PIC 10A

Lecture 8: if statement

if

```
int a,b;

cin >> a >> b;

if (a >= b)
{
    cout >> "First number is bigger";
}
```

If the expression in parenthesis is true then the code block under if gets executed. Otherwise we skip over it.

if – else

```
int a,b;
int max;

cin >> a >> b;

if (a >= b)
{
    max = a;
}
else
{
    max = b;
}

cout << "The larger number is " << max;
```

If the expression in parenthesis is true then the code block under if gets executed otherwise the block under else will get executed. In an if-else structure one or the other code block is always executed!

if – else if

```
int a,b;
cin >> a >> b;

if (a > b)
{
   cout >> "First number is bigger";
}
else if (a == b)
{
   cout >> "They are equal!";
}
```

If the expression in parenthesis is true then the code block under if gets executed and the else if code block is skipped. If it is not true the else if condition is evaluated and if true the code under it is executed. There can be multiple else if statements. At most one of the code blocks in an if structure will ever be done!

if - else if - else

In an if-else if -else structured statement a multiple conditions are checked and if none of the conditions are true then a default else block of code is run:

```
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.";
```

Example

```
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!";
     }
     return 0;
}
```

Same example no 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!”;
}
retun 0;
}
```

Boolean variables (`bool`)

Boolean expressions are expressions that computer can evaluate to be either true or false.

Boolean variables are variables that store one of two values: true or false.

Actually boolean variable stores either 1 for true or 0 for false.

We could write:

```
bool true_or_not = true; // The computer will translate the word true to 1
bool true_or_not = false; // The computer will translate the word false to 0;
bool true_or_not = 5 < 17; // The expression on the right evaluates to true
                           // so value 1 will be assigned to true_or_not
```

How can we use boolean variables?

We will see several examples during the rest of the quarter. For now let us look at some simple examples:

```
bool raining;
raining = true;

if (raining)
     {
     cout << "Better take an umbrella with you.\n";
     cout << "It is wet outside\n";
     }
if (!raining)
     cout << "It is a nice day.\n";
```

Nested if statements example

```
int age;
bool is_with_parent;

cout << “what is your age?”;
cin >> age;
cout << “Is your parent with you? Type Y or N”;
cin >> response;

if (response == “Y”)
    is_with_parent = true;
else
    is_with_parent = false;

if (age < 17)
   {
     if (is_with_parent)
       cout << "You can see the movie";
     else
        cout << "You can not see the movie";
   }
else
   {
   cout << "you can see the movie";
   }
```

How are numbers casted to bools?

Anything that is not 0 is true!

Only 0 gets converted to false.

Example:

```
if (17)
  cout << "yey!";

if (-1)
   cout << "yey!";

if (0)
  cout << "yey!";
```

Common mistake

```
if(a = b)  // most likely the programmer ment ==
   cout << "a equals b";
else
   cout << "a does not equal b";
```

= is assignment

== is comparison

```
int x=5;
if (x=2)
   cout << “x equals 2”;
```

Above code sets x to 2. Then it evaluates the expression to 2 which is true so the cout line gets run.

Building complex boolean expressions

When we have two simple boolean expressions we can create a more complicated expression by putting them together using the conjunction && (AND).

expression1 && expression2

This resulting expression is true iff both expression1 and expression2 are true.

`TRUE && TRUE`	`TRUE`
`TRUE && FALSE`	`FALSE`
`FALSE && TRUE`	`FALSE`
`FALSE && FALSE`	`FALSE`

Example:

(x < 5) && (x > 2)

This expression is true when x is between 5 and 2 exclusive

Building complex boolean expressions

Another way to join together two boolean expressions is with || (OR).

expression1 || expression2

This resulting expression is true if either expression1 or expression2 is true.

`TRUE || TRUE`	`TRUE`
`TRUE || FALSE`	`TRUE`
`FALSE || TRUE`	`TRUE`
`FALSE || FALSE`	`FALSE`

Example:

```
(x < 0) || (x  > 0)
```

This expression is true when x is not equal to 0.

Example

How do we check if x is between two numbers?

Can we write:

```
(-2 < x <=0)
```

This may not evaluate the way you think.

We have to write:

```
(-2 < x) && (x <=0)
```

! operator (NOT)

! is the negation or NOT operator.

!(expression) has the opposite value of the expressions value

ie.

!(true) is false

!(false) is true

Examples:

!(5 < 2) has the value true

What about?

```
int x;
cin >> x;
if (!x)
    cout << "Yey!"; // For what values of x is there output?
```

Example

```
//Given
int v1=4, v2=5, v3=0;

// Evaluate the line
!((v3=1) && (v1 <= v2) || ! (true && (v1 !=v2)))
```

Solution

int v1=4, v2=5, v3=0;

!((v3=1) && (v1 <= v2) || ! (true && (v1 !=v2)))

!((1) && (1 <= 5) || !(true && (4 !=5)))

I'll use T for true

!(T && T || !(T &&T)

! (T && T || F)

!T

F

Comparing strings

The operators >, == and < can be used to compare strings.

Strings are sorted in lexicographic ordering. This is the order used by dictionaries.

First come numbers:1 2 3 ...
Then upper case letters: A B C …
Then lower case letters: a b c …

```
string name = "Simpson"

if (name == "Simpson") // true
if (name == "simpson") // false
if (name < "Homer") // false
if (name < "homer") // true
if (name < "Simpsons") // true
if (name < "2") // false
if (name < 2) // error
```

Confusing && and ||

It is a common mistake to confuse && and ||

When we want all of the conditions to be true we use &&

When we want any of the conditions to be true we use ||

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”)
```

De Morgan's law

It is not true that some people won't be confused by this code:

```
if (!( (country == “USA” || country ==”CAN”) &&
(state !=”AK” && state !=”HI”) ) )
   shipping_charge = 20.0;
```

To understand a complicated expression like this, we want to understand how to simplify expression like

! (A || B)

and

!(A && B)

De Morgans laws cont.

!(A || B)

Consider:

!(go to movies || watch tv)

This is true when I don't go to the movies AND I don't watch tv.

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

De Morgans laws cont.

!(A && B)

Consider:

!(go to movies && watch tv)

This is true when I don't do both.

But I could go to movies OR watch tv.

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

Application of De Morgan's laws

```
if (!( (country == “USA” || country ==”CAN”) &&
(state !=”AK” && state !=”HI”) ) )
   shipping_charge = 20.0;
```

Set `A` to be `country == “USA” || country ==”CAN”`

Set `B` to be `state !=”AK” && state !=”HI”`

So we get:

```
if (!( A  &&  B) )
   shipping_charge = 20.0;
```

Application of De Morgan's laws

```
if (!( A  &&  B) )
   shipping_charge = 20.0;
```

or

```
if ( !A || !B )
```

!A is !(country == “USA” || country ==”CAN”)

or

```
country != ”USA” && country !=”CAN”
```

Application of De Morgan's laws

`!B` is `!(state !="AK" && state !="HI")`

or

```
state == "AK" || state == "HI"
```

So putting it all together

```
if ((country != "USA" && country !="CAN")   ||
(state =="AK" || state =="HI") ) )
   shipping_charge = 20.0;
```

In English: If the country is not USA and the country is not Canada or the state is Alaska or Hawaii then shipping_charge is 20.