CS 180 - More C++

Announcements

- HW 1 due next Wednesday
- Look for program 1 next week
- Office hours:
  Monday 1:30 - 3:30
  Thurs 11-12
The Command Line

In general, when using the terminal, you will need about 5-6 commands:

- `ls` - list
- `cp sourcefile targetfile`
- `mkdir name`
- `rm dir name`
- `rm filename`
- `cd directoryname`
- `mv sourcefile targetfile`

(plus `g++`, `kate`, `nano`, `vi`, etc...)

SSH username@tung.slu.edu
also nx client
Useful tricks

- Hitting the up arrow gives you the last thing you typed (hitting it again goes to 2nd to last, etc.)
  
- You can then edit the command

- Hitting tab is quick complete

- Kate editors have a built in terminal

- You can use an \& to get you prompt back:  kate myfile &

- .. is this directory, .. is parent

  Ex:  cd ..
  cp ../*.file
Arrays

Python has lists, tuples, etc.
C++ only has arrays.
- Size is fixed
- Type is fixed (not homogenous)

Ex: int numbers[10];
numbers[0] = 56;
numbers[9] = 11;

Numbers[10] = 5;  BAD
Creating variables (cont.)

**Allowed:**

```c
```

**Error:**

```c
int daysInMonth[];
```

**Allowed:**

```c
char greeting[] = "Hello";
```
Operators

Basic numeric operators differ slightly:

```plaintext
>>> a = 5;
>>> myarray [a++] = 11;
>>> myarray [++a] = 11;
>>> mt a = 5;
>>> int b = 6;
>>> float a = float(a) / float(b);
>>> a++; is equivalent
>>> a = a + 1;
>>> a++ = 1;
```
Boolean operators and comparators - VERY different

<table>
<thead>
<tr>
<th>Python</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>and</code></td>
<td><code>&amp;&amp;</code></td>
</tr>
<tr>
<td><code>or</code></td>
<td>`</td>
</tr>
<tr>
<td><code>not</code></td>
<td><code>!</code></td>
</tr>
<tr>
<td><code>a if b else c</code></td>
<td><code>b ? a : c</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boolean Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a &lt; b</code></td>
</tr>
<tr>
<td><code>a &lt;= b</code></td>
</tr>
<tr>
<td><code>a &gt; b</code></td>
</tr>
<tr>
<td><code>a &gt;= b</code></td>
</tr>
<tr>
<td><code>a == b</code></td>
</tr>
<tr>
<td><code>a &lt; b &lt; c</code></td>
</tr>
</tbody>
</table>

true = any thing else
false = 0

bool val = (2==3));
val = 2 + 5;
Control Structures

Last time:
  - while loops
  - functions

while ( )

3
Also have do-while:

```cpp
int number;
do {
cout << "Enter a number from 1 to 10: ";
cin >> number;
} while (number < 1 || number > 10);
```

This is a bit different:

- body of loop is executed once, before repeated condition is checked.
Conditionals

if (bool)
  body 1;
else if
  body 2;
else

Ex: if \( x < 0 \), \( x = -x \).

Note: - don't need brackets if only one line in body
- don't need else
- no else if in C++ — write out else if
Boolean conditionals in if and while statements

If statements can also be written with numeric conditions instead of booleans:

Ex: if (mistakeCount) {
    cout << "There were " << mistakeCount
    << " problems" << endl;
}

mistake count == 0 is false
any other number is true.
Common mistake - what is wrong?

double gpa;
cout << "Enter your gpa: ";
cin >> gpa;
if (gpa == 4.0)
  cout << "Wow!" << endl;

In Python, you'd get an error.

C++ - no error

now gpa is 4.0

gpa is inside boolean
For loops

Example:

```
for (int count = 10; count > 0; count --)
    cout << count << endl;
    cout << "Blastoff!" << endl;
```

Note: int declaration isn't required.

Alternate:

```
int count;
for (count = 10; count > 0; count --)
    cout << count << endl;
```
The main function

Every program defaults to running a special "main" function first.
(In python we just started typing code.)

```c
int main ()
{
    body;
}
```
Input + Output

C++ has several predefined useful classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>Purpose</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>istream</td>
<td>Parent class for all input streams</td>
<td>&lt;iostream&gt;</td>
</tr>
<tr>
<td>ostream</td>
<td>Parent class for all output streams</td>
<td>&lt;iostream&gt;</td>
</tr>
<tr>
<td>iostream</td>
<td>Parent class for streams that can process input and output</td>
<td>&lt;iostream&gt;</td>
</tr>
<tr>
<td>ifstream</td>
<td>Input file stream</td>
<td>&lt;fstream&gt;</td>
</tr>
<tr>
<td>ofstream</td>
<td>Output file stream</td>
<td>&lt;fstream&gt;</td>
</tr>
<tr>
<td>fstream</td>
<td>Input/output file stream</td>
<td>&lt;fstream&gt;</td>
</tr>
<tr>
<td>iostream</td>
<td>String stream for input</td>
<td>&lt;sstream&gt;</td>
</tr>
<tr>
<td>ostringstream</td>
<td>String stream for output</td>
<td>&lt;sstream&gt;</td>
</tr>
<tr>
<td>stringstream</td>
<td>String stream for input and output</td>
<td>&lt;sstream&gt;</td>
</tr>
</tbody>
</table>

(We'll use iostream & fstream the most.)
Using cout + cin

#include <iostream>
using namespace std;  

Notes:
- get cout + cin
- separate distinct variables by `>>` or `<<`
- use endl instead of "\n"
Examples

Python

1. `print "Hello"
2. print
3. print "Hello,", first
4. print first, last  # automatic space
5. print total
6. print str(total) + "."  # no space
7. print "Wait...",  # space; no newline
8. print "Done"

C++

1. `cout << "Hello" << endl;
2. cout << endl;
3. cout << "Hello, " << first << endl;
4. cout << first << " " << last << endl;
5. cout << total << endl;
6. cout << total << "." << endl;
7. cout << "Wait... ";  // no newline
8. cout << "Done" << endl;

Figure 7: Demonstration of console output in Python and C++. We assume that variables first and last have previously been defined as strings, and that total is an integer.