Unit 2

Introduction to C

- Overview and History of C
- Features, Advantages and Disadvantages of C
- Structure of C Program, Compiling Process
- Character set used in C, Data types, Variables. C Tokens (Keywords, Identifier, Constants, Operators), Header files, Library function
- Preprocessor Directives, Escape Sequence, Comments
- Input Output Operation
    - Formatted input/output function [printf(), scanf()]
    - Unformatted input/output function [getchar(), putchar(), gets(), puts(), getc(), putc()]

Introduction to C

- C is a general-purpose, procedural programming language.
- It was developed by Dennis Ritchie at Bell Labs in 1972 for UNIX OS.
- C is known for its efficiency, portability, and low-level access to system resources.
- C is widely used for developing operating systems, device drivers, and other low-level applications.
- C has influenced many other programming languages, including C++, Java, and Python.
- It is still widely used today by programmers around the world.

Features of C

Simple: C is a simple language in the sense that it provides a structured approach (to break the problem into parts), the rich set of library functions, data types, etc.

- **Structured programming language**: We can break the program into parts using functions. So, it is easy to understand and modify. Functions also provide code reusability.
- **Procedural language**: C is a procedural language, which means it follows a step-by-step approach to solve a problem.
- **High performance:** C is a highly efficient language, making it a popular choice for developing high-performance applications.
- **Portability**: C code can be easily ported to different platforms, making it a highly portable language.
- **Support for system programming:** C was originally designed for system programming, and it is still widely used for developing operating systems, device drivers, and other low-level applications.

- **Influence on other languages**: C has influenced many other programming languages, including C++, Java, and Python.
- **Mid-level programming language**: Although, C is intended to do low-level programming. It is used to develop system applications such as kernel, driver, etc. It also supports the features of a high-level language. That is why it is **known as mid-level language**.
- **Rich Library:** C provides a lot of inbuilt functions that make the development fast.
- **Extensible**: C language is extensible because it can easily adopt new features.

Application of C

Sure, here are simplified sentences for each of the applications of C programming language:

- System programming: C is used for developing operating systems, device drivers, and other low-level applications.
- Embedded systems: C is used for developing microcontrollers and other small electronic devices.
- Game development: C is used for creating high-performance games with low-level access to hardware.
- Scientific computing: C is used for handling large amounts of data in scientific research.
- Database systems: C is used for managing data and interacting with hardware at a low level.
- Graphics and multimedia: C is used for processing images, videos, and other multimedia.
- Networking: C is used for developing network protocols and communication software.
- Compiler development: C is used for developing compilers and other software tools.
- Web development: C is used for developing web servers and other web-related applications.
- Artificial intelligence and machine learning: C is used for developing efficient algorithms in AI and machine learning.

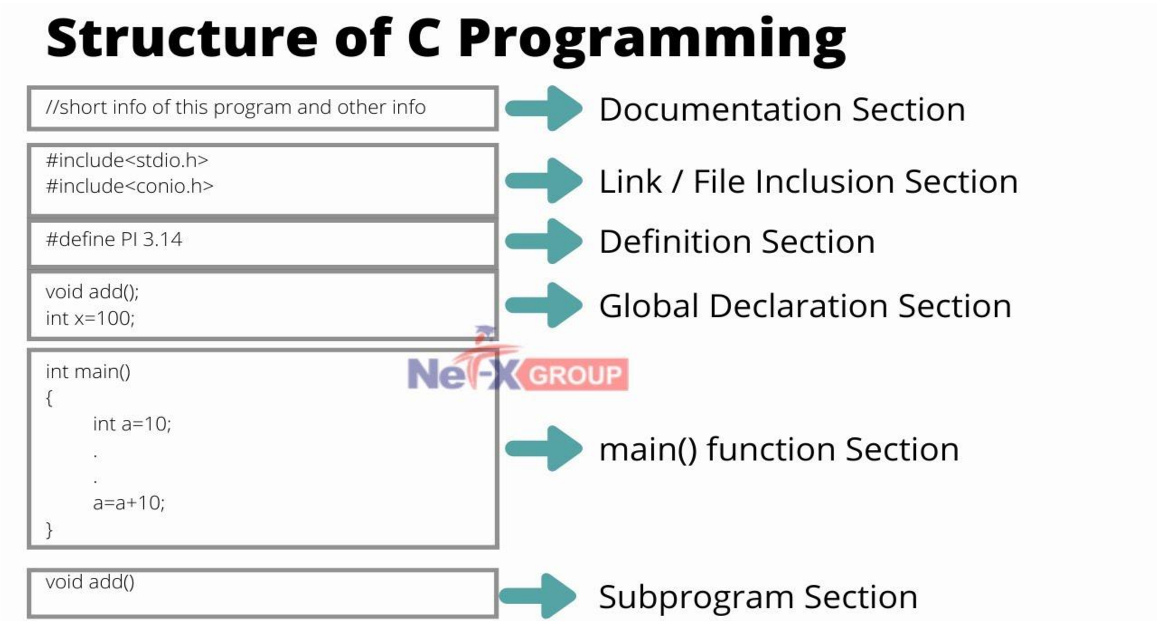
Advantages:

1. Efficiency: C is a fast language that can run code quickly and use system resources well.
2. Portability: C code can be easily moved to different platforms, so it's a good language for different devices.
3. Low-level programming: With C, you can control hardware better by accessing system resources at a low level.
4. Flexibility: C has many ways to work with different types of data, so it's flexible for different kinds of programs.
5. Large community: Lots of people use C and share code and knowledge with each other, so it has a big community of programmers.

Disadvantages:

1. C can be hard to learn, especially for beginners.
2. C programming can be hard because it lacks modern concepts like object-oriented programming.
3. C doesn't have all the modern features of newer programming languages, like automatic memory management or built-in support for multi-threading.
4. It is not popular choice to create modern apps, games and web apps.

Structure of C program



Structure of C Programming
Documentation Section
Link / File Inclusion Section
Definition Section
Global Declaration Section
main() function Section
Subprogram Section

```
//short info of this program and other info

#include<stdio.h>
#include<conio.h>

#define PI 3.14

void add();
int x=100;

int main()
{
        int a=10;
        .
        .
        a=a+10;
}

void add()
```

Section	Description
Documentation	• Consists of the description of the program, programmer's name, and creation date. • These are generally written in the form of comments.
Link	• All header files are included in this section which contains different functions from the libraries. • A copy of these header files is inserted into code before compilation.
Definition	• Includes preprocessor directive, which contains symbolic constants. E.g.: #define allows us to use constants in our code.

Section	Description
	- It replaces all the constants with its value in the code.
Global Declaration	- Includes declaration of global variables, function declarations, static global variables, and functions.
main() Function	- For every C program, the execution starts from the main() function. - It is mandatory to include a main() function in every C program.
Subprograms	- Includes all user-defined functions (functions the user provides). - These are called in the main() function.

Examples of C program

Program 1: A C program to print hello world.

```c
#include <stdio.h>

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

Program 2: A C program to add two numbers.

```c
#include <stdio.h>

int main()
{
    int num1, num2, sum;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    sum = num1 + num2;

    printf("The sum of is: %d", sum);
    return 0;
}
```

Program 3: A C program to calculate Simple Interest.

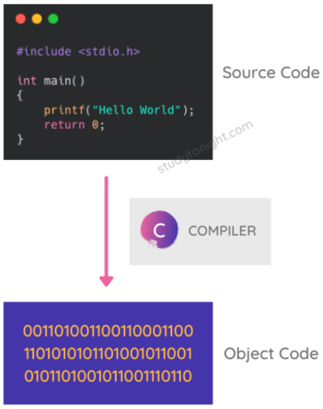
```
#include <stdio.h>

int main()
{
    float principle, rate, time, interest;

    printf("Enter principle, rate, and time: ");
    scanf("%f %f %f", &principle, &rate, &time);

    interest = (principle * rate * time) / 100;

    printf("Simple interest = %f\n", interest);

    return 0;
}
```

Program 4: A C program to convert Fahrenheit to Celsius.

```
#include <stdio.h>

int main()
{
    float fahrenheit, celsius;

    printf("Enter temperature in Fahrenheit: ");
    scanf("%f", &fahrenheit);

    celsius = (fahrenheit - 32) * 5 / 9;

    printf("%.2f Fahrenheit is equal to %.2f Celsius\n", fahrenheit, celsius);

    return 0;
}
```

Compiling process in C

The compilation is a process of converting the source code into object code. It is done with the help of the compiler.



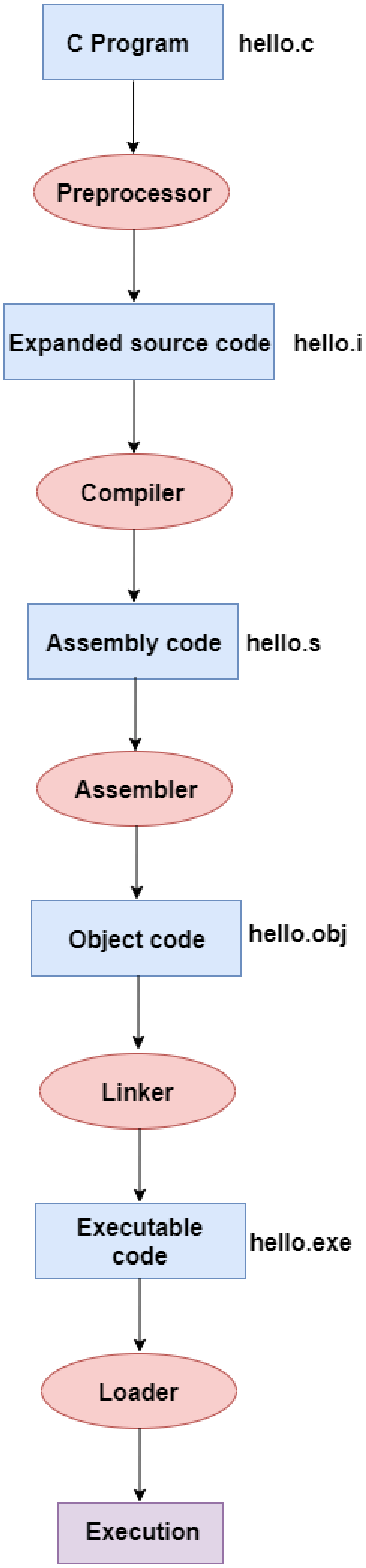
```c
#include <stdio.h>

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

```
0011010011001100011OO
110101010110100101100 1
010110100101100111O110
```

The compilation process in C involves following stages, including:

1. **Preprocessing**: In this stage, the preprocessor examines the source code and performs tasks such as including header files, replacing macros with their definitions, and removing comments.
2. **Compilation**: In this stage, the compiler takes the preprocessed source code and converts it into assembly code or machine code.
3. **Assembly**: In this stage, the assembler converts the assembly code into object code, which contains machine instructions and data.
4. **Linking**: In this stage, the linker combines the object code with other necessary object code and libraries to create an executable file.

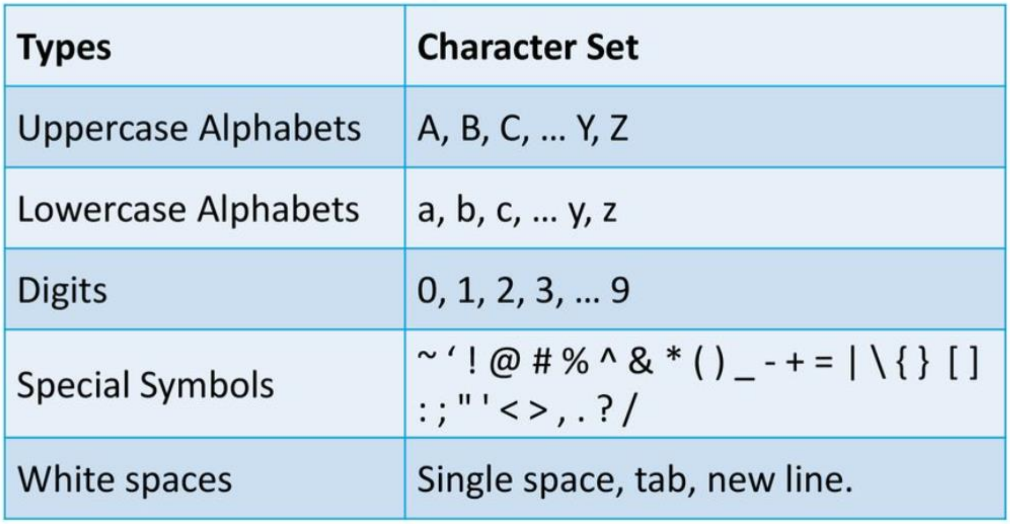
Compilation steps



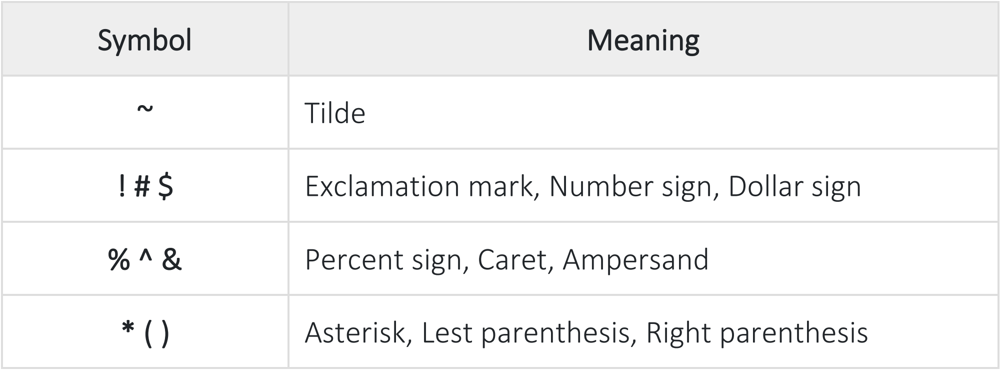
- Firstly, the input file, i.e., hello.c, is passed to the preprocessor, and the preprocessor converts the source code into expanded source code.
- The extension of the expanded source code would be hello.i.
- The expanded source code is passed to the compiler, and the compiler converts this expanded source code into assembly code.
- The extension of the assembly code would be hello.s.
- This assembly code is then sent to the assembler, which converts the assembly code into object code.
- After the creation of an object code, the linker creates the executable file.
- The loader will then load the executable file for the execution.

Character set in C

The character set used in C are:



Types	**Character Set**
Uppercase Alphabets	A, B, C, … Y, Z
Lowercase Alphabets	a, b, c, … y, z
Digits	0, 1, 2, 3, … 9
Special Symbols	~ ‘ ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /
White spaces	Single space, tab, new line.



Symbol	Meaning
~	Tilde
! # $	Exclamation mark, Number sign, Dollar sign
% ^ &	Percent sign, Caret, Ampersand
* ( )	Asterisk, Lest parenthesis, Right parenthesis

Symbol	Meaning
_ + ,	Underscore, Plus sign, Comma
. / |	Period, Slash, Vertical bar
\ ` -	Backslash, Apostrophe, Minus sign
= < >	Equal to sign, Opening angle bracket, Closing angle bracket
? { }	Question mark, Left brace, Right brace
[ ] :	Left bracket, Right bracket, Colon
" ;	Quotation mark, Semicolon

Data types in C

- A data type specifies the type of data.
- In programming, a data type is a classification of data based on the type of value it represents.
- The C programming has 5 primary data types:
  - i. Character (char)
  - ii. Integer (int)
  - iii. Floating-point (float)
  - iv. Double (double)
  - v. Void (void)

1. Character
   - We use the keyword **char** for character data type.
   - It is used to store single bit characters and occupies 1 byte of memory.
   - We can store alphabets from A-Z(and a-z) and 0-9 digits using char.
   - For example,
     ```
     char a = 'a';
     char b = 'A';
     char c = '0';
     char d = 0; //error
     ```
2. Integer
   - We use the keyword **int** for integer data type.
   - The int data type is used to store non-fractional numbers which includes positive, negative and zero values.

- The range of int is -2,147,483,648 to 2,147,483,647.
- It occupies 2 or 4 bytes of memory, depending on the system.
- For example,
  int a = 5550;
  int b = -90,
  int c = 0;
  int d = -0.5; //invalid

3. Floating-point
   - We use the keyword **float** for floating-point data type.
   - float is used to store decimal numbers.
   - It occupies 4 bytes of memory and ranges from 1e-37 to 1e+37. For example,
     float a = 0.05;
     float b = -0.005.
     float c = 1; // it will become c = 1.000000 because of type-casting
4. Double
   - We use the keyword **double** for double data type.
   - double is used to store decimal numbers.
   - It occupies 8 bytes of memory and ranges from 1e-37 to 1e+37.
   - Example:
     double a = 10.09;
     double b = -67.9;
5. Void
   - This means no value.
   - This data type is mostly used when we define functions.
   - The void data type is used when a function does not return anything.
   - It occupies 0 bytes of memory.
   - We use the **void** keyword for void data type.
   - Example:
     void function() {
       //your code goes here
     }

Example program: A C program to demonstrate data types in C.

```
#include <stdio.h>

int main()
{
    int num = 10;
    char letter = 'A';
    float fnum = 3.14;
    double dnum = 2.718;
    void *ptr;
    printf("Integer: %d\n", num);
    printf("Character: %c\n", letter);
    printf("Float: %f\n", fnum);
    printf("Double: %lf\n", dnum);
    printf("Void Pointer: %p\n", ptr);

    return 0;
}
```

Variable in C

- Variables are containers for storing data values, like numbers and characters.
- A variable is a name of the memory location.
- The value of a variable can be changed and it can be used many times.

Variable declaration

- To declare a variable in C, you specify the data type followed by the variable name, like this:
- Syntax: data_type variable_name;
- Example: int x;

Variable initialization

- Variable can be initialized with following syntax:
- Variable_name = value;
- Example: x = 10;

**We can declare and initialize variable at the same time:

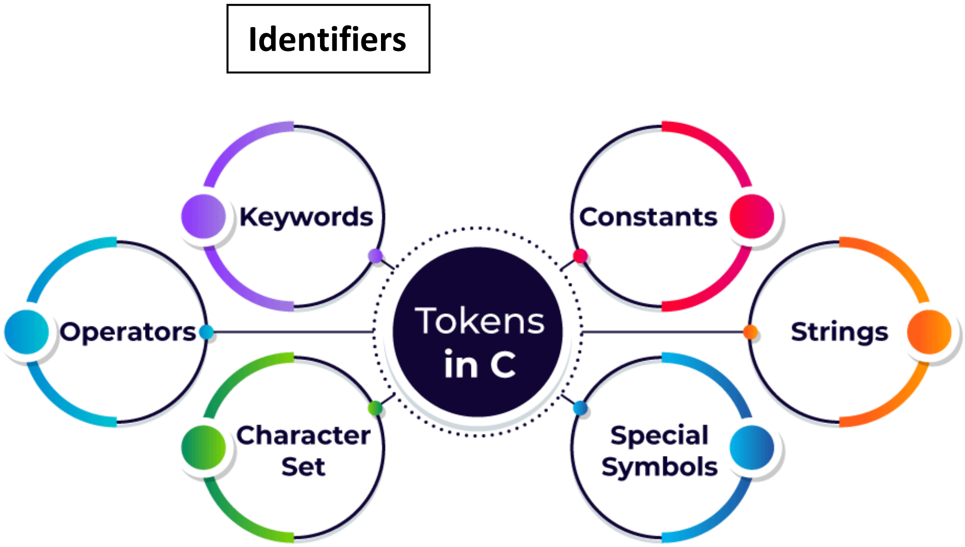
Example: int num = 10;

Rules for naming variables:

- The name must begin with a letter (upper or lowercase) or underscore character _.
- The name can only contain letters (upper or lowercase), digits, and underscore characters _.
- The name cannot be a keyword (reserved word) in the C language.

- Names are case sensitive (myVar and myvar are different variables)
- Names cannot contain whitespaces or special characters like !, #, %, etc.

C Tokens



- In C programming, a token is the smallest individual unit in a program that has a meaning or a purpose.
- The following are the different types of tokens in C programming:

1. Keywords
    - These are predefined or reserved words that have a specific meaning in the language, such as int, float, if, else, while, etc.
    - They cannot be used as the variable names.
    - C language supports 32 keywords given below:

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

2. Identifiers

- These are names given to variables, functions, and user-defined types.
- An identifier can consist of letters, digits, and underscore (_) and must begin with a letter or an underscore.
- The length of the identifiers should not be more than 31 characters.
- Example:
  - int age;
  - float salary;
  - void printMessage();
  - int numbers[10];

3. Constants

- These are fixed values that do not change during program execution.
- Here are two of the ways in which we can declare a constant:
- By using a #define pre-processor
- By using a const keyword
- Here is a list of the types of constants that we use in the C language:

Type of Constant	Example
Floating-point constant	25.7, 87.4, 13.9, etc.
Integer constant	20, 41, 94, etc.
Hexadecimal constant	0x5x, 0x3y, 0x8z, etc.
Octal constant	033, 099, 077, 011, etc.
String constant	“c++”, “.net”, “java”, etc.
Character constant	‘p’, ‘q’, ‘r’, etc.

4. String:

- Strings in C are always represented as an array of characters having null character '\0' at the end of the string.
- Strings in C are enclosed within double quotes, while characters are enclosed within single characters.
- The size of a string is a number of characters that the string contains.
- Now, we describe the strings in different ways:
  - char a[10] = "javatpoint"; // The compiler allocates the 10 bytes to the 'a' array.
  - char a[] = "javatpoint"; // The compiler allocates the memory at the run time.
  - char a[10] = {'j','a','v','a','t','p','o','i','n','t','\0'}; // String is represented in the form of characters.

5. Operators

- An operator is a symbol that tells the compiler to perform specific mathematical or logical functions such as +, -, *, /, %, &&, ||, etc.

6. Special symbols

- We also use some of the special characters in the C language, and all of them hold a special meaning that we cannot use for any other purpose.

i. () parenthesis is used in functions.
ii. [ ] Square brackets are used in arrays.
iii. (,) Commas separate statements, functions and variables.
iv. { } Curly braces enclose code blocks or loops.
v. (*) Asterisk represents pointers and multiplication.
vi. (#) Hash/preprocessor – We use it for the preprocessor directive.
vii. (.) Period is used to access a member of a union or a structure.
viii. (~) Tilde is used as destructor to free memory.

Header files in C

- A header file is a file with extension **.h** which contains C function declarations and macro definitions to be shared between several source files.
- A header file is included in a C program using the preprocessor directive #include.
- There are two types of header files in C programming: system header files and user-defined header files.
- System Header Files:
  - System header files are enclosed in angle brackets (<>) when included in a program.
  - Examples of system header files include stdio.h, stdlib.h, string.h, math.h, etc.
- User-Defined Header Files:
  - User-defined header files are enclosed in double quotes (" ") when included in a program.
  - Examples of user-defined header files include myheader.h, utils.h, etc.

Some of system header files are:

Sr.No.	Header Files & Description
1	**stdio.h** Input/Output functions
2	**conio.h** Console Input/Output functions
3	**stdlib.h** General utility functions
4	**math.h** Mathematics functions
5	**string.h** String functions
6	**ctype.h** Character handling functions
7	**time.h** Date and time functions

Library function

- In C programming, library functions are pre-written functions that are provided in libraries.
- Each function here performs a specific operation. We can use this library functions to get the pre-defined output.
- These are grouped together and placed in a common location called library.
- All C standard library functions are declared by using many header files.
- C library functions are grouped into several categories, including:
- Input/Output Functions: printf, scanf, fopen, fclose, fread, fwrite, fgets, fputs, and more.
- String Manipulation Functions: strcpy, strcat, strcmp, strlen, strchr, strstr, and more.
- Mathematical Functions: sin, cos, tan, exp, log, sqrt, pow, and more.

Preprocessor Directives

- Preprocessor directives are the statements that begins with a # symbol.
- These statements are processed by the preprocessor before the actual compilation of the program.
- The C preprocessor is a micro-processor that is used by compiler to transform your code before compilation.
- Here are some examples of commonly used preprocessor directives in C:

- **#include:** This directive is used to include header files in a C program.

- **#define:** This directive is used to define constants in a C program. For example, #define PI 3.1415926 defines the constant PI with the value 3.1415926.
- **#error:** This directive is used to generate an error message during compilation. For example, #error "Invalid input" generates an error message with the text "Invalid input".

Escape Sequence

- Character combinations consisting of a backslash (\) followed by a letter or by a combination of digits are called "escape sequences."
- Here are some common escape sequences in C:

1. \n : newline
2. \t : horizontal tab
3. \\ : backslash
4. \' : single quote
5. \" : double quote
6. \r : carriage return
7. \b : backspace
8. \f : form feed

Here's an example program that demonstrates the use of escape sequences:

```
#include <stdio.h>
int main()
{
    printf("Hello,\tWorld!\n");
    // Output: Hello,   World!

    printf("My name is \"John\".\n");
    // Output: My name is "John".

    printf("This is a backslash: \\ \n");
    // Output: This is a backslash: \

    return 0;
}
```

Comments

- In C programming, comments are used to add explanations or documentation to the source code.
- Comments are ignored by the compiler and have no effect on the execution of the program.
- There are two types of comments in C:

Single-line comments: These comments start with // and continue until the end of the line. For example:

```c
// This is a single-line comment
int x = 10; // This is another single-line comment
```

Multi-line comments: These comments start with /* and end with */.

- For example:

```c
/* This is a multi-line comment
    that can span multiple lines
    and include multiple statements. */

int x = 10;  /* This is another multi-line
```