BCA- 1st SEM

Introduction to Programming using C

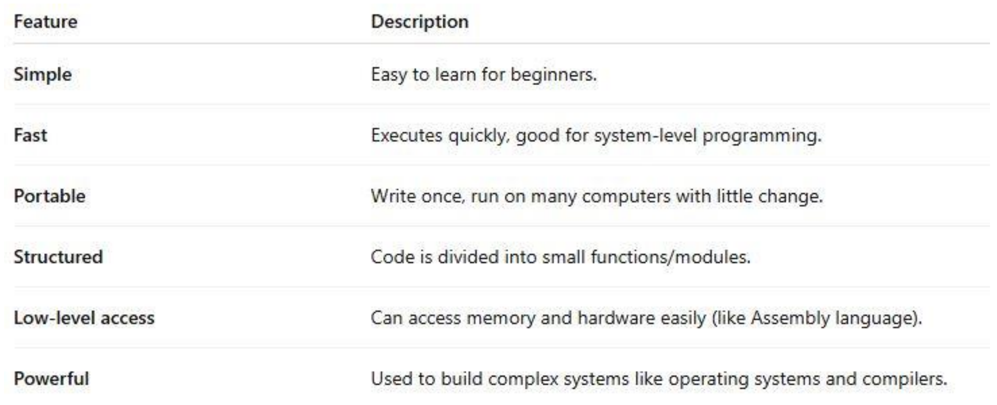
Unit-1 (Notes)

What is c Programming language?

C is a high-level programming language developed in the early 1970s by Dennis Ritchie at Bell Labs. It is one of the most popular and powerful programming languages.

C is a simple and fast programming language used to create software, apps, operating systems (like Windows or Linux), and games.

Features of C:



Feature	Description
Simple	Easy to learn for beginners.
Fast	Executes quickly, good for system-level programming.
Portable	Write once, run on many computers with little change.
Structured	Code is divided into small functions/modules.
Low-level access	Can access memory and hardware easily (like Assembly language).
Powerful	Used to build complex systems like operating systems and compilers.

Where is C used?

- Operating Systems (e.g., Linux)
- Embedded Systems
- Game Development
- Compilers and Interpreters
- Database Systems
- System Drivers

Example:

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

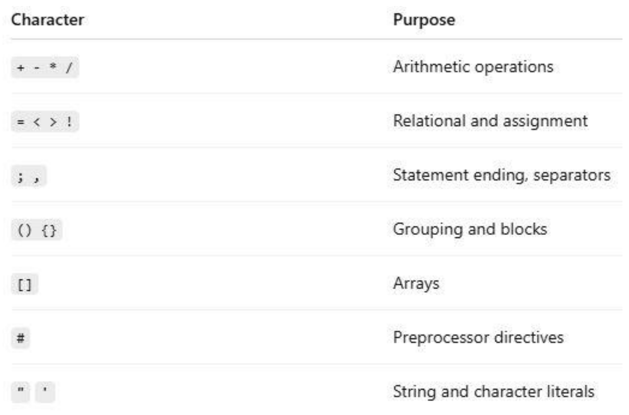
```
int main() {
   printf("Hello, World!");
   return 0;
}
```

Define C Character set:

The Character Set in C refers to all the letters, digits, and symbols that you can use to write your program. It is like the alphabet of the C language.

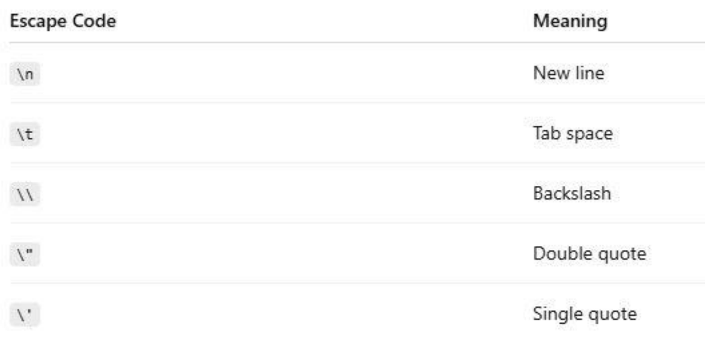
Types of Characters in C:

1. **Letters:** (Uppercase: A to Z, Lowercase: a to z)
2. **Digits:** 0 to 9
3. **Special Characters**



Character	Purpose
`+ - * /`	Arithmetic operations
`= < > !`	Relational and assignment
`; ,`	Statement ending, separators
`() {}`	Grouping and blocks
`[]`	Arrays
`#`	Preprocessor directives
`"` `'`	String and character literals

4. **Escape Sequences**



Escape Code	Meaning
`\n`	New line
`\t`	Tab space
`\\`	Backslash
`\"`	Double quote
`\'`	Single quote

What is an Identifier?

An identifier is a user-defined name given to elements in a program so the programmer can refer to them later. An identifier is the name used to identify elements like:

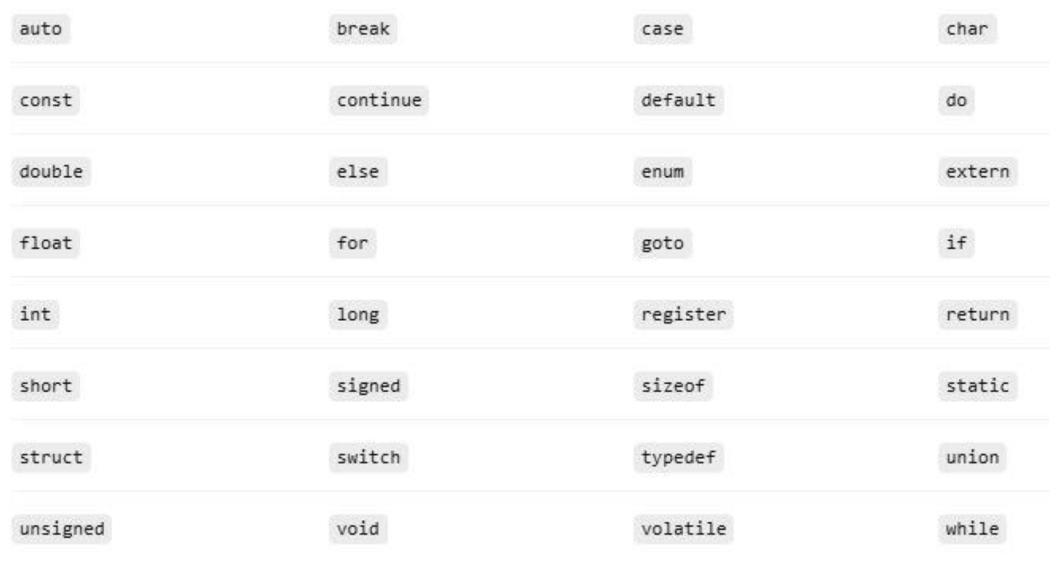
- **Variables**
- **Functions**
- **Arrays**
- **Structures**
- **Constants**

Rules for Identifiers:

a) Must begin with a letter (A-Z or a-z) or underscore (_) only
b) Cannot start with a digit
c) Cannot use **C keywords** (like int, if, else, break, switch etc.)
d) Can contain **letters, digits, and underscores only** (e.g. aAb_18c)
e) Case-sensitive (e.g. A not is equal a)
f) No special characters like @, #, $, %, etc.

What is a Keyword?

Keywords are predefined words in C that are used to perform specific tasks or define the structure of the program. In other words a keyword is a reserved word in C that has a special meaning to the compiler. Keywords are predefined words in C that are used to perform specific tasks or define the structure of the program.

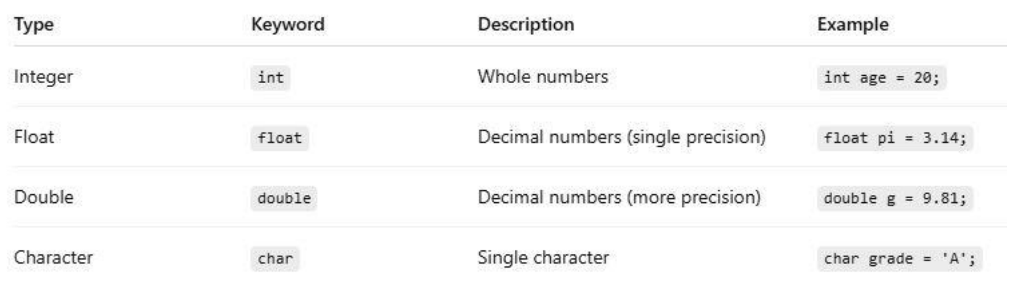


`auto`	`break`	`case`	`char`
`const`	`continue`	`default`	`do`
`double`	`else`	`enum`	`extern`
`float`	`for`	`goto`	`if`
`int`	`long`	`register`	`return`
`short`	`signed`	`sizeof`	`static`
`struct`	`switch`	`typedef`	`union`
`unsigned`	`void`	`volatile`	`while`

What are Data Types?

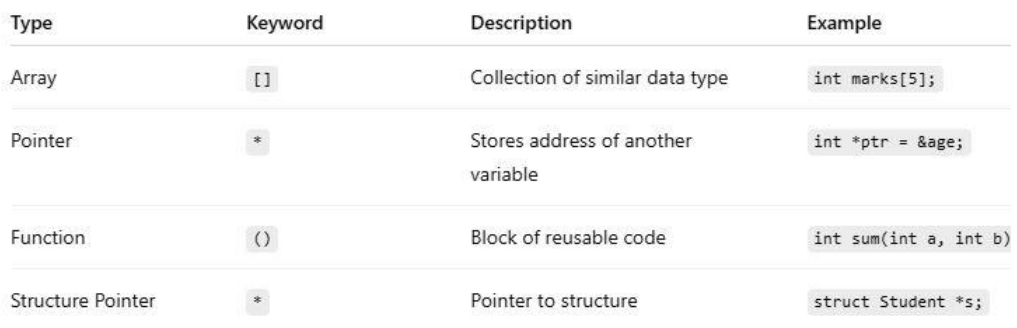
Data types in C tell the compiler what kind of data (like number, character, decimal) a variable can store and how much memory it needs.

1. Basic (Primary) Data Types:



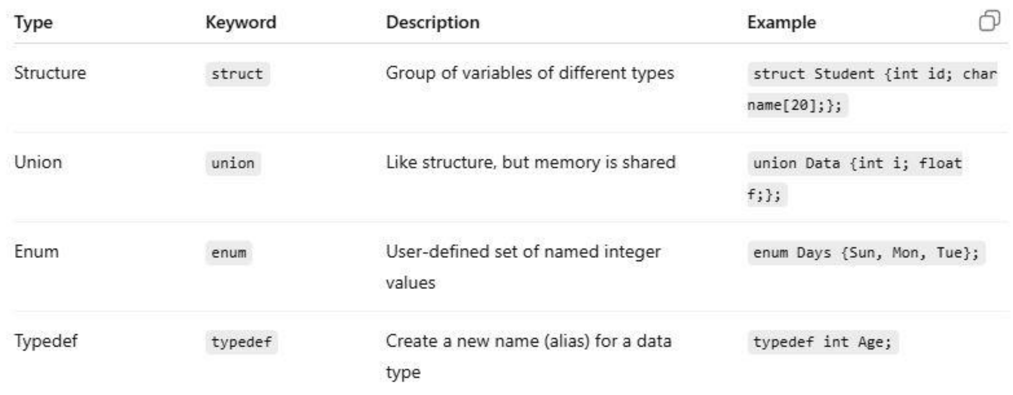
Type	Keyword	Description	Example
Integer	`int`	Whole numbers	`int age = 20;`
Float	`float`	Decimal numbers (single precision)	`float pi = 3.14;`
Double	`double`	Decimal numbers (more precision)	`double g = 9.81;`
Character	`char`	Single character	`char grade = 'A';`

2. Derived Data Types



Type	Keyword	Description	Example
Array	`[]`	Collection of similar data type	`int marks[5];`
Pointer	`*`	Stores address of another variable	`int *ptr = &age;`
Function	`()`	Block of reusable code	`int sum(int a, int b)`
Structure Pointer	`*`	Pointer to structure	`struct Student *s;`

3. User Defined



Type	Keyword	Description	Example
Structure	`struct`	Group of variables of different types	`struct Student {int id; char name[20];};`
Union	`union`	Like structure, but memory is shared	`union Data {int i; float f;};`
Enum	`enum`	User-defined set of named integer values	`enum Days {Sun, Mon, Tue};`
Typedef	`typedef`	Create a new name (alias) for a data type	`typedef int Age;`

Constants:

Constants are fixed values that cannot be modified during program execution. They are defined using the const keyword or the #define preprocessor directive.

Types of Constants:

- **Integer Constants**: e.g., 10, -5,3,-1
- **Floating-point Constants**: e.g., 3.14, -0.001
- **Character Constants**: e.g., 'A', '\n'
- **String Constants**: e.g., "Hello"
- **Symbolic Constants**: Defined using #define or const.

Statements:

Statements are instructions in a C program that perform actions. They are executed sequentially unless controlled by loops, conditionals, or jumps.

Types of Statements:

- Expression Statements: Expressions followed by a semicolon (e.g., x = 5;).
- Compound Statements: Blocks of statements enclosed in {}.
- Control Statements: if, else, for, while, switch, etc.
- Jump Statements: break, continue, return, goto.
- Declaration Statements: Variable declarations (e.g., int x;).

Operators:

In the C programming language, operators are special symbols or keywords used to perform operations on variables and values (operands). Operators allow you to manipulate data, perform calculations, make comparisons, and control program flow. C supports a rich set of operators, categorized based on their functionality.

1. Arithmetic Operators

Perform mathematical operations on numeric operands.

Operator	**Description**	**Example**	**Result**
+	Addition	5 + 3	8
-	Subtraction	5 - 3	2
*	Multiplication	5 * 3	15
/	Division	15 / 3	5 (integer division for int)
%	Modulus (remainder)	17 % 5	2

2. Relational Operators

Compare two operands, returning 1 (true) or 0 (false).

Operator	Description	Example	Result
==	Equal to	5 == 3	0 (false)
!=	Not equal to	5 != 3	1 (true)
>	Greater than	5 > 3	1 (true)
<	Less than	5 < 3	0 (false)
>=	Greater than or equal to	5 >= 5	1 (true)
<=	Less than or equal to	5 <= 3	0 (false)

3. Logical Operator

Perform logical operations on boolean expressions, returning 1 (true) or 0 (false).

Operator	**Description**	**Example**	**Result**
&&	**Logical AND**	**(5 > 3) && (2 < 4)**	**1 (true)**
`		**`**	**Logical OR**
!	**Logical NOT**	**!(5 > 3)**	**0 (false)**

4. Bitwise Operators

Manipulate individual bits of integer operands.

Operator	**Description**	**Example**	**Result**
&	Bitwise AND	5 & 3	1 (0101 & 0011 = 0001)
`	`	Bitwise OR	`5
^	Bitwise XOR	5 ^ 3	6 (0101 ^ 0011 = 0110)
~	Bitwise NOT	~5	-6 (inverts bits, depends on system)
<<	Left Shift	5 << 1	10 (0101 << 1 = 1010)
>>	Right Shift	5 >> 1	2 (0101 >> 1 = 0010)

5. Assignment Operators

Assign values to variables, often combining with other operations.

Operator	**Description**	**Example**	**Equivalent To**
=	**Assign**	**a = 5**	**a = 5**
+=	**Add and assign**	**a += 3**	**a = a + 3**
-=	**Subtract and assign**	**a -= 3**	**a = a - 3**
*=	**Multiply and assign**	**a *= 3**	**a = a * 3**
/=	**Divide and assign**	**a /= 3**	**a = a / 3**
%=	**Modulus and assign**	**a %= 3**	**a = a % 3**
&=	**Bitwise AND and assign**	**a &= 3**	**a = a & 3**
`	=`	**Bitwise OR and assign**	`a
^=	**Bitwise XOR and assign**	**a ^= 3**	**a = a ^ 3**
<<=	**Left shift and assign**	**a <<= 1**	**a = a << 1**
>>=	**Right shift and assign**	**a >>= 1**	**a = a >> 1**

6. Increment and Decrement Operators

Increase or decrease a variable's value by 1.

Operator	Description	Example	Result
++	Increment	a++ or ++a	a = a + 1
--	Decrement	a-- or --a	a = a - 1

7. Conditional (Ternary) Operator

The ternary operator (?:) is a shorthand for an if-else statement. It evaluates a condition and returns one of two values.

Syntax: condition ? expression1 : expression2

- If condition is true, evaluates expression1; otherwise, evaluates expression2.

Example:

```
#include <stdio.h>

int main() {

   int a = 10, b = 20;

   int max = (a > b) ? a : b; // If a > b, max = a; else max = b

   printf("Max: %d\n", max);

   return 0;

}
```

8. Comma Operator

The comma operator (,) evaluates multiple expressions and returns the result of the last expression. It is often used in for loops.
Syntax: expression1, expression2, ..., expression

Example:
```
#include <stdio.h>
```

```
int main() {
    int a = 5, b;
    b = (a += 2, a * 3); // Evaluates a += 2 (a = 7), then a * 3 (7 * 3 = 21)
    printf("a: %d, b: %d\n", a, b);
    return 0;
}
```

9. Unary Operator

"Unary operator" is the most accurate and widely used term in C programming and computer science to describe operators that take a single operand (e.g., ++, --, !, ~, &, *, sizeof, (type)).

MCQS

Question 1
What is a valid identifier?
a) my_var
b) v
c) int
d) g@Name
(Answer: a)

Question 2
Which is a keyword?
a) total
b) if
c) myData
d) x1
(Answer: b)

Question 3
What is the size of int on most systems?
a) 2 bytes
b) 4 bytes
c) 8 bytes
d) 1 byte
(Answer: b)

Question 5
Which declares a constant?
a) int x = 10;
b) const int x = 10;
c) x = 10;
d) var x = 10;
(Answer: b)

Question 6

What is an array?

a) Single variable

b) Collection of variables

c) Function

d) Keyword

(Answer: b)

Question 7

Which operator increments a value?

a) -

b) ++

c) *

d) /

(Answer: b)

Question 9

Which does a == b check?

a) Assignment

b) Equality

c) Addition

d) Increment

(Answer: b)

Question 10

Which is a logical operator?

a) +

b) ||

c) *

d) %

(Answer: b)

Question 11

What does a + b do?

a) Subtracts b from a

b) Adds b to a

c) Multiplies a by b

d) Divides a by b

(Answer: b)

Question 12

What is the ternary operator?

a) +

b) ?:

c) ||

d) &&

(Answer: b)

Question 35 (Page 1)

What encloses a compound statement?

a) ()

b) {}

c) []

d) <>

(Answer: b)

Question 36 (Page 1)

What is the result of 5 % 2?

a) 0

b) 1

c) 2

d) 5

(Answer: b)

Question 37 (Page 1)

Which is true for if (a < b && a < c)?

a) a is largest

b) a is smallest

c) b is smallest

d) c is largest

(Answer: b)

Question 39 (Page 1)

What does !true return?

a) true

b) false

c) 0

d) 1

(Answer: b)

Question 40 (Page 1)
Which declares an array of 3 integers?
a) int arr[3];
b) int arr = 3;
c) arr[3]
d) int arr(3);
(Answer: a)

Question 42 (Page 1)
Which is a valid variable name?
a) 123abc
b) __abc
c) for
d) 12ab
(Answer: b)

Question 43 (Page 1)
What is float used for?
a) Integers
b) Decimals
c) Characters
d) Strings
(Answer: b)

Question 44 (Page 1)
Which statement ends with a semicolon?
a) if
b) while
c) x = 5;
d) for
(Answer: c)

Question 23
What does #define create?
a) Variable
b) Constant
c) Function
d) Loop
(Answer: b)

Question 24
Which operator shifts bits left?
a) >>
b) <<
c) &
d) |
(Answer: b)

Question 25
What is the output of 3 * 4?
a) 7
b) 12
c) 1
d) 0
(Answer: b)

Question 26
Which checks if two values are not equal?
a) ==
b) !=
c) >
d) <
(Answer: b)

Question 27
What is a compound statement example?
a) x = 5
b) { x = 5; y = 10; }
c) if (x > 0)
d) ++x
(Answer: b)

Question 28
What is int a[5]?
a) 5 variables
b) 4 variables
c) 6 variables
d) 1 variable
(Answer: a)

Question 29
Which operator decrements?
a) +
b) --
c) +=
d) *=
(Answer: b)

Question 30
What does a > b return?
a) Smaller value
b) Larger value
c) Sum
d) Difference
(Answer: b)

Question 31
Which is not a data type?
a) int
b) float
c) while
d) char
(Answer: c)

Question 32
What does ~ do?
a) ANDs bits
b) Inverts bits
c) ORs bits
d) Shifts bits
(Answer: b)

Question 33
Which is true for a || b?
a) True if both false
b) True if either true
c) Always false
d) Always true
(Answer: b)

Question 34
What is the result of 10 / 3?
a) 3.3333
b) 3
c) 10
d) 0
(Answer: b)

Question 35 (Page 2)
Which declares a variable?
a) const x = 5;
b) int x = 5;
c) #define x 5
d) x = 5;
(Answer: b)

Question 36 (Page 2)
What does a && b require?
a) One true
b) Both true
c) One false
d) Both false
(Answer: b)

Question 37 (Page 2)
Which operator divides?
a) *
b) /
c) &
d) +
(Answer: b)

Question 38
What is char used for?
a) Numbers
b) Characters
c) Decimals
d) Arrays
(Answer: b)

Question 39 (Page 2)
Which is a valid expression?
a) int x
b) x + y
c) if (x > 0)
d) (x > 5)
(Answer: b)

Question 40 (Page 2)
What does a -= b do?
a) Adds b to a
b) Subtracts b from a
c) Multiplies a by b
d) Divides a by b
(Answer: b)

Question 41
Which shifts bits right?
a) <<
b) >>
c) &
d) |
(Answer: b)

Question 42 (Page 2)
What is the output of !false?
a) false
b) true
c) 0
d) 1
(Answer: b)

Question 43 (Page 2)
Which is not an operator?
a) >
b) int
c) %
d) &
(Answer: b)

Question 44 (Page 2)
What does a -= b do?
a) Adds b to a
b) Subtracts b from a
c) Multiplies a by b
d) Divides a by b
(Answer: b)

Question 45
Which checks if a is less than b?
a) a > b
b) a << b
c) a < b
d) a = b
(Answer: c)

Question 46
What is #define MAX 100?
a) Variable
b) Constant
c) Function
d) Loop
(Answer: b)

Question 47
Which is true for a != b?
a) a equals b
b) a not equals b
c) a greater than b
d) a less than b
(Answer: b)