Selection Control Structures

Outline
- Boolean Expressions
- Boolean Operators
- Relational Operators
- if – else statement

Control Structures
- Flow of programming execution control is sequential unless a “control structure” is used to change that
- there are 2 general types of control structures:
  - Selection (also called branching)
  - Repetition (also called looping)

C++ control structures
- Selection
  - if
  - if...else
  - switch
- Repetition
  - for loop
  - while loop
  - do...while loop
Conditional and Logical Expressions

- Control Structures use conditional (relational) and logical (Boolean) expressions.
- Conditional expressions use relational operators:
  - < <= > >= == !=
- Boolean expressions use logical operators:
  - ! && ||

bool Data Type

- Result value from either a relational expression or Boolean expression is either true or false.
- In C++, the data type bool is a built-in type consisting of just 2 Boolean values, the constants true and false.
- We can declare (Boolean) variables of type bool to hold Boolean values true or false.
- In C++, the value 0 represents false and any nonzero (1) represents true.

Logical (Boolean) Expressions

A Boolean expression is an expression consists of:
- Conditional expressions with relational operators
- A Boolean variable or constant
- Boolean expressions with Boolean operators.

Conditional expressions with relational operators

- Conditional expression is in the form
  
  Expression1 Operator Expression2

Examples:
- Score1 > Score2
- SideA + SideB == SideC
- Length != 0
- CoefB*CoefB - 4*CoefA*CoefC > 0.0
- Count <= 10
- Response == 'Y'
Relational Operators

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

Example

```c
int x, y;
x = 4;
y = 6;
```

**EXPRESSION** | **VALUE**
---|---
$x < y$ | true
$x + 2 < y$ | false
$x != y$ | true
$x + 3 >= y$ | true
$y == x$ | false
$y == x+2$ | true
$y = x$ | 4 (true)

Boolean variable or constant

```c
bool Done, Flag, Result, Test;
de double Score1, Score2, Score3;
int SideA, SideB, SideC;
```

Done = true;
Flag = false;
Test = Score1 > Score2;
Result = SideA + SideB == SideC;
Result = Done && Flag;

Boolean Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp; &amp;</td>
<td>AND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>NOT</td>
</tr>
</tbody>
</table>
### AND (&&) Operator

<table>
<thead>
<tr>
<th>Value of X</th>
<th>Value of Y</th>
<th>Value of X &amp;&amp; Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

### OR (||) Operator

| Value of X | Value of Y | Value of X || Y |
|------------|------------|---------------|
| true       | true       | true          |
| true       | false      | true          |
| false      | true       | true          |
| false      | false      | false         |

### Not (!) Operator

<table>
<thead>
<tr>
<th>Value of X</th>
<th>Value of !X</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

### Precedence of Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Highest precedence</td>
</tr>
<tr>
<td>* / %</td>
<td>Left</td>
</tr>
<tr>
<td>+ -</td>
<td>Left</td>
</tr>
<tr>
<td>&lt; &lt;= &gt; &gt;=</td>
<td>Left</td>
</tr>
<tr>
<td>== !=</td>
<td>Left</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>Lowest precedence</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Right</td>
</tr>
<tr>
<td>* / %</td>
<td>Left</td>
</tr>
<tr>
<td>+ -</td>
<td>Left</td>
</tr>
<tr>
<td>&lt; &lt;= &gt; &gt;=</td>
<td>Left</td>
</tr>
<tr>
<td>== !=</td>
<td>Left</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>Right</td>
</tr>
</tbody>
</table>
Examples

(\text{Response==}'Y'\text{'}) \text{ || } (\text{Response==}'y'\text{'})
(\text{Count} > 10) \text{ && } (\text{Response} == \text{'Y'}\text{'})

!Done

Example

```
int age, height;

age = 25;
height = 70;

!(age < 10) \text{ true}
!(height > 60) \text{ false}
```

```
Example

```c
int age, weight;
age = 25;
weight = 145;
```

Comparing Strings

- Two objects of type string (or a string object and a C string) can be compared using the relational operators.
- A character-by-character comparison is made using the ASCII character set values.
- If all the characters are equal, then the 2 strings are equal. Otherwise, the string with the character with smaller ASCII value is the “lesser” string.

Write an expression for each

- `taxRate` is over 25% and `income` is less than $20000
  
  \[(\text{taxRate} > 0.25) \&\& (\text{income} < 20000)\]
- `temperature` is less than or equal to 75 or `humidity` is less than 70%
  
  \[(\text{temperature} \leq 75) \|| (\text{humidity} < 70)\]
- `age` is over 21 and `age` is less than 60
  
  \[(\text{age} > 21) \&\& (\text{age} < 60)\]
- `age` is 21 or 22
  
  \[(\text{age} == 21) \|| (\text{age} == 22)\]

Example

```c
string myState;
string yourState;
myState = "Texas";
yourState = "Maryland";
myState == yourState
myState > yourState
myState == "Texas"
myState < "Texas"
```
WARNING about Expressions in C++

- "Boolean expression" means an expression whose value is true or false
- An expression is any valid combination of operators and operands
- Each expression has a value
- This can lead to UNEXPECTED RESULTS
- Construct your expressions CAREFULLY
- Use of parentheses is encouraged
- Otherwise, use precedence chart to determine order

CS 1410 Comp Sci with C++  Selection Control Structures  25

What went wrong?

- This is only supposed to display "HEALTHY AIR" if the air quality index is between 50 and 80.
- But when you tested it, it displayed "HEALTHY AIR" when the index was 35.

```cpp
int AQIndex;
AQIndex = 35;

if (50 < AQIndex < 80)
    cout << "HEALTHY AIR";
```

Analysis of Situation

```
AQIndex = 35;

According to the precedence chart, the expression

(50 < AQIndex < 80) means

(50 < AQIndex) < 80 because < is left associative

(50 < AQIndex) is false (has value 0)

(0 < 80) is true.
```

Corrected Version

```cpp
int AQIndex;
AQIndex = 35;

if ((50 < AQIndex) && (AQIndex < 80))
    cout << "HEALTHY AIR";
```
Comparing Real Values

- Do not compare floating point values for equality, compare them for near-equality.

```cpp
float myNumber;
float yourNumber;

cin >> myNumber;
cin >> yourNumber;

if (fabs(myNumber - yourNumber) < 0.00001)
    cout << "They are close enough!" << endl;
```

Selection statements

Selection statements are used to choose an action depending on the current situation in your program as it is running.

Control Structure

- A selection statement is a control structure used to (alter the sequential flow of control) choose an action depending on the current situation in your program as it is running.
- `if-else` statement

Syntax of `if-else` Statement

```cpp
if (Boolean expression)
    Statement1;
else
    Statement2;
```

The else part can be dropped if there is no second choice.
Expressions

Control structure use logical expressions which may include

6 Relational Operators
<  <=  >  >=  ==  !=

3 Logical Operators
!  &&  ||

What can go wrong here?
float average;
float total;
int howMany;

average = total / howMany;

Example

cout<<"Enter your weight";
cin>>Weight;
if (Weight < 300)
  cout<<"You are okay.";
else
  cout<<"You are over Weight.";

Improved Version

float average,
float total;
int howMany;

if (howMany > 0)
{
  average = total / howMany;
  cout << average;
}
else
  cout << "No prices were entered";
Use of blocks recommended

```cpp
if ( Expression )
{
    "if clause"
}
else
{
    "else clause"
}
```

Selection Statement if-else

```
true
expression
false
```

Selection Statement if-else

```
true
Weight < 300
false
```

Example

```cpp
int carDoors, driverAge;
float premium, monthlyPayment;
...
if ((carDoors == 4) && (driverAge > 24))
{
    premium = 650.00;
    cout << " LOW RISK ";
}
else
{
    premium = 1200.00;
    cout << " HIGH RISK ";
}
monthlyPayment = premium / 12.0 + 5.00;
..."
What happens if you omit braces?

```
if (((carDoors == 4) && (driverAge > 24))
   premium = 650.00;
   cout << " LOW RISK ";
else
   premium = 1200.00;
   cout << " HIGH RISK ";
monthlyPayment = premium / 12.0 + 5.00;
```

COMPILE ERROR OCCURS. The "if clause" is the single statement following the if.

---

Single statement in if and else clause

Braces can only be omitted when each clause is a single statement

```
if (lastInitial <= 'K')
   volume = 1;
else
   volume = 2;
```

cout << "Look it up in volume 
" << volume << " of NYC phone book";

---

Example

- Assign value .25 to `discountRate` and assign value 10.00 to `shipCost` if `purchase` is over 100.00
- Otherwise, assign value .15 to `discountRate` and assign value 5.00 to `shipCost`
- Either way, calculate `totalBill`

---

Code for Example

```
if (purchase > 100.00)
{
    discountRate = 0.25;
    shipCost = 10.00;
}
else
{
    discountRate = 0.15;
    shipCost = 5.00;
}
totalBill = purchase *(1.0 - discountRate) + shipCost;
```
if Syntax

```c
if (Boolean expression)
    Statement1;
```

- NOTE: Statement can be a single statement, a null statement, or a block.

Example

```c
cout << "Enter a number";
cin >> Number;
if (Number < 0)
{
    cout << "The number that you enter is negative."
    cout << "\nEnter another number."
    cin >> Number;
}
```

Selection Statement if

- true
- expression
- false
- statement

Terminating your program

```c
int number;
cout << "Enter a non-zero number ";
cin >> number;
if (number == 0)
{
    cout << "Bad input. Program terminated ";
    return 1;
}
// otherwise continue processing
```
Write if or if-else

- If taxCode is 'T', increase price by adding taxRate times price to it.
  
  ```cpp
  if (taxCode == 'T')
      price = price + taxRate * price;
  ```

- If code has value 1, read values for income and taxRate from myInfile, and calculate and display taxDue as their product.
  ```cpp
  if (code == 1)
  {
      myInfile >> income >> taxRate;
      taxDue = income * taxRate;
      cout << taxDue;
  }
  ```

- If A is strictly between 0 and 5, set B equal to 1/A, otherwise set B equal to A.
  ```cpp
  if (A > 0 && A < 5)
  {
      B = 1/A;
  }
  ```

What output? and Why?

```cpp
int age;
age = 20;
if (age == 16)
{
    cout << "Did you get driver's license?"
;}
```

These are equivalent. Why?

```cpp
if (number == 0) if (! number )
{
    .
    .
    .
    .
} else
{
    .
    .
    .
    .
}
```

Each expression is only true when number has value 0.
What output? and Why?

```cpp
int age;
age = 20;
if (age == 16)
    cout << "Did you get driver's license?";  
```

What output? and Why?

```cpp
int age;
age = 30;
if (age < 18)
    cout << "Do you drive?";
    cout << "Too young to vote";
```

What output? and Why?

```cpp
int code;
code = 0;
if (!code)
    cout << "Yesterday";
else
    cout << "Tomorrow";
```

What output? and Why?

```cpp
int number;
number = 2;
if (number == 0)
    cout << "Zero value";
else
    cout << "Non-zero value";
```
Note

Both the if clause and the else clause of an if-else statement can contain any kind of statement, including another selection statement.

Nested Selection

Multi-alternative is also called multi-way branching, and can be accomplished by using NESTED if or if-else statements.

Nested if-else Statement

```c++
if (Boolean expression)
    Statement1;
else if (Boolean expression)
    Statement2;
else
    Statement3;
```

Nested if Statements

- Each Expression is evaluated in sequence, until some Expression is found that is true.
- Only the specific Statement following that particular true Expression is executed.
- If no Expression is true, the Statement following the final else is executed.
- Actually, the final else and final Statement are optional. If omitted, and no Expression is true, then no Statement is executed.
Nested Selections

```cpp
if (creditsEarned >= 90)
    cout << "SENIOR STATUS ";
else if (creditsEarned >= 60)
    cout << "JUNIOR STATUS ";
else if (creditsEarned >= 30)
    cout << "SOPHOMORE STATUS ";
else
    cout << "FRESHMAN STATUS ";
```

Example

```cpp
if (Average >= 90)
    cout<<"Your grade is an A";
else if (Average >= 80)
    cout<<"Your grade is a B";
else if (Average >= 70)
    cout<<"Your grade is a C";
else if (Average >= 60)
    cout<<"Your grade is a D";
else
    cout<<"Your grade is an F";
```

Writing Nested if Statements

- Display one word to describe the int value of number as "Positive", "Negative", or "Zero"

```cpp
if (number > 0)
    cout << "Positive";
else if (number < 0)
    cout << "Negative";
else
    cout << "Zero";
```

Writing Nested if Statements

- Your city classifies a pollution index
  - less than 35 as "Pleasant",
  - 35 through 60 as "Unpleasant",
  - and above 60 as "Health Hazard."
- Display the correct description of the
  - pollution index value.

```cpp
if (index < 35)
    cout << "Pleasant";
else if (index <= 60)
    cout << "Unpleasant";
else
    cout << "Health Hazard";
```
Example

Every Monday thru Friday you go to class
When it is raining you take an umbrella
But on the weekend, what you do depends on the weather
If it is raining you read in bed
Otherwise, you have fun outdoors

Code

```cpp
// Program tells how to spend your day
#include <iostream>
using namespace std;

void main (void)
{
  int day; char  raining;
  cout << "Enter day (use 1 for Sunday)";
  cin >> day;
  cout << "Is it raining? (Y/N)";
  cin >> raining;
  if  ((day == 1) || (day == 7)) { // Sat or Sun
    if  (raining == 'Y')
      cout << "Read in bed";
    else
      cout << "Have fun outdoors";
  }
  else
  {
    cout << "Go to class ";
    if  (raining == 'Y')
      cout << "Take an umbrella";
  }
}
```

Caution

In the absence of braces, an **else** is always paired with the closest preceding **if** that doesn’t already have an **else** paired with it

Example

```cpp
float average;
average = 100.0;
if  (average >= 60.0)
{
  if  (average < 70.0)
    cout << "Marginal PASS";
  else
    cout << "FAIL";
}
else
  cout << "FAIL";
```

FAIL is printed; WHY?
The compiler ignores indentation and pairs the else with the second if
Each I/O stream has a state (condition)

- An input stream enters fail state when you
  - try to read invalid input data
  - try to open a file which does not exist
  - try to read beyond the end of the file
- An output stream enters fail state when you
  - try to create a file with an invalid name
  - try to create a file on a write-protected disk
  - try to create a file on a full disk

How can you tell the state?

- The stream identifier can be used as if it were a Boolean variable. It has value false (meaning the last I/O operation on that stream failed) when the stream is in fail state.
- When you use a file stream, you should check on its state.

Checking on the I/O State

```c++
fstream myOutfile;
myOutfile.open ("A:\myOut.dat", ios::out);
if (! myOutfile)
{
    cout << "File opening error. "
    << "Program terminated." << endl;
    return 1;
} // otherwise send output to myOutfile
```

Testing Selection Control Structures

- to test a program with branches, use enough data sets so that every branch is executed at least once
- this is called minimum complete coverage
Testing Often Combines Two Approaches

<table>
<thead>
<tr>
<th>WHITE BOX TESTING</th>
<th>BLACK BOX TESTING</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Code Coverage</strong></td>
<td><strong>Data Coverage</strong></td>
</tr>
<tr>
<td>Allows us to see the program code while designing the tests, so that data values at the boundaries, and possibly middle values, can be tested.</td>
<td>Tries to test as many allowable data values as possible without regard to program code.</td>
</tr>
</tbody>
</table>

How to Test a Program

- design and implement a **test plan**
- a **test plan** is a document that specifies the test cases to try, the reason for each, and the expected output
- implement the test plan by verifying that the program outputs the predicted results

Body Mass Index Problem

**Problem** Implement a measure called the Body Mass Index (BMI), which computes a ratio of your weight and height, which has become a popular tool to determine an appropriate weight. The formula for non-metric values is

$$\text{BMI} = \frac{\text{weight} \times 703}{\text{height}^2}$$
What is the BMI?

BMI correlates with body fat, which can be used to determine if a weight is unhealthy for a certain height. Do a search of the Internet for "body mass index" and you will find more than a million hits. In these references, the formula remains the same but the interpretation varies somewhat, depending on age and sex. Here is the most commonly used generic interpretation.

<table>
<thead>
<tr>
<th>BMI</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 20</td>
<td>Underweight</td>
</tr>
<tr>
<td>20-25</td>
<td>Normal</td>
</tr>
<tr>
<td>26-30</td>
<td>Overweight</td>
</tr>
<tr>
<td>over 30</td>
<td>Obese</td>
</tr>
</tbody>
</table>

Algorithm

Get Data

Level 1

Prompt for weight
Read weight
Prompt for height
Read height

Test Data

IF weight < 0 OR height < 0
Set dataAreOK to false
ELSE
Set dataAreOK to true

Calculate BMI

Set bodyMassIndex to weight * 703 / height

Print "Your BMI is ", bodyMassIndex, "."

Print "Interpretation and instructions."

IF bodyMassIndex < 20
Print "Underweight: Have a milk shake."
ELSE IF bodyMassIndex < 26
Print "Normal: Have a glass of milk."
ELSE IF bodyMassIndex < 30
Print "Overweight: Have a glass of iced tea."
ELSE
Print "Obese: See your doctor."

C++ Program

#include <iostream>
using namespace std;

int main()
{
    const int BMI_CONSTANT = 703; // Non-metric constant
    float weight; // Weight in pounds
    float height; // Height in inches
    float bodyMassIndex; // Appropriate BMI
    bool dataAreOK; // True if data OK

    // BMI Program
    // This program calculates the body mass index (BMI)
    // given a weight in pounds and a height in inches and
    // prints a health message based on the BMI.
    //**************************************************
    // BMI Program
    // This program calculates the body mass index (BMI)
    // given a weight in pounds and a height in inches and
    // prints a health message based on the BMI.
    //**************************************************
    //**************************************************
    //**************************************************
// Calculate body mass index
bodyMassIndex = weight * BMI_CONSTANT
               / (height * height);
 // Print message indicating status
 cout << "Your BMI is "
       << bodyMassIndex << ". " << endl;
 cout << "Interpretation and instructions. "
       << endl;
 if (bodyMassIndex < 20)
   cout << "Underweight: ...." << endl;
 else if (bodyMassIndex <= 25)
   cout << "Normal: ...." << endl;
 else if (bodyMassIndex <= 30)
   cout << "Overweight:...." << endl;
 else
   cout << "Obese: ...." << endl;
 return 0;
}

Testing the BMI Program

- There is no testing in this program, but there should be!!
- Should you use white box or black box testing?
- What test should be included?
- Where should they be inserted?