Anjuman — I — Islam's

M.H SABOO SIDDIK COLLEGE OF ENGINEERING

8, Saboo Siddik Polytechnic Rd., Byculla, Mumbai — 400 008.

Department of Basic Science and Humanity

Module-01

Subject: C Programming **Sem: I**

◆ Language:

A computer language is a medium of communication between humans (programmers) and computers.

Types of Computer Languages

1. Machine Language

- **Definition**: The lowest-level programming language, directly understood by the computer's hardware.
- **Form**: Written in **binary digits (0 and 1)**.
- **Advantage**: Very fast and efficient for the machine.
- **Disadvantage**: Very difficult for humans to read, write, and debug.

2. Assembly Language

- **Definition**: A low-level language that uses **mnemonic codes** instead of binary numbers.
- **Form**: Human-readable instructions that represent machine operations.
- **Example**:
- ADD A, B
- MOV B, C
- **Advantage**: Easier to understand than machine language.
- **Disadvantage**: Still hardware-dependent and requires knowledge of computer architecture.

3. High-Level Language

- **Definition**: A programming language that uses **English-like statements** for writing instructions.
- **Form**: More abstract, closer to human language.
- **Process**: Programs written in high-level languages are **translated into machine language** using a **compiler, interpreter, or translator**.
- **Examples**: C, C++, Java, Python.
- **Advantage**: Easy to learn, write, and maintain.
- **Disadvantage**: Slower than machine language (due to translation step).

Concept	Primary Function
Compiler	Translates the entire source code to machine code **before** execution.
Interpreter	Translates and executes the source code **line-by-line**, during runtime.
Assembler	Converts assembly language mnemonics into machine code.
Linker	Combines multiple object files and libraries into a single executable.
Loader	Copies the executable program from storage into memory to be run by the CPU.

◆ What is Programming?

Computer programming is a medium for us to communicate with computers. Just as we use languages like Hindi or English to communicate with each other, programming is a way for us to deliver instructions to a computer.

What is C?

- C is a programming language.
- It is one of the oldest, most influential, and widely used programming languages.
- C was developed by **Dennis Ritchie** at **AT&T's Bell Labs** in the **USA in 1972**.

✓ ALGORITHM

An algorithm is a step-by-step procedure to solve a problem or accomplish a task.

Characteristics (FEATURES)

- Finite: Must terminate after a finite number of steps
- Effective: Each step must be basic and executable
- Ambiguity-free: Unambiguous and clear instructions
- Traceable: Can be followed step by step
- Unique input: Zero or more inputs
- Result-oriented: Must produce at least one output
- Explicit: Well-defined and precise

Representation Methods

- ✓ Natural Language (English)
- ✓ Pseudocode (Structured English)
- ✓ Flowchart (Visual Diagram)
- ✓ Program Code (C, C++, Java, etc.)

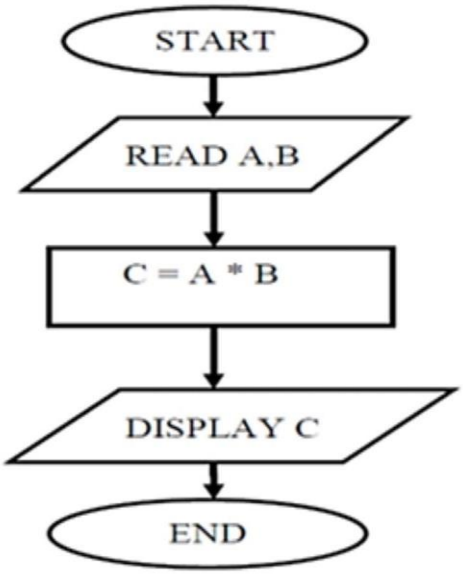
Example: Algorithm to Multiplication of Two Numbers

STEP 1: START
STEP 2: READ A, B
STEP 3: SET C=A*B
STEP 4: PRINT C
STEP 5: STOP

✓ FLOWCHART

A flowchart is a **visual representation** of an algorithm using standard symbols.

Symbol	Meaning
	Start/Stop
	Process
	Input/Output
	Decision/Branching
	Connector
	Flow



◆ 1.1 Character Set, Identifiers, Keywords, Data Types, Constants, Variables

✅ Character Set

- Letters: A–Z, a–z
- Digits: 0–9
- Special characters: + - * / = ~ ! @ # $ % ^ & * ( ) _ { } [ ] ; : ' " < > , . ? \ |
- White spaces: space, tab, newline

✅ Identifiers

- Names used for variables, functions, arrays, etc.
- Rules:

1. Variable Name Must Begin with a Letter or Underscore
   Valid: age, _count
   Invalid: 1stNum, @value

⚠️ Although a variable can start with _, it's not recommended for user-defined names (reserved for system/internal use).

2. Variable Names Can Contain Letters, Digits, and Underscores
   Valid: total_marks, x1, data_2
   Invalid: total-marks (hyphen not allowed), my var (space not allowed)
3. Variable Names Are Case Sensitive
   Age, AGE, and age are different variables.
4. Variable Must Be Declared Before Use
   int x = 10; // OK
   y = 5; // ❌ Error: y is undeclared
5. No Reserved Keywords as Variable Names
   Invalid: int, float, return, for, etc.
   int int = 5; // ❌ invalid: 'int' is a keyword
6. Each Variable Must Have a Specific Data Type
   int age; // correctfloat salary; // correct

You must declare the data type (int, char, float, etc.) before using a variable in C.

7. Optional: Additional Good Practices
   Initialize variables before use:
   int count = 0;
   Use meaningful names for readability:
   float pi; // better than fint score; // better than s

✅ Keywords

- Reserved words in C (cannot be used as identifiers). E.g.:
- **int, float, return, if, else, for, while, break, continue, void, char, long, double, short, sizeof, struct**

✅ Data Types

Basic Data Types in C

Data Type	Keyword	Size (Typical)	Format Specifier	Range (Signed)
Integer	int	4 bytes	%d or %i	-2,147,483,648 to 2,147,483,647
Character	char	1 byte	%c	-128 to 127
Float	float	4 bytes	%f	~±3.4e±38 (7 digits precision)
Double	double	8 bytes	%lf	~±1.7e±308 (15 digits precision)

Integer Type Variants

Type	Size	Signed Range	Unsigned Range	Format Specifier
short	2 bytes	−32,768 to 32,767	0 to 65,535	%hd / %hu
int	4 bytes	−2.14B to 2.14B	0 to 4.29B	%d / %u
long	4 bytes	Same as int (in most systems)		%ld / %lu
long long	8 bytes	±9 quintillion	0 to 18 quintillion	%lld / %llu

Note: Use unsigned before the type for positive-only numbers.

Floating-Point Types

Type	Size	Precision	Format Specifier
float	4 bytes	~6 – 7 decimal digits	%f
double	8 bytes	~15 decimal digits	%lf
long double	12 – 16 bytes	~18 – 21 decimal digits	%Lf

Character Types

Type	Size	Range	Format Specifier
char	1 byte	−128 to 127 (signed) / 0 – 255 (unsigned)	%c
unsigned char	1 byte	0 to 255	%c

✅ **Constants**

- Literal constants: 10, 3.14, 'A', "Hello"
- #define constants:
  - **#define PI 3.14**
- const keyword:
  - **const** int x = 100;

✅ **Variables**

- Containers for storing data values.
- Must be declared before use.

```
int age;
float price;
char grade.
```

🔹 1.2 Operators & Expressions

✅ **Arithmetic Operators**

Operator	Description	Example
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b **(FOR QUOTIENT)**
%	Modulus	a % b **(FOR REMAINDER)**

✅ Relational Operators

Operator	Description	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater or equal	a >= b
<=	Less or equal	a <= b

✅ Logical Operators

Operator	Description	Example
&&	Logical AND	(a > 5 && b < 10)
||	Logical OR	(a > 5 || b < 10)
!	Logical NOT	!a

✅ Assignment Operators

= += -= *= /= %=

✅ Unary Operators

++a, --a, -a, +a, !a

✅ Conditional Operator

(condition) ? true_expr : false_expr;

✅ Bitwise Operators

& | ^ ~ << >>

✅ Comma Operator

int a = (1, 2, 3); // a = 3

✅ Other Operators

- sizeof(): Returns size in bytes
- &: Address of variable
- *: Pointer dereference
- ->: Access structure member via pointer

✅ Expressions

- Combination of variables, constants, operators
- int c = a + b * 2;

✅ Statements

- Instructions that perform actions

```
a = 5;
printf("Hello");
```

✅ Library Functions

- Built-in functions in C standard library
- printf(), scanf(), strcpy(), strlen(), pow()

✅ Preprocessor

- Instructions that begin with #
- #include <stdio.h> // Includes header file
- #define PI 3.14 // Macro definition

🔹 1.3 Data Input and Output

✅ Character I/O

```
char ch;
ch = getchar();   // input one character
putchar(ch);      // output one character
```

String I/O

```
char str[50];
gets(str);          // input string (unsafe)
puts(str);          // output string
```

Formatted I/O

```
int a;
float b;
char c;

scanf("%d %f %c", &a, &b, &c);  // input
printf("%d %.2f %c", a, b, c);  // output
```

Structure of a C Program

```
#include <stdio.h>      // Preprocessor Directive

int main() {            // Main Function
   int a = 10;          // Variable Declaration
   printf("%d", a);     // Output
   return 0;            // Return Statement
}
```

Quick Tips

- ✓ Every statement ends with a semicolon;
- ✓ main() is the entry point of a C program
- ✓ Use #include to include header files like stdio.h, math.h, etc.
- ✓ Always initialize variables before using them

Comments

Type	Syntax	Description
Single-line	// This is a comment	For short, inline explanations
Multi-line	/* This is a multi-line comment */	For longer descriptions or blocks

Escape Sequences

Escape Code	Description	Example in Code	Output
\n	New line	printf("Line 1\\nLine 2");	Line 1Line 2
\t	Horizontal tab	printf("A\\tB");	A B
****	Backslash	printf("C:\\\\Path");	C:\Path
\'	Single quote	printf("\\'Hello\\'");	'Hello'
\"	Double quote	printf("\\\"Hi\\\"");	"Hi"
\r	Carriage return	printf("Hello\\rWorld");	World
\b	Backspace	printf("AB\\bC");	AC
\a	Alert (beep sound)	printf("\\a");	(Makes a beep sound)
\f	Form feed	Rarely used	(Page break effect)
\v	Vertical tab	Rarely used	(Vertical space)
\0	Null character (end of string)	Used in strings	N/A (invisible)

Format Specifiers

Specifier	Type	Description	Example Value
%d	int	Signed decimal integer	10, -25
%u	unsigned int	Unsigned decimal integer	25
%f	float/double	Decimal floating-point	3.14, -0.5
%.nf	float/double	Floating-point with n decimals	%.2f → 3.14
%c	char	Single character	'A'
%s	string	Null-terminated character array	"Hello"
%ld	long int	Long signed integer	1234567890
%lu	unsigned long	Unsigned long integer	4000000000
%lf	double	Double precision float	3.1415926
%p	pointer address	Memory address	0x7ffe...
%%	literal %	Prints a percent sign	%

Practice Questions:

- WAP in C to Program to print Hello World!
- WAP in C to add two numbers by taking values from user.
- WAP in C to calculate area of triangle. **(Hint: area = (b*h)/2;)**
- WAP in C to swap two numbers using temporary variable.
- WAP in C to Calculate area of rectangle. **(Hint: area = length * width;)**
- WAP in C to Calculate area of Circle. **(Hint: area = PI * radius * radius;)**
- WAP in C to Calculate volume of a cylinder. **(Hint: volume = PI * radius * radius * height;)**
- WAP in C to Calculate square of a number. **(Hint: square = number * number;)**
- WAP in C to Calculate average of three numbers. **(Hint: average = (num1 + num2 + num3) / 3;)**
- WAP in C to accept number from user and find remainder after dividing it by 2 and 3.
  (Hint: remainder2 = number % 2;
  remainder3 = number % 3;)
- WAP in C to accept two digit number from user and display it in reverse order.
  (Hint: int tens = number / 10;
  int ones = number % 10;
  reversed = ones * 10 + tens;)
- WAP in C to accept float number and display integer part using type casting operator.
  (Hint: scanf("%f", &num);
  intPart = (int)num;)
- WAP in C to accept number and display equivalent ASCII using type casting.
  (Hint: int num;
  char asciiChar;
  scanf("%d", &num);
  asciiChar = (char)num;)

- Find output:

```
#include <stdio.h>

int main() {
   int age = 25;            // Variable
   const float pi = 3.14;   // Constant
   char grade = 'A';        // Character variable

   printf("Age: %d\n", age);
   printf("Pi: %.2f\n", pi);
   printf("Grade: %c\n", grade);

   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = 5, b = 3;
   int sum = a + b;
   int rel = (a > b);         // Relational
   int logical = (a > 0 && b > 0); // Logical
   int conditional = (a > b) ? a : b; // Conditional

   printf("Sum: %d\n", sum);
   printf("Is a > b? %d\n", rel);
   printf("Logical AND result: %d\n", logical);
   printf("Conditional (max): %d\n", conditional);

   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = 5, b = 8;
   printf("a == b: %d\n", a == b);
   printf("a != b: %d\n", a != b);
   printf("a < b: %d\n", a < b);
   printf("a > b: %d\n", a > b);
   printf("a <= b: %d\n", a <= b);
   printf("a >= b: %d\n", a >= b);
   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = 5, b = 0;
   printf("(a > 0 && b > 0) = %d\n", (a > 0 && b > 0));
   printf("(a > 0 || b > 0) = %d\n", (a > 0 || b > 0));
   printf("!(a > b) = %d\n", !(a > b));
   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = 5;
   printf("a = %d\n", a);
   printf("++a = %d\n", ++a);  // pre-increment
   printf("a++ = %d\n", a++);  // post-increment
   printf("a after post-increment = %d\n", a);
   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   unsigned int a = 5, b = 9;
   printf("a & b = %d\n", a & b);
   printf("a | b = %d\n", a | b);
   printf("a ^ b = %d\n", a ^ b);
   printf("~a = %d\n", ~a);
   printf("a << 1 = %d\n", a << 1);
   printf("b >> 1 = %d\n", b >> 1);
   return 0;
}
```

- Find output:

```
#include <stdio.h>

int main() {
   int a = (printf("Hello, "), 10 + 20);
   printf("\na = %d\n", a);
   return 0;
}
```

Aim: WAP in C to Calculate area of rectangle. (Hint: area = length * width;)

```c
#include <stdio.h>

int main() {
    float length, width, area;

    // Input
    printf("Enter the length of the rectangle: ");
    scanf("%f", &length);

    printf("Enter the width of the rectangle: ");
    scanf("%f", &width);

    // Calculation
    area = length * width;

    // Output
    printf("The area of the rectangle is: %.2f\n", area);

    return 0;
}
```

- WAP in C to Calculate area of Circle. **(Hint: area = PI * radius * radius;)**

```c
#include <stdio.h>

#define PI 3.14159  // Defining constant for π

int main() {
    float radius, area;

    // Input
    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);

    // Calculation
    area = PI * radius * radius;

    // Output
    printf("The area of the circle is: %.2f\n", area);

    return 0;
}
```

- WAP in C to accept number from user and find remainder after dividing it by 2 and 3.
  (Hint: remainder2 = number % 2;
  remainder3 = number % 3;)

```c
#include <stdio.h>

int main() {
    int number, remainder2, remainder3;

    // Input
    printf("Enter a number: ");
    scanf("%d", &number);

    // Calculations
    remainder2 = number % 2;
    remainder3 = number % 3;

    // Output
    printf("Remainder when divided by 2: %d\n", remainder2);
    printf("Remainder when divided by 3: %d\n", remainder3);

    return 0;
}
```

- WAP in C to accept float number and display integer part using type casting operator.
  (Hint: scanf("%f", &num);
  intPart = (int)num;)

```c
#include <stdio.h>

int main() {
    float num;
    int intPart;

    // Input
    printf("Enter a float number: ");
    scanf("%f", &num);

    // Type casting
    intPart = (int)num;

    // Output
    printf("Integer part of %.2f is: %d\n", num, intPart);

    return 0;
}
```

- WAP in C to Calculate Square Root of a number

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

int main() {
    double num, result;

    printf("Enter a number: ");
    scanf("%lf", &num);

    result = sqrt(num);
    printf("Square root of %.2lf is %.2lf\n", num, result);

    return 0;
}
```

"The only way to learn a new programming language is by writing programs in it." – Dennis Ritchie