

Problem Solving With C++

Mohammad A. Rajabi

Dept. of Geomatics Eng.

University of Tehran

Cell: 0912 132 5823

Email: marajabi@ut.ac.ir

<http://www.marajabi.com>

Unfinished Business ... Casts

(force an expression to be of a specific type)

```
#include <iostream>
using namespace std;

int main() // print i and i/2 with fractions
{
    int i;

    for(i=1; i<=100; ++i )
        cout << i << "/ 2 is: " << (float) i / 2 << '\n';

    return 0;
}
```

Program Control Statements

- *selection* e.g. (if, switch)
- *iteration* e.g. (for, while, do-while)
- *jump* e.g. (break, continue, return, goto)

Revisiting *if*

- *Any* valid C++ expression can be used to control the `if`
- The controlling expression must evaluate to either a true or a false result

if ... example

```
// Divide the first number by the second.

#include <iostream>
using namespace std;

int main()
{
    int a, b;

    cout << "Enter two numbers: ";
    cin >> a >> b;

    if(b) cout << a/b << '\n';
    else cout << "Cannot divide by zero.\n";

    return 0;
}
```

Nested ifs

- Nested `if` is an `if` statement that is the target of another `if` or an `else`
- an `else` statement always refers to the nearest `if` statement that is within the same block

Nested if ... example

```
if(i) {  
    if(j) statement1;  
    if(k) statement2; // this if  
    else statement3; // is associated with this else  
}  
else statement4; // associated with if(i)
```

Nested if ... a concrete example

```
// Magic Number program: 2nd improvement.

#include <iostream>
#include <cstdlib> // for older compilers, use <stdlib.h>
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.";
        if(guess > magic) cout <<" Your guess is too high.\n";
        else cout << " Your guess is too low.\n";
    }

    return 0;
}
```

if-else-if Ladder



```
if(condition)
    statement;
else if(condition)
    statement;
else if(condition)
    statement;
.
.
.
else
    statement;
```

if-else-if Ladder ... example

- Example:

```
if(num_credits < 0)
{
    cout << "Come on, get real" << endl;
}
else if (num_credits < 12)
{
    cout << "Part-time student" << endl;
}
else if (num_credits < 18)
{
    cout << "Full-time student" << endl;
}
else
{
    cout << "Glutton for punishment" << endl;
}
```

The switch statement

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

- Switch is C++'s multiway decision statement
- The switch expression must evaluate to either a character or an integer value

The switch statement ... contd

- The (optional) default statement sequence is executed if no matches are found
- switch can test only for equality between the switch expression and the case constant *(this differs from the if conditional expression)*

Switch ... example

```
// Demonstrate the switch using a simple "help" program.
#include <iostream>
using namespace std;

int main()
{
    int choice;

    cout << "Help on:\n\n";
    cout << "1. for\n";
    cout << "2. if\n";
    cout << "3. switch\n\n";

    cout << "Enter choice: (1-3): ";
    cin >> choice;

    switch(choice) {
        case 1:
            cout << "for is C++'s most versatile loop.\n";
            break;
        case 2:
            cout << "if is C++'s conditional branch statement.\n";
            break;
        case 3:
            cout << "switch is C++'s multi-way branch statement.\n";
            break;
        default:
            cout << "You must enter a number between 1 and 3.\n";
    }

    return 0;
}
```

Ah! ... break;

```
#include <iostream>
using namespace std;

int main()
{
    int i;

    for(i=0; i<5; i++) {
        switch(i) {
            case 0: cout << "less than 1\n";
            case 1: cout << "less than 2\n";
            case 2: cout << "less than 3\n";
            case 3: cout << "less than 4\n";
            case 4: cout << "less than 5\n";
        }
        cout << '\n';
    }

    return 0;
}
```

No break; ???

```
switch(i) {  
    case 1:  
    case 2:  
    case 3: do_something();  
        break;  
    case 4: do_something_else();  
        break;
```

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': // ...  
}
```

Some variations on the for loop

```
for (x=0, y=10; x<=10; ++x, --y)  
    cout << x << ' ' << y << '\n';
```

Some variations on the for loop

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
    int i;

    // print numbers until a key is pressed
    for(i=0; !kbhit(); i++) cout << i << ' ';

    return 0;
}
```

Some variations on the for loop

```
#include <iostream>
using namespace std;

int main()
{
    int x;

    for(x=0; x != 123; ) {
        cout << "Enter a number: ";
        cin >> x;
    }

    return 0;
}
```

The infinite loop

```
for (;;)
{
    //...
}
```

Time Delay Loops

(the for loop is a fascinating loop)

```
for (x=0; x<1000; x++) ;
```

While loops

- Loops of various sorts are used to repeat a set of statements some number of times.
 - There are three basic ways to implement loops in C++
 - while loops
 - do-while loops
 - for loops
 - While statements have 2 parts:
 - The loop condition
 - The loop body
 - While loops loop executing the loop body until the loop condition becomes false.

While loops

- If the loop condition is TRUE
 - The loop body is executed.
 - The loop condition is evaluated again.
- If the loop condition is FALSE
 - The loop body is not executed

What's wrong with this?

```
x = 10;  
while ( x > 0 );  
{  
    cout << x << endl;  
    x--;  
}
```

Do-while loops

- **Do-while loops**

- Do-while loops are very much like while loops, **except**
- that the loop body is executed once before the loop condition is ever evaluated.
- After that, they are exactly the same.
- Example:

```
do
{
    cout << "This is a test";
} while (0);
```

– Compare this to:

```
while (0)
{
    cout << "This is a test";
```

Using `continue;`

- Force an early iteration of a loop
- skip code between the `continue;` statement and the expression that controls the loop

continue; ... example

```
#include <iostream>
using namespace std;

int main()
{
    int x;

    for(x=0; x<=100; x++) {
        if(x%2) continue;
        cout << x << ' ';
    }

    return 0;
}
```

```
continue;
```

- In the case of a `for` loop the increment part of the loop is performed and then the conditional expression is executed

Break; to exit loops

- Force an immediate exit from a loop, bypassing the loop's conditional test
- Program control resumes at the next statement following the loop

Break; ... example 1

```
#include <iostream>
using namespace std;

int main()
{
    int t;

    for(t=0; t<100; t++) {
        if(t==10) break;
        cout << t << ' ';
    }

    return 0;
}
```

Break;

- A `break;` in a nested loop will cause an exit from the innermost loop, *not all loops*,

Break; ... example 2

```
#include <iostream>
using namespace std;

int main()
{
    int t, count;

    for(t=0; t<100; t++) {
        count = 1;
        for(;;) {
            cout << count << ' ';
            count++;
            if(count==10) break;
        }
        cout << '\n';
    }

    return 0;
}
```

Program to find Prime Numbers

```
/* This program finds the prime numbers from
   2 to 1000.
*/
#include <iostream>
using namespace std;

int main()
{
    int i, j;

    for(i=2; i<1000; i++) {
        for(j=2; j <= i/2; j++)
            if(!(i%j)) break; // if factor found, not prime
        if(j>i/2) cout << i << " is prime\n";
    }

    return 0;
}
```

The goto statement

C++'s unconditional branch statement

```
for (...) {  
    for (...) {  
        while (...) {  
            if (...) goto stop;  
            .  
            .  
            .  
        }  
    }  
}  
stop:  
    cout << "Error in program.\n";
```

A comprehensive example

```
// Magic Number program: Final improvement.
#include <iostream>
#include <cstdlib> // for older compilers, use <stdlib.h>
using namespace std;

void play(int m);

int main()
{
    int option;
    int magic;
    magic = rand();

    do {
        cout << "1. Get a new magic number\n";
        cout << "2. Play\n";
        cout << "3. Quit\n";
        do {
            cout << "Enter your choice: ";
            cin >> option;
        } while(option<1 || option>3);

        switch(option) {
            case 1:
                magic = rand();
                break;
            case 2:
                play(magic);
                break;
            case 3:
                cout << "Goodbye\n";
                break;
        }
    } while(option!=3);

    return 0;
}
```

```

// Play the game.
void play(int m)
{
    int t, x;

    for(t=0; t<100; t++) {
        cout << "Guess the number: ";
        cin >> x;
        if(x==m) {
            cout << "** Right **\n";
            return;
        }
        else
            if(x<m) cout << "Too low.\n";
            else cout << "Too high.\n";
    }
    cout << "You've used up all your guesses. Try again.\n";
}

```