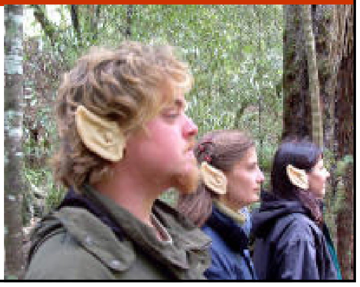
Lecture 20: File I/O

PIC 10A

Todd Wittman



Sec 9.1: File Streams

- Recall cin/cout are the I/O streams for the console window.
- We can create our own I/O streams to read/write data different places.
- In particular, we can create streams to read from and write to text files using the <fstream> library.
- Declaring a stream is a lot like creating a variable. I like to call my file I/O streams fin & fout. *(Other names work too!)*

```
    ifstream fin;      // A file input stream.
    ofstream fout;   // A file output stream.
```

- We can then use this stream to open a file and perform reading and writing procedures just like cin/cout.

```
int x;      fin >> x;      //Read x from the file associated with fin.
fout << x;    //Write x to fout's file, could be a different file.
```

A Basic Example

- Write hello to the text file “out.txt”.

```
#include <fstream>          //Need the file I/O library.
using namespace std;

int main() {
    ofstream fout;             //Create an output stream.
    fout.open(“out.txt");      //Open the file out.txt for writing.
    fout << "Hello Middle Earth!";
    fout.close();              //Close the file when we’re done.
    return 0;
}
```

Opening Files

- Suppose we want to read a list of integers in the text file “list.txt”.

list.txt

```
42        15          -32         12 88
96              -12               2
33 44           -88 -22 37             2              1 0 1 2 3
```

- The numbers could be separated by spaces or line breaks.
- First we need to create the input stream and open the file for reading with this stream.

```
ifstream fin;
fin.open(“list.txt”);
```

- This associates the file “list.txt” with fin.

Opening Files

- We could also have opened a file name given by the user.

```
string file_name;
cin >> file_name;
ifstream fin;
fin.open( file_name.c_str( ) );
```

- We add c_str( ) to convert the string to a char array.
- We need this because open( ) is an older function that doesn't recognize strings.

Reading Files

- To read the first #: `int x; fin >> x; // Now x=42.`
- We can read the whole file with a loop and place the numbers in an array or vector.
- If we don't know how many numbers there are ahead of time, probably best to use a vector.
- Recall (cin>>x) is actually a boolean. Returns false if there was no integer x to read. Our ifstream fin works likewise.

```
vector<int> list;     // Creates empty int vector.
int x;
while (fin >> x)
        list.push_back(x);     // Adds x to the end.
```

- When you're done reading, remember to close the file.

```
fin.close( );
```

Putting It All Together

- Ex Read the integers in a file given by the user and then output the average to the console.

```
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
using namespace std;
string file_name;
cout << "Enter file name to read: ";
cin >> file_name;
ifstream fin;
fin.open(file_name.c_str( ));
vector<int> list;
int x;
while (fin >> x)
    list.push_back(x);
fin.close( );
int total=0;
for (int i=0; i < list.size(); i++)
    total += list[i];
double average = (double) total /
    list.size();
cout << "The average is "
    << average << "\n";
```

•Did we really need to use a vector in this example?

Reading File to Parallel Vectors

- Recall we talked about storing products' name, price, and score in 3 parallel vectors.
- Let's read the data in the file "games.txt" to 3 parallel vectors name, price, and score.
- Then we'll output each product's bang-for-the-buck ratio.

games.txt

```
Wii  249.99 8
PC 1299  5
Mac  800   6
PS3  600  3
XBox360   499.49  7
Atari2600   2.99   10
```

	Name	Price	Score
0	Wii	249.99	8
1	PC	1299	5
2	Mac	800	6
3	PS3	600	3
4	XBox360	499.49	7
5	Atari2600	2.99	10

Reading File to Parallel Vectors

```
ifstream fin;
fin.open("games.txt");
vector<string> name;
vector<double> price;
vector<int> score;
string s;   double d;   int i;
while (fin >> s >> d >> i) {
     name.push_back(s);
     price.push_back(d);
     score.push_back(i);
}
fin.close( );
for (int j = 0; j < name.size( ); j++)
     cout << name[j] << "\'s ratio is " << price[j] / score[j] << "\n";
```

Stops reading when it doesn't find a string-double-int combo to read or runs into the end of the file.

	Name	Price	Score
0	Wii	249.99	8
1	PC	1299	5
2	Mac	800	6
3	PS3	600	3
4	XBox360	499.49	7
5	Atari2600	2.99	10

Writing to a File

- Writing to a file is very similar, except we create an output stream and the arrows << go the other way.
- Let's write out the first 100 integers to the file "first100.txt".

```
ofstream fout;
fout.open("first100.txt");
for (int i = 1; i <= 100; i++)
        fout << i << "\n";
fout.close( );
```

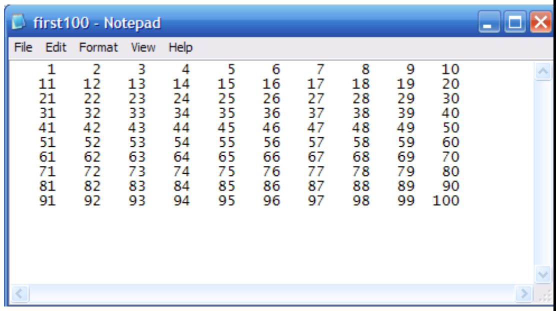
- If the file "first100.txt" did not exist, it will be created.
- If the file did exist, we will overwrite what was there previously. All existing data will be lost.
- If you want to <u>append</u> the data rather than overwrite, use

```
fout.open ("first100.txt", ios::app);
```

Formatting Output

- Recall that we can line up our output in columns using the setw function in <iomanip>. What does the code below do?

```
#include <fstream>
#include <iomanip>
using namespace std;
int main ( ) {
    ofstream fout;
    fout.open("first100.txt");
    for (int i = 1; i <= 100; i++) {
        fout << setw(5) << i;
        if (i%10 == 0)
            fout << "\n";
    }
    fout.close( );
    return 0;
}
```



```
    1    2    3    4    5    6    7    8    9   10
   11   12   13   14   15   16   17   18   19   20
   21   22   23   24   25   26   27   28   29   30
   31   32   33   34   35   36   37   38   39   40
   41   42   43   44   45   46   47   48   49   50
   51   52   53   54   55   56   57   58   59   60
   61   62   63   64   65   66   67   68   69   70
   71   72   73   74   75   76   77   78   79   80
   81   82   83   84   85   86   87   88   89   90
   91   92   93   94   95   96   97   98   99  100
```

We can also control # decimal places on doubles with setprecision(int).

<u>Example</u>: Pig Latin

- In Pig Latin, if the first letter of a word is a consonant it is moved to the end and the suffix "-ay" is added.

computer programming ➡ omputercay rogrammingpay

- If the first letter is a vowel (aeiou), we just add "-ay" to the end.

arthur also under ➡ arthuray alsoay underay

- Let's write a program that reads a text file given by the user and converts it to Pig Latin. The output is written to a file also supplied by the user.
- For this example, suppose the file contains all words, all lower-case, no punctuation. The output is all one line.
- *(We'll handle the harder cases later.)*
- oday eway eallyray eednay otay useay ectorsvay?

Example: Pig Latin

- The following function takes a string and returns the Pig Latin equivalent of that word.

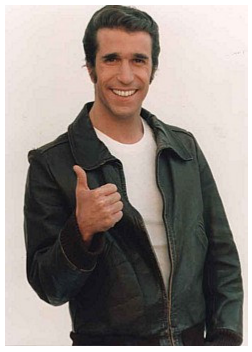
translate2PigLatin ("hello") → returns "ellohay"

translate2PigLatin ("awesome") → returns "aweseomeay"

```
string translate2PigLatin (string word) {
    if (word[0]=='a' || word[0]=='e' || word[0]=='i'
                        || word[0]=='o' || word[0]=='u')
        return word + "ay";
    else
        return word.substr(1,word.length()-1) + word[0] + "ay";
}
```

Example: Pig Latin

```
string in_file, out_file;
cout << "Enter the file to read: ";
cin >> in_file;
cout << "Enter the destination file: ";
cin >> out_file;
ifstream fin;
fin.open(in_file.c_str( ));
ofstream fout;
fout.open(out_file.c_str( ));
string word;
while (fin >> word) {
    word = translate2PigLatin(word);
    fout << word << " ";
}
fin.close( );
fout.close( );
```

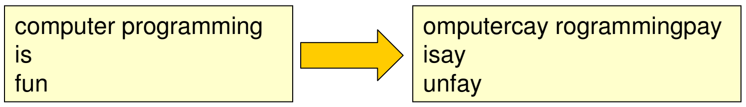


The get Function

- We could also read the file one char at a time using ifstream member function get(char).

```
char c;
fin.get(c);
```

- This places the character in fin's current position into the char variable c.
- If you don't like what you saw, you could back up one character with `fin.unget( );`
- This is useful for detecting line breaks: `if (c=='\n')`
- Suppose we want our Pig Latin translator to output line breaks in the same place as the input file.



```
computer programming
is
fun
```

```
omputercay rogrammingpay
isay
unfay
```

Detecting Line Breaks

- To put line breaks in the corresponding position, we just add a few lines to our Pig Latin example. *(Assumes line break occurs immediately after the word.)*

```
string word;
char c;
while (fin >> word) {
    word = translate2PigLatin(word);
    fout << word << " ";
    fin.get(c);
    if (c=='\n')
        fout << "\n";
}
```

Detecting Capitals and Punctuation

- What if we want to do the harder case with punctuation and capital letters?

  "Hello!" → "Ellohay!"
- First we want to detect punctuation and capital letters.
- The two functions below might help us. (#include <cctype>)

```
bool checkPunctuation (string word) {
      char c = word[word.size()-1];
      if (c=='.' || c==',' || c==';' || c==':' || c=='!' || c=='?')
                return true;
      else
                return false;
}
```

```
bool checkCapital (string word) {
                char c = word[0];
                return isupper(c);
}
```

- What assumptions do we make on the word here?

```
string word;
char c;
while (fin >> word) {
      bool hasPunctuation = checkPunctuation(word);
      char mark;
      if (hasPunctuation) {
                mark = word[word.length()-1];
                word = word.substr(0,word.length()-1);
      }
      bool hasCaps = checkCapital(word);
      if (hasCaps)
                word[0] = tolower(word[0]);
      word = translate2PigLatin (word);
      if (hasCaps)
                word[0] = toupper(word[0]);
      if (hasPunctuation)
                word.push_back(mark);
      fout << word << " ";
      fin.get(c);
      if (c=='\n')
                fout << "\n";
}
```

Check for punctuation at end. Remove punctuation mark (temporarily) if there is one.

Check for capital letter at front. Change to lower-case (temporarily).

Translate the word.

If it had capital originally, capitalize first letter.

If it originally had punctuation, add mark to end. Note strings are vectors!!!

Check for line break as before.