An Introduction to C Language

(Quadrant 1- Quadrant 4)

By

Dr. Parul Saxena
Convener and Head,
Department of Computer Science,
Soban Singh Jeena University,
Campus Almora, 263-601
Uttarakhand, India
E-mail:-parul_saxena@yahoo.com

Month & Year: March 2022

Quadrant 1

Introduction to C

- **Initially Basic Combined Programming Language (BCPL) was developed at Cambridge University in 1960's.**

- **Later Dennis Ritchie modified BCPL to make it available for commercial purpose at Bell Laboratories in 1970.**

Character Set

- **Letters :** A...Z a…z
- **Digits :** 0 1 2 3 4 5 6 7 8 9
- **Special Characters :**

  . , ; : ? , “ ! | / \ ~ _ $ %

  # & ^ + - * < > ( ) { } [ ]

- **White Space Characters :**

  Blank Space Horizontal Tab Carriage Return

  New Line Form Feed

Keywords

Keywords are the reserve words which have standard and predefined meanings in C. These can be used only for their intended purpose.

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	for		

Identifiers

- **Identifiers are the names given to various program elements such as variables, functions or any user defined data.**
- **It may consists letters, digits, and the underscore.**
- **It must begin with a letter, some compilers also permit underscore as the first character.**
- **No comma or blanks are allowed.**
- **The identifier should not be a keyword**
- **In some compilers only first eight characters are recognizable, while in other compilers first 31 characters are recognizable.**

Data Types

Primary Data Type :

Integer Data Types	Size
short	1 Byte
int	2 Bytes
long	4 Bytes
Floating Point Data Types	
float	4 Bytes
double	8 Bytes
Character Data Types	
char	1 Byte

Integer Constants

- **Decimal Integer Constants may contain digits 0-9, if the number contains more than two digits then first digit should not be zero.**

  32 907 892 2009

- **Octal Integer Constants may contain digits 0-**

  7, the first digit should be zero.

  0 03 0432 0777

Contd…

- **Hexadecimal Integer Constants contain digits 0-9 and A-F(or a-f) and begin with 0x or 0X.**

  0x26ef, 0xAB, 0x9972, 0x356

Following are some invalid integer constants:

09, A20, 078, 0xPQ

User Defined Data Types

- **typedef < type> <identifier>;**

  typedef int units;

  units number1, number2;

- **enum <identifier>{value1,value2,….value n};**

  enum day {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday};

  enum day birthday;

  Birthday=Monday;

Derived Data Types

- **Array**
- **String**
- **Structure**
- **Union**

Quadrant 2

Operators in C

Arithmetic Operators

- **Addition** +
- **Subtraction** -
- **Multiplication** *
- **Division** /
- **Modulo Division** %
- **Increment Operator** ++
- **Decrement Operator** --

Relational Operators

- **Equal to** ==
- **Not Equal to** !=
- **Less than** <
- **Greater than** >
- **Less than or Equal to** <=
- **Greater than or Equal to** >=

Logical Operators

- **Logical AND** **&&**
- **Logical OR** **||**
- **Logical Not** **!**

Conditional and Cast Operator

- **Conditional Operator ? :**

Variable=(exp1)? exp2:exp3;

Example

Max= a>b? a:b;

- **Cast Operator**

(data type) expression

Example

(int)(4.5+7.2) =11

Assignment Operators

Assignment Equal to =

Variable = <expression>

Operator Equal to op=

Variable op= expression

⇒Variable=Variable op expression

+= -= *= /= %=

Precedence and Associativity of operators

Operators	Associativity
++ -- ! sizeof (type)	R->L
* / %	L->R
+ -	L->R
< > <= >=	L->R
== !=	L->R
&&	L->R
||	L->R
? :	R->L
= op=	R->L

Data Input and Output

Commonly used conversion characters for data Input and Output

%c	to read and write a Single Character
%d	to read and write a decimal integer
%e	to read and write floating point value in exponent form
%f	to read and write a floating point value
%g	to read and write a floating point value in both forms
%h	to read and write a short integer
%i	to read and write a decimal, octal or hexadecimal integer
%o	to read and write an octal integer
%s	to read and write an string
%u	to read and write an unsigned integer

scanf and printf statement

The scanf Statement

scanf (“control string”, arg1, arg2, arg3,……);

```
int a,b,c;
float p,q;
scanf(“%d%d%d”, &a,&b,&c);
scanf(“%f%f”,&p,&q);
```

The printf Statement

printf(“String”);

printf(“control string”, arg1, arg2,agr3,…….);

Structure of a C Program

- Documentation Section
- Link Section
- Definition Section
- Global Declaration Section
- Main Function Section
    - Declaration Part
    - Executable Part
- Subprogram Section
  - Function1
  - Function2
  - ……
  - Function n

Simple C Program

```
//Program to find the area of rectangle
#include<stdio.h>
#include<conio.h>
void main()
{ float length, breadth, area;
printf("Enter the length and breadth of the rectangle");
scanf("%f%f",&length,&breadth);
area=length*breadth;
printf("The area of the rectangle is %f ", area);
}
```

Control Structures

If statement

```
If(expression 1)
{statement1;
Statement2;
…….. }
else if (expression 2)
{ statement1;
Statement2;
……..}
……..
else if(expression n)
{statement1;
Statement2;
……..}
else
{ statement1;
Statement2;
……..}
```

Example of if statement

```
If(marks>=80)
        printf(“Pass with Grade A”);
else if(marks>=70)
        printf(“Pass with Grade B”);
else if(marks>=60)
        printf(“Pass with Grade C”);
else if(marks>=50)
        printf(“Pass with Grade D”);
else
        printf(“Fail”);
```

switch Statement

```
switch(expression)
{case value1: statement1;
                    statement2;
                     ……
                      break;
case value 2: statement1;
                    statement2;
                     ……
                     break;
……..
Case value n: statement1;
                   statement2;
                    ……
                    break;
default:       statement1;
                  statement2;
                          ……..
}
```

Example of switch statement

```
switch(marks/10)
{case 10:
  case 9:
  case 8: printf(“Pass with Grade A”); break;
  case 7: printf(“Pass with Grade B”); break;
  case 6: printf(“Pass with Grade C”);break;
  case 5: printf(“Pass with Grade D”);break;
default: printf(“Fail”);
```

Quadrant 3

while statement

```
While(conditional statement)
{
statement1;
statement2;
.............
}
```

Example while statement

```
int a=1;
while(a<=10)
{
printf("Welcome");
a=a+1;
}
```

do while statement

```
do
{
statement1;
Statement2;
}
while( conditional expression);
```

Example do while statement

```
int a=1;
do {
printf("Welcome");
a=a+1;
} while(a<=10);
```

for statement

```
for (initialization statements; conditional
    expressions; increment/ decrement statements)
{
Statement1;
Statement2;
……
}
```

Example for statement

```
for (i=1;i<=10;i++)
{
printf(“Welcome”);
}
```

break statement

The break statement is used to terminate the enclosing loop and to exit from the switch statement. It causes a transfer of control out of the loop or switch.

Syntax

```
break;
```

Example break statements

```
#include<stdio.h>
#include<conio.h>
void main()
{ int a,n;
printf("Enter any number");
scanf("%d",&n);
for(a=2;a<n;a++)
    if(n%a==0) break;
If(a==n) printf(“ Number is Prime”);
else printf(“Number is not Prime”);
getch();}
```

continue statement

The continue statement causes the loop to be continued with the next iteration after skipping the statements, which are followed by the continue statement;

```
continue;
```

continue statement

```
#include<stdio.h>
#include<conio.h>
void main()
{ int n, num, Oddsum=0;
printf("Enter how many numbers ? ");
scanf("%d", &n);
for(int i=1;i<=n;i++)
{   printf("Enter Number : ");
 scanf("%d",&num);
 if (num%2==0) continue;
  Oddsum +=num; }
printf("The sum of all odd numbers= %d", Oddsum);
getch();
}
```

Arrays

Array is the collection of similar type of elements. Each element is recognized by the index of array. The general form of array representation is as follows:

Syntax

<data type> < Array name> [size1][size2]……[size n];

```
int Number[10];
float Height[10];
Int matrix[3][3];
```

Contd…

int A[10];

index varies from 0 to size-1

10	20	30	40	50	60	70	80	90	100

A[0]=10; A[5]=60;

A[1]=20; A[6]=70;

A[2]=30; A[7]=80;

A[3]=40; A[8]=90;

A[4]=50; A[9]=100;

String

String is an array of characters terminated by null character. Any group of characters defined between double quotation marks is a string constant.

"God is Great" is the example of string.

Syntax

```
char name[30];
name = "John Smith";
```

name[0]='J'	name[1]='o'	name[2]='h'	name[3]='n'
name[4]=' '	name[5]='S'	name[6]='m'	name[7]='i'
name[8]='t'	name[9]='h'	name[10]='\0';	

Reading and Writing Strings

The gets statement is used to read multiword string.

```
gets(string);
```

The puts statement is used to print the multiword string.

```
puts(string);
```

String Handling Functions

- strlen function returns the length of the string.

  strlen(string)

- strcat function concatenates the content of string 2 to string1.

  strcat(string 1, string2)

Contd…

- strcpy function copies the content of string 2 to string 1

  strcpy(string1,string2)

- strcmp function compares two strings and returns integer value equal to the difference of ASCII value of the first non matching character.

  strcmp(string1,string2)

Quadrant 4

Functions

A function is a self contained program segment that carries out some specific task. A function behaves like a frame in which we pass the arguments and get the desired result. The function defined once may be used again and again.

Syntax of a function

return type function name (argument list)

{ local variable declaration part;

executable part;

}

Program using Function

```
#include<stdio.h>
#include<conio.h>
float area(float radius)
{ float a;
  a= pi*radius*radius;
  return a;}
void main()
{ int I; float r;
for( i=1;i<=10;i++)
{ printf("Enter Radius"); scanf("%f", &r);
printf("The area of the circle with radius %f is %f", r, area(r));
 }getch();
}
```

Recursion

Recursion is the process in which a function calls itself repeatedly, until some specified condition is satisfied. The result of any recursive function is defined in terms of the previous result.

```
long fact(int n)
{ if(n<=1)return 1;
else return(n* fact(n-1)); }
```

Structures

Structure is the collection of data items, which may be of different data types.

Syntax

```
struct <structure Name>
{
  data type     dataitem1;
   data type    dataitem2;
…….
};
```

Contd…

```
struct Book
{ int ISBN;
   char Title[30];
    char Author[30];
};
struct Book book1,book2;
```

Membership Operator

The different data items of the structure are referred by membership operator as follows:

book1.ISBN=40089;

book1.Title="Introduction to C";

book1.Author= "D. M. Ritchie";

book2.ISBN=20005;

book1.Title="Let Us C";

book2.Author= "Yashwant Kanetkar";

Array of Structures

```
struct student
{ int Idno;
char Name[30];
int Marks[5];
};
Struct student ClassGIS[60];
ClassGIS[1].Idno=01;
ClassGIS[1].Name=“Priya”;
```

Pointers

A Pointer is a variable that contains the address of a variable.

Variable	Value	Address
amount	200	1000
P	1000	1050

```
int   *p;
If p=&amount;
then  *p=amount; or amount= *p;
```

Passing Pointers to a Function

```
#include<stdio.h>
#include<conio.h>
exchange(&x,&y);
void main()
{ int x, y;
   x=10;y=20;
printf("Initial Values x= %d , y=%d ",x,y);
exchange(&x,&y);
printf("Initial Values x= %d , y=%d ",x,y);}
```

```
exchange(int *a, int *b)
{
Int temp, p;
temp= *a;
*a=*b;
*b=temp;
}
```

Thank You