Chapter 7

Operators, Expressions and Control Structures

7.1 Introduction

All operators in C are also valid in C++. In addition, C++ introduces some new operators. They are following:

- **<< Insertion operator:** It prints the contents of the variable on its right to the output screen.
- **>> Extraction operator:** It takes the value from the keyboard and assign it to the variable on its right.
- **:: Scope resolution operator:** C++ is a block-structured language. Same variable name can be used in different blocks. The scope of variable is in between the point of its declaration and end of the block containing the declaration. A variable declared inside a block is local to that block. The scope resolution operator is used to access global version of a variable.

Program 7.1: Scope resolution operator

```
#include<iostream>
using namespace std;
int x=10;              //global variable
int  main()
{
    int x=20;          //x re-declared , local to main
    {
            cout<<"Inner block\n";
            int x=30;        //x declared again, local to inner block
            cout<<"x="<<x<<"\n";
            cout<<"::x="<<::x<<"\n";
    }
    cout<<"Outer block\n";
    cout<<"x="<<x<<"\n";
    cout<<"::x="<<::x<<"\n";
    return 0;
}
```

The output the program 7.1 would be:

```
Inner block
x=30
::x=10
Outer block
```

```
x=20
::x=10
```

- **new operator:** The operator allocates sufficient amount of memory to data object at run time. For example

```
int *p = new int;
```

The above statement allocates sufficient amount of memory to integer data object at run time.

- **delete operator:** The operator de-allocates the memory when the data object is no longer needed. So that the released memory can be reused by the other programs.

  For example

```
delete p;
```

  The above statement de-allocates the memory pointed by the pointer variable p.

7.2 Expressions and their types

An expression is a combination of operators, constants and variables arranged as per the rule of the language. There are following types of expressions:

- **Constant expressions:** It consists of only constant values. For example 20+10*5.2

- **Integral expressions**: Those expressions which produce integer results after implementing implicit and explicit type conversions. Examples:

  x+y*10
  x+'a'
  5+int(7.5)
  where x and y are integer variables.

- **Float expressions:** Those expressions which produce floating-point results after implementing implicit and explicit type conversions. Examples:

  a+b/5
  7+float(10)
  where a and b are float type variables.

- **Pointer expressions:** Pointer expressions produce address values.

Examples:
ptr=&x;
ptr+1
where x is a variable and ptr is a pointer.

- **Relational expressions:** Those expressions which produce Boolean type results that is either true or false. Examples:

  x<=y
  a==b

- **Logical expressions:** Those expressions which combines two or more relational expressions and produce Boolean type result. Examples:

  x>y && x==5
  a==20 || y==10

- **Bitwise expressions:** These type of expressions are used to manipulate data at bit level. They are used for testing or shifting bits. Examples:

  a<<3 // shift three bits position to left
  x>>1 // shift one bit position to right

- **Special Assignment Expressions:**
  Chained assignment
  a=b=10;
  First 10 is assigned to b then to a.

  Embedded assignment
  a=(b=20)+5;
  (b=20) is an assignment expression called embedded assignment. Here, the value 20 is assigned to b and then the result is assigned to a.

  Compound assignment
  It is a combination of the assignment operator and a binary arithmetic operator.

  For example:
  The expression
  a=a+5;
  can be written as
  a+=5;

The operator += is called compound assignment operator or short-hand operator.

7.3 Operator precedence and associativity

If more than operators are involved in an expression, C++ language has predefined rules of priority for the operators. The operator with higher priority will execute before the operators with lower priority. This rule is called operator precedence.

If two or more operators with same precedence are present in an expression, the order in which they execute is called associativity of operators. The complete list of C++ operators with their precedence from highest to lowest and associativity is given in table 7.1.

Table 7.1 Operator precedence and associativity

Operator precedence	Associativity
::	Left to right
->, ., (), [], ++, --, ~, !, unary+, unary-, unary*	Left to right
Unary &, (type), sizeof, new, delete	Right to left
*, /, %	Left to right
+, -	Left to right
<<, >>	Left to right
<, <=, >, >=	Left to right
==, !=	Left to right
&	Left to right
^	Left to right
|	Left to right
&&	Left to right
||	Left to right
?:	Left to right
=, *=, /=, %=, +=	Right to left
<<=, >>=, &=, ^=, |=, ,(comma)	Left to right

There are three types of control structures:

(i) Sequence structure
(ii) Selection structure
(iii) Loop structure

C++ supports all the three basic control structures and implements them using various control statements.

(i) **Sequence structure:** Statements are executed sequentially as they are written in program. Example:

```
- - - - - - - -
statement1;
statement2;
statement3;
- - - - - - -
```

(ii) **Selection structure:** Two or more paths of execution out of which one is selected based on a condition.
Examples:

The if statement

```
if(expression is true)
{
        statements;
}
```

The if-else statement

```
if(expression is true)
{
        statements;
}
else
{
        statements;
}
```

The switch statement

```
switch(expression)
{
        case 1: statements;
                break;
        case 2: statements;
                break;
```

```
        case 3: statements;
                break;
        default : statements;
}
```

(iii) Loop structure: Statements are executed zero or more times.

Examples:

The for statement

The for loop is used when an action is to repeated for predefined number of times.

```
for(initial value; test condition; increment/decrement)
{
    statements;
}
```

The while statement

The statements within the while are executed till the condition is true. It is pre-test condition loop.

```
while(condition is true)
{
        statements;
}
```

The do-while statement

The loop is executed at least one time. It is post-test condition loop.

```
do
{
        statements;
}while(condition is true);
```

Important Points

- All operators in C are also valid in C++.
- C++ is a block-structured language.
- An expression is a combination of operators, constants and variables arranged as per the rule of the language.

- The operator with higher precedence will execute before the operators with lower precedence.
- C++ supports all the three basic control structures and implements them using various control statements.

Practice Questions

Objective type questions:

Q.1 Operator that prints the contents of the variable on its right to the output screen is

A. << B. >>

C. :: D. &

Q.2 Operator that allocates sufficient amount of memory to data object at run time is

A. Insertion operator B. Extraction operator

C. new operator D. delete operator

Q.3 In the expression **a=(b=20)+5;** the value of variable 'a' will be

A. 20 B. 25

C. 5 D. 30

Q.4 Which is a short-hand assignment operator?

A. += B. - =

C. * = D. All of these

Q.5 Selection structure is implemented by which control statement?

A. if statement B. if-else statement

C. switch statement D. All of these

Q.6 Loop structure is implemented by which control statement?

A. for statement B. while statement

C. do-while statement D. All of these

Very Short Answer Type Questions

Q.1 Define operator precedence.
Q.2 Define associativity of operators.
Q.3 What are the different types of control structures?
Q.4 What are expressions?

Short Answer Type Questions

Q.1 What are the uses of scope resolution operator?
Q.2 What are the uses of new and delete operators?
Q.3 Explain how selection control structure is implemented in C++.

Essay Type Questions

Q.1 Explain different types of expressions with examples.
Q.2 Explain various types of looping statements.

Answer Key

1. A	2. C	3. B
4. D	5. D	6. D