Relational Operators, and the If Statement

9.1 Combined Assignments

- Last time we discovered combined assignments such as:
  - `a /= b + c;`
- Which of the following long forms is equivalent to the above form?
  - `a = a / b + c;`
  - `a = a / (b + c);`
- Assume that: `a = b = 6, c = 9`

Relational Operators (4.1)

- So far, we can Input, Output and Calculate
- How can we explore relationships between data?
  - Is your grade greater than 90%?
  - Is it hotter or colder today than yesterday?
  - Do I have enough US dollars to get 100 Euros?

Relational Operators, Explained!

Operator	*Meaning*
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

- All are binary operators
- Left to right associativity

Relational Expression

- An expression is a statement that has value
- Relational expression: an expression that uses a Relational Operator
  - its value is a Boolean value (True or False)

```
int x = 9, y = 42;
```

  - `x > y`
  - `y == x // y = x; is the assignment operator`
  - `x <= (x * y + 99)`

Precedence (page 1125)

Precedence Operators (Highest to Lowest)	
(unary negation) -	
* / %	Arithmetic Operators
+ -	
> >= < <=	Relational Operators
== !=	
= += -= *= /= %=	Assignment Operators

9.2 Practice

- What is the value of the following Relational Expressions?

```
int x = 99, y = 42;
```

  - `x > y`
  - `y <= x`
  - `y != x`
  - `x == (x + 1)`
  - `y == y + 1`

Boolean value (True or False)

- How does the computer represent True and False?
- New data type: `bool`

```
bool bTValue = true;  // 1
bool bFValue = false; // 0
```


9.3 Practice

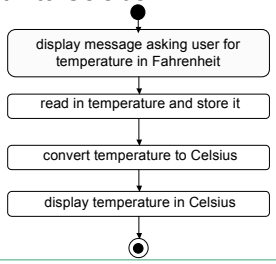
```
bool bvalue;
int x = 5, y = 10;
bvalue = x > y; // value = ??
bvalue = x == y; // value = ??
bvalue = x == y - 5; // value = ??

// what does this output look like?
cout << "Value is: " << bvalue;
```

UML Activity Diagrams

- UML: Unified Modelling Language
- Used to represent algorithms that will later be translated into code
- Give the programmer a visual representation of the solution to a problem
- Can also help the programmer see a solution to a problem

Example

- Write a program that will read in the current temperature in degrees Fahrenheit and convert it into Celsius



The `if` Statement (4.2)

- We execute each statement in our program in order
- What `if` we only want to execute a statement sometimes?
- The `if` Statement!

```
int x=5, y=10;

if(x > y)
{
   // do stuff
}
```

Formally defined

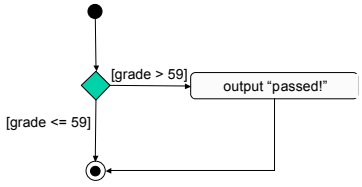
```
if(expression)
{
   statement 1;
   statement 2;
   . . .
   statement n;
}
```

Just like a **function**, start at the top and execute in order to the bottom

- What is an expression?

UML Activity Diagram

```
if( grade > 59 )
{
  cout << "passed!";
}
```



9.4 Practice

```
int x = 5, y = 10;
bool bValue = x > y;

if (bValue)
{
  cout << "value is True" << endl;
}
if (x < y)
{
  cout << x << " < " << y;
  cout << " is true" << endl;
}
```

Coding Standards

```
if(expression)
{
   statement 1;
}
```

```
if(expression)
   statement 1;
```

If you only have ONE statement in the body of the if, the { } are optional in C++.

For this class, the { } must **ALWAYS** be used. Not using { } will result In a loss of style points.

The { } must also be on their own line.

Why?

9.5 More on Truth

- Expressions that evaluate to non-zero are considered `true`

```
int x = 5, y = 0;
if (x + y)
{ // This will be executed
  cout << "x + y is True" << endl;
}
if (y)
{ // This will NOT be executed
  cout << "y is True" << endl;
}
```


9.6 Example

- Your local bookstore has asked you to write a program to help them determine the cost of shipping of customers orders. If the order is $30 or less then shipping will cost $5, if the order is over $30 then shipping will be $3.

Solution

```
double order, shipping;

cout << "Enter the amount of the order: ";
cin >> order;

if( order <= 30 )
{
   shipping = 5.00;
}

if( order > 30 )
{
   shipping = 3.00;
}

cout << "The cost of shipping is $"
     << shipping << endl;

return 0;
```


9.7 Practice

- The bookstore has now changed it's shipping policy so that
  - If the order is $30 or less, shipping is $5
  - If the order is over $30 but less than or equal to $50, shipping is $3
  - If the order is over $50 then shipping is $2
- What would we need to change in the program?

Floating Point and Relational Operators

- Floating point math may not work out as you expect because of round off errors.
- In Math
  - 6 * 2/3 = 4
- In C++, where 0.66666 is equivalent to 2/3
  - 6.0 * 0.66666 =
  - 6.0 * 0.66667 =
  - 6.0 * 0.666666 =
  - 6.0 * (2.0 / 3.0) =

9.8 Example (page 180)

```
double result;

result = 6.0 * 0.666666;

if ( result == 4.0 )
{
  cout << "result == 4.0" << endl;
}

cout << setprecision(6) << fixed;
cout << result << endl;
cout << setprecision(2) << result;
cout << endl;
```


Example

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{

    double result;
    result = 6.0 * 0.666666;
    if( result == 4.0 )
    {
        cout <<" TRUE, result does equal 4.0" <<endl;
    }
    cout <<setprecision(6) <<fixed<< result <<endl;
    cout <<setprecision(2) <<fixed<< result <<endl;
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
3.999996
4.00
Press any key to continue . . .
```


Floating Points in Relational Operators

- How can we get around this problem?

Boolean Flags

- We have seen how to store the value of a relational expression to a `bool` variable.

```
bool bIsSquare = ( length == width);
if ( bIsSquare ) {
}
```

- Why would you want to do this?
- Why not use the relational expression directly?

Boolean Flags

- This use of a `bool` variable is called a flag.
- It is used to keep track of a condition so that the expression is evaluated only once