UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructuras de Datos I

Segundo cuatrimestre de 2018

09 de Abril de 2018

Taller de Entrada/Salida - Vectores

1. Indicaciones

El objetivo del taller es resolver ejercicios en los cuáles tendrán que trabajar con vectores, entrada y salida estándar y manipulación de archivos.

Podrán encontrar el contenido del taller **(labo03_src.zip)**en la sección de descargas de la materia . Una vez descomprimido el archivo, se encontrarán con el árbol de directorios siguiente:

- `template-alumnos` : directorio raiz del taller.
- `template-alumnos/src`: conteniendo el cógido de las funciones de los ejercicios.
- `template-alumnos/archivos`: archivos para utilizar de ejemplos en las funciones de los ejercicios.

Se encontrarán también con varios archivos, entre los que se destacan:

- `src/ejercicios.cpp` : aquí deben completar las implementaciones de los problemas. Se dan algunos ejemplos de los ejercicios.
- `src/ejercicios.h`: headers de los problemas. Pueden agregar headers de auxiliares si lo necesitaran.
- `archivos/in/`: aquí se encuentran los archivos de entrada que utilizan como entrada de las funciones. Los archivos que representan secuencias de números contienen a los mismos separados por espacios.
- `archivos/out/`: los archivos de salida de las funciones se crean en esta carpeta.

Para utilizarlo, crear un nuevo proyecto en CLION, eligiendo como destino el directorio `template-alumnos`, y agregar al mismo `ejercicios.cpp` y `ejercicos.h`.

Importante!!!
En el archivo CMakeList.txt es preciso agregar la siguiente línea

```
set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR})
```

para que el compilador ubique el ejecutable en el folder `template-alumnos` y funciones las rutas relativas (`archivos/in`...)
Verificar que al compilar el ejecutable esta en el directorio raiz. Si no es asi, cambiar el `set` por:
`set(EXECUTABLE_OUTPUT_PATH ${CMAKE_CURRENT_SOURCE_DIR})`

2. Ejercicios

Aclaraciónes:

- salvo en el caso que lo pida explícitamente, no "convertir" los datos del archivo de entrada en un vector.
- Para todos los ejercicios que requieran utilizar la entrada y/o salida estándar escribir el código necesario en main.cpp para realizar las acciones pedidas.

2.1. bool divide(vector<int> v, int a)

Dados un vector v y un int n, decide si n divide a todos los numeros de a.

2.2. int mayor(vector<int> v)

Dado un vector v, devuelve el valor maximo encontrado.

2.3. vector<int> reverso(vector<int> v)

Dado un vector v, devuelve el reverso.

2.4. vector<int> rotar(vector<int> v, int k)

Dado un vector v y un entero k, rotar k posiciones los elementos de v. <1,2,3,4,5,6> rotado 2, deberia dar <3,4,5,6,1,2>.

2.5. bool estaOrdenado(vector<int> v)

Dado un vector v de enteros, devuelve verdadero si esta ordenado (ya sea creciente o decrecientemente).

2.6. void mostrarVector(vector<int> v)

Dado un vector de enteros muestra por pantalla su contenido. Ejemplo: si el vector es <1, 2, 5, 65> se debe mostrar en pantalla [1, 2, 5, 65].

2.7. vector<int> leerVector(string nombreArchivo)

Dado un archivo que contiene una secuencia de números enteros con el formato (por ejemplo): 1 2 34 4 45, lo lee y devuelve un vector con los números en el mismo orden.

2.8. void guardarVector(vector<int> v, string nombreArchivo)

Dado un vector de enteros y el nombre de un archivo de salida, escribe al vector en el archivo cuyo nombre se recibe como parámetro. Ejemplo: si el vector es <1, 2, 5, 65> el archivo contiene [1, 2, 5, 65]

2.9. vector<int> factoresPrimos(int n)

Dado un entero que debe ser leído de la entrada estándar, escribir la función factoresPrimos que devuelve un vector con los factores primos del entero. Mostrar el resultado por pantalla.

2.10. int elementoMedio(vector<int> v)

Dado un vector de enteros encontrar el primer elemento de izquierda a derecha tal que los elementos a su izquierda suman más que los que están a su derecha. Ejemplo: <1, 2, 3, 4> el resultado es 3 porque (1+2) < 3 + 4 y (1 + 2 + 3) > 4. El vector de entrada debe ser leído desde un archivo y el resultado debe ser mostrado por pantalla.

2.11. vector <int,int> cantApariciones(string nombreArchivo);

Dado un archivo que contiene una lista de números, contar la cantidad de apariciones de cada uno y devolver un vector de tuplas en el que la primer posición es el número y la segunda la cantidad de apariciones del mismo, el resultado no debe contener números repetidos. Por ejemplo, si el vector es <1, 2, 1, 1, 4> el vector de salida tiene que ser [(1, 3), (2, 1), (4, 1)]. Además, crear en un archivo en el directorio **out** con el mismo nombre del archivo de entrada a la función, de manera de tener una línea por cada número encontrado, un espacio y su cantidad de apariciones. Ejemplo anterior:
linea 1: '1 3'
linea 2: '2 1'
etc.
Utilizar los archivos `10000NumerosEntre1y50.in` y `cantidadApariciones.in`.

2.12. int cantidadAparicionesDePalabra(string nombreArchivo, string palabra)

Ingresar por consola una palabra a buscar y el nombre de un archivo de texto y devolver la cantidad de apariciones de la palabra en el archivo. Mostrar el resultado por pantalla.
Para testear el ejercicio pueden usar el archivo `cantidadAparicionesDePalabra.in`.

2.13. void promedio(string nombreArchivoIn1, string nombreArchivoIn2, string nombreArchivoOut)

Dados dos archivos en los que cada uno contiene una secuencia de enteros de la misma longitud, guardar el promedio de cada par de números que se encuentran en la misma "posición" en el archivo de salida. Ejemplo: si tenemos dos secuencias s1 = <1, 2, 3, 4>, s2 = <1, 25, 3, 12> el resultado debe ser [1, 13.5, 3, 8] En archivos/in/ se encuentra `promedio1.in` y `promedio2.in`. Cada archivo contiene 100 números random entre 1 y 10.

2.14. void ordenarSecuencias(string nombreArchivoIn1, string nombreArchivoIn2, string nombreArchivoOut)

Dados dos archivos en los que cada uno contiene una secuencia de enteros ordenada, ordenarlos y guardar el resultado en el archivo de salida. Ejemplo: si tenemos dos secuencias s1 = <1, 4, 8, 19>, s2 = <3, 25, 31> el resultado debe ser [1, 3, 4, 8, 25, 31].

En archivos/in/ se encuentra `ordenarSecuencia1.in` y `ordenarSecuencia2.in`. Cada archivo contiene 5000 números ordenados entre 1 y 1000. El primer archivo contiene los números impares en el rango y el segundo los pares.

2.15. void maxPerfecto(string nombreArchivo)

Dado un ñombreArchivo'conteniendo muchos numeros, busca entre ellos el mayor número perfecto y lo imprime por pantalla. Los numeros perfectos son enteros tales que su valor es igual a la suma de sus divisores. Por ejemplo, 6 = 3+2+1... 28=1+2+4+7+14, etc...

2.16. void palindromos(string nombreArchivo, string nombreArchivoOut)

Dado un archivo de texto, lo filtra y devuelve los palindromos en el archivo salida, uno por linea.

2.17. void estadisticas(string nombreArchivo)

Dado un archivo de texto, busca la palabra de mayor longitud. Luego muestra por pantalla las estadisticas de cantidad de palabras con longitud 1, 2, 3... hasta el maximo.
Ej: Palabras de longitud 1: 100
Palabras de longitud 2: 12
Palabras de longitud 3: 6
... etc

2.18. vector<int> intersecion()

Función que pide al usuario que se ingrese por teclado dos nombres de archivos que contengan solo números enteros, luego calcula la intersección y la devuelve como vector.