C Programming Language

Author:
Youssef Lamkhantar

1.Introduction

- C is a general-purpose, procedural programming language.
- Developed by Dennis Ritchie in 1972 at Bell Labs.
- Known for its efficiency and control, often used for system software, operating systems, and embedded programming.

2.Key Features of C:

- Simple and easy to learn syntax.
- Procedural language with a rich set of built-in operators and functions.
- Supports structured programming.

2.Key Features of C:

- Low-level memory access through pointers.
- Portability and flexibility for system-level programming.
- Fast execution due to its close relationship with assembly language.

3.Variables and Data Types:

- **Variables:** Used to store data; each has a specific type.
- **Data Types:**
  - **Basic types:** `int`, `float`, `char`, `double`.
  - **Derived types:** Arrays, Pointers, Structures, Unions.
  - **Void:** Used for functions that do not return a value.

4.Operators

- **Arithmetic Operators:** `+`, `-`, `*`, `/`, `%`
- **Relational Operators:** `==`, `!=`, `>`, `<`, `>=`, `<=`
- **Logical Operators:** `&&`, `||`, `!`
- **Bitwise Operators:** `&`, `|`, `^`, `<<`, `>>`
- **Assignment Operators:** `=`, `+=`, `-=`, etc.

5.Control Structures

- **Conditional Statements:** `if`, `else`, `switch`
- **Loops:**
  - `for`: Repeats a statement a specific number of times.

5.Control Structures

- `while`: Repeats a statement while a condition is true.
- `do-while`: Similar to `while` but ensures the statement executes at least once.

- **Jump Statements:** `break`, `continue`, `return`, `goto`

6.Functions

- Functions break programs into smaller, reusable code blocks.
- **Defining Functions:** Syntax, return types, parameters.
- **Library Functions:** Standard functions like `printf`, `scanf`, `malloc`, etc.
- **Recursion:** Functions that call themselves.

7. Standard Library Functions

Input/Output Functions:

- `printf()`: Prints output to the console.
- `scanf()`: Reads formatted input from the user.
- `gets()`: Reads a string from the user (deprecated due to security risks).
- `puts()`: Writes a string to the console.
- `fscanf()`: Reads formatted data from a file.
- `fprintf()`: Writes formatted data to a file.
- `fgets()`: Reads a string from a file.
- `fputs()`: Writes a string to a file.

7. Standard Library Functions

String Handling Functions:

- `strlen()`: Returns the length of a string.
- `strcpy()`: Copies a string to another.
- `strcat()`: Concatenates two strings.
- `strcmp()`: Compares two strings.
- `strncpy()`: Copies a specific number of characters from one string to another.
- `strstr()`: Finds the first occurrence of a substring.

7. Standard Library Functions

Memory Management Functions:

- `malloc()`: Allocates dynamic memory.
- `calloc()`: Allocates and zeroes out dynamic memory.
- `free()`: Frees dynamically allocated memory.
- `realloc()`: Reallocates dynamic memory.

7. Standard Library Functions

Mathematical Functions:

- `abs()`: Returns the absolute value of an integer.
- `pow()`: Returns the value of a number raised to a power.
- `sqrt()`: Returns the square root of a number.
- `sin()`, `cos()`, `tan()`: Trigonometric functions.
- `ceil()`, `floor()`: Rounds up or down a number.
- `log()`: Returns the natural logarithm.

7. Standard Library Functions

File Handling Functions:

- `fopen()`: Opens a file.
- `fclose()`: Closes a file.
- `fwrite()`: Writes to a file.
- `fread()`: Reads from a file.
- `fflush()`: Flushes the output buffer to a file.
- `ftell()`: Returns the current file position.
- `fseek()`: Moves the file pointer to a specific position.

7. Standard Library Functions

Character Handling Functions:

- `isalpha()`: Checks if a character is alphabetic.
- `isdigit()`: Checks if a character is a digit.
- `isalnum()`: Checks if a character is alphanumeric.
- `tolower()`: Converts a character to lowercase.
- `toupper()`: Converts a character to uppercase.

7. Standard Library Functions

Time Functions:

- `time()`: Returns the current time.
- `clock()`: Returns processor time.
- `difftime()`: Calculates the difference between two times.
- `localtime()`: Converts the time to local time.
- `strftime()`: Formats the date and time.

7. Standard Library Functions

Utility Functions:

- `exit()`: Terminates the program.
- `system()`: Executes a system command.
- `assert()`: Used for debugging, checks conditions.

8.Pointers

- Variables that store memory addresses.
- Used for dynamic memory management, passing arguments by reference, and arrays.
- Syntax: `int *ptr;`
- **Pointer arithmetic** and **NULL pointers**.

9.Arrays and Strings

- **Arrays:** Collection of variables of the same type.
  - Syntax: `int arr[5];`
  - Can be one-dimensional or multi-dimensional (2D arrays).
- **Strings:** Arrays of characters ending with a null character `\0`.

10.Structures and Unions

- **Structures:** User-defined data types that group variables of different types.
  - Syntax: `struct { int age; char name[20]; };`
- **Unions:** Similar to structures but with shared memory for all members.

11.Memory Management

- **Dynamic Memory Allocation:** Functions like `malloc()`, `calloc()`, `free()`, `realloc()`.
- **Static vs. Dynamic memory allocation.**
- **Memory Leaks:** How to avoid them.

12.File Handling

- Opening, reading, writing, and closing files using standard functions like `fopen()`, `fclose()`, `fscanf()`, `fprintf()`, etc.
- Modes: Read (`r`), Write (`w`), Append (`a`), etc.

13.Preprocessors

- **Macros:** Defined using `#define`.
- **Include Files:** Use `#include` to include header files.
- Conditional Compilation using `#ifdef`, `#ifndef`.

14.Debugging and Error Handling

- Handling errors using functions like `perror()`, `strerror()`.
- Common errors in C: Segmentation faults, memory access errors, buffer overflows.
- Tools for debugging: `gdb`, `valgrind`.

15.Applications of C

- System software like operating systems (Linux is written in C).
- Embedded systems programming.
- Game development.
- Database engines (like MySQL).
- Compilers and interpreters.