies that a variable can store both positive and negative values (default for integers).	
sizeof	Returns the size (in bytes) of a data type or object.	

C programming notes quick revision

70-day challenge

static	Specifies that a variable has local scope but retains its value between function calls. Also used for limiting visibility of functions/variables within the file.	
struct	Defines a structure, which is a collection of different data types grouped together.	
switch	Starts a switch statement, which is used for multi-way branching based on the value of a variable.	
typedef	Creates a new name (alias) for an existing data type.	
union	Defines a union, which is a data structure where all members share the same memory location.	
unsigned	Specifies that a variable can only store non-negative values (i.e., no sign bit).	
void	Specifies a function that does not return a value or indicates that a pointer is of unknown type.	
volatile	Indicates that a variable's value can change unexpectedly, often used in hardware or signal handling.	
while	Starts a while loop, which executes the block of code as long as the condition is true.	

Notes:

- **auto**: By default, local variables in C are **auto** variables, meaning they are stored in the stack. The **auto** keyword is rarely used explicitly.
- **register**: This is a hint to the compiler to try to store the variable in a CPU register for faster access. However, modern compilers optimize this automatically, and its usage is less important today.
- **typedef**: This keyword is used to create aliases for data types. For example, **typedef unsigned long ulong**; allows you to use **ulong** instead of **unsigned long**.
- **volatile**: This keyword is essential when dealing with variables that can be modified outside the program flow (e.g., hardware registers or variables in interrupt routines).

C programming notes quick revision

70-day challenge

4. Identifiers & Naming Rules

- Identifiers are names given to various program elements such as variables, functions, arrays, etc.
- Naming Rules for identifiers (specific to C or C-like languages) include:
 - An identifier must start with a letter (A-Z or a-z) or an underscore (_).
 - After the first character, it can contain letters, digits (0-9), or underscores.
 - It must not be a keyword.
 - C identifiers are case-sensitive (e.g., **Variable** and **variable** are different).
 - They cannot start with a number.

5. Constants

- Constants are values that do not change throughout the execution of a program.
- Types of Constants:
 - Integer constants: E.g., **100**, **-50**
 - Floating-point constants: E.g., **3.14**, **-0.001**
 - Character constants: E.g., **'A'**, **'b'**
 - String constants: E.g., **"Hello, World!"**
- Constants can be declared in C using **#define** (preprocessor directive) or **const** keyword
- Source code

```
#define PI 3.14 // Preprocessor constant
```

```
const int max_size = 100; // Constant variable
```

6. Type Qualifiers

- Type qualifiers are keywords in C/C++ that modify the behavior or properties of data types.

C programming notes quick revision

70-day challenge

- Common type qualifiers:
 - **const**: Specifies that a variable's value cannot be changed after it is initialized.
 - **volatile**: Specifies that the value of a variable can change unexpectedly, often used with hardware-level programming or when dealing with memory-mapped registers.

Source code **+**

```
const int x = 10; // x cannot be modified
```

```
volatile int flag; // flag can be modified unexpectedly (e.g., hardware interrupt)
```