Lecture 3 Notes

- Recall the following program from lecture 2

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

int main()
{
cout << "How many hours did you work? ";
double hoursworked;
cin >> hoursworked;

cout << "What is your hourly rate of pay? ";
double payRate;
cin >> payRate;
if (payRate < 15.00)
cout << "Ask for a raise!" << endl;

double amtEarned = hoursworked * payRate;
cout.setf(ios::fixed);
cout.precision(2);
cout << "You earned $" << amtEarned << endl;

double withholdingRate;

if (payRate >= 18.00)
withholdingRate = 0.10;

else
withholdingRate = 0.05;

cout << "$" << (withholdingRate * amtEarned) << " will be withheld." << endl;
}
```

- Instead of saying `else withholdingRate = 0.05`, we can **remove the else statement** and initialize it to be 0.05 at the top
    - Will have the same outputs

```
...

double withholdingRate = 0.05; // initialized to be 0.05

if (payRate >= 18.00)
withholdingRate = 0.10; // ignores the initial value if payRate >= 18.00

...
```

- Your code will be **easier to read** if you break it into sections with space in between
    - Sections should be separated by function
    - Similar to paragraphs
- A nicely annotated program, separated by section, should look similar to the one shown on the next page
    - The green markings `// sample text` are annotations for your own purposes

- Sample program with annotations and good organization

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

int main()
{

// Gather input data

cout << "How many hours did you work? ";
double hoursworked;
cin >> hoursworked;

cout << "What is your hourly rate of pay? ";
double payRate;
cin >> payRate;

// Compute and print earnings

double amtEarned = hoursworked * payRate;
cout.setf(ios::fixed);
cout.precision(2);
cout << "You earned $" << amtEarned << endl;

// Compute and print withholding

double withholdingRate
if (payRate >= 18.00)
withholdingRate = 0.10;
else
withholdingRate = 0.05;

cout << "$" << (withholdingRate * amtEarned) << " will be withheld." << endl;
}
```

- If you are writing a program with **certain variables that change frequently**, define them at the top of the program

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

int main()
{
/* Variables that frequently change are defined here. That way, you are able to
quickly edit it without having to go through the whole program. */

double PAYRATE_THRESHOLD = 18.00;
double HIGH_WITHHOLDING_RATE = 0.10;
double LOW_WITHHOLDING_RATE = 0.05;

...

double withholdingRate
if (payRate >= PAYRATE_THRESHOLD)
withholdingRate = HIGH_WITHHOLDING_RATE; // Sets it equal to defined variable
else
withholdingRate = LOW_WITHHOLDING_RATE; // Sets it equal to defined variable

cout << "$" << (withholdingRate * amtEarned) << " will be withheld." << endl;
}
```

- If you want an initialized variable to remain constant and **ignore** any changes, use `const` before declaring it
  - If you try to change a `const`, the program will not compile

```
...
// "const" is used so that these values are set and cannot be changed

const double PAYRATE_THRESHOLD = 18.00;
const double HIGH_WITHHOLDING_RATE = 0.10;
const double LOW_WITHHOLDING_RATE = 0.05;
...

PAYRATE_THRESHOLD = 19.00; // Wants to change the "const" defined earlier
// Program will not compile!!!!
...
```

- The following program will compile, but it will output wrong results
  - This is because the **else statement** will pair up with the **nearest** `if` **that is not paired**

```
...
string citizenship;
int age;

...
// get values for these variables
...

if (citizenship == "US")

      if (age >= 18)
            cout << "You can vote in U.S. elections" << endl;

// This else pairs up with “if (age >= 18)”
else
      cout << "You are not a U.S. citizen" << endl;
...
```

- The solution is to use **curly braces** to isolate the if-statement in between

```
...
string citizenship;
int age;

...
// get values for these variables
...

if (citizenship == "US")
{
if (age >= 18)
cout << "You can vote in U.S. elections" << endl;
}
/* This else pairs up with "if (citizenship == "US")" since the if-statement
above is protected by curly braces */
else
cout << "You are not a U.S. citizen" << endl;
...
```

- You can also use if-statements with complex conditions using "or" (`||`) or "and" (`&&`)

```
if (citizenship == "US" || citizenship == "Canada")
cout << "The country code is 1" << endl;
```

```
if (citizenship == "US" && age >= 18)
cout << "You are eligible to vote in U.S. elections" << endl;
```

```
if (roll == 2 || roll == 3 || roll == 12)
cout << "You lose!" << endl;
```

- `&&` has higher precedence than `||`
- The following statements are equivalent, and they say:
    - If your major is CS, you are eligible for something
    - If your major is MATH, you also need at least a 3.2 gpa to be eligible

```
if (major == "CS" || major == "MATH" && gpa >= 3.2)

if (major == "CS" || (major == "MATH" && gpa >= 3.2))
```

- The following statements are not equivalent to the previous ones; they say:
    - If your major is CS and have a 3.2 or higher gpa, you are eligible for something
    - If your major is CS and have a 3.2 or higher gpa, you are also eligible

```
if ((major == "CS" || major == "MATH") && gpa >= 3.2)
```

- The following program is a **common mistake**
  - The lines in bold are the outputs

```
...
int n = 17;
cout << "n is " << n << endl;

if (n = 0)
cout << "n is zero" << endl;

else
cout << "n is not zero; n is " << n << endl;

n is not zero; n is 0
...
```

- Here, `n` is reassigned to be `2`
  - It will fall under the `else` case since the if-statement is basically nonexistent

- Remember that:
  - `=` is an assignment statement
  - `==` is an equal sign in an if-statement

- When writing complex if-statements, write down the conditions completely

```
if (citizenship == "US" || citizenship = "Canada") // Valid statement

if (citizenship == "US" || == "Canada") // Error! Won't compile.

if (citizenship == "US" || "Canada") // Won't do what you want
```

- More examples below:

```
if (rating < 1 || rating > 10)
cout << "Rating must be from 1 to 10" << endl; // Valid statement

if (rating < 1 || > 10) // Won't compile

if (rating >= 1 && rating <= 10)
cout << "Rating is OK" << endl; // Valid statement

if (1 <= rating <= 10) // Error!

if (a/b + c/d < 10) // Works unless b or d is equal to 0

if (b != 0 && d != 0 && a/b + c/d < 10) // Valid since equal precedence is read
from left to right
```

- In the last if-statement in the code above, if b is equal to 0, then the condition `a/b + c/d < 10` will not be evaluated

- If a statement is false, all proceeding conditions will not be evaluated

```
if (a/b + c/d < 10 && b != 0 && d != 0)
```

- The program above will crash since `a/b + c/d < 10` is evaluated before subsequent conditions

In summary,

`A && B`	Evaluate A.	If A is true, evaluate B, result is A and B. If A is false, the result is false, and B is not even evaluated.
`A || B`	Evaluate A.	If A is true, the result is true, and B is not even evaluated. If A is false, evaluate B, result is A or B.

- Sometimes, we want the shorter lines of commands to be on top to make it easier to read
- Let's change the following program:

```
if (citizenship == "US" && age >= 18)
{
...
...
...
}

else
cout << "You cannot vote in U.S. elections" << endl;
```

- You cannot just use the opposite operators!

```
if (citizenship != "US" && age < 18) // Not what we want!!!
cout << "You cannot vote in U.S. elections" << endl;

else
{
...
...
...
}
```

- Pay attention to the && and ||'s
- The following program switches the && to an || and is logically equivalent to the program at the top of this page

```
if (citizenship != "US" || age < 18)
cout << "You cannot vote in U.S. elections" << endl; // Correct!

else
{
...
...
...
}
```

- The **De Morgan Laws** state that when you take the opposite, you need to switch the `&&` and `||`'s

not (A AND B)	… turns into ...	(not A) or (not B)
not (A OR B)	… turns into ...	(not A) AND (not B)

- We also need to consider the inclusivities of greater-than and less-than signs

not (a <= b)	… turns into ...	a > b
not (a < b)	… turns into ...	a >= b
not (a >= b)	… turns into ...	a < b
not (a > b)	… turns into ...	a <= b
not (a == b)	… turns into ...	a != b
not (a != b)	… turns into ...	a == b

- Let's write a program that categorizes income into the following categories:

Low:	< 30,000	
Medium:	>= 30,000	and < 100,000
High:	>= 100,000	and < 500,000
Very high:	>= 500,000	

```
if (income < 30000)
cout << "Low";
else
{
if (income >= 30000 && income < 100000)
cout << "Middle";
else
{
if (income >= 100000 && income < 500000)
cout << "High";
else
{
if (income >= 500000)
cout << "Very high";
}}}
```

- If your income is low, for instance, the lower portions of the program don't even matter

- This structure is called an **if-ladder**

- You can take out the if-statement at the bottom
    - If the program runs the if-statement at the bottom, your income must be greater than 500,000

```
if (income < 30000)
cout << "Low";
else
{
if (income >= 30000 && income < 100000)
cout << "Middle";
else
{
if (income >= 100000 && income < 500000)
cout << "High";
else
cout << "Very high";
}}
```