Contents

- Introduction to Programming
- Source code Vs Object Code
- Algorithm
- Flowchart
- Pseudo code
- C features
- Tokens
- Operators in C

Introduction To Programming

- A **program** is a set of instructions that tells the computer to do various things.
- A **programming language** is a vocabulary and set of grammatical rules for instructing a computer to perform specific tasks.
- **Computer programming** is the process of designing and building an executable computer program for accomplishing a specific task.
- The **source code** of a program is written in one or more programming languages.
- **Object Code** is Machine code generated by language translator.

Source Code Vs Object Code

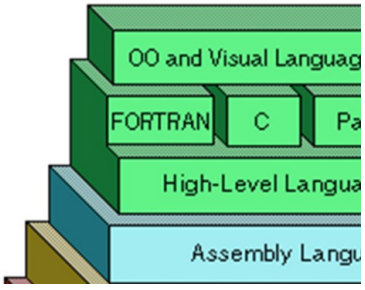
Source Code

- Source code is in the form of Text form.
- Source code is Human Readable Code.
- Source code is Generated by Human or Programmer.
- Source code is received by Compiler as an Input.

Object Code

- Object Code is in the form of Binary Numbers.
- Object Code is in Machine Readable formats.
- Object Code is Generated by Compiler as a Output.

Levels of Programming Languages



Levels of Programming Languages

- **MACHINE LANGUAGE:** It is also called Low Level Language. The language closest to the hardware. Each unique computer has a unique machine language. A machine language program is **made up of a series of binary patterns** (e.g., 01011100). Machine language programs are directly *executable*. Programming in machine language is difficult for the human programmer.
- **ASSEMBLY LANGUAGE:** . The machine language instructions are replaced with simple **pneumonic abbreviations (e.g., ADD, MOV).** Thus assembly languages are unique to a specific computer (machine). Translation is accomplished by a computer program known as an **Assembler.**
- **HIGH LEVEL LANGUAGES:** High-level languages, like **C,C++, JAVA** etc., are more English-like. High-level languages also require translation to machine language before execution. This translation is accomplished by either **a compiler or an interpreter.**
- ***Compilers*** translate the entire source code program before execution.(Eg: C, C++)
- ***Interpreters*** translate source code programs one line at a time. (Eg: Python). Interpreters are more interactive than compilers.

Algorithm

- To write a logical step-by-step method to solve the problem is called **algorithm.**
- An algorithm is a procedure for solving problems.
- Algorithms can be represented by natural languages, pseudo code and flowcharts, etc.

Qualities of a good algorithm

- Inputs and outputs should be defined precisely.
- Each steps in algorithm should be clear and unambiguous.
- Algorithm should be most effective among many different ways to solve a problem.
- An algorithm shouldn't have computer code. Instead, the algorithm should be written in such a way that, it can be used in similar programming languages.

Example (Largest of Three Numbers)

```
Step 1: Start
Step 2: Declare variables a,b and c
Step 3: Read variables a,b and c.
Step 4: If a>b
            If a>c
                Display a is the
            Else
                Display c is the
        Else
            If b>c
```

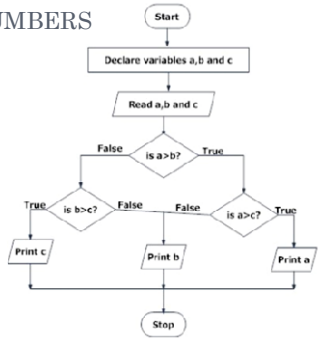
FLOWCHART

- A flowchart is the graphical or pictorial representation of an algorithm with the help of different symbols, shapes and arrows in order to demonstrate a process or a program.
- Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

SYMBOLS IN FLOWCHART

Symbol	Purpose	Description
→	Flow line	Used to indicate the flow of l
⬭	Terminal(Stop/Start)	Used to represent start and e
▱	Input/Output	Used for input and output op
▭	Processing	Used for airthmetic operations

EXAMPLE
FLOWCHART TO FIND LARGEST OF THREE NUMBERS



PSEUDO CODE

- **Pseudocode** is an informal high-level description of the operating principle of a computer program or other algorithm.
- Pseudocode is not actual programming language. It uses short phrases to write code for programs before you actually create it in a specific language.
- No standard for pseudocode syntax exists, as a program in pseudocode is not an executable program.
- It is a combination of human language and programming language

EXAMPLE

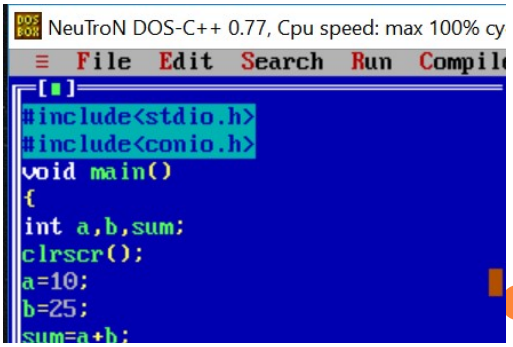
If
    student's grade is greater than or equal to 60
    Print "passed"
else
    Print "failed"
endif

```
IF (A > B)
    THEN Print A + "i
    ELSE Print B + "i
```

GENERAL ASPECT OF 'C'

- C is a procedural programming language. It was initially developed by Dennis Ritchie between 1969 to 1973 at Bell Telephone Laboratories, Inc.
- It was mainly developed as a system programming language to write operating.
- C is a High level , general –purpose structured programming language. Instructions of C consists of terms that are very closely same to algebraic expressions, consisting of certain English keywords such as if, else, for ,do and while
- C contains certain additional features that allows it to be used at a lower level , acting as bridge between machine language and the high level languages.

C WINDOW



ADVANTAGES OF C

- Easy to learn
- Structured language
- It produces efficient programs
- It can handle low-level activities
- It can be compiled on a variety of computer platforms.

THE CHARACTER SET OF 'C'

Character set is a set of alphabets, letters and some special characters that are valid in C language.

Alphabets a to z, A to Z

Numeric 0,1 to 9

Special Symbols {,},[,],?,+,-,*,/,%,!,;,and more

TOKENS: The words formed from the character set are building blocks of C and are sometimes known as tokens. These tokens **represent the individual entity of language.**

Types of Tokens:

1) Identifiers 2)Keywords 3)Constants

4) Operators 5)Punctuation Symbols

IDENTIFIERS

- Identifiers are names for entities in a C program, such as variables, arrays, functions, structures, unions and labels.
- An identifier can be composed only of uppercase, lowercase letters, underscore and digits, but should start only with an alphabet or an underscore.
- A 'C' program consist of two types of identifiers , user defined and system defined.
- Both Upper and lowercase letters can be used.

RULES FOR CONSTRUCTING IDENTIFIERS (NAMING RULES)

- The first character in an identifier must be an alphabet or an underscore and can be followed only by any number alphabets, or digits or underscores.
- They must not begin with a digit.
- Uppercase and lowercase letters are distinct. That is, identifiers are case sensitive.
- Commas or blank spaces are not allowed within an identifier.
- Keywords cannot be used as an identifier.
- Identifiers should not be of length more than 31 characters.
- Identifiers must be meaningful, short, quickly and easily typed and easily read.

EXAMPLES

- **Valid identifiers**:
  total
  sum
  average
  _x
  y_
  mark_1
  x1
- **Invalid identifiers:**
  - begins with a digit 1x
  - reserved word char
  - special character x+y

KEYWORDS

- Keywords are reserved words of the language.
- Keywords are nothing but system defined identifiers.
- They have specific meaning in the language and cannot be used by the programmer as variable or constant names.
- C is case senitive, Keywords are used in lower case.
- 32 Keywords in C Programming.

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

Differentiate between Keywords words and identifiers

Keyword	Identifier
Predefined-word	User-defined word
Must be written in lowercase only	Can written in lowercase and uppercase
Has fixed meaning	Must be meaningful in the program
Whose meaning has already been explained to the C compiler	Whose meaning not explained to the C compiler
Combination of alphabetic characters	Combination of alphanumeric characters
Used only for it intended purpose	Used for required purpose
Underscore character is not considered as a letter	Underscore character is considered as a letter

VARIABLES AND CONSTANTS

Variable

- In programming, a variable is a container (storage area) to hold data.
- To indicate the storage area, each variable should be given a unique name (identifier).

int playerScore = 95;

Here, playerScore is a variable of integer type. The variable is assigned value: 95.

Constants/Literals

- Constants refer to fixed values that the program may not alter during its execution. These fixed values are also called **literals**.
- Constants can be of any of the basic data types like *an integer constant, a floating constant, a character constant, or a string literal.*
- For example: 1, 2.5, "C programming is easy", etc.
- As mentioned, an identifier also can be defined as a constant.

const double PI = 3.14

TYPES OF CONSTANTS

Integer constants

- A integer constant is a numeric constant (associated with number) without any fractional or exponential part. There are three types of integer constants in C programming:
  - decimal constant(base 10)
  - octal constant(base 8)
  - hexadecimal constant(base 16)

Floating-point constants

- A floating point constant is a numeric constant that has either a fractional form or an exponent form. For example: 2.0,0.0000234,-0.22E-5

Character constants

- A character constant is a constant which uses single quotation around characters. For example: 'a', 'l', 'm', 'F'

String constants

- String constants are the constants which are enclosed in a pair of double-quote marks. For example: "good" ,"x","Earth is round\n"

Escape Sequences

Sometimes, it is necessary to use characters which cannot be typed or has special meaning in C programming. For example: newline(enter), tab, question mark etc. In order to use these characters, escape sequence is used.

- For example: \n is used for newline. The backslash (\ ) causes "escape" from the normal way the characters are interpreted by the compiler.Escape

Sequences	Character
\b	Backspace
\f	Form feed
\n	Newline
\r	Return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\'	Single quotation mark
\"	Double quotation mark
\?	Question mark
\0	Null character

Operators in C: An operator is a symbol which operates on a value or a variable. For example: + is an operator to perform addition.

C programming has wide range of operators to perform various operations. For better understanding of operators, these operators can be classified as:

- Arithmetic Operators
- Increment and Decrement Operators
- Assignment Operators
- Relational Operators
- Logical Operators
- Conditional Operators
- Bitwise Operators
- Special Operators

Arithmetic Operator

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	remainder after division(modulo division)

Increment and Decrement Operators

1. C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.
2. Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1.
3. These two operators are unary operators, meaning they only operate on a single operand.

eg. int a=10, b=100
++a = 11
--b = 99

C Assignment Operators

- An assignment operator is used for assigning a value to a variable. The most common assignment operator is =

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

C Relational Operators

- A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.
- Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 returns 0
>	Greater than	5 > 3 returns 1
<	Less than	5 < 3 returns 0
!=	Not equal to	5 != 3 returns 1
>=	Greater than or equal to	5 >= 3 returns 1
<=	Less than or equal to	5 <= 3 return 0

Logical Operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

Bitwise Operators

- Bitwise operator works on bits and perform bit-by-bit operation.

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = -60, i.e,. 1100 0100 in 2's complement form.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

Special Operators

Operator	Description	Example
sizeof()	Returns the size of a variable.	sizeof(a), where a is integer, will return 4.
&	Returns the address of a variable.	&a; returns the actual address of the variable.
*	Pointer to a variable.	*a;
? :	Conditional Expression.	If Condition is true ? then value X : otherwise value Y

References

- https://www.programiz.com/article/algorithm-programming
- https://www.tutorialspoint.com/cprogramming/c_operators.htm
- LET US C, YASHWANT KANETKAR
- Basics of C Programming, 2011, by J.B. Dixit