C language for biginner:

Introduction:

C is a general-purpose programming language created by Dennis Ritchie at the Bell Laboratories in 1972. It is a very popular language, despite being old. The main reason for its popularity is because it is a fundamental language in the field of computer science.

Why Learn C?

- It is one of the most popular programming languages in the world
- If you know C, you will have no problem learning other popular programming languages such as Java, Python, C++, C#, etc, as the syntax is similar
- C is very fast, compared to other programming languages, like Java and Python
- C is very versatile; it can be used in both applications and technologies

Structure of a C Program:

Every C program consists of one or more functions.

A function contains statements that specify the computing operations to be done.

Basic Syntax:

- Statements in C are terminated by a semicolon (;).
  Example:
  Copy code
  int x = 5;
  printf("Hello, world!");
- Curly braces ({}) are used to define blocks of code.
- Comments can be added using /* */ for multi-line comments or // for single-line comments.
  Example:
  // This is a single-line comment

```
/*
This is a multi-line comment
It can span multiple lines
*/
```

Data Types:

C supports various data types such as int, float, double, char, etc.

Example:

int score = 100;

float price = 10.99;

char grade = 'A';

Each data type has a specific range and memory allocation.

specific range and memory allocation:

interger types:

char: Typically 1 byte. Range: -128 to 127 or 0 to 255 (if unsigned).

short: Typically 2 bytes. Range: -32768 to 32767 or 0 to 65535 (if unsigned).

int: Typically 4 bytes. Range: -2147483648 to 2147483647 or 0 to 4294967295 (if unsigned).

long: Typically 4 or 8 bytes. Range: -2147483648 to 2147483647 or 0 to 4294967295 (if unsigned).

long long: Typically 8 bytes. Range: -(2^63) to (2^63 - 1) or 0 to (2^64 - 1) (if unsigned).

Floating-Point Types:

float: Typically 4 bytes. Range: approximately ±3.40282347e+38F (6-7 significant decimal digits).

double: Typically 8 bytes. Range: approximately ±1.79769313486231570e+308 (15 significant decimal digits).

long double: Varies by system. Range: extended precision compared to double.

Other Types:

void: Typically used for indicating a function that does not return a value. No storage allocated.

_Bool: Typically 1 byte. Range: 0 or 1.

Arrays:

Memory allocated depends on the size of the array and the data type of its elements. For example, an array of 10 integers would typically allocate 40 bytes (assuming an int is 4 bytes).

int numbers[5] = {1, 2, 3, 4, 5};

Pointers:

Typically 4 or 8 bytes on 32-bit or 64-bit systems respectively, but this can vary. Memory allocated depends on the size of the data type being pointed to.

Example:

nt *ptr;

int x = 10;

ptr = &x;

Structures:

Memory allocated depends on the size and alignment of each member of the structure.

Unions:

Memory allocated is the size of the largest member of the union

Variables:

Variables are used to store data temporarily in a program.

Before using a variable, it needs to be declared with its data type.

Example:

```
int age;

float height = 5.9;
```

Operators:

C supports various types of operators such as arithmetic operators (+, -, *, /), relational operators (==, !=, >, <), logical operators (&&, ||, !), etc.

Control Flow:

Control flow statements like if-else, switch-case, while, do-while, and for loops are used to control the flow of execution in a program.

```
if (x > 0) {
   printf("Positive");
} else {
   printf("Non-positive");
}
```

Functions:

Functions are blocks of code that perform a specific task.

A C program may contain one or more functions, but it must have at least one main() function which is the entry point of the program.

Example:

```
int add(int a, int b) {
   return a + b;
}
```

Strings:

Strings in C are arrays of characters terminated by a null character '\0'.

Standard library functions are available for string manipulation like strcpy(), strcat(), strlen(), etc.

Input and Output:

Input can be obtained from the user using functions like scanf(), and output can be displayed to the user using functions like printf().

Example:

```
int num;

printf("Enter a number: ");

scanf("%d", &num)
```

Header Files:

Header files contain function prototypes and declarations used in a C program.

Standard header files like <stdio.h>, <stdlib.h>, <string.h>, etc., provide functions for input/output, memory allocation, string manipulation, etc.

Memory Management:

C allows dynamic memory allocation and deallocation using functions like malloc(), calloc(), realloc(), and free().

Error Handling:

Error handling in C is typically done using conditional statements and error codes returned by functions.

Best Practices:

Follow proper naming conventions, indent code for readability, use comments to explain complex sections, and avoid using global variables unless necessary