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

editors

- emacs, vi or nano
- g++
- make (later)
- man command

manual

> man ls

> ls -lart

Also:

- CS page has info on connecting (Dennis + 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 nomachine, & 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

```
if (bool) {
    body1;
} else {
    body 2;
}
```

Ex:

```
if (x < 0)
   x = −x;
```

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

```
if (      )
    cout
else if (      )
```

These can get a bit ugly!

```
if (cond 1)
if (cond 2)
   {code;}
else
   {code;}
if (cond 3)
if (cond 4)
if (cond 5)
   {code;}
else
   {code;}
```

```
if (cond1)
  if (cond2)
    . . .
  else
    . . .
if (cond3)
  if (cond4)
    if (cond5)
      code
    else
      code
```

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

Booleans & whiles/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 w/ conditionals / booleans:

"Feature" 1: bools are really ints, & 0 is false (only)

"Feature" 2: operator = chains

```
x = y = 5;
```

So - a common bug:

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

(= for T/F)

Do-While loops

- A variant of whiles that executes body before checking condition

```
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

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

(int: word; main: function name; (): no inputs; return 0;: optional)

Other functions:

```
int myAdd( int x, int y) {
    return (x+y);
    // code here won't run
}
```

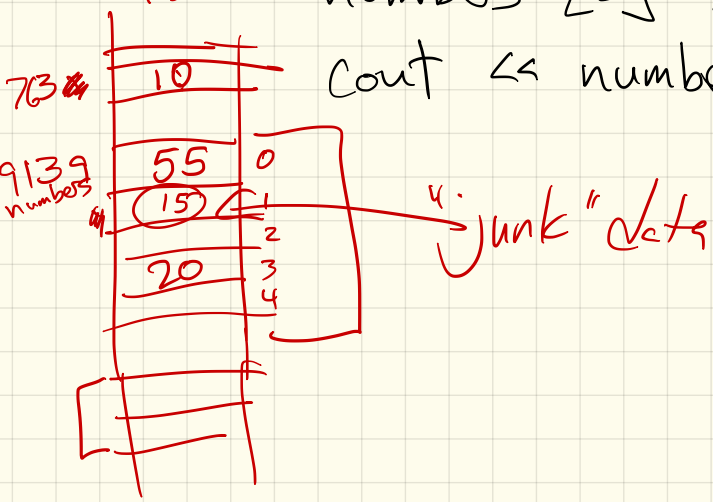
Arrays

- 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 is → `numbers[0] = 55;`
`numbers[3] = 20;`

`cout << number[0] << endl;`

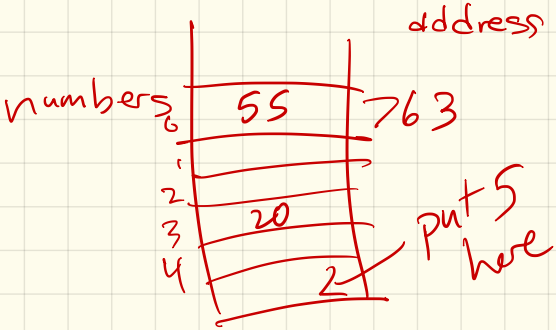


Caution:

- Seg faults will be a problem!

Ex:

```
int numbers[5];
numbers[0] = 55;
numbers[3] = 20;
numbers[5] = 5;
```



Creating arrays

no brackets

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

```
daysInMonth[2]
```

Error:

no fixed size

```
int daysInMonth[];
```

One exception:

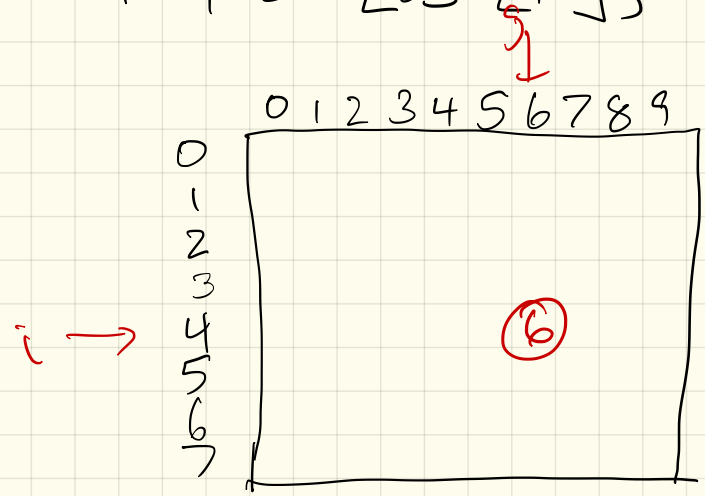
```
char greeting[] = "Hello";
```

Reason:

Strings are char arrays

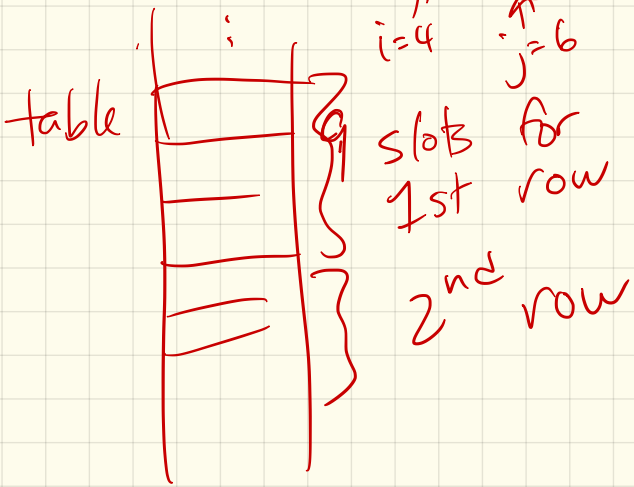
Multi-d arrays.

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



table[i][j] = 6

i=4 j=6



I/O:

C++ has classes to handle I/O:

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>

Figure 6: Various input and output stream classes.

Most common : I/O stream

Python

```
1  print "Hello"
2  print                          # blank line
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;                  // blank line
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.

Notes:

- #include <iostream>
  using namespace std;
- get cin, cout
  >> <<

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