Programación en C/C++

Práctica voluntaria de C :

“Análisis de un texto”

David Rozas Domingo
I.T.I.Sistemas

1. Descripción del algoritmo

La idea general del algoritmo es :

- Abrimos fichero
- Leemos carácter
- Si hemos formado palabra
  - Insertamos en nuestra lista de palabras
- Si no,
  - Si es una letra, lo agregamos a nuestra palabra auxiliar
  - Si no, se descarta dicho carácter
- Mientras haya caracteres que leer

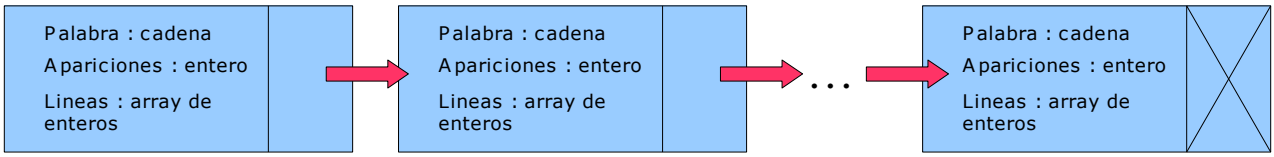
[Se omiten comprobaciones que se realizan en el código para simplificar el esquema, como la transformación de mayúsculas a minúsculas, la forma en que se produce el recuento de líneas, etc.]

En cuanto a la lógica de las funciones, cabe destacar la de la función insertar, que realiza gran parte de la labor.

- Mientras no sea el último y sea menor
  - Recorremos elementos
- Si esa palabra ya existe, actualizar nº apariciones (y quizá nº de línea)
- Si no existe, creamos un nuevo nodo con ella en su posición correspondiente

2. Estructuras de datos

La estructura de datos utilizada ha sido una lista dinámica (dado que desconocemos el tamaño del texto), con nodos en los que almacenamos la palabra, el nº de apariciones, y un array con los números de líneas.



3. Otras pruebas realizadas

Además de pruebas con el texto del enunciado, se han realizado pruebas con otros textos con el fin de hacer pruebas con textos más extensos, distintos salteados de linea, etc.
Se encuentran tanto en formato fin de línea DOS (CRLF) como en formato fin de línea Unix (LF). Tanto los textos, como sus ficheros de análisis, se encuentran en el disco con los nombres de : texto1.txt, texto1_analisis.res, neruda.txt, neruda_analisis.res, etc.

4. Código fuente

El código fuente se encuentra en un único archivo y es el siguiente :

```
/************************************************************
      Asignatura : Programacion en C/C++
      Practica : Practica voluntaria de C. "Analisis de un texto"
      Autor : David Rozas Domingo
      Ultima revision : Sun Apr 23 19:19:33 2006
************************************************************/

/*Descripcion : Realiza el analisis de las palabras de un fichero de texto
                        contado su numero de apariciones y las lineas donde
aparece*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAX_CAR_RUTA 100
#define MAX_CAR_PALABRA 50
#define MAX_NUM_LINEA 100


struct tipoNodo{
      char palabra[MAX_CAR_PALABRA];
      int linea[MAX_NUM_LINEA];
      int ultLinea;
      int apariciones;
      struct tipoNodo *sig;
      };
typedef struct tipoNodo tNodo;


void mostrar (tNodo *lista);
void insertar(tNodo **lista, char palabra[], int linea);
void liberar(tNodo *lista);
int existeNLinea(int *linea, int ultLinea, int nuevaLinea);
void pintaLineas(int *linea, int ultLinea);
void guardaLineas(FILE *fichero_destino, int *linea, int ultLinea);
void guardarFichero(tNodo *lista, char rutaOrigen[]);

int main(void)
{


      tNodo *lista;
      char palabraAux[MAX_CAR_PALABRA];
      char ruta[MAX_CAR_RUTA];
      lista = NULL;
      FILE *fichero;
      char car,carAnt;
      int nLinea = 1;
      char verResultado;

      strcpy(palabraAux,"");

      printf("\t\t A N A L I Z A D O R   D E   T E X T O S \n");
      printf("\t\t ====================================== \n\n\n");
      do
      {
            printf("Introduzca la ruta del fichero a analizar : ");
```

```
            scanf("%s",ruta);
            fichero = fopen(ruta, "rt");
            if(fichero==NULL)
                  printf("¡Esa ruta no existe!\n");

      }while(fichero==NULL);

      while(!feof(fichero))
      {
            /*Guardamos el car anterior, para ignorar las lineas blancas
intermedias en la cuenta*/
            carAnt = car;
            car = fgetc(fichero);

            if (car != EOF)
            {
                  if (isalpha(car))
                  {
                        /*Lo convertimos a minusculas*/
                        car = tolower(car);
                        strncat(palabraAux,&(car),1);
                  }else if (car==' '||car=='\n'){

                        if (strcmp(palabraAux,"")!=0)
                        {
                              insertar(&lista,palabraAux,nLinea);
                              strcpy(palabraAux,"");
                        }

                        if (car=='\n'&&carAnt!='\n')
                              nLinea++;
                  }
            }
      }

      guardarFichero(lista,ruta);


      do
      {
            printf("¿Mostrar los resultados por pantalla? (S/N) : ");
            scanf("%c",&verResultado);//Para recoger "caracter basura"
            scanf("%c",&verResultado);
      }while(verResultado!='S' && verResultado!='N');

      if (verResultado=='S')
            mostrar(lista);

      liberar(lista);

  return 0;
}


/*Inserta una linea en la lista de forma ordenada.
Si la palabra ya existe agregamos el numero de linea,
si no, secrea un nuevo nodo con esa palabra*/
void insertar (tNodo **lista, char palabra[], int linea)
{
      tNodo *pActual, *pNuevo, *pAnterior;

      pActual = *lista;
      pAnterior = *lista;
```

```
	/*Mientras no sea el ultimo, y sea menor*/
	while((pActual !=NULL) && (strcmp(pActual->palabra,palabra)<0))
	{
		pAnterior = pActual;
		pActual = pActual->sig;
	}


	/*Si ya existia, almacenamos num de linea*/
	if(pActual != NULL && strcmp(pActual->palabra,palabra)==0)
	{
		/*Si dicha linea existe, solo aumentamos el indice de apariciones*/
		if(existeNLinea(pActual->linea,pActual->ultLinea,linea))
		{
			pActual->apariciones++;
		}else{
			/*Si no existe, guardamos tb la linea en si*/
			pActual->linea[pActual->ultLinea] = linea;
			pActual->ultLinea++;
			pActual->apariciones++;
		}

	}else{
		/*Si la palabra no existe, insertamos ordenadamente*/
		pNuevo = malloc(sizeof(tNodo));
		if (pNuevo!=NULL)
		{
			strcpy(pNuevo->palabra,palabra);
			pNuevo->ultLinea = 0;
			pNuevo->linea[pNuevo->ultLinea] = linea;
			pNuevo->ultLinea++;
			pNuevo->apariciones++;

			if((pAnterior==NULL) || (pAnterior==pActual))
			{
				/*Insertamos al principio*/
				pNuevo->sig = pAnterior;
				*lista = pNuevo;
			}else{
				/*Insertamos entre medias o al final*/
				pNuevo->sig = pActual;
				pAnterior->sig = pNuevo;
			}
		}
	}

}


/*Recorre la lista mostrando su contenido por la salida estandar*/
void mostrar (tNodo *lista)
{
	int i;
	while(lista != NULL)
	{
		printf("%s : ", lista->palabra);

		if(lista->apariciones>=2)
			printf("%i apariciones; ", lista->apariciones);
		else
			printf("%i aparicion; ", lista->apariciones);

		if (lista->ultLinea>=2)
```

```
                printf("lineas: ");
            else
                printf("linea: ");

            pintaLineas(lista->linea,lista->ultLinea);

            lista = lista->sig;
        }
}

void guardarFichero(tNodo *lista, char rutaOrigen[])
{
      char rutaDestino[MAX_CAR_RUTA];
      int i;

      /*Preparamos la ruta de salida*/
      strcpy(rutaDestino, strtok(rutaOrigen,"."));
      strcat(rutaDestino, "_analisis.res");

      FILE *fichero_destino = fopen(rutaDestino, "wt");


      while(lista != NULL)
      {
            fprintf(fichero_destino,"%s : ", lista->palabra);

            if(lista->apariciones>=2)
                fprintf(fichero_destino,"%i apariciones; ", lista-
>apariciones);
            else
                fprintf(fichero_destino,"%i aparicion; ", lista->apariciones);

            if (lista->ultLinea>=2)
                fprintf(fichero_destino,"lineas: ");
            else
                fprintf(fichero_destino,"linea: ");

            guardaLineas(fichero_destino,lista->linea,lista->ultLinea);

            lista = lista->sig;
      }

      fclose(fichero_destino);
      printf("Analisis realizado satisfactoriamente.\n");
      printf("Los resultados se han guardado en : %s \n",rutaDestino);
}

/*Libera todos los nodos de la lista*/
void liberar (tNodo *lista)
{
      tNodo *pAux;

      while(lista != NULL)
      {
            pAux = lista;
            lista = lista->sig;
            free(pAux);
      }

}
```

```
/*Determina si una nuevaLinea a ingresar ya esta en el array de lineas*/
int existeNLinea(int *linea, int ultLinea, int nuevaLinea)
{
	int i=0;
	int encontrado = 0;

	while((i<ultLinea) && (encontrado==0))
	{
		if(linea[i]==nuevaLinea)
			encontrado = 1;
		i++;
	}

	return encontrado;
}

/*Muestra el contenido del array de lineas en el formato especificado*/
void pintaLineas(int *linea, int ultLinea)
{
	int i;

	for(i=0; i<ultLinea; i++)
	{
		if(i<(ultLinea-1))
			printf("%i, ",linea[i]);
		else
			printf("%i\n",linea[i]);
	}

}
/*Guarda el contenido del array de lineas en el fichero, en el formato
especificado*/
void guardaLineas(FILE *fichero_destino,int *linea, int ultLinea)
{
	int i;

	for(i=0; i<ultLinea; i++)
	{
		if(i<(ultLinea-1))
			fprintf(fichero_destino,"%i, ",linea[i]);
		else
			fprintf(fichero_destino,"%i\n",linea[i]);
	}

}
```

6. Herramientas utilizadas

Para realizar esta práctica se ha utilizado :

- Compilador gcc
- IDE Anjuta
- Open Office Writer (para la realización de la memoria)