Práctica voluntaria de memoria dinámica
Laboratorio de Programación 2. Grupo A

16 de abril de 2007

Objetivo

El objetivo de la práctica consiste en escribir un programa para obtener las palabras que aparecen en un texto, sin repeticiones y ordenadas según la ordenación del tipo `char` (no alfabéticamente).

El programa comenzará leyendo el texto de un fichero, y creará una lista con todas las palabras que aparezcan. Posteriormente, se eliminarán las palabras repetidas y se ordenará la lista. Finalmente, el contenido de la lista se escribirá en un fichero de texto.

Representación de la lista de palabras

Representaremos las listas mediante una secuencia de nodos doblemente enlazados con cabecera. La cabecera contendrá un puntero (punto de anclaje), un entero `num_elems` con la longitud de la lista, y otro entero que indica la posición del elemento al que apunta el punto de anclaje (es decir, el punto de anclaje no apunta necesariamente al primer elemento). Los elementos se numeran desde 1 hasta `num_elems`.

Supondremos que las palabras del texto tienen a lo más `MAX_PALABRA` caracteres. El tipo base de la lista es un *array* de `MAX_PALABRA+1` caracteres.

Las definiciones de tipos son las siguientes:

```
const int MAX_PALABRA= 32;
typedef char TPalabra[MAX_PALABRA+1];

typedef TPalabra TBase;

struct TNodo;

typedef TNodo* PNodo;

struct TNodo {
   TBase info;
   PNodo sig;
   PNodo ant;
};

struct TLista {
   PNodo p;
   int   pos;
   int   num_elems;
};
```

Creación de la lista de palabras

El programa comienza leyendo el texto un fichero cuyo nombre se solicita por teclado. Todas las palabras que aparecen en el texto tienen una longitud máxima de `MAX_PALABRA` caracteres y están separadas entre sí por blancos (espacios, tabuladores y saltos de línea). Además, las palabras están formadas exclusivamente por letras. En el texto no aparecen otros caracteres aparte de las letras que forman las palabras y los blancos que las separan; es decir, no hay símbolos de puntuación, números, etc.

El siguiente texto es un ejemplo de entrada válida:

```
El objetivo de la práctica consiste en escribir
un programa para leer un texto y obtener la lista
de las palabras que aparecen en éste sin repeticiones y
ordenadas según la ordenación del tipo char
no alfabéticamente
```

El texto debe leerse palabra a palabra. Cada palabra debe almacenarse en la posición *i*-ésima de la lista que se elige aleatoriamente mediante la expresión:

```
i = 1 + (rand() % (longitud(l)+1));
```

donde `longitud(l)` es la longitud de la lista `l` y `rand()` es una función definida de la biblioteca `cstdlid` que devuelve un entero aleatorio entre 0 y `RAND_MAX`. La expresión anterior asegura que:

$$1 \leq i \leq longitud(l) + 1$$

es decir, que la posición *i*-ésima está entre el principio (1) y el final más uno ($longitud(l) + 1$) de la lista. Esto nos permite insertar en cualquier posición de la lista.

Para insertar en la posición *i*-ésima de una lista usaremos la función `inserta_en`:

```
void inserta_en(TLista & l, TBase x, int i)
```

que inserta `x` en la posición `i` de la lista `l`.

Eliminación de repeticiones y ordenación

Una vez creada una lista con las palabras que aparecen en el texto, eliminaremos palabras repetidas aplicando la función `purga()`:

```
void purga(TLista & l)
```

que elimina los elementos repetidos de la lista `l`.

Tras eliminar repeticiones, ordenaremos la lista aplicando cualquiera de los métodos de ordenación estudiados. Para comparar las palabras durante la ordenación puedes utilizar la función `strcmp()` definida en la biblioteca `cstring`:

```
int strcmp(const char *S1, const char *S2)
```

que devuelve 0 si `S1==S2`, 1 si `S1>S2` y -1 si `S1<S2`.

Escritura de la lista de palabras

Al terminar correctamente los pasos anteriores, la lista debe contener las palabras que aparecen en el texto, sin repeticiones y ordenadas según el tipo `char` de la máquina. Esta lista debe escribirse en un fichero de texto cuyo nombre debe solicitarse por teclado. En este fichero debe aparecer una palabra en cada línea.

Destrucción de la lista de palabras

La lista de palabras se destruye borrando sus elementos aleatoriamente. Para ello, utilizaremos la función `borrar()`:

```
void borrar(TLista & l, int i)
```

que borra el elemento *i*-ésimo de la lista `l`, donde $1 \leq i \leq longitud(l)$.

Normas de realización y entrega

La práctica debe realizarse individualmente.

La fecha tope de entrega será el **martes 15 de mayo**. No se aceptarán prácticas después de esa fecha.

Se debe entregar un listado completo del programa y un disco con todo el código fuente. El fichero C++ debe comenzar con un comentario que identifique a los autores de la práctica.

El listado y el disco se recogerán al comenzar la clase del martes 15 de mayo.

Ejemplo de uso

```
Laboratorio de Programación 2
Primero A de Ingeniería de Telecomunicación
Práctica voluntaria de memoria dinámica

Alumno: García Soria, Sandra

texto: ejemplo.txt
palabras: ejemplo.palabras
```

Si el fichero `ejemplo.txt` contiene el texto:

```
El objetivo de la práctica consiste en escribir
un programa para leer un texto y obtener la lista
de las palabras que aparecen en éste sin repeticiones y
ordenadas según la ordenación del tipo char
no alfabéticamente
```

en el fichero `ejemplo.palabras` se debe obtener:

```
El
alfabéticamente
aparecen
char
consiste
de
del
en
escribir
la
las
leer
lista
no
objetivo
obtener
ordenación
ordenadas
palabras
para
programa
práctica
que
repeticiones
según
sin
texto
tipo
un
y
éste
```

donde se observa que las mayúsculas preceden a las minúsculas y éstas a las acentuadas.