Programming fundamentals

Lecture 7: Switch statement, Nested if, loops

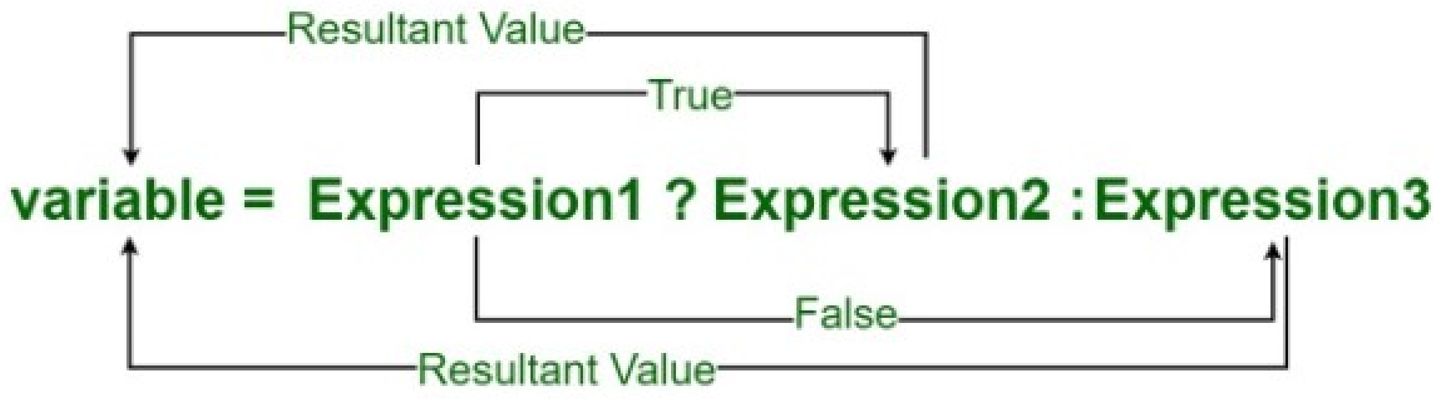
Recap

- operators, and
- conditional statements

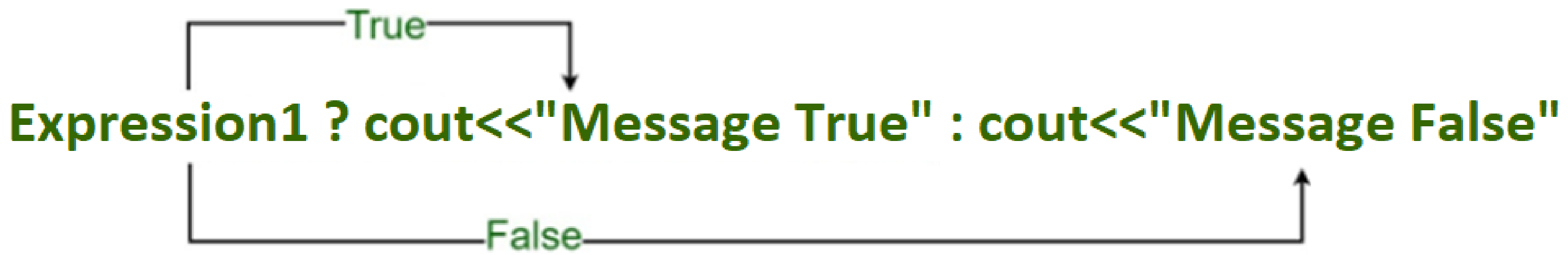
Agenda

- Conditional operator
- Nested if
- Switch Statement, Break Statement, goto Statement
- Repetitive structure: while loop, break/continue

Conditional operator



OR



Consider this code:

```
if ( n1 > n2 )
{
  max = n1;
}
else
{
   max = n2;
}
```

OR

```
#include <iostream>
Using namespace std;
int main()
{
// variable declaration
int n1 = 5, n2 = 10, max;

// Largest among n1 and n2
max = (n1 > n2) ? n1 : n2;

// Print the largest number
Cout<<"Largest number between“;
Cout<<n1<< n2<< max<<endl;

return 0;
}
```

```
cout << ( grade >= 60 ? "Passed" : "Failed" );
```

OR

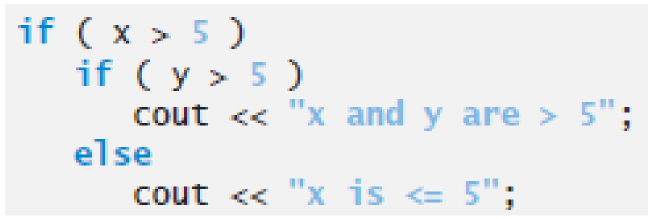
```
grade >= 60 ? cout << "Passed" : cout << "Failed";
```

If we want to print something

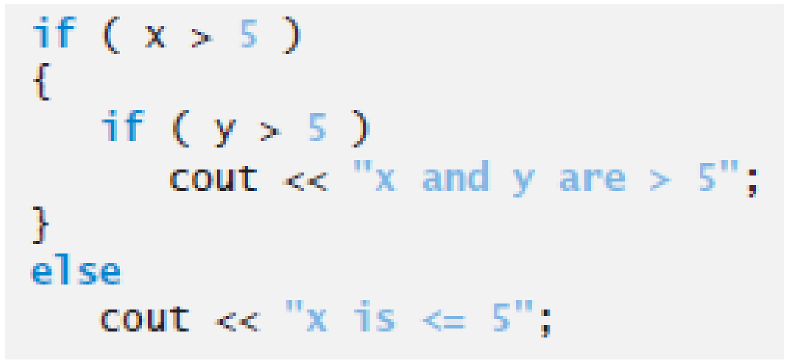
```
#include <iostream>
#include <string>
using namespace std;
int main() {
    int n1 = 5, n2 = 10, max;

    // Largest among n1 and n2
    string m = (n1 > n2) ? "n1 is greater" : "n2 is greater";
    cout << m;
return 0;
}
```

Nested if…else *Statements*

- if…else selection statements *inside* other if…else selection statements
- C++ compiler always associates an else with the *immediately preceding*

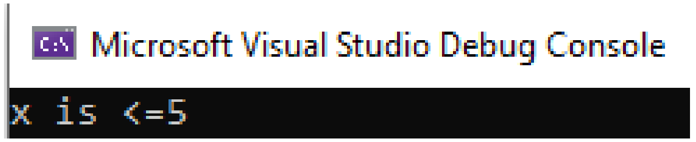


```
if ( x > 5 )
   if ( y > 5 )
      cout << "x and y are > 5";
   else
      cout << "x is <= 5";
```



```
if ( x > 5 )
{
   if ( y > 5 )
      cout << "x and y are > 5";
}
else
   cout << "x is <= 5";
```

```
int x = 9,y=3;
```



Microsoft Visual Studio Debug Console

```
x is <=5
```

Some built-in functions

- Rand()   #include <cstdlib>
- **sqrt( )**, **sin( )**, **cos( )**, **tan( )**, **log( )**, **ceil( )**, and **floor( )**   #include<cmath>.
- Generating random number between a range

Example: nested if

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
	int magic; // magic number
	int guess; // user's guess
	magic = rand(); // get a random number
	cout << "Enter your guess: ";
	cin >> guess;
	if (guess == magic) {
		cout << "** Right **\n";
		cout << magic << " is the magic number.\n";
	}
	else {
		cout << "...Sorry, you're wrong.";
		// use a nested if statement
		if (guess > magic)
				cout << " Your guess is too high.\n";
		else
				cout << " Your guess is too low.\n";
		}
	return 0;
}
```

Switch statement

- the value of an expression is successively tested against a list of integer or character constants.
- When a match is found, the statement sequence associated with that match is executed. The general form of the **switch** statement is

```
switch(expression) {
    case constant1:
            statement sequence
            break;
    case constant2:
            statement sequence
            break;
    case constant3:
            statement sequence
            break;
            .
            .
            .
    default:
            statement sequence
}
```

- **break** statement causes program flow to exit from the entire **switch** statement and resume at the next statement outside the **switch**
- The **switch** expression must evaluate to either a character or an integer value. (Floating-point expressions, for example, are not allowed.) Frequently, the expression controlling the **switch** is simply a variable.
- The **default** statement sequence is performed if no matches are found. The **default** is optional; if it is not present, no action takes place if all matches fail.
- When a match is found, the statements associated with that **case** are executed until the **break** is encountered or, in the case of the **default** or the last **case**, the end of the **switch** is reached.

Difference from if statements

- The **switch** differs from the **if** in that **switch** can test only for equality (i.e., for matches between the **switch** expression and the **case** constants)

```
#include <iostream>
using namespace std;
int main()
{
    int num;
    cin >> num;
    switch (num)
    {
        case 1:
                cout << "You entered 1\n";
                break;
        case 2:
                cout << "You entered 2\n";
                break;
        case 3:
                cout << "You entered 3\n";
                break;
        default:
                cout << "Default case.\n";
    }
    return 0;
}
```

Nested switch

```
switch(ch1)
{
    case 'A’:
        cout << "This A is part of outer switch";
        switch(ch2) {
            case 'A':
                cout << "This A is part of inner switch";
                break;
            case 'B': // ...
        }
        break;
    case 'B': // ...
```

Goto statement

```
#include <iostream>
using namespace std;
int main()
{
    int num;
    cin >> num;
    abc:
    cout << "control shifted after goto"<<endl;
    switch (num) {
        case 1:
            cout << "You entered 1\n";
            break;
        case 2:
            cout << "You entered 2\n";
            break;
        case 3:
            cout << "You entered 3\n";
            goto abc;
                break;
        default:
                cout << "Default case.\n";
    }

    return 0;
}
```

Loops

- While
- Do-while
- for

While loop

```cpp
#include <iostream>
using namespace std;
int main()
{
    int x = 0;
    cout << x << endl;
    cout << x+1 << endl;
    cout << x + 2 << endl;
    cout << x + 3 << endl;
    cout << x + 4 << endl;
    cout << x + 5 << endl;
    cout << x + 6 << endl;
    cout << x + 7 << endl;
    cout << x + 8 << endl;
    cout << x + 9 << endl;
    cout << x + 10 << endl;
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int main()
{
    int x = 0;
    while (x != 11)
    {
        cout << x<<endl;
        x += 1;
    }
    return 0;
}
```

Another example..

```
#include <iostream>
using namespace std;
int main()
{
    int num = 0;
    while (num != 100)
    {
        cout << "Enter a number (100 to stop): ";
        cin >> num;
    }
    return 0;
}
```

Infinite loop

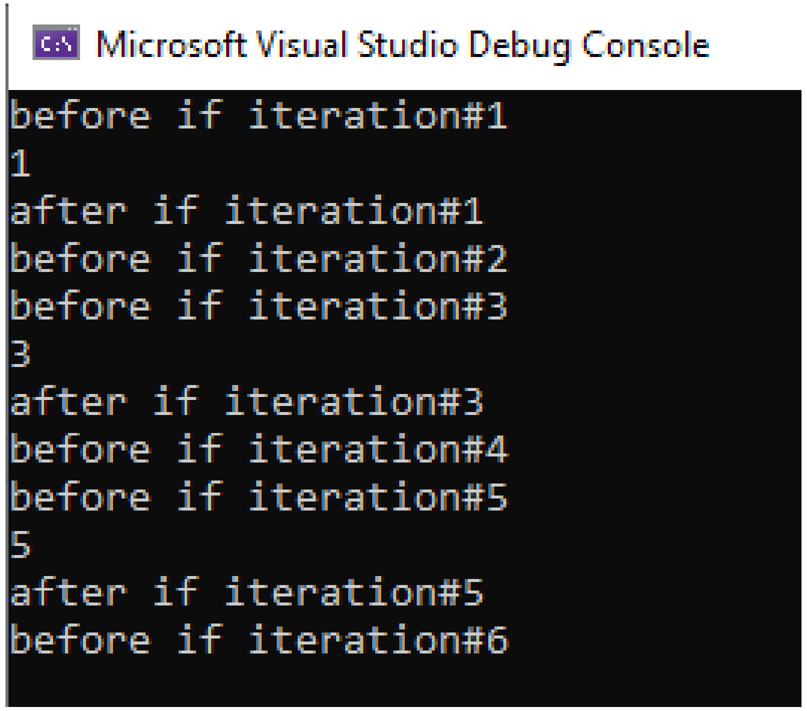
```
#include <iostream>
using namespace std;
int main()
{
    int x = 0;
    while (1)
    {
        cout << x<<endl;
        x += 1;
    }
    return 0;
}
```

continue

- To bypass the loop’s normal control structure
- The **continue** statement forces the next iteration of the loop to take place, skipping any code between itself and the conditional expression that controls the loop

```cpp
#include <iostream>
using namespace std;
int main()
{
int x = 0; int num = 0;
while(x <= 5)
{
    x++;
    cout << "before if iteration#" << x << endl;
    if ((num++) % 2==1) continue;
    cout << x << endl;
    cout << "after if iteration#" << x << endl;

}
return 0;
}
```

These statements will be skipped if **continue** gets executed


Microsoft Visual Studio Debug Console

```
before if iteration#1
1
after if iteration#1
before if iteration#2
before if iteration#3
3
after if iteration#3
before if iteration#4
before if iteration#5
5
after if iteration#5
before if iteration#6
```

Using break to Exit Loops

```
#include <iostream>
using namespace std;
int main()
{
int x = 0; int num = 0;
while(x <= 5)
{
    if ((num++) % 2 == 1)
        break;
    cout << "Num: " << num << endl;

}
return 0;
}
```

Recommended reads

- Dietal & Dietal
    - Chapter 4