

# Basics of 'C'

# General Aspect of 'C'

- C was originally developed in the 1970s, by Dennis Ritchie at Bell Telephone Laboratories, Inc.
- 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.
- This allows C to be used for system programming as well as for applications programming

# The Character set of 'C'

C language consist of some characters set, numbers and some special symbols. The character set of C consist of all the alphabets of English language. C consist of

Alphabets      a to z, A to Z

Numeric        0,1 to 9

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

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. The following different types of token are used in C

- |                |                       |             |
|----------------|-----------------------|-------------|
| 1) Identifiers | 2)Keywords            | 3)Constants |
| 4) Operators   | 5)Punctuation Symbols |             |

# Identifiers

- A 'C' program consist of two types of elements , user defined and system defined. Idetifiers is nothing but a name given to these eleme
- nts.
- An identifier is a word used by a programmer to name a variable , function, or label.
- identifiers consist of letters and digits, in any order, except that the first charecter or lable.
- Identifiers consist of letters and digits if any order,except that the first charecter must be letter.
- Both Upper and lowercase letters can be used

# Keywords

- Keywords are nothing but system defined identifiers.
- Keywords are reserved words of the language.
- They have specific meaning in the language and cannot be used by the programmer as variable or constant names
- C is case sensitive, it means these must be used as it is
- 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

# Hello World Program

```
#include<stdio.h>//Header file
/* Following is an example of a basic
C program*/
int main() {
printf("Hello World");
return 0;
}
```

# Variables

- A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.
- The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C is case-sensitive. There are following basic variable types –

Type	Description
• char	Typically a single octet(one byte). This is an integer type.
• int	The most natural size of integer for the machine.
• float	A single-precision floating point value.
• double	A double-precision floating point value.
• void	Represents the absence of type.

# Constants

- A constant is a value or an identifier whose value cannot be altered in a program. For example: 1, 2.5,
- As mentioned, an identifier also can be defined as a constant. eg. `const double PI = 3.14`
- Here, PI is a constant. Basically what it means is that, PI and 3.14 is same for this program.

## 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)



# Constants

## 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

# Sample Program

```
#include<stdio.h>
int main() {
int age=20;
float pi=3.14;
char star='c';
printf("age is %d\n", age);//integer example
printf("value of pi is \'%f'\n", pi);//float
numbers example
printf("char looks like this:\t %c ",
star);//characters example
return 0;}
```

# Output

```
age is 20
```

```
value of pi is '3.140000'
```

```
char looks like this:    c
```

# Sample Program

```
#include <stdio.h>
int main()
{
    int num;
    char ch;
    float f;
    printf("\n\nEnter the Character: ");
    scanf("%c", &ch);
    printf("\nEntered character is: %c\n", ch);
    printf("Enter the integer: ");
    scanf("%d", &num);
    printf("\nEntered integer is: %d", num);
    printf("\nEnter the float: ");
    scanf("%f", &f);
    printf("\nEntered float is: %f", f);
    return 0;}
```

# Output

```
Enter the Character: a
Entered character is: a
Enter the integer: 10
Entered integer is: 10
Enter the float: 10.5
Entered float is: 10.500000
```

# 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

# Sample Program

```
int main() {  
    int a = 1, b = 2, c = 3;  
    a = b + c;  
    printf("%d \n",a);  
    printf("%d \n", a/ b);  
    printf("%d \n", (-a) % b);  
    printf("%f \n", (float)a/ b);  
    printf("%f\n", (float)a + 3);  
    printf("%.2f\n", (float)a + 3);  
    printf("a=%d\n",a);  
    printf("a++=%d\n",a++);  
    printf("a=%d\n",a);  
    printf("++a=%d\n",++a);  
    a-=b;  
    printf("a after \'a-=b\'=%d\n",a);  
    printf("\'a==b\'=%d",a==b);  
    return 0;}
```

```
5  
2  
-1  
2.500000  
8.000000  
8.00  
a=5  
a++=5  
a=6  
++a=7  
a after 'a-=b'=5  
'a==b'=0
```

# Logical Operators

Operators	Example/Description
&& (logical AND)	<code>(x&gt;5)&amp;&amp;(y&lt;5)</code> It returns true when both conditions are true
(logical OR)	<code>(x&gt;=10)   (y&gt;=10)</code> It returns true when at-least one of the condition is true
! (logical NOT)	<code>!((x&gt;5)&amp;&amp;(y&lt;5))</code> It reverses the state of the operand “((x>5) && (y<5))” If “((x>5) && (y<5))” is true, logical NOT operator makes it false

# Sample Program

```
#include <stdio.h>
int main()
{
    int m=40,n=20;
    int o=20,p=30;
    if (m>n && m !=0)
    { printf("&& Operator : Both conditions are true\n"); }
    if (o>p || p!=20)
    { printf("|| Operator : Only one condition is true\n"); }
    if (!(m>n && m !=0))
    { printf("! Operator : Both conditions are true\n"); }
    else
    { printf("! Operator : Both conditions are true. " \
        "But, status is inverted as false\n");
    }
}
```

# Output

```
&& Operator : Both conditions are true  
|| Operator : Only one condition is true  
! Operator : Both conditions are true. But, status is inverted as false
```

# Bitwise Operator

Operators	Meaning of operators
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR
~	Bitwise complement
<<	Shift left
>>	Shift right



# Sample Program

```
#include <stdio.h>
int main()
{
    int a = 5, b = 9;// a = 5(00000101), b = 9(00001001)
    printf("a = %d, b = %d\n", a, b);
    printf("a&b = %d\n", a & b);// The result is 00000001;
    printf("a|b = %d\n", a | b);// The result is 00001101
    printf("a^b = %d\n", a ^ b);// The result is 00001100
    printf("~a = %d\n", a = ~a);// The result is 11111010
    printf("b<<1 = %d\n", b << 1);// The result is 00010010
    printf("b>>1 = %d\n", b >> 1);// The result is 00000100
    return 0;
}
```

# Output

```
a = 5, b = 9
```

```
a&b = 1
```

```
a|b = 13
```

```
a^b = 12
```

```
~a = -6
```

```
b<<1 = 18
```

```
b>>1 = 4
```

# Special Operator

Operators	Description
&	This is used to get the address of the variable. Example : &a will give address of a.
*	This is used as pointer to a variable. Example : * a where, * is pointer to the variable a.
Sizeof ()	This gives the size of the variable. Example : size of (char) will give us 1.

# Sample Program

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    char a;
```

```
    int b;
```

```
    float c;
```

```
    double d;
```

```
    printf("size of char %lu\n",sizeof(a));
```

```
    printf("size of int %lu\n",sizeof(b));
```

```
    printf("size of float %lu\n",sizeof(c));
```

```
    printf("size of double %lu\n",sizeof(d));
```

```
    printf("address of a %p\n",&a);
```

```
    return 0;
```

```
}
```

# Output

```
size of char 1  
size of int 4  
size of float 4  
size of double 8  
address of a 0x7ffee78e2347  
|
```

# Loops

Sr.No.	Loop Type & Description
1	<p>while loop <a href="#">↗</a></p> <p>Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.</p>
2	<p>for loop <a href="#">↗</a></p> <p>Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.</p>
3	<p>do...while loop <a href="#">↗</a></p> <p>It is more like a while statement, except that it tests the condition at the end of the loop body.</p>
4	<p>nested loops <a href="#">↗</a></p> <p>You can use one or more loops inside any other while, for, or do..while loop.</p>

# Sample Program(While)

```
#include <stdio.h>

int main () {
    int a = 10;
    while( a < 20 ) {
        printf("value of a: %d\n", a);
        a++;
    }
    return 0;
}
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

# Sample Program(Do While)

```
#include <stdio.h>

int main () {
    int a = 10;
    do {
        printf("value of a: %d\n", a);
        a = a + 1;
    }while( a < 20 );
    return 0;
}
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```



# Sample Program(For Loop)

```
#include <stdio.h>
```

```
int main () {
```

```
    int a;
```

```
    for( a = 10; a < 20; a = a + 1 ){  
        printf("value of a: %d\n", a);
```

```
    }
```

```
    return 0;
```

```
}
```

```
value of a: 10
```

```
value of a: 11
```

```
value of a: 12
```

```
value of a: 13
```

```
value of a: 14
```

```
value of a: 15
```

```
value of a: 16
```

```
value of a: 17
```

```
value of a: 18
```

```
value of a: 19
```

# Sample Program(Nested For Loop)

```
#include <stdio.h>
```

```
int main () {  
    int i, j;  
    for(i = 2; i<50; i++) {  
        for(j = 2; j <= (i/j); j++)  
            if(!(i%j)) break; // if factor found,  
not prime  
        if(j > (i/j)) printf("%d is prime\n",  
i);  
    }  
    return 0;  
}
```

```
2 is prime  
3 is prime  
5 is prime  
7 is prime  
11 is prime  
13 is prime  
17 is prime  
19 is prime  
23 is prime  
29 is prime  
31 is prime  
37 is prime  
41 is prime  
43 is prime  
47 is prime
```






# Sample Program(Nested For Loop)

```
#include <stdio.h>
```

```
int main () {  
    int i, j;  
    for(i = 2; i<50; i++) {  
        for(j = 2; j <= (i/j); j++)  
            if(!(i%j)) break; // if factor found,  
not prime  
        if(j > (i/j)) printf("%d is prime\n",  
i);  
    }  
    return 0;  
}
```

```
2 is prime  
3 is prime  
5 is prime  
7 is prime  
11 is prime  
13 is prime  
17 is prime  
19 is prime  
23 is prime  
29 is prime  
31 is prime  
37 is prime  
41 is prime  
43 is prime  
47 is prime
```

# Decision statement

Sr.No.	Statement & Description
1	<p>if statement </p> <p>An <b>if statement</b> consists of a boolean expression followed by one or more statements.</p>
2	<p>if...else statement </p> <p>An <b>if statement</b> can be followed by an optional <b>else statement</b>, which executes when the Boolean expression is false.</p>
3	<p>nested if statements </p> <p>You can use one <b>if</b> or <b>else if</b> statement inside another <b>if</b> or <b>else if</b> statement(s).</p>
4	<p>switch statement </p> <p>A <b>switch</b> statement allows a variable to be tested for equality against a list of values.</p>
5	<p>nested switch statements </p> <p>You can use one <b>switch</b> statement inside another <b>switch</b> statement(s).</p>

# Sample Program

```
#include <stdio.h>
```

```
int main () {  
    int a = 100;  
    int b = 200;  
    if( a == 100 ) {  
        if( b == 200 ) {  
            printf("Value of a is 100 and b is  
200\n" );  
        }  
    }  
    printf("Exact value of a is : %d\n", a  
);  
    printf("Exact value of b is : %d\n", b  
);  
    return 0;  
}
```

```
Value of a is 100 and b is 200  
Exact value of a is : 100  
Exact value of b is : 200
```

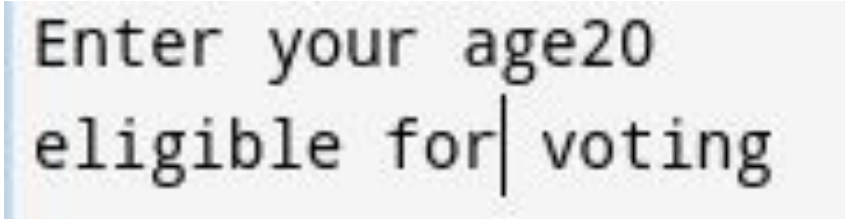
# Sample Program

```
#include <stdio.h>
int main () {
    char grade = 'B';
    switch(grade) {
        case 'A' :
            printf("Excellent!\n" );
            break;
        case 'B' :
        case 'C' :
            printf("Well done\n" );
            break;
        case 'F' :
            printf("Better try again\n" );
            break;
        default :
            printf("Invalid grade\n" );
    }
    printf("Your grade is  %c\n", grade );
    return 0;
}
```

Well done  
Your grade is B

# Sample Program(? operator)

```
#include <stdio.h>
int main()
{
    int age; // variable declaration
    printf("Enter your age");
    scanf("%d",&age); // taking user
    input for age variable
    (age>=18)? (printf("eligible for
    voting")) : (printf("not eligible for
    voting")); // conditional operator
    return 0;
}
```

A screenshot of a terminal window with a light gray background. It shows the output of the program: "Enter your age20" on the first line and "eligible for| voting" on the second line. A vertical cursor is positioned between "for" and "voting" on the second line.

Enter your age20  
eligible for| voting