

Unit 2

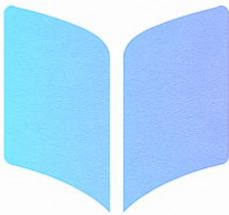
C Language

Introduction of C Programming

1. C is a programming language developed at AT & T's Bell laboratories of USA in 1972.
2. It was designed & written by Dennis Ritchie.
3. It is initially designed for operating System UNIX.

Structure of C Program

```
Comments (Documentation) – (optional)
Preprocessor directives (Link or Definition) – (compulsory)
Global variables declaration – (optional)
main () – (compulsory)
{
Local Variable declaration;
Statements;
-----
-----
}
Func1() – (optional)
{
Local Variable declaration;
Statements;
-----
-----
}
```



Comments

It is used for documentation purpose or understanding purpose.

Syntax: `/* ----- comment ----- */`

Preprocessor Directives

They are used for linking of file or data to our program.

Types:

- (a) **#include**: It is used to include or add header file or library file.

- Syntax: `#include < header_file >`

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

- (b) **#define**: It is used to define macro name & macro expansion. macro is the code, which will execute in program again & again.

- Syntax: `#define macro_name macro_expansion`

E.g. `#define P printf("hello")`

It is also used to define constant.

- Syntax: `#define constant value`

E.g. `#define PI 3.143`

main () Function

This is the place where the execution of the C program begins. Without this function, your C program cannot execute. Next comes the opening brace '{', which indicates the beginning of the function. The closing brace '}' indicates the end of the function.

Local Variable

Scope of these variables is only within the function or body in which they declared.

Global Variable

Scope of these variables is within whole program.

Func1()

This is user defined function or subprogram, used to perform subtask during execution of program. It is optional to used.

Rules of Writing C Program

- C is a case sensitive language → ABC, Abc, abc etc.
- All statement is terminated with semi colon (;).
E.g. `a = 10; printf (" ");`
- All C program is a collection of function and it contains at least one function called main () function
- Declaration of all variable is must before using them.

Special Symbol

,	Comma	&	Ampersand
.	Period (dot)	^	Caret
;	Semicolon	*	Asterisk
:	colon	-	Minus Sign
?	Question Mark	+	Plus Sign
'	Apostrophe (single quote)	<	Opening Angle (Less than sign)
"	Quotation Marks (Double quote)	>	Closing Angle (Greater than sign)
!	Exclamation Mark	(Left Parenthesis
	Vertical Bar)	Right Parenthesis
/	Forward Slash	[Left Bracket
\	Backward slash]	Right Bracket
~	Tilde	{	Left Brace
-	Underscore	}	Right Bracket
\$	Dollar Sign	#	Number Sign (Hash)
%	Percentage Sign	@	At the rate of

CHARACTER SET OF C

Character denotes any alphabet, digit or symbol or special symbol used to represent Information.

- *Alphabet*: A to Z (uppercase), A to z (lowercase)
- *Digits*: 0 to 9
- *Special symbol*: (<) less than, (>) greater than, (%) percentage, (,) comma, () blank space

KEYWORDS

The alphabets, digit, and underscore when properly combined forms Keyword.

- Keywords are the words whose meaning has already been explained to the compiler.
- Keywords are used for specific meaning.
- Keyword cannot be used as new variable name.
- There are 32 keyword available in C.

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

CONSTANTS

Any alphabet number and special symbol combined to form constant. It is the value which cannot be changed at the execution time of program.

Types of constants => 1. numeric 2. Character 3. string 4. enumerated 5. symbolic

Integer

It is number that has an integer value and do not have any floating-point value.

Types of integers: decimal (base 10), octal (base 8), and hexadecimal (base 16).

- It must have at least one digit.
- It must not have decimal point.
- It can be either negative or positive.
- No commas or blanks are allowed within an integer.
- *Range of integer:* -32768 to 32767
- E.g. 426, -123, 0, +124 etc.

Unsigned integer: negative values are not allowed means value which is greater than or equal than 0.

Long integer: Has bigger size which require more memory as compared to normal integer.

Real Constant (Float Point)

Real constant is often called floating point constant. Real constant could be written in two form which is fractional and exponential form.

Feature:

- It must have a decimal point. E.g. +3253.4, 423.0, -4.432
- A real no. may also be expressed in exponential notation.

Mantissa e exponent or mantissa E exponent

E.g. 213.65 → 2.1365e2

- In exponent notation e2 means multiply by 10².
- The mantissa is either a decimal notation or an integer but the exponent is an integer no. with a sign + or - sign.

Eg. + 25.8e5, 0.05E - 3.

- Range of floating-point constant is 3.4 e-38 to 3.4 e 38.

Character Constant

A character constant is a single alphabet, which is written in the single inverted commas. The maximum length of character constant can be one character.

E.g. 'A', '5', '=' etc

String constant

String constant is a sequence of character enclose in double quotes.

- (a) Character may be letters numbers special character and blank space.

E.g. "Samar", "143" etc.

- (b) '\0' null character automatically places in the end of string at the time of storage of the string.

Symbolic constant

A symbolic constant is a name that substitutes for a sequence of characters. This is generally written at the beginning of the program.

Method-I:

```
#define PI 3.14
#define MAX 300
#define MIN 10
```

Method-II: *const data_type constant_name = value;*

E.g. *Const float rate = 21.5;*

Escape Sequence

Some characters are used with back slash (\). This combination is used for special Function called escape sequences.

E.g. \t: horizontal tab

\n: new line

\b: back space

\0: null (used in case of string)

\r: return carriage

VARIABLES

The alphabets, digit, and underscore when properly combined forms variables.

1. It may be varied during the program execution.
2. They are names given to location in memory.
3. These locations contains integer, real or character constants.

Declaring a Variable: A variable must be declared before it is used in the program.

Syntax: *data type variable_name;*

E.g. *int x; float a, b, c; char = ch;*

Initialization of variable:

```
x = 10; y = 2.5; ch = 'r';
```

Combine statement of declaration & initialization of variable is also allowed.

```
int x = 10; float y = 2.5; char ch = 'r';
```

DATA TYPES

Data Type	Size (Typical)	Range	Example	Use Case
int	2 or 4 bytes	-32,768 to 32,767 (2 bytes)- 2,147,483,648 to 2,147,483,647 (4 bytes)	int a = 100;	Whole numbers (positive or negative)
float	4 bytes	±3.4e-38 to ±3.4e+38 (6-7 digits precision)	float b = 3.14;	Decimal numbers, scientific calculations
char	1 byte	-128 to 127 (ASCII values)	char c = 'A';	Storing single characters or letters
double	8 bytes	±1.7e-308 to ±1.7e+308 (15- 16 digits precision)	double d = 3.1415926535;	High-precision floating-point values

- The actual size of each data type may vary depending on the compiler and system architecture (e.g., 16-bit, 32-bit, or 64-bit systems).
- *double* provides higher precision than *float*, used in complex or financial calculations.
- *char* stores both characters and ASCII numeric values.

COMMENTS

Comments are placed in any program for understanding purpose of function group of line etc.

1. *Single Line Comment*: This started with double slash `"/"`.

E.g. `// Hello Samar //`

2. *Multi Line Comment*: This comment enclosed by `/*` and `*/`

E.g. `/* hello samar */`

`/* a = 3; B = a + c; C = a/b; */`

OPERATORS

Operator is a symbol that tells the computer to perform certain mathematical or logical manipulations.

1. Arithmetic Operator

- (a) *Unary Arithmetic*: It requires only one operand.

E.g. `+a`, `-b`, changes the sign of the operands.

- (b) *Binary Arithmetic*: It requires two operands.

`+` plus addition

`-` minus subtraction

`*` asterisk multiplication

`/` slash division

`%` modulus it gives remainder after division

- (c) *Integer Arithmetic*: All the operands in an expression are integer type and this expression always returns an integer value.

Expression	Result
14+5	19
12*5	60
15%7	1

- `%` operator used only with integer operands.
- when the modulo operation perform, the sign of result is always the sign of the first operand.

Expression	Result
14%5	4
-12%5	-2
-15%-7	-1
15%-7	1

- (d) *Floating Point or Real Arithmetic*: All the operands in an expression are real type and this expression always returns a real value.

Expression	Result
14.5+5.0	19.5
12.0/5.0	6.0

- (e) *Mixed Mode Arithmetic*: When one of the operands is real and the other is integer in the expression and this expression always returns a real value.

Expression	Result
14.5+5	19.5

- (f) *Priority Of Operator*:

1st `*` / `%` left to right

2nd `+` `-` left to right

3rd `=` assignment

E.g. `A * B + C/D`

1. multiplication 2. division 3. addition

2. Relational Operator

These are used for compare data values. it always returns 0 for false and 1 for true.

Operator	Name	Example	Result
<	Less than	5<6	1
<=	Less than or equal to	6<=5	0
>	Greater than	6>5	1
>=	Greater than or equal to	6>=7	0
==	Equal to	5==4	0
!=	Not equal to	5!=4	1

3. Logical Operator

These are used to check more than one condition and more decision. And always return 0 for false and 1 for true.

Operator	Name	Example	Result
&&	Logical AND	(5<6) && (4>3)	1
	Logical OR	(6<=5) (5<5)	0
!	Logical NOT	!(6>5)	0

Truth table:

Expression -I	Expression -II	&&(AND)	(OR)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Expression	! (NOT)
0	1
1	0

4. Assignment Operator

This is used for stored or assign the value to variable.

Syntax: "="

E.g. a=5; a=a+5;

Shorthand assignment statement:

$Var1\ op = var2/expression/value = Var1 = var1\ op\ var2/expression/value$

E.g. $A = A + B; \Rightarrow A += B;$

$A = A - B; \Rightarrow A -= B;$

$A = A * B; \Rightarrow A *= B;$

$A = A \% B; \Rightarrow A \% = B;$

$A = A + B + C; \Rightarrow A += B + C$

5. Increment & Decrement Operator

These are also called unary operator.

$++$ Increment, $--$ Decrement

Syntax:

$++\ Variable$ pre-increment

$--\ Variable$ pre-decrement

$Variable\ ++$ post-increment

$Variable\ --$ post-decrement

E.g. $a = 5$

Prefix: $b = ++a\ result \rightarrow a = 6 / b = 6$

$b = --a\ result \rightarrow a = 4 / b = 4$

Postfix: $b = a++\ result \rightarrow a = 6 / b = 5$

$b = a--\ result \rightarrow a = 4 / b = 5$

E.g. $a = 1, b = 2, c = 3$

$s = a++++b---c \Rightarrow a = 2, b = 3, c = 2, s = 2$

6. Bitwise Operator

These operators manipulate the data of bit level. These operation work only with the integer type data.

Operator	Name
&	Bit-wise AND
	Bit-wise OR
^	Bit-wise XOR
<<	Left shift
>>	Right shift
~	One's complement

```
int a= 12, b=5, c;
```

```
int a = 12, b = 5, c;
```

```
a = 12 -> 0000 0000 0000 1100
```

```
b = 5 -> 0000 0000 0000 0101
```

Bitwise AND: $c = a \& b$

```
a = 0000 0000 0000 1100
```

```
b = 0000 0000 0000 0101
```

```
a & b = 0000 0000 0000 0100 (4)
```

Bitwise OR: $c = a|b$

```
a|b = 0000 0000 0000 1101 (13)
```

Bitwise XOR: $c = a \wedge b$

```
a ^ b = 0000 0000 0000 1001 (9)
```

Bitwise Left Shift: $c = a << 1$

It means insert a new bit (0) into sequence of bits at right most position.

```
a << 1 = 0000 0000 0001 1000 (24)
```

Bitwise Right Shift: $c = a >> 1$

It means insert a new bit (0) into sequence of bits at left most position.

```
a >> 1 = 0000 0000 0000 0110 (6)
```

Note: The value of variable will be double if left shift is processed and half if right shift is processed.

7. Conditional Operator (Turnery Operator)

It is represented by the symbol '?' and ':'. It requires three operands so it is also called turnery operator.

Syntax: $Operand1 ? Operand2 : Operand3$

- In the conditional operation, the operand1 (expression1) is tested, if it is true then operand2 is evaluated otherwise operand3 is evaluated.

8. Special Operator

a) *Sizeof Operator:* it is used to find the size of variable or basic data type (in byte).

Syntax: $num = sizeof(variable_name);$

E.g. $num = sizeof(int);$

It gives the size of integer (2 byte)

```
num = sizeof(a);
```

b) *Comma Operator:* It is used to sequence of expression which are performed from left to right. A comma operator has the lowest precedence among all C operators.

E.g. $a = (x = 5, x + 2);$ it executes in the following order:

1. Value 5 is assigned to x.
2. X is incremented by 2.
3. The value of expression $x+2$ (7) is assigned to variable a.