Programming in C

Nepal College of Information Technology

Course Instructor: Er. Rabina Chaudhary

Operators in C Programming Language

Operators:

- A operator is a symbol that operates on a certain data type or data item
- The data items or variables in which the operator operates are called operands
- Operators are used in program to perform certain mathematical or logical manipulations
- Example:

  8 + 10, here + is operator and it operates on 8 and 10, 8 and 10 are called operands

- An expression is combination of variables, constants and operators written according to syntax of programming language

Operators can be classified into three types on the basis of number of operands:

1. Unary operator
2. Binary operator
3. Ternary operator

1. Unary Operator:

- Operators which require only one operand are unary operators
- Example: increment operator (++)
  decrement operator (--)
  unary plus (+)
  unary minus (-)

2. Binary Operators:

- Operators which require two operators are binary operators

Example : binary plus operator(+),
binary minus operator (-),
multiplication operator(*),
division operator(/),
greater than operator(>) ,etc

a+b

a-b

a/b, etc

3. Ternary operand:

- Operator that operates on three operands

  ?:

  example:

  (a>b)? a:b;

Classification of operator on the basis of work / utility :

1. Arithmetic operator
2. Logical operator
3. Assignment operator
4. Relational operator
5. Increment / decrement operator
6. Conditional operator
7. Special operator

1. Arithmetic operator:

- It includes the operators that are used to perform mathematical operations

1. Plus operator , A+B
2. Minus operator, A-B , -a
3. Multiplication operator, A*B
4. Division operator, A/B
5. Modulo operator, A%B

integer and integer gives integer
Integer and float gives float
Float and integer gives float
Float and float gives float

2. Assignment operators :

- These operators are used to assign particular values or result of expression to a variable.
- Example: ‘=‘
- A=b;

- It can also be associated with arithmetic operators
- Example: += ,-=,*=,/=
- a-=b; //equivalent to, a=a-b;
- a+=b; //equivalent to, a=a+b;
- a*=b; //equivalent to, a=a*b;
- a/=b; //equivalent to, a=a/b;
- a%=b; //equivalent to, a=a%b;

3. Conditional operators:

- The operator **?:** is called conditional operator
- It takes three operands so also called ternary operator

- Syntax:

  Expression1 ? True expression : false expression;

Example: maximum of two numbers using conditional operator

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

void main()
{
        int numberOne,numberTwo, result;
        printf("Enter two numbers :");
        scanf("%d%d",&numberOne, &numberTwo);
        result= (numberOne>numberTwo)? numberOne : numberTwo ;
        printf("\nThe larger vaue among %d and %d is %d",numberOne, numberTwo, result);
  getch();
}
```

Example: maximum of two numbers using conditional operator

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

void main()
{
        int numberOne,numberTwo;
        printf("Enter two numbers :");
        scanf("%d%d",&numberOne, &numberTwo);

        (numberOne>numberTwo)    ?
                                        printf("\n The larger vaue among %d and %d is
                                        %d",numberOne, numberTwo, numberOne)
                                :       printf("\nThe larger vaue among %d and %d is %d",numberOne,
                                        numberTwo, numberTwo) ;

  getch();
}
```

Practical:

1. Write a program to check if input number is positive or negative.

4. Relational Operators:

- These operators are used to compare two operands
- Can be used to compare :
    1. one constant with another constant
    2. constant with variable and vice versa
    3. variable with another variable

5. Relational Operators:

Operator	Operator Name (meaning)
<	Less than
>	Greater than
<=	Less than or equals to
>=	Greater than or equals to
==	Equals to
!=	Not equals to

5. Logical operators:

- These operators are used to perform the logical operation on the given expressions.
- Three logical operators are

1. logical AND denoted as &&
2. Logical OR denoted as ||
3. Logical NOT denotes as !

Logical operators:

Expression is denoted as A and B,
1=true, 0=false

A	B	A&&B	A||B	!A	!B
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

6. Increment Operator and Decrement Operator:

- Increment operator is used to increase the value of operand by one
- Decrement operator is used to decrease the value of operand by one

- They operate one operand only hence also called unary operator
- Syntax for operator is :

    ++ variable

    variable ++

    --variable

    variable--

Increment Operator:

- Increment can divided to

1. Pre increment :

- Increment operator used in form ++variable is pre increment
- Operator is used before operand
- In pre increment the value of associated operand is increased and then value is utilized in expression

Increment Operator:

2. Post increment:

- Increment operator used in form variable++ is post increment
- Operator is used after operand
- In post increment the value of associated variable is utilized in expression and then increased

Decrement operator:

- Decrement can divided to

1. Pre decrement :

- decrement operator used in form --variable is pre decrement
- Operator is used before operand
- In pre decrement the value of associated operand is decreased and then value is utilized in expression

Decrement Operator:

2. Post decrement:

- decrement operator used in form variable-- is post decrement
- Operator is used after operand
- In post decrement the value of associated variable is utilized in expression and then decreased

7. Special Operators:

a. Comma operator
b. sizeof operator
c. Address of operator

Comma operator: ,

- Comma operator is used to link related expressions together
- Denoted by “ , ”

Example:

```
result=(numberOne=100, numberTwo=200, numberOne + numberTwo);
```

sizeof operator:

- It returns the memory allocated by its operand
- Syntax:

  sizeof(operand)

Address of operator:

- It gives the address of a particular memory location
- It is represented by “&”
- Also called ampersand operator

Operator precedence and associativity:

Precedence	Operator Type	Operator	Associativity
1. Highest	Function call Subscript operator	( ) [ ]	Left to right
2.	**Unary operators** Unary plus Unary minus Increment Decrement Address of Size of Logical NOT Indirection operator	 + - ++ -- &, sizeof() ! *	Right to left
3.	**Arithmetic Operator** Multiplication Division Modulus	 * / %	Left to right

Operator precedence and associativity:

Precedence	Operator Type	Operator	Associativity
4.	Arithmetic Operator Addition Subtraction	 + -	Left to right
5.	Less than Greater than Less than equals to Greater than equals to	< > <= >=	Left to right
6.	Equals to Not equals to	== !=	Left to right
7.	Logical AND	&&	Left to right
8.	Logical OR	||	Left to right
9.	Assignment operators	= += -= /= %=	Right to left
10	Comma operator	,	Left to right

Find the value of A.

1. A = 9-45/5+8*12-100
2. A = 13 * 40 / 50 + 9 / 19 – 4 – 25 + 6 % 4
3. A + = A * B where, A = 50 and B = 90
4. A/ = A – B where , initial value of A = 50 and B = 90
5. A = 4 + 1 / 5 + 3.14 * 34 * 5 +1
6. A = X++ + ++X where, initial value of X=45

Find the output of :

```
#include<stdio.h>

 void main()
                {
                        int x,y,z;
                        x=20;
                        y=18;
                        z=0;
                         y=x++ + ++x;
                        printf("x=%d\ny=%d\nz=%d\n\n",x,y,z);
                        z=++x + y++;
                        printf("x=%d\ny=%d\nz=%d\n\n",x,y,z);
                        x=x + y + z--;
                        printf("x=%d\ny=%d\nz=%d",x,y,z);

}
```

Expression:

- Expression is combination of variables, constants and operators arranged according to the syntax of language.
- Result of expression is assigned to a variable using assignment statement of the form:

  Destination_Variable = Expression;

- **Example :**
  1. **x = 1 + a + 34 ;**
  2. **x = b – a * (c / d) ;**
  3. **x = ((a + b) * c) / d;**

Type Conversion In Expression:

1. Implicit type conversion:

- also called automatic type conversion
- When operands of operators are of different types in expression, the lower types are automatically converted to higher type and the result is of higher type

- When assigning the result of expression to a variable type

  Destination_variable = expression ;

  The final result of expression is converted to the type of the variable on the left of assignment operator

  Note: this also implies when assigning one type variable to another type

Conversion Hierarchy:

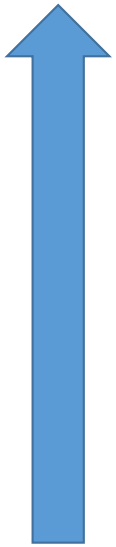
long double

double

float

int

char



Type Conversion In Expression:

2. Explicit type conversion :

- Converting one data type to another data type forcibly by the programmer
- Syntax for explicit type conversion :

(type_name) expression;

where type_name is any data type of C Programming language

and expression can be constant, variable or expression