Topic 3
Selection

Department of
Engineering Physics

University of Gaziantep

Course web page
www.gantep.edu.tr/~bingul/ep241

Sep 2013
Introduction

This lecture covers the following topics:

- Relational and logical operators
- Boolean expressions
- The `if` structure
- The `if .. else` structure
- The `if .. else if .. else` structure
- The `switch` statement
- The `?` Operator
Relational Operators

Control statements use *relation operators* to compare two objects. There are six relational operators as follows:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>x &lt; y</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
<td>x &lt;= y</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>x &gt; y</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
<td>x &gt;= y</td>
</tr>
<tr>
<td>==</td>
<td>equal to</td>
<td>x == y</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
<td>x != y</td>
</tr>
</tbody>
</table>

Example:

```java
if ( b != 0 ) c = a/b;
```

*control structure using a relational operator*
Logical Operators

*Compound* relation expressions can be formed by *logical operators*:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>logical AND, conjunction. Both sides must be true for the result to be true</td>
<td>( x &gt; 2 \land \text{and} \ y = 3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>Logical NOT, negation</td>
<td>( ! (x&gt;0) )</td>
</tr>
</tbody>
</table>

Example:

```java
if ( b != 0 && a > 0 ) c = a/b;
```

*control structure using a compound relational operator*
Results for the `&&` and `||` operators:

| X     | Y       | X && Y (AND) | X || Y (OR) |
|-------|---------|--------------|-------------|
| true  | true    | true         | true        |
| true  | false   | false        | true        |
| false | true    | false        | true        |
| false | false   | false        | false       |

```java
if (b != 0 && a > 0) c = a/b;
```
Expressions that evaluate to **true** or **false** are called *Boolean*.

We can form Boolean expressions inside control statements (previous page) or in the form of assignments as follows:

```c
int x=1, y=2, s;
bool u, z = true, t, w;
u = x > 3;
z = x <= y && y > 0;
t = y <= 0 || z;
w = !s;
s = 2 > 1;
```

Note that variables *u, z, t, and w* are declared as type *bool* and so can represent the states *true* and *false*.

Also *literal constants* *true* and *false* can be used in assignments and relational operations.

### Results

- \( u = \text{false} \) since \( 1>3 \) is false.
- \( z = \text{true} \) since \( 1\leq 2 \) and \( 2>0 \) are both true.
- \( t = \text{true} \) since \( z \) is true.
- \( w = \text{false} \) since \( s \) is true, therefore its negation is false.
- \( s = 1 = \text{true} \) since \( 2>1 \) (integer representation! see next).
5. if structure

The if statement allows conditional execution; the general form is:

```plaintext
if (condition) {
    statements
    
}
```

If `condition` is *true* then the block defined by the braces `{...}` is executed.

If `statements` is a single statement then the braces can be omitted:

```plaintext
if (condition)
    single-statement
```

Variable `c` is assigned only if the condition is *true*. But, the output statement will be executed in *any case*. 
6. if .. else structure

The **if** .. **else** structure allows both outcomes of a selection to be defined.

The general form is:

```cpp
if (condition) {
    statements1
    ...
}
else {
    statements2
    ...
}
```

If *condition* is *true* then the first block is executed, otherwise (false) the second block is executed.

```cpp
if ( x+y != 0. ) {
    c = 1/(x+y);
    cout << "c = " << c << endl;
} else {
    cout << "c is undefined! " << endl;
}
```
An example program using the `if .. else` structure.

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

int main() {
    double c, x, y;
    cin >> x >> y;
    if (x + y != 0.0) {
        c = 1 / (x + y);
        cout << "c = " << c;
    } else {
        cout << "c is undefined!";
    }
}
```

```
1 2
3 -3
```

```
c = 0.333333
```

```
c is undefined!
```
A leap year is a year in which one extra day (February 29) is added to the regular calendar. Most of us know that the leap years are years that are divisible by 4. For example 1992 and 1996 are leap years.

But this rule does not work generally. For example centennial years are not leap years. For example 1800 and 1900 are not leap years.

A year is called the leap year if

- it is divisible by 4 and but not divisible by 100
- or it is divisible by 400

Write a program that reads a year and outputs whether it is leap year or not.

Solution will be given in the lecture.
More levels of selection can be added with the `else if` statement.

Add as many blocks as you need.

This is executed if none of the above conditions are true.

```java
if (condition1) {
   .
   statements1
   .
} else if (condition2) {
   .
   statements2
   .
} else if (condition3) {
   .
   statements3
   .
} else {
   .
   statements4
   .
}
```
Problem 2: Quadratic Roots

Consider the quadratic equation:

\[ f(x) = a \ x^2 + b \ x + c \]

The roots are the values of \( x \) such that \( f(x) = 0 \).

Analytical solution:

\[ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

Three cases for the result \( b^2 > 4ac \)

i) \( b^2 > 4ac \) there are two roots.

ii) \( b^2 = 4ac \) there is one root.

iii) \( b^2 < 4ac \) the roots are imaginary.

Examples
we can use these results to validate our program

i) \((x-4)(x+2) = 0\)
when \( x = 4, \ x = -2 \)

\[ f(x) = x^2 - 2 \ x - 8 \]
\[ a = 1, \ b = -2, \ c = -8 \]

\[ x = 2/2 \pm \sqrt{0}/2 = 2 \]

ii) \((x-2)(x-2) = 0\)
when \( x = 2 \)

\[ f(x) = x^2 - 4 \ x + 4 \]
\[ a = 1, \ b = -4, \ c = -4 \]

\[ x = 4/2 \pm \sqrt{0}/2 = 2 \]
#include <iostream>
#include <cmath>
using namespace std;

int main() {

double a, b, c;
cin >> a >> b >> c;

double Delta = b*b - 4*a*c;

if ( Delta < 0. ) {
    cout << "The roots are imaginary!" << endl;
} else if ( Delta == 0. ) {
    double x1 = -b / (2*a);
    cout << "The root is " << x1 << endl;
} else {
    double x1 = ( -b - sqrt(Delta) ) / (2*a);
    double x2 = ( -b + sqrt(Delta) ) / (2*a);
    cout << "The two roots are " << x1 << " and " << x2 << endl;
}
}
Consider the composite function:

\[ f(x) = \begin{cases} 
2 & \text{for } x < 3 \\
\frac{2x}{3} & \text{for } 3 \leq x \leq 6 \\
4 & \text{for } x > 6 
\end{cases} \]

Write a program that inputs a value for \( x \) and outputs the corresponding value of \( f(x) \).
#include <iostream>
using namespace std;

int main() {
    double x, f;
    cout << "input x: ";
    cin >> x;
    if (x < 3.) f = 2.0;
    else if (x < 6.) f = 2.0/3.0*x;
    else f = 4.0;
    cout << "f(" << x << ") = " << f << endl;
    return 0;
}

Example outputs

input x: 0  f(0) = 2
input x: 1  f(1) = 2
input x: 2  f(2) = 2
input x: 3  f(3) = 2
input x: 4  f(4) = 2.66667
input x: 5  f(5) = 3.33333
input x: 6  f(6) = 4
input x: 7  f(7) = 4
Problem 4: Flow Chart

Implement the following flow chart into a C++ program.

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

int main() {
    string H;
    double h;
    cin >> h;

    if (h<1.0)   H = "small";
    else if (h<10.) H = "medium";
    else         H = "large";

    cout << H << endl;
}
```
8. **switch Statement**

This is an alternative for the `if .. else if .. else` structure. General form:

```java
switch (expression)
{
    case constant1:
        group of statements 1;
        break;
    case constant2:
        group of statements 2;
        break;
    ...
    default:
        default group of statements;
}
```
int main(){

    int classCode;
    cin >> classCode;

    switch(classCode){
        case 1:
            cout << "Freshman\n";
            break;
        case 2:
            cout << "Sophomore\n";
            break;
        case 3:
            cout << "Junior\n";
            break;
        case 4:
            cout << "Graduate\n";
            break;
        default:
            cout << "bad code\n";
    }
}

int main(){

    int classCode;
    cin >> classCode;

    if(classCode==1){
        cout << "Freshman\n";
    }
    else if(classCode==2){
        cout << "Sophomore\n";
    }
    else if(classCode==3){
        cout << "Junior\n";
    }
    else if(classCode==4){
        cout << "Graduate\n";
    }
    else{
        cout << "bad code\n";
    }
}
9. ? Operator

The ? operator (*conditional expression operator*) provides a concise form of the *if .. else* structure.

The general form is:

```
( condition ) ? expression1 : expression2;
```

The value produced by this operation is either expression1 or expression2 depending on condition being true or false.

Example:

```
max = ( x > y ) ? x : y;
```

is equivalent to

```
if ( x > y )
    max = x;
else
    max = y;
```
Problem 5: Even or Odd

a) Write an algorithm  
b) draw a flow chart and  
c) write a C++ program

to input an integer number and output whether it is even or not. Use the ? operator.

Solution will be given in the lecture.
Problem 6: Bungee Jumping

Consider the bungee jumping as displayed in Fig. At full stretch, the elastic rope of original length $L$ stretches to $L + x$. For a person whose weight is $W$ and a cord with a stiffness $K$, the extension $x$ is given by this formula:

$$x = \frac{W}{K} + \sqrt{\frac{W^2}{K^2} + \frac{2WL}{K}}$$

Assume that the height of the cliff and person are $H = 150$ m and $h = 2$ m respectively, the stiffness is $K = 20$ N/m and the unstretched length is $L = 40$ m.

(a) Write a program to input the weight of the jumper in Newton and output the value for $L + x + h$. Also, print a message if the jumper is able to bungee jump safely or not.

(b) What is the output of your program for $W = 800$ N?

Solution will be given in the lecture.
Homeworks

1. Draw a flow-chart and write a program to input two integer numbers \( p \) and \( q \) and output whether \( p \) is divisible by \( q \) or not if \( q \) is non-zero. Use the \( \div \) operator.

2. Write a program that reads a grade A, B, C, D, or F and then prints "excellent", "good", "fair", "poor", or "failure". Use a switch statement.

3. Write a program to input coefficients of a quadratic equation of the form: \( ax^2 + bx + c = 0 \) and output the roots of the equation for all possible the cases: real roots, complex roots and \( a = 0 \).

Examples:
* a=1, b=0, c= -4 \( \Rightarrow \) \( x_1 = 2.0 \) and \( x_2 = -2.0 \)
* a=0, b=4, c= -2 \( \Rightarrow \) \( x_1 = x_2 = 0.5 \)
* a=1, b=1, c= 1 \( \Rightarrow \) \( x_1 = -0.5-0.866i \) and \( x_2 = -0.5+0.866i \)