

Day 1

Topics

Algorithm and Flowchart, Translator and its types, Applications of C programming , transition from algorithm to program, Syntax, logical errors and Runtime errors, Keywords, identifiers, constant, Data types, Type conversion, Operators and their types, Precedence and associativity,

ALGORITHM

- Algorithm refers to the logic of the program. It is a step by step description of how to arrive at the solution of the problem.
- An algorithm is a complete, detailed and precise step by step method for solving a problem independently of the hardware and software.

Characteristics of a good algorithm are:

- **Input** – the algorithm receives input.
- **Output** – the algorithm produces output.
- **Finiteness** – the algorithm stops after a finite number of instructions are executed.
- **Precision** – the steps are precisely stated (defined).
- **Uniqueness** – results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
- **Generality** – the algorithm applies to a set of inputs.






Example: Algorithm to find sum of two numbers:

1. Begin
2. Input the value of A and B
3. $SUM = A + B$
4. Display SUM
5. End.

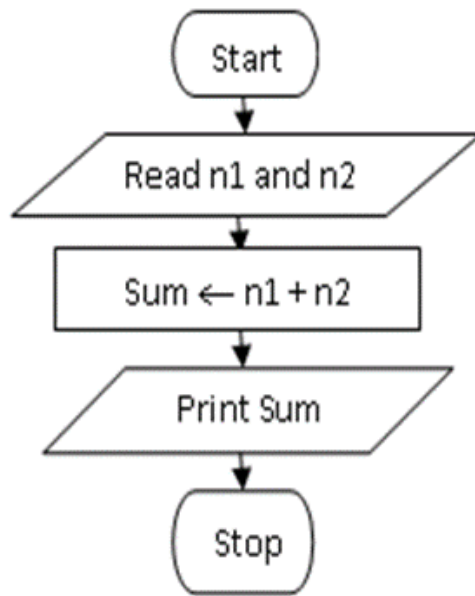
FLOWCHART

- A flowchart is a pictorial representation of an algorithm or process.

Symbols used:

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

Example: Flow-Chart to find sum of two numbers



Questions: Write an algorithms and design flow charts for the following:

1. Find average of three numbers.
2. Swap two numbers using: i. Using third variable ii. Without using third variable
3. Find area and perimeter of a circle.
4. Check if a number is divisible by 7 or not.
5. Among three numbers: a,b and c, find the greatest number.
6. Check if a number is an Armstrong number or not.
7. Find Factorial of a number.

COMPUTER LANGUAGES

- Computer language or programming language is a coded syntax used by computer programmers to communicate with a computer.
- There are three types of Computer Languages
 - Machine Language
 - Assembly Language
 - High level Language

Machine Language

- It is a low level language.
- Instructions written in the form of 0 and 1.
- Communicate directly with the computer.
- Machine language or machine code is the native language directly understood by the computer's central processing unit or CPU.
- This type of computer language is not easy to understand for humans, as it only uses a binary system.
- **Advantages:**
 - Computation speed is very fast.
 - Directly understandable by computer.
- **Disadvantages:**
 - Writing a program is very time consuming.
 - Error correction is a tedious process.

Assembly Level Language

- It is also a low level language.
- Written in the form of symbolic codes (mnemonics).
- Symbolic codes are like- ADD, SUB, MUL, DIV, LOAD, STORE, etc.
- Assembly Level Language is a set of codes that can run directly on the computer's processor.
- This type of language is most appropriate in writing operating systems and maintaining desktop applications.
- With the assembly level language, it is easier for a programmer to define commands. It is easier to understand and use as compared to machine language.
- **Advantages**
 - Easier to understand as compared to machine code.
 - Easy to locate and correct errors.
- **Disadvantages**
 - Like machine language it is also machine dependent.
 - Programmers should have knowledge of hardware.

High Level Language

- High Level Languages are user-friendly languages which are similar to English with vocabulary of words and symbols.
- These are easier to learn and require less time to write.
- They are problem oriented rather than 'machine' based.
- Programs written in a high-level language can be translated into many machine languages and therefore can run on any computer for which there exists an appropriate translator.
- Examples: COBOL, FORTRAN, PASCAL, C, C++, JAVA etc.
- **Advantages**
 - User-friendly.
 - Easier to learn.
 - Require less time to code.
 - Easier to maintain.
- **Disadvantages**
 - Slower than low level language.
 - Needs a translator.

LANGUAGE TRANSLATORS

- Used to convert one programming language to another.
- Computers understand only Machine Language. Most of the programming is done in High level languages as it is much easier to code in these languages.
- The need is there to translate a high level language program into machine language.
- The system programs which perform this job are called language translators.
- They are classified into three categories
 - Assembler
 - Interpreter
 - Compiler

Assembler

- This language processor converts the program written in assembly language into machine language.

Interpreter

- It converts High level language into low level.
- It translates line by line.

Compiler

- It converts High level language into low level.
- It converts entire program at once.

Applications of C programming

Applications of C programming is a powerful and versatile language used in various domains due to its efficiency, performance, and flexibility. Here are some key applications of C programming:

1. System Software Development

- **Operating Systems:** Many operating systems, including Unix, Linux, and parts of Windows, are written in C. C provides the low-level access to memory and system processes needed for OS development.
- **Embedded Systems:** C is widely used in embedded systems, which are specialized computing systems that perform dedicated functions within larger systems (e.g., automotive systems, medical devices, consumer electronics).

2. Application Software Development

- **Desktop Applications:** C is used for developing high-performance applications such as graphics editors (e.g., Adobe Photoshop), office suites, and video editors.

- **Compilers and Interpreters:** Compilers for various programming languages, including the C compiler itself, are often implemented in C due to its efficiency and control over system resources.

3. Hardware Drivers and Firmware

- **Device Drivers:** C is used to develop drivers that allow the operating system to communicate with hardware devices like printers, display adapters, and storage devices.
- **Firmware:** Firmware, which provides low-level control for the device's specific hardware, is often written in C to ensure efficient operation and compatibility.

4. Network Programming

- **Network Protocols:** Many network protocols and their implementations are written in C. This includes protocols used for communication over the internet (e.g., TCP/IP).
- **Server and Client Software:** Servers and client software that handle network communications, such as web servers and mail servers, often utilize C for its performance benefits.

5. Game Development

- **Game Engines:** C and its derivative C++ are used in the development of game engines that require real-time processing and high performance.
- **Graphics and Simulation:** C is used to write high-performance graphics and simulation code that is fundamental in game development and other graphic-intensive applications.

6. Database Systems

- **Database Management Systems (DBMS):** Several DBMS systems, such as MySQL, are written in C. C allows fine-tuning of performance-critical database operations and memory management.

7. Scientific Computing

- **Numerical Analysis and Computation:** C is used in scientific computing for tasks that require intensive numerical computation and precise control over hardware.
- **Simulation and Modeling:** Scientific simulations and models that require high performance and efficiency are often written in C.

8. Embedded Software in Consumer Electronics

- **Telecommunications:** C is used in the software that runs on telecommunications devices, including modems and routers.
- **Consumer Devices:** Many consumer electronics, such as smart TVs, microwave ovens, and washing machines, use C for their embedded software.

9. IoT (Internet of Things)

- **IoT Devices:** C is commonly used in programming IoT devices that require efficient code to operate with limited resources (e.g., sensors, actuators, smart home devices).

10. Utility Programs

- **Text Editors:** Many text editors, which are critical tools for software development, are written in C.
- **System Utilities:** Utilities like disk defragmenters, file managers, and performance monitoring tools are often developed using C.

Transitioning from an algorithm to a program

1. Define the Problem

Before writing an algorithm, clearly define the problem you are trying to solve. Understand the inputs, desired outputs, and any constraints or requirements.

2. Design the Algorithm

An algorithm is a step-by-step procedure to solve a problem. It should be clear, unambiguous, and finite. Here are the steps to design an algorithm:

Identify Inputs and Outputs: Specify what data you will receive as input and what data you need to output.

Outline Steps: Break down the problem into a series of logical steps. Use pseudocode or flowcharts to represent these steps.

Consider Edge Cases: Think about possible edge cases and how your algorithm will handle them.

Iterate and Refine: Review and refine your algorithm to ensure it covers all aspects of the problem.

3. Choose a Programming Language

Select an appropriate programming language based on factors such as performance requirements, ease of use, and specific features needed for the task.

4. Write the Program

Translate the algorithm into a program using the syntax and features of your chosen programming language. Here's how you can do this step-by-step:

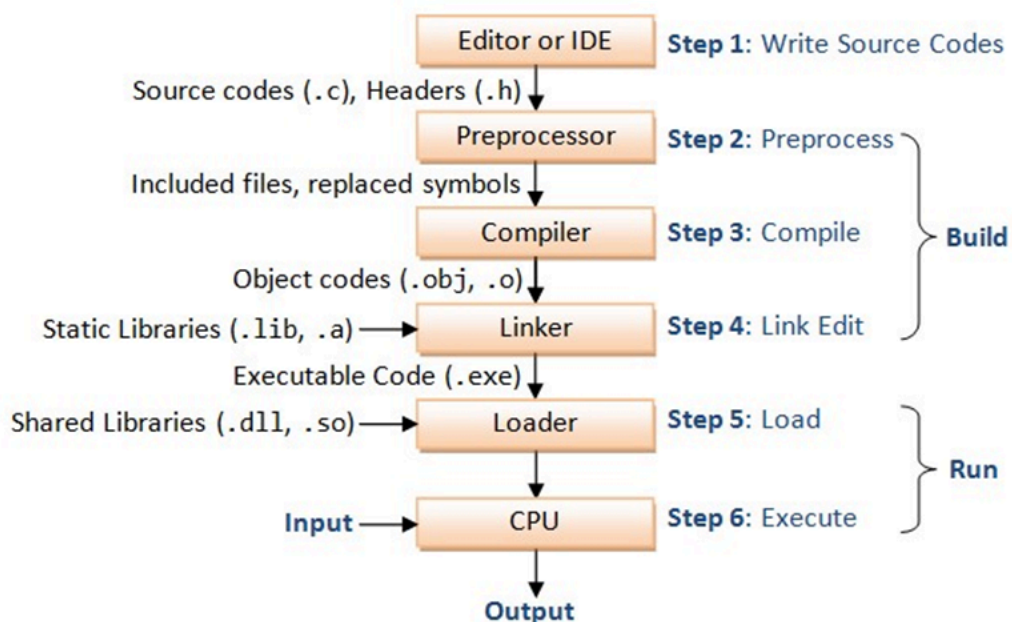
Translate the Algorithm to Code

Convert each step of your algorithm into corresponding code. For the example algorithm, here's how you might write it in Python and C.

5. Test the Program

Test the program if it matches with all the test cases.

Steps in Program Execution



Syntax errors, logical errors and Runtime errors

Syntax errors

- Syntax errors occur when the rules of the C language are violated. These are detected by the compiler during the compilation process.

- Common Causes:
 - Misspelled keywords
 - Missing semicolons at the end of statements
 - Mismatched parentheses, brackets, or braces
 - Incorrect use of operators

Logical errors

- Logical errors occur when the program compiles and runs, but the output is not as expected. These are errors in the logic or algorithm used in the program.
- Common Causes:
 - Incorrect algorithm implementation
 - Wrong conditions in loops or if statements
 - Incorrect use of variables or operators

Runtime Errors

- Runtime errors occur while the program is running. These errors are not detected during compilation and typically cause the program to terminate abnormally.
- Common Causes:
 - Division by zero
 - Null pointer dereferencing
 - Array index out of bounds
 - Memory allocation failures

C programming:

Example 1: use semicolon ; at the end of every statement.

```
#include<stdio.h>

void main() {
    printf("Hello World");
}
```

Example 2: variable declaration is important before using them. Use %d for int and %f for float.

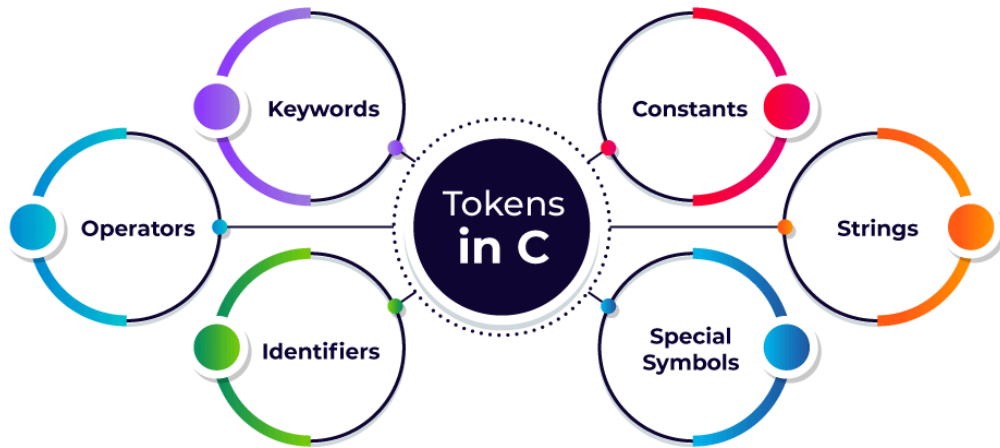
```
#include<stdio.h>

void main() {
    int x =7;
    float y = 5.25;
    printf("value of x = %d \n", x);
    printf("value of y = %f", y);
}
```

Features of C

1. Procedural Language
2. Fast and Efficient
3. Modularity
4. General-Purpose Language
5. Rich set of built-in Operators
6. Libraries with Rich Functions
7. Middle-Level Language: C is High level language with some features of low level language, hence, it is called middle level language.
8. Portability: Programs that are written in C language can run and compile on any system with either no or small changes.
9. Easy to Extend

C Tokens



Keywords

Keywords are predefined or reserved words that have special meanings to the compiler. These are part of the syntax and cannot be used as identifiers in the program. A list of keywords in C or reserved words in the C programming language are mentioned below:

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

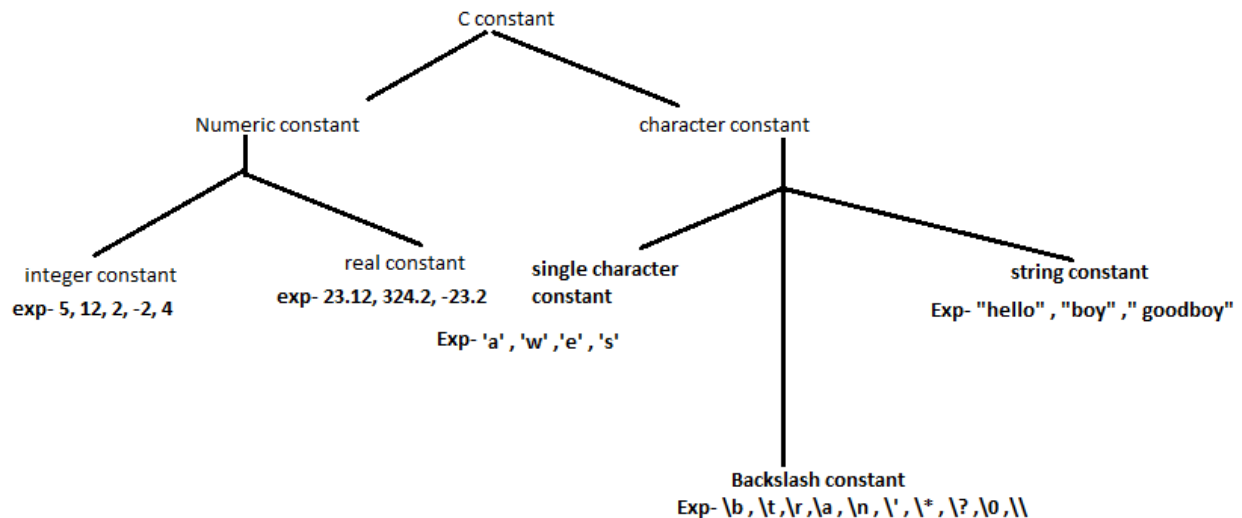
Identifier

Identifier is a name used to identify a variable, function, or array.

Rules for constructing C identifiers

1. Identifiers must start with a letter or an underscore.
2. Identifiers can only contain letters, digits, and underscores.
3. No Commas or blank spaces within an identifier.
4. Identifiers are case-sensitive.
5. Identifiers cannot be a reserved keyword.
6. The length of the identifiers should not be more than 31 characters.

C-Constants



Operators

C language provides a wide range of operators that can be classified into 6 types based on their functionality:

1. Arithmetic Operators

2. Assignment Operators
3. Increment Decrement Operators
4. Relational Operators
5. Logical Operators
6. Bitwise Operators
7. Conditional Operator
8. Special Operator

1. Arithmetic Operators:

For a=9, b=3

OPERATION	OPERATOR	SYNTAX	COMMENT	RESULT
Multiply	*	a * b	result = a * b	27
Divide	/	a / b	result = a / b	3
Addition	+	a + b	result = a + b	12
Subtraction	-	a - b	result = a - b	6
Modulus	%	a % b	result = a % b	0

2. Assignment Operators

OPERATOR	MEANING	EXAMPLE
=	ASSIGNMENT	A=5 WILL ASSIGN 5 TO A

3. Increment Decrement Operators

OPERATOR	MEANING	EXAMPLE
++	INCREMENT	A++ INCREMENTS THE VALUE OF A TO 1
--	DECREMENT	A- DECREMENTS THE VALUE OF A TO 1

4. Relational operators

OPERATOR	MEANING	EXAMPLE
<	LESS THAN	3 < 5 GIVES 1
>	GREATER THAN	7 > 9 GIVES 0
>=	LESS THAN OR EQUAL TO	100 >= 100 GIVES 1
<=	GREATER THAN EQUAL TO	50 >=100 GIVES 0
==	IS EQUAL TO	5==5 GIVES 1
!=	NOT EQUAL TO	5!=5 GIVES 0

5. Logical Operators

OPERATOR	MEANING	EXAMPLE
&&	LOGICAL AND	(6>5) && (7>9) GIVES 0
	LOGICAL OR	(6>5) (7>9) GIVES 1
!	LOGICAL NOT	!(6>5) GIVES 0

6. Bitwise Operators

OPERATOR	MEANING	EXAMPLE
&	BITWISE AND	5 & 6 GIVES 4
	BITWISE OR	5 6 GIVES 7
^	BITWISE X-OR	5 ^ 6 GIVES 3
<<	BITWISE LEFT SHIFT	5<<2 GIVES 20
>>	BITWISE RIGHT SHIFT	5>>2 GIVES 1
~	BITWISE NOT	~6 GIVES -7

How 5 & 6 gives 4?

Convert decimal 5 and 4 in binary. Perform bitwise AND operation. Convert result back into Decimal.

5 = 0101
6 = 0110
5 & 6 = 0100 = 4

7. Conditional Operator

Also called ternary operator as it takes 3 operands.

exp1 ? exp2 : exp3

Example:

```
#include<stdio.h>

void main() {
    int a = 12, b = 7, c;
    c = a > b ? a : b;
    printf("value of c = %d ", c);
}
```

8. Special Operators

These will be discussed later when we will be using them in upcoming chapters.

- A. , comma operator
- B. . member access operator
- C. sizeof() operator
- D. * dereferencing operator

Operator Precedence and Associativity

- Both are used to determine the order of evaluation of Operators in an expression.
- Precedence is always applied before associativity.
- Associativity can be left to right or right to left.

Example1: (precedence)

3 + 7 * 2 (Here * has higher precedence than +. So * will be evaluated first)

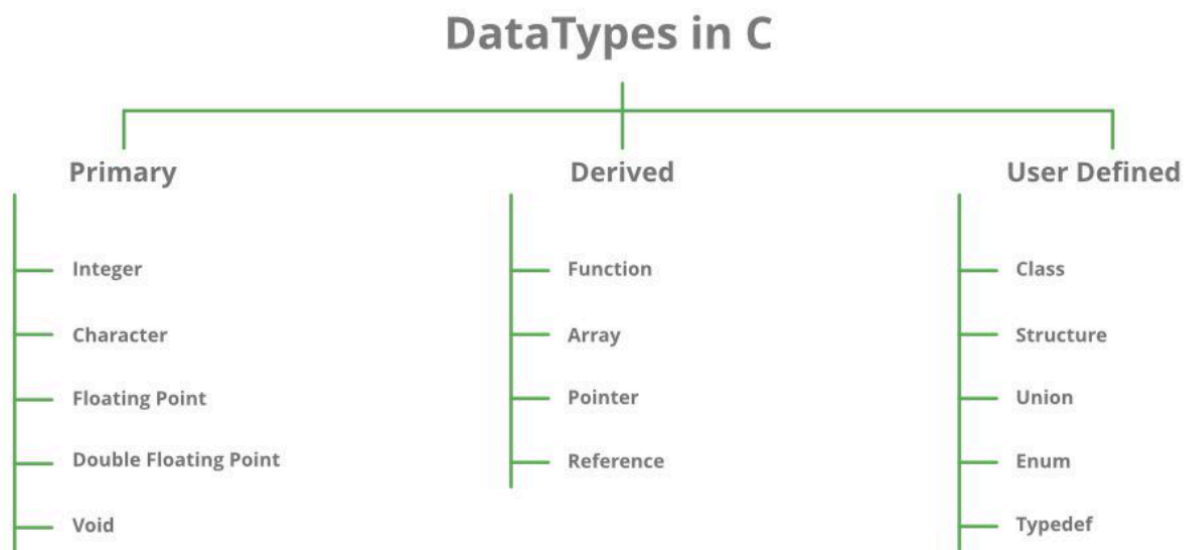
Example2: (associativity)

3 - 4 + 6 (Here + and - both have the same precedence so according to associativity of arithmetic operators, which is left to right, in this example - will be evaluated before +)

Example3: (associativity)

3 + 4 - 6 (Here + and - both have the same precedence so according to associativity of arithmetic operators, which is left to right, in this example + will be evaluated before -)

Data Types

**Primary or Fundamental data types:**

(Derived and user defined data types will be discussed later during the course)

Data Type	Size (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
long long int	8	$-(2^{63})$ to $(2^{63})-1$	%lld
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	1.2E-38 to 3.4E+38	%f
double	8	1.7E-308 to 1.7E+308	%lf

long double	16	3.4E-4932 to 1.1E+4932	%Lf
--------------------	----	---------------------------	-----

Type Conversion in C

- Type conversion in C is the process of converting one data type to another.

Type of Type conversion

There are two types of Conversion:

1. Implicit Type Conversion

- Also known as 'automatic type conversion'.
- Done by the compiler on its own, without any external trigger from the user.
- Generally takes place when in an expression more than one data type is present. In such conditions type conversion (type promotion) takes place to avoid loss of data.

Example:

```
#include <stdio.h>
int main()
{
    int x = 10; // integer x
    char y = 'a'; // character c

    // y implicitly converted to int. ASCII
    // value of 'a' is 97
    x = x + y;

    printf("x = %d, x);
    return 0;
}
```

2. Explicit Type conversion

- This process is also called type casting and it is user-defined.
- Here the user can typecast the result to make it of a particular data type.
- Syntax

(type) expression

Example :

```
#include <stdio.h>
int main()
{
    double x = 1.2;

    // Explicit conversion from double to int
    int sum = (int)x + 1;

    printf("sum = %d", sum);

    return 0;
}
```

Escape Sequence

- Escape sequences in C are combinations of characters that represent special characters or actions.
- Escape sequences start with a backslash (\) followed by a character or combination of characters.

Escape Sequences	Meaning
\'	Single Quote
\"	Double Quote
\\	Backslash
\0	Null
\a	Bell
\b	Backspace
\f	form Feed
\n	Newline
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab

Programs for day1 (Hands-on):

1. Find average of three numbers
2. Find Area and perimeter of Circle.
3. Find Area of triangle
 - a. using base and height
 - b. using Heron's formula
4. Convert Fahrenheit temperature into celsius. $C = (F-32) * 5 / 9$
5. Program to swap two numbers.