CSCI 123 Introduction to Programming Concepts in C++

Brad Rippe

Even More Flow
Variables

• int age;

• This is a **declaration**. Makes the name and the type of variable known to the program.

• int age = 22;

• This is a **definition or initialization**. It is also a declaration, so it defines the variable type and name, in addition to initializing it to a value.
Multiple Assignment and Combined Assignment

• $=\,$ is an operator that can be used $> 1$ time in an expression:

$$x = y = z = 5;$$

• Value of $=\,$ is the value that is assigned

• Associates right to left:

$$x = (y = (z = 5));$$

- Value is 5
- Value is 5
- Value is 5
Constant Variables

• variable whose content cannot be changed during program execution
• Used for representing constant values with descriptive names:
  
  ```
  const double TAX_RATE = 0.0675;
  const int NUM_STATES = 50;
  ```
• Often named in uppercase letters

• Unmodifiable for the life of the program.
Variable Types

• What’s the difference here?

‘C’

“C”

• Is there a difference?
variables

• Char is 8 bits
• May contain one of the values from the *ascii* character codes

• String depends of the number of characters
• Size is determined when we define the variable

99 \0

Null Terminator
A few questions?

```cpp
int x = 22;
// what's happening here
if(3 < x < 7) {
    cout << (3 < x < 7) << endl;
    cout << "Executing if statements\n";
}
cout << "outside of if statements\n";
```
Logical Operators

```cpp
if( temp > 68 && temp < 72 || temp > 100) {
    cout << "Temperature " << temp;
}
```

- `&&` has a higher precedence than the `||` operator

- equivalent to:
  ```cpp
  if(((temp > 68 && temp < 72) || temp > 100)
  ```
Negation Operator

- ! – the not operator
- Returns the inverse of the condition or expression

```c
int temp = 72;  // Evaluates to true
if (!temp)
    bool registered = false;  // Converts the value to false
```

```
if (!registered)  // Evaluates to false
    // Converts the value to true
```
What happens?

```c
int iVal = 0;
iVal = 3.541 + 3;
```

- Your compiler should complain by giving you a warning
- Loss of data due to the fact that an int uses 4 bytes and doubles use 8 bytes.
What is output?

```cpp
int c = 5;
cout << c << endl;
cout << c++ << endl;
cout << c << endl;

c = 5;
cout << c << endl;
cout << ++c << endl;
cout << c << endl;
```
What is output here?

```cpp
for(int p = 0; p < 10; p++) {
    if(p < 5) {
        continue;
    }
    if(p > 5)
        break;
    cout << "This is p " << p << endl;
}
```
For loop

- **Initialization statement** – initializes a counter variable (*counter-controlled loop*). Executed once when the loop begins
- **Continuation condition** – determines if the body of the loop will be executed. As long as the condition remains true the body of the loop is executed
- **Increment/Decrement statement.** The counter is modified. The counter can be incremented or decrement (this doesn’t have to be by 1)
Omitting parts

for(;;) {
    cout << “catch me if you can!!!\n”;}

• Initialization can be omitted if initialization occurs else where
• The condition is omitted, it’s the equivalent of typing true. If this is the case then the loop must have a break or a return statement to terminate the loop
• The expression for the counter can be omitted if the body handles changing the value of the counter or the loop must be terminated with a break or return statement
What happens?

```cpp
for(int j = 0; j < 5; j++) {
    cout << "outer: j is " << j << endl;
    for(int i = 4; i > 0; i--) {
        cout << "\tinner: i is " << i << endl;
    }
}
```
Nested Loops - Notes

• **Inner loop** goes through all repetitions for each repetition of outer loop

• Inner loop repetitions complete sooner than outer loop

• Total number of repetitions for inner loop is product of number of repetitions of the two loops. In previous example, inner loop repeats 20 times
How do we know?

• Step through the code on a piece of paper

• Type it into your editor and step through the code in the debugger
Preprocessor

• `#include` directive  
  - Inserts the contents of the file specified `< >`  
  - Runs before your code gets to the compiler  
  - Searches in the directory of the file then the include path (`%PATH%` or `$PATH` ENV VAR – type “echo $PATH” on linux)

• More after we cover header files