Programming With C
Fundamentals of 'C' including data types, operators and I/O statements

Gagan Deep

Founder & Director

Rozy Computech Services, Kurukshetra

Email – rozygag@yahoo.com

www.rozyph.com

Contents of Discussion

- Language & Programming Language
- About C
- Character Set, Identifiers & Statements
- Data Types
- Operators
- Input Output Functions
- Basic Programming Examples

Language

- Language is a way of Communication Between Two like Hindi, English, Punjabi, Marathi, Tamil etc.
- If we want to communicate other then our own languages then we can communicate in either of two ways
    - Either Learn
    - Use Translator

Programming Language

- Way of Communication between Human and Computer like Machine language, Assembly Language, High Level Language.
- Machine Language and Assembly Languages are low level languages because these don't work like our own languages.
- High level languages(HLL) works like our own languages like Hindi, English and others that's why these are known as HLL.

About C

- C is a Programming Language
- C Language is a High Level Language (because it works like natural language). Here we uses English like statements.
- C Language also have a properties of low level languages
- That's why some of us says – This is Middle Level Language.

About C

- C is a general purpose programming language.
- C was originally designed for and implemented on the UNIX operating system on the DEC PDP-11, by Dennis Ritchie.
- C is a case sensitive language A≠ a

Character Set / Alphabets

- First Thing we learned in any language is **Alphabets** – Similarly in Programming Languages is **Character Set**.
- C character Set includes **alphabets** like a-z, A-Z, Digits like 0-9, special **symbols** like !, @, #, $, %, ...-, + etc., and some other **characters** which are not available on keyboard <=, >=, !=, == etc.
- Print Characters- which are known as **Escape sequences** like '\n', '\t', '\b' etc...

Words

- Second thing we always learn in any language are words. Similarly we learn here in C. Here we say words as **Identifiers** and **Reserve words.**
- **Identifier** Gagan, Rajesh, Saminder, Meenu, etc. are our names which identifies us, and some other names are like Table, chair, Fan, Tube which we always we are using for some objects.
- First Type of Identifiers are known as Variable in any Language or we can say in C and second type of names are known as Constants.
- Some other words are known as **Standard words** like min, max etc.

Rules for Naming Identifiers

- First Character is Letter not digit
- Second character it may be letter or digit
- Special characters (other then letters or digits) are not allowed Except underscore(_)
- Length of Identifiers 8, 31 etc. – depends upon compiler

Type of Identifiers – Data Type

- Like we have our Gender -Male / Female. Similarly in each language we have some types of identifier known as Data types.
- What data type does is categorize data.
- These basic categorizations or data types are integer(int),
- Real(float),
- Character (char)

- char – 1 byte - -128 to 127
- Character can declare as
  char c;
- int – 2 bytes - Range (-32768 to 32767)
- Integer can declare as
  int a;
- float – 4 bytes - $3.4X10^{-38}$ to $3.4X10^{38}$
- Real can declare as
  float f;

Why int's range is -32768 to 32767 ?

Integer Representation

- Most of the computers use 2 bytes to store an integer.
- The left most bit, out of sixteen bits, is used to indicate the sign of the integer and is called Sign bit.
- The other 15 bits are used to store the given integer, a 0 in the **sign bit** position indicates a positive integer and 1 in this position means the integer stored is negative.

Integer Representation

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0/1															
Sign Bit	15 bits for Number representation														
	Maximum Number														
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$+2^{15}-1 = 32767$															
Minimum Number															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$-2^{15} = -32768$															
Unsigned Numbers are numbers without sign i.e. First bit is used for number itself.															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Range is 0 to $2^{16}-1$ i.e. 0 to 65535															

If we want bigger number then the ranges defined.

- char – strings
  String is collection of character
- int – short, long, unsigned, unsigned long
  Short is of 1 byte/ 2 bytes(depends upon compilers)
  Long is of 4 bytes(double precession(size))
- float - double
  Double is of 8 Bytes

Constants

- **int** – Decimal, Octal, Hexadecimal
- Decimal int – 77 equals 77
- Octal int – 077 equals 63
- Hexadecimal int – 0x77 equals 119
- Unsigned int – 45678U
- Unsigned long – 243567849UL
- **Float** – 25.67 or 25.67 x 10^3
- **Char** - ‘a’
- **String** – “Rozy”
- E.g.
- const int a=77;

Symbolic Constants

- Symbolic constant is a name that substitute for a sequence of characters.
- The character may represent a numeric constant, a character constant or a string constant.
- When a program is compiled, each occurrence of a symbolic constant is replaced by its corresponding character sequence.
- E.g.

  ```
  #define X    5
  #define PI   3.1417
  ```
- Symbolic constant does not end with semicolon.

Sentences / Statements

- Third thing we learn in any language are sentences, here in programming these are known as statements.
- Statements are the instructions given to the computer to perform any kind of action.
- Statements form the smallest executable unit within a C++ program.
- As sentences terminated by some full stop(., I), similarly, Statements are terminated with a semicolon (;).

Types of Statements

- Simple Statements
- Compound Statements
- Control Statements

Simple Statement – Single statement. The simplest statement is the empty or null statement. e.g.

```
;  // it is a null statement.
```

Examples of simple statement are as

```
x=y;  x=x+y;
scanf("%d", &ajay);
printf("%d", ajay);
```

- **Compound Statement** – paragraph (Block) A compound statement in C++ is a sequence of statements enclosed by a pair of braces ({}). e.g.,

  { statement 1;

  statement 2; },

A compound statement is treated as a single unit or statement.

- **Control Statements** – which controls flow of statements like decision/ branching and loops

Comments

- How we can understand any thing easily.
- The answer is – if that is explained properly or written in a explained manner.
- Comments are used to provide our program a good readability. Comment helps user to read and understand the programs easily. Comments are only for users not for the compilers.
- We can put one line comment or many line comments
- For one line - // Any thing written after this will not executed by compiler
- For many line - /* Number of lines of comment */

Operators

- Operators are tokens that trigger some computation when applied to variable and other objects in an expression. Classifications of Operators are

Depending upon the number of operands we have :

- **Unary operators** : These operators require just 1 operand. e.g, - (minus sign), ++, --, sizeof , casting.
- **Binary operators**: These operators take 2 operands. e.g. +, -, *, / , <, >, !=, <=
- Ternary operators: These operators take 3 operands, e.g. Conditional operator (? :) e.g. (A >B?5:6)

Operators by Operations

- Arithmetic Operators : Integer and Real
  Integer : +,-, *, /, %
  Real : +, -, *, /
- Relational Operators : <, >, <=, >=,
- Equality Operators : !=, ==
- Logical Operators : && (AND), ||(OR), !(NOT)
- Assignment Arithmetic Operators : +=, *=, /= etc. e.g. a+=5 is equivalent to a=a+5. These are also known as **Compound Assignment or Shorthand's**

Operators by Operations

- Increment Operator

Pre increment : ++a ++a means a = a +1 Pre means before execution of statement e.g. If a=5;	Post increment : a++ a++ means a = a +1 Post means after execution of statement e.g. If a=5;

printf("%d", a); //returns 5
printf("%d", ++a); //returns 6
printf("%d", a); //returns 6
	printf("%d", a); //returns 5
	printf("%d", a++); //returns 5
	printf("%d", a); //returns 6

- Decrement Operators : Similarly pre and post decrement operators e.g. - -a; and a- -;

Hierarchy / Precedence of Operators with their Associativity

Operators Category	Operators	Associativity
Unary Operators	-, ++, - - , !, sizeof , (type)	Right → Left (R→L)
Arithmetic	* , / , %,	Left → Right (L → R)
	+ , -	
Relational	< , <=, >, >=	L → R
Equality	== , !=	L → R
Logical	&&	L → R
	||	L → R
Assignment	=, +=, -=, *=, /=, %=	R → L

Low-Level Programming – Bit Wise Operator

- The One's (1's) complement operator (~)
- Another Unary Operator
- Causes the bits of its operator to ve inverted (reversed) so that 1s become 0s and vice versa.
- This operator always precedes its operand.
- e.g. int a=12;//12 binary equivalent is 1100
  ~a //gives result as 0011
- Generally , the operand will be an unsigned octal or an unsigned hexadecimal quantity, though this is not a firm requirement.

The Logical Bitwise Operator

- The Bitwise and expression (&)

  When both are true then true otherwise false means returns 1 only if both are 1 otherwise 0.
- The bitwise or expression (|)

  When both are false then false otherwise true means returns 0 if both are 0 otherwise 1
- The bitwise exclusive or expression (^)

  If both are complement to each other then true otherwise false means when one is 1 and other is 0 then returns 1 otherwise 0

Table for Logical Bitwise operators

a	b	a&b	a|b	a^b
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Examples of Bitwise operators

int a = 12, b=13

Binary equivalents are

a=1100

b=1101

&	|	^
1100 &1101 Result is 1100	1100 |1101 Result is 1101	1100 ^1101 Result is 0001

Bitwise Shift Operators

- Shift Left (<<) & Shift Right(>>)
- Each operator requires two operands. The first is an integer type operand that represents the bit pattern to be shifted. The second is an unsigned integer that indicates the number of displacements.
- int a=15; //binary representation is 00001111
- b=2;
- a<<b; gives 00111100 the result is 60 means

  $a * 2^b$

- a>>b; gives 00000011 the result is 3 means

  $a / 2^b$

Data Input and Output

- Input/Output – may be interactive or non interactive.
- In C you can input/output using input and output library function.
- Functions – User Defined and Library Function
- These I/O Library functions are getchar(), putchar(), scanf(), printf(), gets() and puts().
- These six functions permits the transfer of information between the computer and the standard I/O devices(e.g. Keyboard, VDU etc.)
- An I/O fxs. Can be accessed from anywhere within the program simply by writing the function name.

I/O Statements

- First we discuss about Single Character and Strings I/O Functions

Single Character I/O Functions are getchar() and putchar ()	String I/O Functions are gets() and puts()

char c;
Input statement like this
c = getchar();
 	
Output statement like this
putchar(c);
	char name[20];
	Input statement like this
	gets(name);
	
	Output statement like this
	puts(name);

scanf()

- With the help of scanf() we can enter any type of data and mixed data. In general terms scanf() function is written as

  scanf(control string, arg1, arg2, arg3…, argn);
- **Control String** consists of individual groups of characters, with one character group for each input data item.
- Each character group must begin with a percent(%) sign and followed by **conversion character** which indicates the type of corresponding data item.

Commonly used conversion characters for data input are

- **c** for single character,
- **d** is for decimal integer,
- **e** for floating point value,
- f for floating point value,
- **l** is for long etc….

- The arguments are written as variables, arrays, whose types match the corresponding character group in the control string.
- Each variable must be preceded by an ampersand (&).
- String name should not begin with &.

Examples

```
char name[20];
int  roll;  float marks;

scanf(" %s %d %f", name, &roll, &marks);
```

- **Formatted scanf() function**

```
int a,b,c;

scanf("%3d, %3d %3d", &a, &b, &c);
```

In the above statement all a,b and c can take maximum of 3 digits.

printf()

- It is similar to input function scanf(), except that its purpose is to display data rather than to enter it into the computer.
- Also there is no ampersand (**&**) symbol before args.

```
printf(“ %s \n %d \n %f”, name, roll, marks);
```

Formatted printf() function

- example

```
int a,b,c;
printf("%3d, %3d %3d", a, b, c);
```

- In the above statement all a,b and c can display minimum of 3 digits or spaces instead of digits.

THANKS!

If you have any queries you can contact me at :

rozygag@yahoo.com