Lecture 8: The if Statement

PIC 10A
Todd Wittman

if Statement Example

- Programs should be able to adapt to user input and branch in different directions.
- A popular 1960s war protest slogan was "Don't trust anyone over 30."
- How do we check if we can trust someone?

```
int age;
cout << "Enter your age: ";
cin >> age;
if (age < 31)
   cout << "I trust you.\n";
if (age > 30)
   cout << "I DON'T trust you.\n";
```

What is a boolean?

- A boolean statement is a true / false statement.
  - (2<3) is a true statement
  - (2>3) is a false statement
- Later we'll learn about boolean (*bool*) variables, which take on only true or false values. Could also do 1(T) / 0(F).

All the same.

```
bool my_boolean = (2 < 3);
bool my_boolean = true;
bool my_boolean = 1;
```

- Named after logician George Boole (1815-1864).

Sec 3.1: The *if* statement

- An if statement will execute the statements that follow only if the boolean condition in parentheses is TRUE.

```
if (statement is TRUE)  {
        **STATEMENTS**
}
```

Notice the indentation of the closing }. Lines up with the if statement.

```
int x = 2;
if (x < 3)  {
   cout << "hi";
}
```

Outputs "hi"

```
int x = 4;
if (x < 3)  {
        cout << "hi";
}
```

Outputs nothing nada zip

The braces { }

- The braces { } tell us what statements to run if the statement is true.
- Without the braces, only the first line following the if statement is executed.

```
int x = 2;
if (x > 3)  {
   cout << "My ";
   cout << "precious!";
}
```

Outputs nothing.

```
int x = 2;
if ( x > 3)
      cout << "My";
      cout << "precious";
```

Outputs "precious"

- On the right, would have been better not to indent that last line. It's not part of the if statement.

Sec 3.3: The if/else statement

- We can pair an if statement with an else statement. The if block runs when the statement is true. And the else block runs when it's false.

```
int x;
cin >> x;
if (x>0)
     cout << "Your number is positive.";
else
     cout << "Your number is not positive.";
```

- Notice the indentation makes it clear what goes with each block.

The if/else statement

- We can chain the if/else statements.

```
int x;
cin >> x;
if (x>0)
        cout << "Your number is positive.";
else if (x<0)
        cout << "Your number is negative.";
else
        cout << "Your number was zero.";
```

- You could write a bunch of else if statements to handle every possible case.
- The final else handles the "NONE OF THE ABOVE" case, whatever possibility remains.

Example of if/else statement

- An advertisement for a job agency on The Onion website.

Sec 3.2: The Relational Operators

C++	Math	Meaning
>	>	Greater than
>=	≥	Greater than or equal to
<	<	Less than
<=	≤	Less than or equal to
==	=	Equals
!=	≠	Does not equal

1+1 == 2 is TRUE
3 >= 3 is TRUE
1+1 != 3 is TRUE

1+1 == 3 is FALSE
3 > 3 is FALSE
1+1 != 2 is FALSE

Example of multiple if/else's

```
int main ( )  {
    int score;
    cout << "Enter your score: ";
    cin >> score;
    if ( score >= 90 )  {
          cout << "Your grade is A.\n";
          cout << "Good work!";
    }
    else if ( score >= 80 )
          cout << "Your grade is B.";
    else if ( score >= 70 )
          cout << "Your grade is C.";
    else if (score >= 60 )
          cout << "Your grade is D.";
    else  {
          cout << "You fail.\n";
          cout << "See you again next semester!";
    }
}
```

Same code without indenting

```
int main ( )  {
int score;
cout << "Enter your score: ";
cin >> score;
if ( score >= 90 )  {
cout << "Your grade is A.\n";
cout << "Good work!";
}
else if ( score >= 80 )
cout << "Your grade is B.";
else if ( score >= 70 )
cout << "Your grade is C.";
else if (score >= 60 )
cout << "Your grade is D.";
else  {
cout << "You fail.\n";
cout << "See you again next semester!";
}
}
```

Confusing = and ==

- We use == to compare two values because = is used for assignment.
- x == 2 Tests whether x equals 2.
- x = 2 Assigns the value 2 to x.
- Using = in an if statement is a common mistake.
- You won't get a compile error, instead the computer will do something very strange.

```
if (x = 2)
```

- Sets the value of x to be 2 and then always executes the if block. As long as the value assigned is non-zero, treats that as true.
- Visual Studio gives a warning, but still compiles.

Comparing floating point numbers

- Be careful when comparing decimals.

```
double r = sqrt(2);
if ( r*r == 2)
        cout << "r squared is 2";
```

- Outputs nothing, even though it should.
- Because of roundoff error, we have

  sqrt(2) * sqrt(2) = 2.0000000000000004 ≠ 2

- One solution is to check if our answer is really really close to 2.

```
if ( r*r > 1.9999999 && r*r < 2.0000001)
```

- Better solution would be

```
if ( fabs( r*r - 2) < epsilon )
```

FOR A SMALL epsilon

Comparing Strings

- We can use the operators >,<,= for strings, but it's a little tricky.
- The strings are sorted in lexicographic order, like in a dictionary *(see p.125)*.
- But numbers come first, then uppercase letters, then lowercase: 123... then ABC... then abc...

```
string name = "hobbit";
    if (name == "hobbit")        TRUE
    if (name == "Hobbit")        FALSE
    if (name < "orc")            TRUE
    if (name < "Orc")            FALSE
    if (name < "hobbits")        TRUE
    if (name < "2")              FALSE
    if (name < 2)                ERROR
```

Conjuctions

C++	Meaning	When is it true?
&&	and	Both statments must be true
||	or	At least one of the statements must be true
!	not	Tests whether something is not true

To test whether x is between 1 and 5 (1 ≤x ≤5), use

```
if ( x>=1 && x<=5)
```

To test whether x is less than 1 or greater than 5, use

```
if ( x<1 || x>5)
```

To test whether x is not 3

```
if !(x==3)
```

OR

```
if (x != 3)
```

Correct the following bad code

1.)
```
if (x=1)
        cout << x;
```

2.)
```
if x > 0 then
        cout << x;
```

3.)
```
if (name=="Frodo" or "frodo")
        cout << name;
```

4.)
```
if (x and y = 0)
        cout <<"Both x and y are zero.";
```

5.)
```
if (name == "Frodo")
        cout << "Hi Frodo!\n";
        cout << "How's the ring?";
```

Corrected statements

1.)
```
if (x==1)
        cout << x;
```

2.)
```
if (x > 0)
        cout << x;
```

3.)
```
if (name==“Frodo” || name==“frodo”)
        cout << name;
```

4.)
```
if (x==0 && y==0)
        cout <<“Both x and y are zero.”;
```

5.)
```
if (name == “Frodo”)   {
        cout << “Hi Frodo!\n”;
        cout << “How’s the ring?”;
}
```

Confusing && and ||

- Using && instead of || (and vice-versa) is a common error.
- Suppose we ask the user if they want to play a game again.

```
string response;
cout << "Do you want to play again? (y/n) ";
cin >> response;
```

- How do we check their response?

```
if (response==“y” && response==“Y”)
if (response==“y” || response==“Y”)
```

Sec 3.5: Boolean Operations

- Be careful not to confuse && and ||.

  if (response==“y” && response==“Y”) WRONG!

- The boolean statement above will never be true.
- Ex: Find people named Frodo but not Frodo Baggins, e.g. “Frodo Smith”. Which of the if statements below is correct?
- if (name!=“Frodo Baggins”)
- if (name!=“Frodo Baggins” || first_name==“Frodo”)
- if (name!=“Frodo Baggins” && first_name==“Frodo”)
- if (first_name==“Frodo” || last_name!=“Baggins”)
- if (first_name==“Frodo” && last_name!=“Baggins”)

THESE 2 ARE CORRECT

Sec 3.5: De Morgan’s Law

- If you think about it, you can convince yourself of two statements:

  !(A && B) is the same as !A || !B

  !(A || B) is the same as !A && !B

- Q: What did you have for dinner last night?
- A: Not (pizza AND beer).
- So it means I might’ve had pizza or I might’ve had beer, but not both together. So that’s the same as: (not pizza) OR (not beer).

Sec 3.5: De Morgan's Law

- The take-home message is that you have to be cautious when negating a statement.
- Ex: Check if we're inside the continental U.S.

```
if !(state=="Alaska" && state=="Hawaii")     //WRONG!
if !(state=="Alaska" || state=="Hawaii")     //RIGHT!
```

- I avoid negations when possible.
- Ex How can we say the following more simply?

```
if !(x>3 && x < 4)
```

R3.13: Simplify Simplify Simplify

- Use DeMorgan's Law to simplify the following boolean statements.

a.) `!(x>0 && y>0)`

`(x <= 0 || y <= 0)`

b.) `!( x != 0 || y != 0 )`

`(x == 0 && y == 0)`

c.)
```
!(country=="USA" && state != "HI" &&
                state != "AK")
(country != "USA" || state=="HI" || state=="AK")
```

Preview: The *while* Statement

- Sometimes we want to repeat a block of code as long as a certain condition is true.
- The *while* statement is just a repeated *if* statement.

Why do we initalize age=1?

```
int age = 1;
cout << "I trust people with ages ";
while (age <= 30) {
   cout << age << "  ";
   age++;
}
```