

Text File IO

Class 17

Data

- variables are storage locations for data in RAM
- RAM is volatile
- its contents vanish when the program ends
- to make data persist across different runs of a program
- and across different programs
- we store data in files on disk

Files

- a file on disk is strictly a sequence of bytes
- when you ask the operating system for some stuff from a file, you just get raw bytes
- it is up to you, the programmer, how to **interpret** those bytes

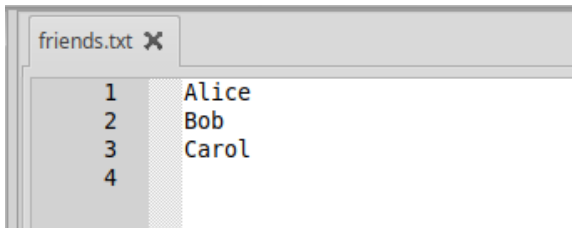
Files

- a file on disk is strictly a sequence of bytes
- when you ask the operating system for some stuff from a file, you just get raw bytes
- it is up to you, the programmer, how to interpret those bytes
- there are two main flavors of file
 1. binary files
 2. text files
- **every** file is a binary file in the sense that it contains bytes
- text files, however, contain **only** bytes that correspond to ASCII characters
- one of those bytes represents the **newline** character, interpreted as the end of a line
- thus text files are easy to interpret as a sequence of **lines** each of which is a sequence of **characters**

Characters

- to **represent** a character, a program must use an **encoding**
- an encoding is an agreement about which bit pattern will represent which character
- e.g., let us agree that in a character context the byte 0100 0001 (which is 65_{10} or $0x41$) will represent 'A'
- by default, C++ uses the ASCII encoding scheme, which we have seen several times

Text Files



what we see in an editor

A	l	i	c	e	\n	B	o	b	\n	C	a	r	o	l	\n
---	---	---	---	---	----	---	---	---	----	---	---	---	---	---	----

what is really in the disk file

- this is why **every** line of output is terminated with a newline

Binary Files

- all files contain bytes
- the bytes encode some information, that we call data
- binary files contain data that is strictly designed to be read by computer programs
- some are open standard formats, e.g., jpeg and pdf for images
- some are proprietary, e.g., xls for spreadsheets and psd for photoshop

Text Files

- text files contain data that **can** be read either by a human or by a computer program
- working with text files requires the program to be able to
 - read** copy the data from the disk file **into** a program's variables
 - write** copy data from a program's variables **out** to some space on disk

Streams

- the model that C++ uses for working with files is to consider them as **streams** of bytes
- to **read** from a file is to treat the file as an **input** stream of bytes coming in from disk
- to **write** to a file is to treat the file as a destination for an **output** stream of bytes going out to disk

Streams

- the model that C++ uses for working with files is to consider them as streams of bytes
- to read from a file is to treat the file as an input stream of bytes coming in from disk
- to write to a file is to treat the file as a destination for an output stream of bytes going out to disk
- fortunately, we already know how to deal with streams
- for input, we use the stream extraction operator \gg and the function `getline`
- for output, we use the stream insertion operator \ll

Filenames

- files on disk are identified by **filename**
 - by convention a filename consists of **name** and **extension** e.g.,
 - `grades.xls` or `phone plan.cpp`
 - by default, Windows file explorer does not show you the extension
-
- the extension indicates what **kind** of file it is
 - there are hundreds of extensions
 - we will use `.txt` for plain text files
 - we will always assume that the text file is in the **current directory** (the project directory of CodeBlocks), so we will not have to worry about paths which are different on different operating systems

Opening an Input File

- to access a disk file it must first be **opened**
- to open a file means to associate it with a special **file variable**
- the file variable must be of type **ifstream** for an input text file
- this type is defined in the `<fstream>` library

```
#include <fstream>
...
ifstream input_file;
input_file.open("foo.txt");
```

Opening an Output File

```
#include <fstream>
...
ofstream output_file;
output_file.open("foo.txt");
```

- the file variable must be of type **ofstream** for an output text file
- if the file **does not already exist**, it is created in the current directory

Opening an Output File

```
#include <fstream>
...
ofstream output_file;
output_file.open("foo.txt");
```

- the file variable must be of type `ofstream` for an output text file
- if the file does not already exist, it is created in the current directory
- If the file **does** already exist, all its contents are **deleted** by the open function call

Closing a File

- before your program terminates
- you must **close** the file
- this frees up the operating system resources associated with the file
- for output files especially, this will ensure that all write commands are processed accordingly.

```
output_file.close();
```

Programs That Write to Files

- a newly opened file is empty
- each successive output operation appends more data after what has already been written
- see files
 - Program 5-15 (page 274): write a few strings
 - Program 5-16: write strings without newlines (bad)
 - Program 5-17: numeric values entered from the keyboard, written to disk file
 - Program 5-18: string values entered from the keyboard, written to disk file

Programs That Read from Files

- the open function places the **read marker** at the first byte of the file
- as data are extracted from the file, the read marker is advanced toward the end of the file
- cannot back up