Structured Programming Language

Sabbir Hossen

Lecturer

Department of CSE

City University

Email: sabbirhossen00@gmail.com

LinkedIn: https://www.linkedin.com/in/sabbir-hossen-71181115a

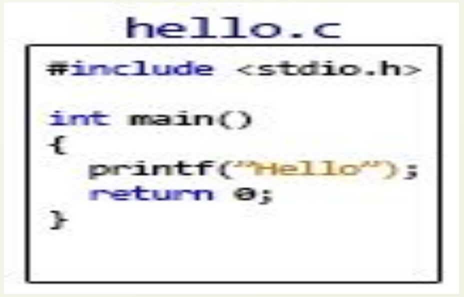
No: 01757007366 (Emergency Call’s only)

Program

Program: A program is a set of instructions that tells the computer (Central Processing Unit) what to do.

Source code: Human readable format of a program is called its Source code. We can modify the source code any number of times.

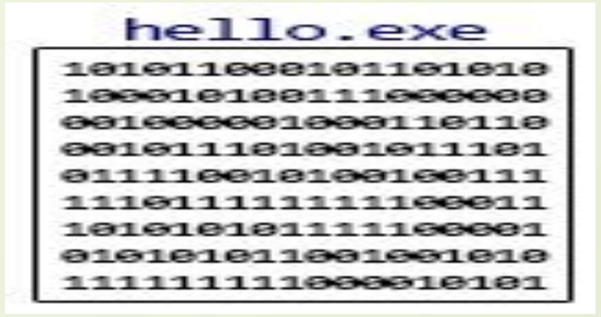
Executable: Binary format or machine readable format of any program is called as its Executable. They are the compiled format of source code and are generally in binary format. We cannot modify the executable of any program.


hello.c

```
#include <stdio.h>

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

Source code


hello.exe

```
101011100010111010101
100010101001111000000
001000000010000110110
001011101001011110101
011111001010100100111
111011111111111100011
101010101011111100001
010101010110010001010
111111111100000101101
```

Executable

Programming language

Programming language: It is a language that provides a medium of communication between computer and human.

Example: C, C++, Java, C#, PHP, JavaScript, Python, Ruby etc.

Categorization of programming language:

1. Low Level Language(LLL)

- Low level languages are programming languages that are directly understood by the computer or requires less interpretation.
- Machine language and Assembly language.
- Low level languages are easily understood by computers but are hard to figure out by humans.
- Processing of any low level languages is much faster than High level languages.
- Examples: BCPL, ALGOL etc.

Programming language

2. High Level Language(HLL)

- High level languages are programming languages that are easily understood by humans and are mostly written in English like statements.
- High level languages requires translation into machine language or low level language hence requires compilers, interpreters to accomplish this translation job.
- Processing of any high level language takes more time compared to low level language because of time elapsed in converting high level instructions into low level instructions.
- Examples: C, C++, Java, C# etc.

Tokens

Tokens: Smallest individual element of a program is called as Token. Everything you see inside a program is a token. There are generally five types of tokens:

- Keyword
- Identifier
- Operator
- Separator
- Literal

Keyword: Keyword is a reserved word whose meaning is already defined by the programming language. We cannot use keyword for any other purpose inside programming. Every programming language have some set of keywords.

Examples: int, do, while, void, return etc(Note: These keywords are common to C).

Keywords

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

Identifier

Identifiers: Identifiers are the name given to different programming elements. Either name given to a variable or a function or any other programming element.

All follow some basic naming conventions listed below:

- Keywords must not be used as an identifier.
- Identifier must begin with an alphabet(a-z A-Z) or an underscore(_) symbol.
- Identifier can contains alphabets(a-z A-Z), digits(0-9) and underscore(_) symbol.
- Identifier must not contain any special character(e.g. !@$*.'[] etc) except underscore(_).

Examples of some valid identifiers:

- num, Num, _num, _Num, num1, Num1, _num1, _Num1, _1num, _1Num, _num_, number_to_add etc.

Examples of some invalid identifiers:

- 1num, number to add, 1_num, num-to-add, num@ etc.

Operators

Operator: Operators are symbols given to any mathematical or logical operation. C language provides a rich set of operators. There are basically seven types of operators in C:

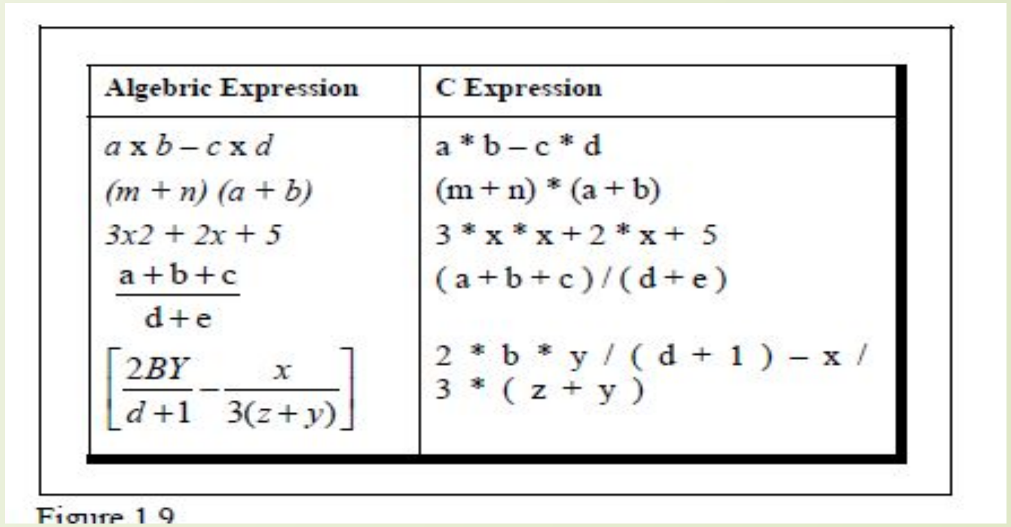
- Arithmetic operator **:** + - * / %
- Assignment operator**:** =
- Relational operator**:** **>** , **<** , **>=**, **<=**, **= =**, **!=**
- Logical operator **:** **&&** || **!**
- Bitwise operator **:** >> << **&** | ^ ~
- Increment/Decrement operator **:** ++ **--**
- Conditional/Ternary operator **:** **? :**
- Miscellaneous operator sizeof()

Operators

Arithmetic operator

Arithmetic operator are used to perform basic arithmetic operations.

Operator	Description	Example
+	Adds two operands.	a + b gives 15
-	Subtracts second operand from first.	a - b gives 5
*	Multiplies two operands.	a * b gives 50
/	Divides two operands.	a / b gives 2
%	Modulus operator divides the first operand from second and returns the remainder. It is generally used for checking divisibility.	a % b gives 0 (As 10/5 will have 0 remainder)



Algebraic Expression	C Expression
$a \times b - c \times d$	a * b – c * d
$(m + n)(a + b)$	(m + n) * (a + b)
$3x2 + 2x + 5$	3 * x * x + 2 * x + 5
$\frac{a+b+c}{d+e}$	(a + b + c) / (d + e)
$\left[\frac{2BY}{d+1} - \frac{x}{3(z+y)}\right]$	2 * b * y / (d + 1) – x / 3 * (z + y)

Figure 1.9

Assignment operator

Assignment operator is used to assign value to a variable. The value is assigned from right to left.

Operator	Description	Example
=	Assigns value from right operand to left operand.	a = 10 will assign 10 in a

Operators

Relational operator

Relational operator is used to check relation between any two operands. Whether any of them is greater, equal or not equal.

Operator	Description	Example
<	If value of left operand is greater than right, returns true else returns false	(a > b) will return true
>	If value of right operand is greater than left, returns true else returns false	(a < b) will return false
= =	If both operands are equal returns true else false	(a == b) will return false
!=	If both operands are not equal returns true else false.	(a != b) will return true
>=	If value of left operand is greater or equal to right operand, returns true else false	(a >= b) will return true
< =	If value of right operand is greater or equal to left operand, returns true else false	(a <= b) will return false

Operators

Logical operator

Logical operator are used to combine two boolean expression together and results in a single boolean value according to the operand and operator used.

Operator	Description	Example
&&	Used to combine two expressions. If both operands are true or Non-Zero, returns true else false	((a>=1) && (a<=10)) will return true since (a>=1) is true and also (a<=10) is true.
||	If any of the operand is true or Non-zero, returns true else false	((a>1) || (a<5)) will return true. As (a>1) is true. Since first operand is true hence there is no need to check for second operand.
!	Logical NOT operator is a unary operator. Returns the complement of the boolean value.	(!(a>1)) will return false. Since (a>1) is true hence its complement is false.

Operators

Increment/Decrement operator

Increment/Decrement operator is a unary operator used to increase an integer value by 1 or decrease it by 1. Increment/decrement operator are of two types Postfix and Prefix.

Operator	Description	Example
++	Increment operator will add 1 to an integer value.	a++ will give 11 ++a will also give 11
--	Decrement operator will subtract 1 from an integer value.	a-- will give 9 --a will also give 9

Operators

Separator

Separators are used to separate different programming elements. The various types of separators used in programming are:

(Space) \t(Tab) \n(New line) . , ; () {} []