Basic Input / Output Functions

There are some input & output functions which are given below:

1. **getchar():** Reading a single character can be done by using the function getchar() . It is an input functions.

   General Format:

   variable_name = getchar();

   Example:
   ```
   char name;
   name = getchar();
   ```

   where, the variable_name is a type char variable containing a character.

2. **putchar():** Writing characters one at a time to the terminal by using the function putchar(). It is an output functions.

   General Format:

   putchar(variable_name);

   Example:
   ```
   answer = 'Y';
   putchar(answer);
   ```

   where, variable_name is a type char variable containing a character. This statement display the character contained in the variable_name at the terminal.

3. **scanf():** Accepting / Reading the input data which is entered by the keyboard by using the scanf() functions. It is an input functions.

   General Format:

   scanf("control string", arg1, arg2, arg3, . . . argn);

   Example: scanf("%d %d", &a, &b);

   where, the control string specifies the field format in which the data is to be entered and arguments arg1, arg2, arg3, . . . argn specify the address of locations where the data is stored. Control string and arguments are separated by commas.

4. **printf():** Printing / Writing the output data to the monitors by using the printf() functions. It is an output functions.

   General Format:

   printf("control string", arg1, arg2, arg3, . . . argn);

Example: printf(" The output is: %d %d", a, b);

Where, control string indicates how many arguments follow and what their types are. The arguments arg1, arg2, arg3, . . . argn are the variables whose values are formatted, and printed according to the specifications of the control string.

Note: 1. %wd = এখানে, %d মানে আমরা Interger type এর data নিব, w মানে আমরা কতসংখ্যার number নিব তা বোঝায়।

2. %.wf = আমরা floating point number এর ক্ষেত্রে use করব। w মানে আমরা দশমিকের পর কতগুলো number নিব তা বোঝায়।

3. %c = আমরা character type এর ক্ষেত্রে use করব।

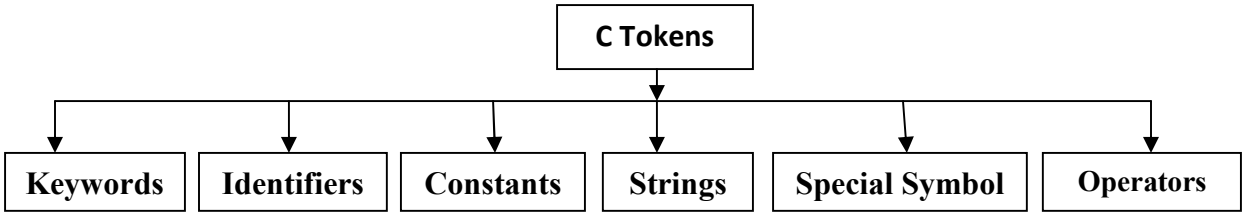
4. %s = আমরা string ("…") এর ক্ষেত্রে input & output হিসাবে use করব।

Character Set: The characters that can be used to form words, numbers and expressions depend upon the computer on which the program is run. The characters in C are grouped into the following categories:

I. Letters,
II. Digits,
III. Special Characters,
IV. White Spaces.

, ; . : ? & ! ' " \ / | ~ _ $ %
are also special characters.

C Tokens: In a passage of text, individual words and punctuation marks are called tokens. Similarly, in a C program, the smallest individual units are known as C Tokens. C has six types of tokens which are given below:



Keywords: All keywords have fixed meanings and these meanings can not be changed. It serve as basic building blocks for program statements. All keywords must be written in lowercase letters.

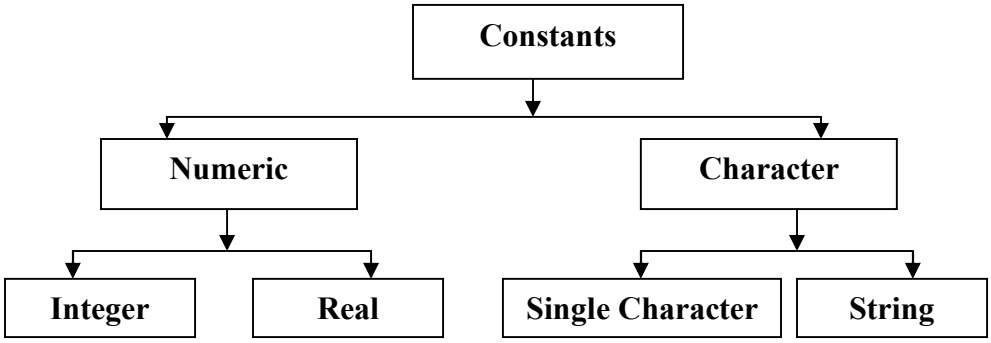
Such as: int, float, double, char, auto, break, case, const, continue, default, do, while, if, else, enum, for, goto, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, extern.

Identifiers: Identifiers refer to the names of variables, functions, and arrays. These are user-defined names, and consist of a sequence of letters and digits, with a letter as a first character. **Both *uppercase and lowercase letters are permitted*.**

Rules For Identifiers:

i. First character must be an alphabet (underscore) .
ii. Must consist of only letters, digits or underscore.
iii. Only first 31 characters are significant.
iv. Can't use a keyword,
v. Must not contain white space.

Constants: Constants refers to the fixed values that do not change during the execution of a program. **Classifications:**



Integer: An integer constants refers to a sequence of digits. There are three types of integers: decimal, octal, hexadecimal integer. Such as: **123, 037, 0AF** etc.

Real: It represents the fractional numbers like as: **12.34** which is called real of floating point numbers. Such as: **12.4, 57.3** etc.

Single Character: A single character constant contains a single character which is enclosed within a pair of single quotation marks. Such as: **'5', 'M'** etc.

String Constants: A string constant is a sequence of characters which is enclosed within a pair of double quotation marks. Such as: **"5", "A"** etc.

Variables: A variable is a data name which is used to store a data value. It remain unchanged during the execution of a program, a variable may take different values at different times during execution. Such as: **John, Delhi,** mark, name, a, x1, **int_add, etc.**

Variable names may consists of letters, digits, and the underscore (_) character.

Rules of Variables:

i. They must begin with a letter. Some systems permit underscore as the first character.

ii. It supports the first eight character out of 31 characters.
iii. Uppercase and lowercase are significant.
iv. It should not be a keyword.
v. White space is not allowed.

Declaration of Variables: After designing suitable variable names, we must declare them to the compiler. Declaration does two things:

i. It tell the compiler what the variable name is.
ii. It specifies what type of data the variable will hold.

Example: **int a, b, add;**

Where **a, b, add** is the variable names & **int** is the data type of all variables.

Data Types: C language is rich in ***data types.*** Storage representations and machine instructions to handle constants differ from machine to machine.

Classifications: There are three types of data:

i. Primary (Fundamental) data types: int, float, char, double, void.
ii. Derived data types: arrays, functions, pointers.
iii. User-defined data types: structures, unions, enumerations.

Character Types: A single character can be defined as a character type data. Characters are usually stored in 8 bits of internal storage. Range is: -128 to 127.

Integer Types: Integers are whole numbers with a range of values supported by a particular machine. Size is: 16 or 32 bits & Range is: -32,768 to 32,768.

Floating Point Types: Floating point numbers are stored in 32 bits with 6 digits of precision. Range is: 3.4e-38 to 3.4e+38.

Double Types: A double data type number uses 64 bits giving a precision of 14 digits. These are known as double precision number. Range: 1.7e-308 to 1.7e+308

Void Types: The void type has no values. This is usually used to specify the type of functions. The type of a function is said to be void when it does not return any value to the calling function.

Declaration of Data types: The syntax for declaring a variable is as follows:

Data_type variable_name1, variable_name2,……., variable_namen;

Example: int a, b, add;

Where a, b, add are the variable name & int is the type of all these variables.

Operators: An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variables.

Classifications: C supports 8 types of operators which are given below:

1. Arithmetic Operators,
2. Relational Operators,
3. Logical Operators,
4. Assignment Operators,
5. Increment & Decrement Operators,
6. Conditional Operators,
7. Bitwise Operators,
8. Special Operators.

Arithmetic Operators: An arithmetic operators are used to perform the arithmetic operations which are listed below:

Arithmetic Operators	Meaning
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

Example: x = a+b ; where a=6, b=8;
Then output is: x = 14.

Where, x = a+b is an expression, x,a,b are called operands and + is called arithmetic operators.

Relational Operators: For example, we may compare the age of two persons or the price of two items and so on. These comparisons can be done with the help of relational operators which are listed below:

Relational Operators	Meaning
<	Is less than
<=	Is less than equal to
>	Is greater than
>=	Is greater than equal to
==	Is equal to
!=	Is not equal to

Example: a < b or 1<20

The value of a relational expression is either one or zero. It is one if the specified

Relation is true and it is zero if the relation is false. So, The result is one (true) for the expression (1 < 20).

Logical Operators: An expression which combines two or more relational expressions is termed as a logical expression. Logical operators are given below:

Logical Operators	Meaning
&&	Logical AND operation
||	Logical OR operation
!	Logical NOT operation

Example: a > b && x == 10

The logical expression given above is true only if a>b is true and x==10 is true. Otherwise false.

Assignment Operators: An assignment operators are used to assign the result of an expression to a variable which are given below:

'=' is an assignment operator. When this operator is formed x+= then it is called shorthand assignment operators.

Shorthand Operators
a+=1
a-=1
a*=n+1
a/=n+1
a%=b

Increment & Decrement Operators: Increment & Decrement operators are unary and they require variable as their operands which are given below:

++ (increment) or -- (decrement)

Rules for ++ and -- operators:

1. When the postfix ++ (or --) is used with a variable in an expression, the expression is evaluated first using the original value of the variable and then the variable is incremented (or decremented) by one.
2. When the prefix ++ (or --) is used with a variable in an expression, the variable is incremented (or decremented) first and then the expression is evaluated using the new value of the variable.

Example: m = 5;

Y = ++m; Means that, A prefix operator first adds 1 to the operand and then the result is assigned to the variable on left. So, the result is: m=6, y = 6.

Example: m = 5;

Y = m++; Means that, A postfix operator first assigns the value to the variable on left and then increments the operand. So, the result is: m=5, y = 6.

Conditional Operators: A ternary operators pair " **? :**" is available in C toconstruct conditional expressions of the form:

exp1 ? exp2 : exp3

Where exp1, exp2 and exp3 are expressions. The operator **? :** works as follows: exp1 is evaluated first. If is non zero (true) then the expression exp2 is evaluated and becomes the value of the expression. If exp1 is false, exp3 is evaluated and its value becomes the value of the expression.

Example:

```
a = 10;
b=15;
x = (a > b) ? a : b ; Means that
if (a > b)
x = a;
else
x = b;
```