CS180 - More C++

Note Title 1/19/2011

Announcements

- HW 1 due next Wednesday
- Look for program 1 next week
- Office hours:

  Monday 1:30 - 3:30

  Thurs 11 - 12

ssh username@turing.slu.edu
also nx client

The Command Line

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

- ls — list
- cp sourcefile targetfile
- mkdir name
- rmdir name — rm -R name
- rm filename
- cd directoryname
- mv sourcefile targetfile
  - cp
  - rm

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

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 autocomplete
- 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

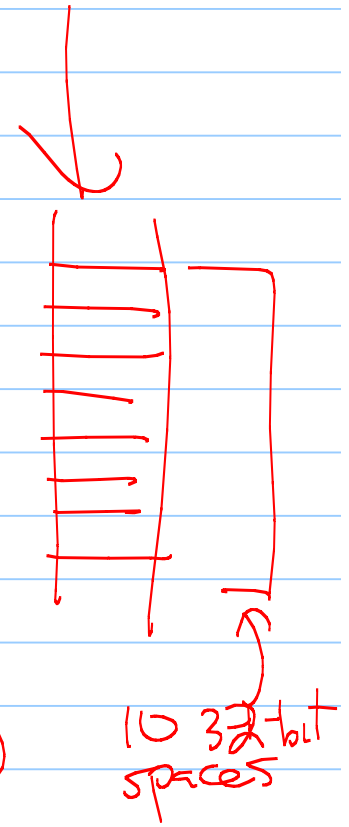
Python has lists, tuples, etc.

C++ only has arrays.

- size is fixed
- type is fixed (& homogenous)

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

numbers[10] = 5;
```



BAD

Creating variables (cont.)

Allowed:

```
int daysInMonth[] = {31, 28, 31, 30, 31, 30,
                     31, 31, 30, 31, 30, 31};
```

Error:

```
int daysInMonth[];
```

no size

Allowed:

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

Operators

Basic numeric operators differ slightly:

Arithmetic Operators		
Python	C++	Description
−a	−a	(unary) negation
a + b	a + b	addition
a − b	a − b	subtraction
a * b	a * b	multiplication
▷ a ** b		exponentiation
a / b	a / b	standard division (depends on type)
▷ a // b		integer division
a % b	a % b	modulus (remainder)
▷	++a	pre-increment operator
▷	a++	post-increment operator
▷	−−a	pre-decrement operator
▷	a−−	post-decrement operator

```
a=5;
myarray[a++]=11;

myarray[++a]=11;

int a=5;
int b=6;
float c;
c=float(a)/float(b);
```

a++; is equivalent to a=a+1;
a+=1;

Boolean operators + comparators - VERY different

Python C++

	Boolean Operators		
▷	**and**	&&	logical and
▷	**or**	||	logical or
▷	**not**	!	logical negation
▷	a **if** b **else** c	b ? a : c	conditional expression

	Comparison Operators		
	a < b	a < b	less than
	a <= b	a <= b	less than or equal to
	a > b	a > b	greater than
	a >= b	a >= b	greater than or equal to
	a == b	a == b	equal
▷	a < b < c	a < b && b < c	chained comparison

true = any thing else

false = 0

```
bool val = (2==3);
val = 2+5;
```

Control Structures

Last time:

- while loops

- functions

while ( ) {

}

Also have do-while:

```
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)
{
   body1;
}
else if
{
   body2;
}
else
```

Ex:
```
if (x < 0)
   x = -x;
```

Note:
- don't need brackets if only one line in body
- don't need else
- no elif in C++ - write out else if

Boolean conditionals in if & while statements

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

Ex
```
if (mistakeCount)
    cout << "There were " << mistakeCount
         << " problems" << endl;
```

mistakeCount == 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;
```

initialization of loop control variable

test condition

updates loop control variable

no semicolon

```
for (; count > 0;
     count--)
```

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.)

```
int main() {
  body;
}
```

Input + Output

C++ has several predefined, useful classes.

Class	Purpose	Library
istream	Parent class for all input streams	<iostream>
ostream	Parent class for all output streams	<iostream>
iostream	Parent class for streams that can process input and output	<iostream>
ifstream	Input file stream	<fstream>
ofstream	Output file stream	<fstream>
fstream	Input/output file stream	<fstream>
istringstream	String stream for input	<sstream>
ostringstream	String stream for output	<sstream>
stringstream	String stream for input and output	<sstream>

(We'll use iostream & fstream the most.)

Using cout & cin

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

loads the library

without this line,

```
std::cin >> value;
std::cout << variable;
```

Notes:
- gets cout & cin
- Separate distinct variables by >> or <<
  - >> cin
  - << cout
- 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.