**C++ Activities and Lecture Notes – Chapter 9**

**Reading and Writing Text Files**

There are a bewildering number of ways to perform file I/O in C and C++. The most modern and easiest is to use C++ streams. Recall that cin and cout are the console stream operators.

|  |  |
| --- | --- |
| To open a file for input in C++: | To open a file for input in Java: |
| #include<iostream>  #include<fstream>  #include<string>  using namespace std;  .  .  .  ifstream fin;  string filename;  int i, j, k;  .  .  .  fin.open(filename);  if (fin.is\_open()){  fin >> i >> j >> k;  }  else{  .  } | import java.io.\*;  .  .  .  String fileName;  String line;  File f;  BufferedReader fin;  .  .  .  try{  f = new File(fileName);  fin = new BufferedReader(new  FileReader(f));  line = fin.readLine();  }  catch (IOException e){  .  } |

The instructor usually names his C++ file input and output streams fin and fout to match the console I/O streams cin and cout. Note that in Java you try to open a file since it might fail; in C++ you should verify that the operation succeeded by calling the is\_open()function.

Use a flat-text editor to create a file of integers, separated by spaces. Then enter and execute the program ch09/maxvfal1.cpp on page 378 of the text. A few pages later, program ch09/maxval2.cpp does much the same thing but shows how to pass a reference to the stream to a function. Enter and test this program as well.

Since this is just another C++ stream all of the stream manipulators in <iomanip> that you learned back in Beginning Programming can also be used with file streams. They are summarized for you in Table 1.

“String streams” are pretty cool; they let you treat a string as a stream and read from it using the string interface. This is convenient when the text file contains, say, numeric data, but you’re forced to read it in a line at a time as a string.

**Command-Line Parameters**

The C and C++ standard allows us to write

int main(void)

to start out our programs. However, the main function is formally defined as…

|  |  |
| --- | --- |
| main function in C and C++ | main function in Java |
| int main  (int argc, char \* argv[]) | public static void main  (String [] args) |

On the C/C++ side the first parameter is the argument count. The second parameter contains the argument values as an array of strings. Two parameters are necessary as C/C++ arrays don’t know their length. “argc” and “argv” are the traditional names for these parameters.

In Java the single parameter contains the argument values as an array of Strings. Java arrays know their size so a second “count” parameter is not necessary. This parameter is traditionally called “args” as a nod to the customary C/C++ parameter names.

Let’s write some short command-line parameter demos in both languages.

|  |
| --- |
| C++ |
| #include<iostream>  using namespace std;  int main(int argc, char \* argv[]){  cout << "You gave me " << argc  << " command-line parameters" << endl;  for(int i = 0; i < argc; ++i){  cout << argv[i] << " ";  }  } |
| Java |
| public class CommandLineDemo{  public static void main (String [] args){  System.out.println("You gave me " + args.length +  " command-line parameters");  for(int i = 0; i < args.length; ++i){  System.out.print(args[i] + " ");  }  }  } |

Enter and run these two programs with the same command-line parameters. How does the output differ?

**Practice Problem #1: Data File Maker**

We wish to make a data file of pseudorandom integers. Our file format will be that the file will contain a single integer, equal to the number of integers in the file, followed by *at least* that many integers.

Write a program that performs the following tasks:

* Prompt the user for an output file name
* Prompt the user for the number of entries to be written into the file
* Prompt the user for the upper limit on the entries (1000 means three-digit values)
* Create the desired file

Command-line parameters:

If there are no command-line parameters, prompt the user for all three entries.

If there is a single command-line parameter, assume that it is the file name and prompt for the two integers.

If there are two command-line parameters, assume that they are the file name and the number of entries to write, and assume 1000 as the upper limit.

If there are three command-line parameters, assume that they are the three required parameters.

If there are more than three command-line parameters, display a usage message and exit.

**Practice Problem #2: Data File Reader**

We wish to read and display the values in a file. The format of the file is that it will contain a single integer, equal to the number of integers in the file, followed by *at least* that many integers. That leading integer is not one of the data values.

This file format allows programs that need these data files to

* Open the file
* Read one integer
* Create an array of the appropriate size
* Read the data

Write a program that performs the following tasks:

* Prompt the user for an input file name
* Read and display the file
  + The display routine should display the entire file if there are 200 or fewer entries,
  + or display the first and last hundred entries if there are more than 200 entries.

Command-line parameters:

If there are no command-line parameters, prompt the user for the file name.

If there is one command-line parameter, assume that it is the file name.

If there are more than two command-line parameters, display a usage message.