OBJECT: To understand Conditional Operator, Logical Operators, break keyword, continue keyword, nested loops, and combination of all these statements.

CONDITIONAL OPERATOR

The conditional operator (? :) is C's only ternary operator; that is, it is the only operator to take three terms. The conditional operator takes three expressions and returns a value:

```
(expression1) ? (expression2) : (expression3)
```

It replaces the following statements of if else structure

```
if(a>b)
c=a;
else
c=b;
```

can be replaced by

```
c = (a>b) ? a : b
```

This line is read as "If expression1 is true, return the value of expression2; otherwise, return the value of expression3." Typically, this value would be assigned to a variable.

PROGRAM 1: Demonstrates conditional operator

```
#include <iostream>
using namespace std;
int main() {
int a,b,c;
cout<<"Enter a, b ";
cin>>a>>b;
c = a > b ? a : b ;
cout<<"Larger number is: "<<c;
return 0;
}
```

PROGRAM 2: Demonstrates conditional operator

```
#include <iostream>
using namespace std;
int main()
{
for(int j=0; j<80; j++) //for every column,
{ //ch is 'x' if column is
char ch = (j%8) ? ' ' : 'x'; //multiple of 8, and
cout << ch; //' ' (space) otherwise
}
return 0;
}
```

PROGRAM 3: Demonstrates nested conditional operator

```
#include <iostream>
using namespace std;
int main() {
int a,b,c;
cout<<"Enter a, b and c: ";
cin>>a>>b>>c;

cout<<"Larger number is: "
    <<((((a>b)? a : b) > c) ? ((a>b)? a : b) : c);
return 0;
}
```

LOGICAL OPERATORS

While relational (comparison) operators can be used to test whether a particular condition is true or false, they can only test one condition at a time. Often we need to know whether multiple conditions are true at once and logical operators provide us with this capability to test multiple conditions. Depending upon the requirement, proper logical operator is used. There are three logical operators:

Operator	Symbol	Form	Operation
Logical NOT	!	!x	true if x is false, or false if x is true
Logical AND	&&	x && y	true if both x and y are true, false otherwise
Logical OR	||	x || y	false if both x and y are false, true otherwise

The operators return true or false, according to the rules of logic:

A	b	a && b
True	true	true
True	false	false
False	true	false
False	false	false

a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

The ! operator is a unary operator, taking only one argument and negating its value:

a	!a
true	false
false	true

Examples using logical operators (assume x = 6 and y = 2):

```
!(x > 2) → false
(x > y) && (y > 0) → true
(x < y) && (y > 0) → false
(x < y) || (y > 0) → true
```

PROGRAM 4: Demonstrates && (AND) operator

```
#include <iostream>
using namespace std;
int main()
{
int age, height;
cout << "Enter age of candidate: "; cin >> age;
cout << "Enter height of candidate in cm: "; cin >> height;
if (age>=18 && height>=160)
cout << "The candidate is selected ";
else
cout << "Sorry! The candidate is not selected ";
return 0;
}
```

PROGRAM 5: Demonstrates || (OR) operator

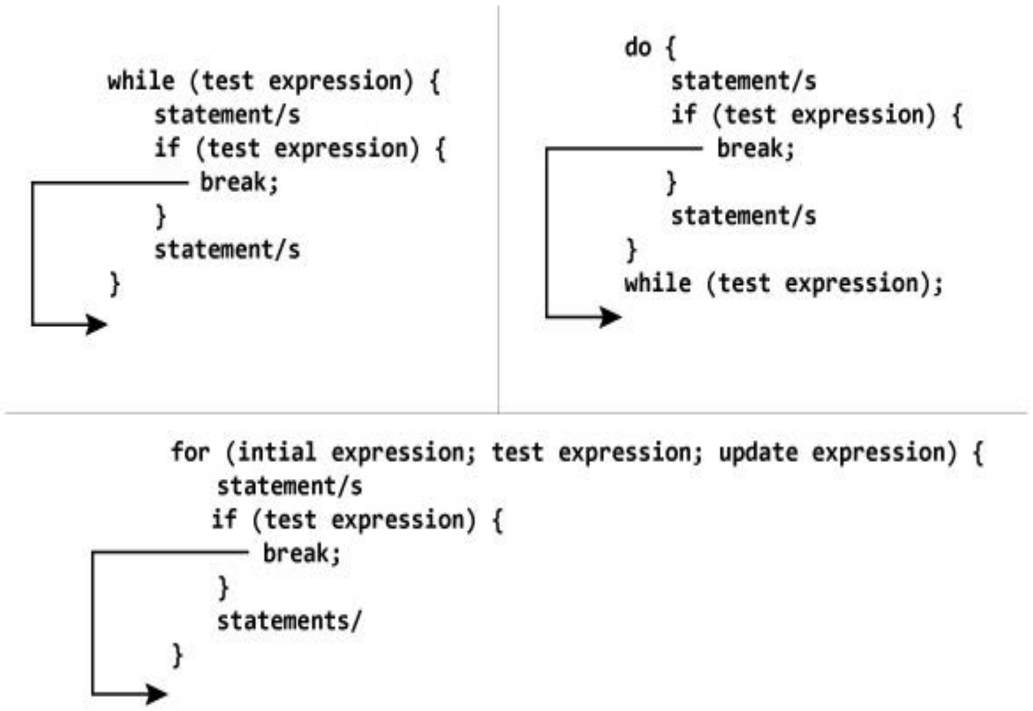
```
#include <iostream>
#include <conio.h>
using namespace std;
int main()  {
int month;
cout<<"Enter month in number: ";   cin>>month;
if(month == 12 || month == 1 || month == 2)
cout<<"It is Winter";
else if(month == 3 || month == 4 || month == 5)
cout<<"It is Spring";
else if(month == 6 || month == 7 || month == 8)
cout<<"It is Summer";
else if(month == 9 || month == 10 || month == 11)
cout<<"It is Autumn";
else
cout<<"You entered wrong Month";
return 0;
}
```

PROGRAM 6: Demonstrates || (OR) operator and nested for loops

```
#include <iostream>
#include<conio.h>
using namespace std;
int main() {
int r, c, k ,j;
char ch;
do{
cout<<"Enter Number of rows:";
cin>>r;
cout<<"Printing Triangle:"<<endl;
for (k=1; k<=r ; k++){
   for (j=1; j<=k; j++)
     cout<<'*';
     cout<<endl;
     }
cout<<"Enter Number of rows:";
cin>>r;
cout<<"Enter Number of columns:";
cin>>c;
cout<<"Printing rectangle:"<<endl;
for (k=1; k<=r ; k++){
   for (j=1; j<=c; j++)
     cout<<'*';
     cout<<endl;
     }
cout<<"Do U want to print more shapes??? (Y / N)";
ch = getche();
}
while(ch == 'y' || ch == 'Y');
return 0;
}
```

BREAK STATEMENT

The break; statement terminates a loop (for, while and do..while loop) and a switch statement immediately when it appears.



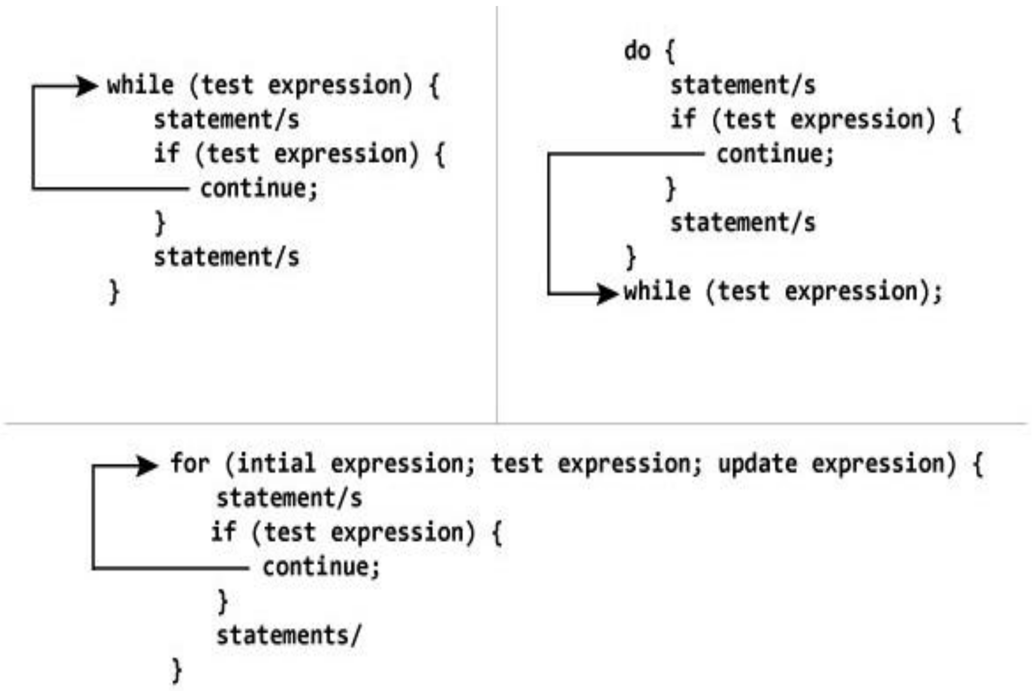
```
while (test expression) {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
```

```
do {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
while (test expression);
```

```
for (intial expression; test expression; update expression) {
    statement/s
    if (test expression) {
        break;
    }
    statements/
}
```

C++ CONTINUE STATEMENT

It is sometimes necessary to skip a certain test condition within a loop. In such case, continue; statement is used in C++ programming.



```
while (test expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
```

```
do {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
while (test expression);
```

```
for (intial expression; test expression; update expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statements/
}
```

PROGRAM 7: Demonstrating the break statement

```
include <iostream>
using namespace std;
int main() {
float number, sum = 0.0;
while (true) {          // test expression is always true
cout << "Enter a number: ";   cin >> number;
if (number != 0.0)
sum += number;
else
break;         // terminates the loop if number equals 0.0
}
cout << "Sum = " << sum;
return 0;
}
```

PROGRAM 8: Demonstrating the continue statement

```
#include <iostream>
using namespace std;
int main() {
long dividend, divisor;
char ch;
do {
cout << "Enter dividend: "; cin >> dividend;
cout << "Enter divisor: ";  cin >> divisor;
if( divisor == 0 )  {              //if attempt to divide by 0,
cout << "Illegal divisor\n";   //display message
continue;                      //go to top of loop
}
cout << "Quotient is " << dividend / divisor;
cout << ", remainder is " << dividend % divisor;
cout << "\nDo another? (y/n): ";  cin >> ch;
} while( ch != 'n' );
return 0;
}
```

Exercise 1:

Write a program to input a character from user and check whether given character is alphabet, digit or a special character. Use if-else for logic and if user press Enter only the program should exit.

```
Input character: a
'a' is alphabet
Input character: A
'A' is alphabet
Input character: #
'#' is special character
Input character: 1
'1' is a number
Input character:
Exit
```

EXERCISE 2:

Write a program which prints the following pattern by using nested loops

```
< - >>>>>>>>>>
<< - >>>>>>>>>
<<< - >>>>>>>>
<<<< - >>>>>>>
<<<<< - >>>>>>
<<<<<< - >>>>>
<<<<<<< - >>>>
<<<<<<<< - >>>
<<<<<<<<< - >>
<<<<<<<<<< - >
```

EXERCISE 3:

Write a program which prints the following pattern by using nested loops

```
     *
    ***
   *****
  *******
 *********
***********
```

Exercise 4:

Write a program which takes a number as input from user between range (1 to 99) and converts the number into words by using multiple switch cases.

```
Enter a number: 55
Fifty Five
Enter a number: 3
Three
Enter a number: 20
Twenty
Enter a Number: 0
Exit
```