CSCI 301
Computer Science 2

Project EXAMPLE: Design Document

Introduction

Formatting text is arranging it on the screen or page in a particular way. This program is a simple text formatter: it reads an input file of text and writes the text to an output file, arranged into lines no longer than a fixed maximum length. The program considers any block of contiguous characters to be a word and white space---blanks, tabs, ends of lines---separates words. The program breaks output lines between words.

The program reads the names of the input and output files and the output line length from the terminal.

Data Structures

The program uses only one data structure, a string (array of characters) called `s`. This array holds each word read from the input file, in turn. A program constant sets the maximum allowed value of the output line length, and the string `s` is declared to be of this maximum length; an output line may be filled by one word. Another program constant sets the minimum allowed value of the output line length. It is assumed that no word in the input file exceeds the line length the user specifies.

A variable keeps track of the current length of the output line. This variable is reset when the program begins each new output line.

Functions

The program uses three functions. Two read the names of the files and open them for input and output, respectively. They continue to prompt for file names until each file is opened successfully. Another function reads the output line length, which must fall between the bounds set by the two program constants. This function, too, continues to prompt for input until the user enters a value within those bounds. The program also calls `strlen()` from the `string.h` library and `eof()` and `fail()` from `fstream.h`

The Main Program

The main program calls the functions that open the input and output files and read the output line length. It then reads words (contiguous blocks of non-blank characters) from the input file into the string `s` one at a time. From `s` it writes them to the output file.

A variable in the main program always holds the length of the current output line so far. If the word in `s` fits on the line without exceeding the output line length, it is printed, and the line length is increased by the word's length. If the next word will not fit on the current line, the program issues an end-of-line, writes the word on the next line, and resets the line length variable to the length of that word. After each word, the program writes a blank if there is room on the line; it increments the line length variable by one in this case.

This process continues until the input file has been exhausted, when the program closes both files and terminates.