Alfonso Cabargas Madrid
alfonso.cabargas@usach.cl
Rosa Barrera Capot
Sunday, June 14, 2015

Informe Técnico

Trabajo Nº1 - Manejo de Archivos y Strings en C++

Descripción del Problema

Se solicita lo siguiente:

Dado dos archivos de tipo texto, se pide a usted que trabaje con las palabras de los archivos, realizando:

- Genere un nuevo archivo que contenga la intersección de palabras de ambos archivos.

- Genere un nuevo archivo que contenga la unión de las palabras de ambos archivos.

- Genere un nuevo archivo que realice la diferencia de las palabras de ambos archivos.

- Genere un nuevo archivo, donde concatene dos palabras, la primera palabra se obtiene del primer archivo y debe terminar en vocal, la segunda palabra se obtiene del segundo archivo y debe comenzar en una consonante. Debe generar tantas palabras como pueda, es decir, se generan palabras hasta que se terminan las palabras que terminan en vocal o las que comienzan con consonante.

- Indique en que archivo se encuentra la palabra que tiene más vocales diferentes, debe mostrar la palabra y decir en que archivo(s) se encuentra.

- Genere una función adicional a las ya solicitadas, que tenga que utilizar palabras de ambos archivos, se premiará la originalidad.

Descripción de las Soluciones a Implementar

Para el primer problema utilizaremos un total de 2 funciones (intersection, isPresent).

La primera, servirá de función “principal” mientras que la segunda servirá para mantener la modularidad del algoritmo.

En **intersection()**, además de realizar las operaciones básicas de I/O y apertura/cierre de archivos, utilizaremos 2 ciclos independientes while para comprobar la presencia de cada palabra en el archivo contrario con el uso de **isPresent()** (Ej: que “Hola” se encuentre en el Archivo1 y el Archivo2) y que a su vez no se encuentre presente en el archivo de salida (Ej: que “Hola” no se encuentre en ArchivoSalida) para así evitar las entradas duplicadas.

Para el segundo problema utilizaremos un total de 2 funciones (joinWords, isPresent).

La primera, servirá de función “principal” mientras que la segunda la reutilizaremos del primer ejercicio y servirá también para mantener la modularidad del algoritmo.

En **joinWords()**, además de realizar las operaciones básicas de I/O y apertura/cierre de archivos, utilizaremos la misma estructura de la función anterior, sólo que esta vez sólo comprobaremos si la palabra se encuentra o no en el archivo de salida para ingresarla y así evitar duplicados.

Para el tercer problema utilizaremos un total de 2 funciones (difference, isPresent).

La primera, servirá de función “principal” mientras que la segunda la reutilizaremos del primer ejercicio y servirá también para mantener la modularidad del algoritmo.

En **difference()**, que es muy similar al primer ejercicio (pero al revés), además de realizar las operaciones básicas de I/O y apertura/cierre de archivos, utilizaremos 2 ciclos independientes while para comprobar la presencia de cada palabra en el archivo contrario con el uso de **isPresent()** (Ej: que “Hola” **no** se encuentre en el Archivo1 y el Archivo2) y que a su vez no se encuentre presente en el archivo de salida (Ej: que “Hola” no se encuentre en ArchivoSalida) para así evitar las entradas duplicadas.

Para el cuarto problema utilizaremos un total de 3 funciones (concatWords, startsWithConsonant, endsWithVowel).

La primera, servirá de función “principal” mientras que las otras 2 servirán para el análisis particular de cada palabra y así cumplir lo pedido en el enunciado.

En **concatWords()**, además de las operaciones básicas de I/O y apertura/cierre de archivos, utilizaremos una estructura **while-if-while-if** (bastante costosa) que permite recorrer ambos archivos y verificar si cumplen las condiciones.

En el primer ciclo **while**, se recorre cada palabra del primer archivo, luego (en el primer **if**) sí la palabra empieza con consonante, ingresamos al segundo **while**, para recorrer el segundo archivo hasta que se encuentre una palabra (en el segundo **if**) que termine con vocal y así almacenarlas (la que empiece con consonante y la que termine con vocal) dentro del archivo de salida concatenadas.

Para el quinto problema utilizaremos un total de 2 funciones (moreVowels, vowels).

La primera, servirá de función “principal” mientras que la segunda cumple la función de retornar el número de vocales contenidas en una palabra.

En **moreVowels()**, se inicializan 2 variables: **moreV** y **fileN** *(ver tablas de variables por función)* que servirán para almacenar el número mayor de vocales y en que archivo se encuentra la palabra que las contiene.

Para lograr el objetivo, utilizaremos variados ciclos **while** además de varios **if**. Debido a la necesidad de encontrar primero cual es el mayor número de vocales en una palabra y el archivo en que se encuentra, caemos en la necesidad de recorrer 1 de los archivos 2 veces.

Además, esta solución difiere de las otras ya que también necesita salida por pantalla de información.

En el primer ciclo recorremos el primer archivo palabra por palabra y si dicha palabra tiene más vocales que el valor actual de **moreV** lo reemplaza. Luego hacemos lo mismo con el segundo archivo.

Después de lo dicho, se realiza la salida por pantalla de los datos importantes (cuantas vocales y en que archivo) para luego recorrer el archivo que la contiene (gracias al valor de **fileN**) y guardar la palabra correspondiente en el archivo de salida.

Lista de Funciones

VARIABLE	TIPO	PARAMETROS DE ENTRADA	FUNCIÓN
intersection	void	string filename1, string filename2	Crea un archivo de texto con la *intersección de palabras** de 2 archivos dados.
joinWords	void	string filename1, string filename2	Crea un archivo de texto con la *unión de palabras*** de 2 archivos dados.
difference	void	string filename1, string filename2	Crea un archivo de texto con la *diferencia de palabras**** de 2 archivos dados.
concatWords	void	string filename1, string filename2	Crea un archivo de texto que concatena palabras que empiezan en consonante del primer archivo con las que terminan en vocal del segundo.
moreVowels	void	string filename1, string filename2	Muestra en que archivo se encuentra la palabra con más vocales, indica el número de vocales de la misma y la copia en un archivo de salida.
bonusTrack	void	string filename1, string filename2	Implementación simple del juego del ahorcado que utiliza como palabra a buscar la encontrada por *moreVowels()*.
optionSwitch	void	int option	Recibe una opción del usuario y ejecuta la función del menú a la que corresponde. Semi-recursiva.
printMenu	void		Imprime el menú del programa principal.
isPresent	int	string word, string filename	Retorna 1 si la palabra buscada se encuentra en el archivo dado.
startsWithConsonant	int	string word	Retorna 1 si la palabra dada empieza con consonante.
endsWithVowel	int	string word	Retorna 1 si la palabra dada termina con vocal
vowels	int	string word	Retorna el número de vocales de la palabra dada.
main	int		Función principal.

* Recorre ambos archivos y determina que palabras están presentes en ambos.
** Recorre ambos archivos y determina que palabras están presentes en ambos. Luego, genera un archivo sin considerar las que se repiten.
*** Recorre ambos archivos y determina que palabras están presentes en sólo uno (a través de excluir las repetidas).

Definición de Variables por Función

intersection(string filename1, string filename2);

VARIABLE	TIPO	FUNCIÓN	RANGO
file1	ifstream	Corresponde al primer archivo dado por el usuario.	
file2	ifstream	Corresponde al segundo archivo dado por el usuario.	
outputfile	ofstream	Corresponde al archivo de salida, su nombre es definido programaticamente.	
word	string	Se utiliza como variable para recorrer ambos archivos *palabra** por palabra.	

* Corresponde al string de caracteres agrupados que no contenga espacios en blancos ni el flag BOF/EOF.

joinWords(string filename1, string filename2);

VARIABLE	TIPO	FUNCIÓN	RANGO
file1	ifstream	Corresponde al primer archivo dado por el usuario.	
file2	ifstream	Corresponde al segundo archivo dado por el usuario.	
outputfile	ofstream	Corresponde al archivo de salida, su nombre es definido programaticamente.	
word	string	Se utiliza como variable para recorrer ambos archivos *palabra** por palabra.	

* Corresponde al string de caracteres agrupados que no contenga espacios en blancos ni el flag BOF/EOF.

difference(string filename1, string filename2);

VARIABLE	TIPO	FUNCIÓN	RANGO
file1	ifstream	Corresponde al primer archivo dado por el usuario.	
file2	ifstream	Corresponde al segundo archivo dado por el usuario.	
outputfile	ofstream	Corresponde al archivo de salida, su nombre es definido programaticamente.	
word	string	Se utiliza como variable para recorrer ambos archivos *palabra** por palabra.	

* Corresponde al string de caracteres agrupados que no contenga espacios en blancos ni el flag BOF/EOF.

concatWords(string filename1, string filename2);

VARIABLE	TIPO	FUNCIÓN	RANGO
file1	ifstream	Corresponde al primer archivo dado por el usuario.	
file2	ifstream	Corresponde al segundo archivo dado por el usuario.	
outputfile	ofstream	Corresponde al archivo de salida, su nombre es definido programaticamente.	
word	string	Se utiliza como variable para recorrer el primer archivo *palabra** por palabra.	
word2	string	Se utiliza como variable para recorrer el segundo archivo *palabra** por palabra.	

* Corresponde al string de caracteres agrupados que no contenga espacios en blancos ni el flag BOF/EOF.

moreVowels(string filename1, string filename2);

VARIABLE	TIPO	FUNCIÓN	RANGO
file1	ifstream	Corresponde al primer archivo dado por el usuario.	
file2	ifstream	Corresponde al segundo archivo dado por el usuario.	
outputfile	ofstream	Corresponde al archivo de salida, su nombre es definido programaticamente.	
word	string	Se utiliza como variable para recorrer ambos archivos *palabra** por palabra.	
moreV	int	Almacena el mayor número de vocales encontradas en una palabra.	0-Inf
fileN	int	Almacena en que archivo se encuentra la palabra con más vocales.	1-2

* Corresponde al string de caracteres agrupados que no contenga espacios en blancos ni el flag BOF/EOF.

bonusTrack(string filename1, string filename2);

VARIABLE	TIPO	FUNCIÓN	RANGO
wfile	ifstream	Corresponde al archivo generado por *moreVowels()*	
outputfile	ofstream	Corresponde al archivo de salida, su nombre es definido programaticamente.	
c	char	Toma el valor dado por el usuario en cada ciclo	
i	int	Variable auxiliar para el ciclo for.	0-Inf
aux	int	Almacena temporalmente el valor de good en cada ejecución para determinar si el usuario ingreso correctamente el carácter o no.	0-Inf
good	int	Almacena el total de letras correctas encontradas por el usuario en el juego en curso.	0-Inf
tryouts	int	Contador que corresponde al número de intentos restantes que le quedan al usuario en el juego en curso.	0-6
word	string	Almacena la palabra del archivo “moreVowels.txt” generado previamente.	

optionSwitch(int option);

VARIABLE	TIPO	FUNCIÓN	RANGO
option	int	Corresponde a la opción seleccionada por el usuario.	0-5

isPresent(string word, string filename);

VARIABLE	TIPO	FUNCIÓN	RANGO
checkfile	ifstream	Corresponde al archivo que se analizará.	
wordcmp	string	Se utiliza para contrastar la palabra recibida como parámetro versus las del archivo que se está analizando.	

startsWithConsonant(string word);

VARIABLE	TIPO	FUNCIÓN	RANGO
i	int	Variable auxiliar para el ciclo for.	0-4
vowels[5]	char	Contiene las 5 vocales.	{‘a’,’e’,’i’,’o’,’u’}

endsWithVowel(string word);

VARIABLE	TIPO	FUNCIÓN	RANGO
i	int	Variable auxiliar para el ciclo for.	0-4
vowels[5]	char	Contiene las 5 vocales.	{‘a’,’e’,’i’,’o’,’u’}

vowels(string word);

VARIABLE	TIPO	FUNCIÓN	RANGO
i	int	Variable auxiliar para el primer ciclo for.	0-Inf
j	int	Variable auxiliar para el segundo ciclo for.	0-4
c	int	Almacena el total de vocales contenidos en la palabra.	0-Inf
vowels[5]	char	Contiene las 5 vocales.	{‘a’,’e’,’i’,’o’,’u’}

Código de Fuente

```
/*
  Copyright (C) 2015  Alfonso Cabargas Madrid <felipe@cabargas.ninja>

  This program is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/

/* DOCUMENTACION

Version del Compilador:         GNU Make 3.81
Version del Sistema Operativo:  Mac OS X 10.10.4-beta
Version del Editor de Texto:    TextMate 2.10-beta

*/

#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <regex>

using namespace std;

//Main functions of the algorithm
void intersection(string filename1, string filename2);
void joinWords(string filename1, string filename2);
void difference(string filename1, string filename2);
void concatWords(string filename1, string filename2);
void moreVowels(string filename1, string filename2);
void bonusTrack(string filename1, string filename2);

//Menu function
void optionSwitch(int option);
void printMenu();

//Auxiliary functions
int isPresent(string word, string filename);
int startsWithConsonant(string word);
int endsWithVowel(string word);
int vowels(string word);

int main(){
      int option;

      cout << "Bienvenido! Ingresa los datos que se te piden: " << endl;

      printMenu();

      cin >> option;
      optionSwitch(option);
```

```
	return 0;
}

void printMenu(){
	cout << endl <<
"================================================================================
===========\n" << endl;

	cout << "Por favor selecciona una de las siguientes opciones: " <<
endl;
	cout << "\t (1) Obtener interseccion de ambos archivos" << endl;
	cout << "\t (2) Obtener union de ambos archivos" << endl;
	cout << "\t (3) Obtener diferencia de ambos archivos" << endl;
	cout << "\t (4) Obtener concatenacion de palabras de ambos archivos"
<< endl;
	cout << "\t (5) Obtener palabra con mas vocales diferentes dentro de
ambos archivos" << endl;
	cout << "\t (6) BONUS TRACK" << endl;

	cout << endl <<
"================================================================================
===========\n" << endl;
}

void optionSwitch(int option){
	string filename1, filename2;

	cout << "\t - Nombre de Archivo 1: ";
	cin >> filename1;
	cout << endl << "\t - Nombre de Archivo 2: ";
	cin >> filename2;

	cout << endl <<
"================================================================================
===========\n" << endl;


	switch(option){
		case 1:
			cout << "Ejecutando funcion 1........." << endl;
			intersection(filename1, filename2);
			cout << "Ejecucion finalizada, tu archivo de salida es
intersection.txt" << endl;
			cout << "Selecciona otra opcion o ingresa 0 para
terminar." << endl;
			printMenu();
			cin >> option;
		break;
		case 2:
			cout << "Ejecutando funcion 2........." << endl;
			joinWords(filename1, filename2);
			cout << "Ejecucion finalizada, tu archivo de salida es
joinWords.txt" << endl;
			cout << "Selecciona otra opcion o ingresa 0 para
terminar." << endl;
			printMenu();
			cin >> option;
		break;
		case 3:
			cout << "Ejecutando funcion 3........." << endl;
			difference(filename1, filename2);
```

```
                  cout << "Ejecucion finalizada, tu archivo de salida es
difference.txt" << endl;
                  cout << "Selecciona otra opcion o ingresa 0 para
terminar." << endl;
                  printMenu();
                  cin >> option;
            break;
            case 4:
                  cout << "Ejecutando funcion 4........." << endl;
                  concatWords(filename1, filename2);
                  cout << "Ejecucion finalizada, tu archivo de salida es
concatWords.txt" << endl;
                  cout << "Selecciona otra opcion o ingresa 0 para
terminar." << endl;
                  printMenu();
                  cin >> option;
            break;
            case 5:
                  cout << "Ejecutando funcion 5........." << endl;
                  moreVowels(filename1, filename2);
                  cout << "Ejecucion finalizada, tu archivo de salida es
moreVowels.txt" << endl;
                  cout << "Selecciona otra opcion o ingresa 0 para
terminar." << endl;
                  printMenu();
                  cin >> option;
            break;
            case 6:
                  cout << "Ejecutando funcion 6........." << endl;
                  bonusTrack(filename1, filename2);
                  cout << "Puedes encontrar un resumen de tu juego en
bonusTrack.txt" << endl;
                  cout << "Selecciona otra opcion o ingresa 0 para
terminar." << endl;
                  printMenu();
                  cin >> option;
            break;
            default:
                  cout << "*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=* FIN DEL
PROGRAMA *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*" << endl;
                  option = 0;
            break;
      }

      if(option != 0){
            optionSwitch(option);
      }
}

void intersection(string filename1, string filename2){
      ifstream file1, file2;
      ofstream outputfile;
      string word;
      string outputfilename = "intersection.txt";

      file1.open(filename1);
      file2.open(filename2);

      while(file1 >> word){
            if(isPresent(word, filename2) && !isPresent(word,
outputfilename)){
                  outputfile.open(outputfilename, ios::app);
                  outputfile << word << "\n";
```

```
                  outputfile.close();
            }
      }

      while(file2 >> word){
            if(isPresent(word, filename1) && !isPresent(word,
outputfilename)){
                  outputfile.open(outputfilename, ios::app);
                  outputfile << word << "\n";
                  outputfile.close();
            }
      }

      file1.close();
      file2.close();
}

void joinWords(string filename1, string filename2){
      ifstream file1, file2;
      ofstream outputfile;
      string word;
      string outputfilename = "joinWords.txt";

      file1.open(filename1);
      file2.open(filename2);

      while(file1 >> word){
            if(!isPresent(word, outputfilename)){
                  outputfile.open(outputfilename, ios::app);
                  outputfile << word << "\n";
                  outputfile.close();
            }
      }

      while(file2 >> word){
            if(!isPresent(word, outputfilename)){
                  outputfile.open(outputfilename, ios::app);
                  outputfile << word << "\n";
                  outputfile.close();
            }
      }

      file1.close();
      file2.close();
}

void difference(string filename1, string filename2){
      ifstream file1, file2;
      ofstream outputfile;
      string word;
      string outputfilename = "difference.txt";

      file1.open(filename1);
      file2.open(filename2);

      while(file1 >> word){
            if(!isPresent(word, filename2) && !isPresent(word,
outputfilename)){
                  outputfile.open(outputfilename, ios::app);
                  outputfile << word << "\n";
                  outputfile.close();
            }
      }
```

```
	while(file2 >> word){
		if(!isPresent(word, filename1) && !isPresent(word,
outputfilename)){
			outputfile.open(outputfilename, ios::app);
			outputfile << word << "\n";
			outputfile.close();
		}
	}

	file1.close();
	file2.close();
}

void concatWords(string filename1, string filename2){
	ifstream file1, file2;
	ofstream outputfile;
	string word, word2;
	string outputfilename = "concatWords.txt";

	file1.open(filename1);
	file2.open(filename2);
	outputfile.open(outputfilename);

	while(file1 >> word){
		if(startsWithConsonant(word)){
			while(file2 >> word2){
				if(endsWithVowel(word2)){
					outputfile << word << word2 << "\n";
				}
			}
		}
	}

	file1.close();
	file2.close();
	outputfile.close();
}

void moreVowels(string filename1, string filename2){
	int moreV = 0;
	int fileN; //stores number of the file that contains that word
	ifstream file1, file2;
	ofstream outputfile;
	string word;
	string outputfilename = "moreVowels.txt";

	file1.open(filename1);
	file2.open(filename2);
	outputfile.open(outputfilename);


	/* FIND GREATEST NUMBER OF VOWELS INSIDE FIRST FILE'S WORDS*/
	while(file1 >> word){
		if(vowels(word) > moreV){
			moreV = vowels(word);
			fileN = 1;
		}
	}
	/* FIND GREATEST NUMBER OF VOWELS INSIDE SECOND FILE'S WORDS*/
	while(file2 >> word){
		if(vowels(word) > moreV){
			moreV = vowels(word);
```

```
                  fileN = 2;
            }
      }

      file1.close();
      file2.close();

      /* GETTING THE WORD */
      cout << "La palabra con mas vocales se encuentra en el archivo " <<
fileN << " y contiene un total de " << moreV << " vocales." << endl;

      file1.open(filename1);
      file2.open(filename2);

      if(fileN == 1){
            while(file1 >> word){
                  if(vowels(word) == moreV){
                        outputfile << word << "\n";
                  }
            }
      } else {
            while(file2 >> word){
                  if(vowels(word) == moreV){
                        outputfile << word << "\n";
                  }
            }
      }

      outputfile.close();
}

void bonusTrack(string filename1, string filename2){
      ifstream wfile;
      ofstream outputfile;
      char c;
      int i, aux, good=0, tryouts=6;
      string word;
      string outputfilename = "bonusTrack.txt";

      outputfile.open(outputfilename);

      moreVowels(filename1, filename2);
      wfile.open("moreVowels.txt");
      wfile >> word;

      cout << "BIENVENIDO AL JUEGO DEL AHORCADO" << endl;
      cout << "================================" << endl;

      while((tryouts>0) && (good!=word.length())){

            cout << "Ingrese una letra: " << endl;
            cin >> c;
            aux = good;

            for(i=0;i<word.length();i++){
                  if(word[i] == c){
                        good++;
                        outputfile << word[i];
                  } else {
                        outputfile << "-";
                  }
            }
```

```
            if(good == aux){
                  tryouts--;
                  cout << "Respuesta equivocada, solo te quedan: " <<
tryouts << " intentos.";
            }

            outputfile << "\n";
      }


      wfile.close();
      outputfile.close();
}

//Aux Functions
int isPresent(string word, string filename){
      ifstream checkfile;
      string wordcmp;

      checkfile.open(filename);
  checkfile.clear();
  checkfile.seekg(0, ios::beg);

      while(checkfile >> wordcmp){
            if(word == wordcmp){
                  return 1;
            }
      }

      checkfile.close();
      return 0;
}

int startsWithConsonant(string word){
      int i;
      char vowels[5] = {'a','e','i','o','u'};

      for(i=0;i<5;i++){
            if(word[0] == vowels[i]){
                  return 0;
            }
      }
      return 1;
}

int endsWithVowel(string word){
      int i;
      char vowels[5] = {'a','e','i','o','u'};

      for(i=0;i<5;i++){
            if(word[word.length()-1] == vowels[i]){
                  return 1;
            }
      }
      return 0;
}

int vowels(string word){
      int i, j, c;
      char vowels[5] = {'a','e','i','o','u'};

      c = 0;
```

```
	for(i=0;i<word.length();i++){
		for(j=0;j<5;j++){
			if(word[i]==vowels[j]){
				c++;
			}
		}
	}

	return c;
}
```