Class 25

Files

Class Methods

- Class type has methods, which are special functions
- Call method:
  - VarName.MethodName(Arguments)

Input from/Output to files

- This is performed using streams
- A stream on a computer performs input/output operations
- It can be viewed as either a destination or a source of indefinitely long characters
- C++ comes with a library called fstream that includes methods for dealing with files

Input from/Output to files

- Class types:
  - ifstream – used to read information from files
  - ofstream – used to create files and to write information to files
- #include<fstream>
- Class methods that apply to both ifstream and ofstream:
  - .open(fileName) – connects a variable to a file
  - .is_open() – checks to see if the stream is connected to the file
  - .close() – closes a file
- ifstream also has .eof() and .get()

Example

```
ofstream f;              // f is used to access our output file
f.open(“out.txt”); // connects f to file out.txt
f << “Writing this to a file” << endl; // inserts content into the file
f.close();               // properly closes the connection to the file
```

Example 1

```
#include<iostream>
#include<fstream>
using namespace std;
int main(){
  ofstream f;
  f.open("out.txt");
  if(!f.is_open()){ // checks to see if f is connected to the file
    cout << "No such file exists." << endl;
    return 0;
  }
  f << "Hello" << endl; // insertion operator
  f << "World" << endl;
  f.close();
  return 0;
}
```

ifstream

- Considerations when reading files:

- Have you reached the end of the file (is there no more data left to read)?
  - Answer this question with .eof()

- What if you would like to read the input character by character?
  - Use .get() to obtain the next character in the file
  - This also reads white space, such as spaces, new lines, etc.
  - Used to obtain very detailed input

Example

```
ifstream f;              // f is used to extract from our input file
f.open(“out.txt”); // connects f to out.txt
string s;
f >> s; // Extracts first string in file connected to f and stores in s
f.close();               // properly closes the connection to the file
```

Example 2

```
#include<iostream>
#include<fstream>
using namespace std;
   int main(){
   ifstream f;
   f.open ("out.txt");
   string x, y;
   f >> x; // extraction operator
   cout << "The first string in your file is " << x << endl;
   f >> y;
   cout << "The next string in your file is " << y << endl;
   f.close();
   return 0;
}
```

Example

```
ifstream f;
f.open(“out.txt”);
while(!f.eof()){     // while you have not yet reached the end of the file
   char x = f.get(); // get next character
   cout << x;        // print character to monitor
}
f.close();
```

The above goes through file f character by character and prints whatever it sees to the monitor

Example 3

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

int main(){
  ifstream f;
  f.open ("out.txt");
  while(!f.eof()){     // while you have not yet reached the end of the file
     char x = f.get(); // get next character
      cout << x;        // print character to monitor
  }
  f.close();
  return 0;
}
```

Arguments to main

- Alternative title line to main
- We can set up a main program to work with arguments
- These are called command-line arguments

mars> ./a.out file1 file2

Main title line for command-line arguments

- Old title line:
  - int main()
- New title line:
  - int main(int argc, char *argv[])
  - This version of the main title line allows main to take command-line arguments
  - argc is the number of inputs
  - argv is an array storing the inputs
    - argv is an array of c-strings

Example 4

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

int main(int argc, char *argv[])){ // Command line arguments are stored in an array called argv

  for(int i = 0; i < argc; i++){

     cout << argv[i] << endl;

  }

  return 0;

}
```

argc, argv

mars>./a.out file1 file2

int main(int argc, char *argv[])

argc: how many things did the user type?

argv: what did the user type?

Example 5

```
int main(int argc, char *argv[]){
   ifstream f1;
   ofstream f2;
   f1.open(argv[1]); // connect f1 to command-line argument
   f2.open(argv[2]); // connect f2 to command-line argument
   while(!f1.eof()){
      char x = f1.get(); // go through f1 char by char
      f2 << x; // and copy each char to f2
   }
   return 0;
}
```