Data Structures

Control structures (cont)

I/O
Course announcements

- Lab due today
- HW - due Tuesday (git instructions are coming)
- Likely in this afternoon
Last time

- loops
- conditionals
Command Line Tips

In general, 5 or 6 commands will go far!

- `ls`
- `cp sourcefile destfile`
- `mkdir name`
- `rmdir name` must be empty
- `cd directory` variants: `cd ~` to get back home
- `mv sourcefile destfile`
- `rm file`

Careful!

Don't ask if you are sure!
Others
- emacs, vi or nano
- `g++`
- `make` (later)
- `man command`

Also:
- CS page has info on connecting (Dennis and I can also help!)
- Many, many resources online
A few tricks

- Hit up arrow gives last command, which you can then edit
- Tab will auto complete file names
- On lab or none machine & gives prompt back
  ie > kate myfile &

- . is current directory
- .. is parent (up one level)
- ~/ is home
- / is root

Ex: > cd ..
    > ./a.out
    > cp .. /file ..
Conditional

```cpp
if (bool) {
    body 1;
} else {
    body 2;
}
```

Ex:

```cpp
if (x < 0)
    x = -x;
```

```cpp
if (groceries.length( ) > 15)
    cout << "Go to the grocery store" << endl;
else if (groceries.contains("milk"))
    cout << "Go to the convenience store" << endl;
```

```cpp
if ( )
    cout
else if ( )
```
These can get a bit ugly!

```c
if (cond 1)
  if (cond 2)
    code;
  else
    code;
else
  code;

if (cond 3)
  if (cond 4)
    if (cond 5)
      code;
    else
      code;
  else
    code;
```

If these conditions become complex, they can lead to tangled code. Considering the possibilities, it's crucial to ensure readability.
Booleans & while/conditionals:

- If & while can both be written with numeric values as the boolean

Reason: bools are really just integers!

Ex: `if (mistakeCount) cout << "error!" << endl;`

0 <=> false
all else is true
An error that crops up with conditionals/booleans:

"Feature" 1: bools are really ints, + 0 is false only

"Feature" 2: operator = chains

\[ x = y = 5; \]

So — a common bug is:

```cpp
double gpa;
cout << "Enter your gpa: ";
cin >> gpa;  // = = for T/F
if (gpa <= 4.0)
    cout << "Wow!" << endl;
```
Do-While loops

- A variant of whiles that executes body before checking condition

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

Every program starts running at its main function.

Syntax

```c
int main ()
{
    // body
    int x = myAdd(10, 15);
    return 0;
}
```

Other functions:

```c
int myAdd (int x, int y)
{
    return (x+y);
}
```

```c
// code here won't run
```
Arrays
- Python has lists, tuples, etc.
- C++ starts with only arrays
  - size is fixed at time of declaration
  - type is fixed (homogeneous)

Ex: int numbers[5];
variable list

numbers[0] = 55;
n numbers[3] = 20;

cout << "number[0] = " endl;

junk data
Caution:
- Seg faults will be a problem!

Ex:
```c
int numbers[5];
numbers[0] = 65;
numbers[3] = 20;
numbers[5] = 5;
```

![Diagram of array indexing]

Address

Numbers

55

763

Put 5 here
Creeting arrays

int days In Month = 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31

days In Month [2]

Error:

int days In month [ ] ;

One exception:

char greeting [ ] = "Hello" ;

Reason:

Strings are char arrays
Multidimensional arrays.

```
int table[8][10];
```

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
table[i][j] = 6
```

```
i = 4, j = 6
```

```
slot for 1st row
```

```
2nd row
```
C++ has classes to handle I/O:

<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>istream</td>
<td>String stream for input</td>
<td>&lt;sstream&gt;</td>
</tr>
<tr>
<td>ostream</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>

Figure 6: Various input and output stream classes.
Most common: I/O stream

---

**Notes:**

- #include `<iostream>
using namespace std;

- get cin, cout

  ```
  char letter;
  cin >> letter;
  ```