Práctica 13
Creación, uso y administración de librerías estáticas en Turbo C

Objetivo
Conocer el proceso de la creación de librerías estátics, su uso y su administración por medio del comando TLIB del Turbo C.

Introducción
Aunque las librerías (o bibliotecas) son parecidas a los archivos objeto, hay una diferencia crucial: no todo el código de una biblioteca se añade al programa. Cuando se enlaza un programa que consiste en varios archivos objeto, todo el código de cada archivo objeto se convierte en parte del programa ejecutable final. Esto ocurre se esté utilizando el código o no. En otras palabras, todos los archivos objeto especificados en el tiempo de enlace se "unen" para formar el programa final. Sin embargo, este no es el caso de las librerías.

Una librería es una colección de funciones. A diferencia de un archivo objeto, un archivo de librería guarda el nombre de cada función, su código objeto e información de reubicación necesaria para el proceso de enlace.

Cuando un programa quiere utilizar funciones contenidas en librerías (usamos indistintamente las palabras librerías y bibliotecas) es necesario asegurar que tenga acceso a los prototipos de esas funciones. Normalmente esto se hace por medio de un archivo de cabecera (extensión **.h**) donde se colocan los prototipos de todas las funciones de la librería. Al tiempo de ligado, el enlazador busca las funciones en las librerías especificadas en la línea de comandos y añade su código objeto al programa. De esta forma, sólo se añadirán al archivo ejecutable el código de aquellas funciones que realmente se utilicen en el programa.

Actividades
1.- Captura de archivos fuente
1.1.- Captura los siguientes archivos (`f1.c`, `f2.c`, `f3.c` y `f4.c`) y compila cada uno de ellos **sin ligarlos**. Usa el comando `tcc -c f1.c` para convertir a objeto el programa fuente `f1.c` y así para los demás archivos.

```
/* --------- f1.c ------------- */
#include<stdlib.h>
void inicializa_random(void)
{
      randomize();
}

/* --------- f2.c -------------*/
#include <stdlib.h>
int obten_sig_random(int x)
{
      return random(x);
}

/* ---------- f3.c ---------- */
#include <math.h>
float cuadratica(float a, float b, float c)
{
      float raiz,x;
      raiz = b*b - 4*a*c;
      if (raiz < 0.0) return -1;
      x = (-1*b + sqrt(raiz))/(2*a);
      return x;
}
```

```
/* --------- f4.c ------------ */
#include<math.h>
float pi(register int intervals)
{
   register double width,sum;
   register int i;
   /* Obtiene el numero de intervalos */
   width = 1.0/intervals;
   /* Efectua el calculo */
   sum = 0;
   for(i=0;i<intervals;++i)
   {
      register double x = (i+0.5)*width;
      sum += 4.0 / (1.0 + x*x);
   }
   sum += width;
   return sum;
}
```

Tamaños `f1.obj: _____, f2.obj: _____, f3.obj: _____, f4.obj: _____.`

2.- Generación y uso de librerias

2.1.- Ahora vamos a crear una librería donde incluimos las funciones contenidas en los primeros tres archivos anteriores. Para esto usamos el programa TLIB del Borland C tecleando el siguiente comando:

```
tlib varios +f1 +f2 +f3, varios.lst
```

Lo que creará el archivo de librería `varios.lib` y el archivo de listado `varios.lst`
Tamaño de estos archivos: `varios.lib`: _____ `varios.lst`: _____

Despliega el contenido del archivo `varios.lst`. (con el comando `more varios.lst`) ¿Cuáles procedimientos están incluidos en la librería y cómo se les llama ahora?

_____
_____
_____

Ahora vamos a analizar un poco al contenido de la librería usando el siguiente comando:

```
tdump varios.lib | more
```

Entre toda la información que se despliega, ¿aparecen los nombres de las funciones incluidas en la librería? Sí: __ No: __

2.2.- Vamos a agregar el procedimiento definido en `f4.c` a la librería `varios.lib`. Para esto tecleamos:

```
tlib varios +f4, varios.lst
```

Lo que modificará los archivos `varios.lib` y `varios.lst`

Nuevo tamaño de estos archivos: `varios.lib`: _____ `varios.lst`: _____

Despliega el contenido del archivo `varios.lst`. ¿Cuáles procedimientos están incluidos en la librería y cómo se les llama ahora?

_____

3.- Uso de las librerías

3.1.- Ahora captura los archivos `varios.h` y `usolib.c` y compila `usolib.c` **sin ligarlo** para depurar errores:

```
/* ------ Archivo: varios.h ------- */
#include<stdio.h>
extern float cuadratica(float,float,float);
extern float pi(register int);

/* ------ Archivo: usolib.c ------- */
#include "varios.h"
int main(void)
{
   float x,y,z,r;
   printf("Teclea tres numeros (separados por comas): ");
   scanf("%f,%f,%f",&x,&y,&z);
   r = cuadratica(x,y,z);
   if (r == -1)
          printf("La ecuacion tiene raices imaginarias.\n");
   else
          printf("La raiz de la funcion con coeficientes %f, %f y %f \
          es %f\n",x,y,z,r);
   printf("Ahora teclea un múltiplo de 10 para calcular pi:");
   scanf("%f",&x);
   r = pi(x);
   printf("El valor aproximado de PI con %f intervalos es: %f",x,r);
   return 0;
}
```

3.2.- Ahora compila `usolib.c` para generar el objeto `usolib.obj` tecleando el siguiente comando:

```
tcc -c usolib.c
```

Tamaño de `usolib.obj`: _____

Para comprobar las funciones definidas en este archivo objeto, ejecuta el comando:

```
tdump /v usolib.obj | more
```

Lo que debe generar un despliegue con mucha información, entre la que se debe encontrar la lista de funciones incluidas, como `printf, scanf, cuadratica` y `pi`, entre otras.

¿Se encuentran las funciones `cuadratica` y `pi` incluidas en el archivo `usolib.obj`? Sí: ___ No: ___

3.3.- Ahora vamos a generar el ejecutable `usolib.exe` con el siguiente comando:

```
tcc -Ic:\tc\include -Lc:\tc\lib usolib.obj varios.lib
```

Tamaño de `usolib.exe`: _____

3.4.- Ejecuta **`usolib`** y comenta brevemente sobre el resultado de la ejecución

_____
_____
_____

4.- Modificaciones al programa que usa la librería

4.1.- ¿Qué tendrías que agregar a los archivos **`varios.h`** y `usolib.c` para que, utilizando las funciones `inicializa_random(void)` y `obten_sig_random(int)` (que ya están incluidas en la librería `varios.lib`) genere una lista de 5 números aleatorios?

Escribe abajo las modificaciones:

_____
_____
_____
_____
_____
_____
_____
_____
_____

4.2.- Generación del archivo ejecutable.

Para generar el nuevo ejecutable, debemos volver a compilar **`usolib.c`** tecleando:

```
tcc -c usolib.c
```

Y volver a ligarlo con la librería **`varios.lib`** tecleando:

```
tcc -Ic:\tc\include -Lc:\tc\lib usolib.obj varios.lib
```

4.3.- Ejecuta el nuevo **`usolib.exe`** y comenta sobre los resultados de su ejecución:

_____
_____
_____
_____

5.- Modificación de las funciones de la librería.

5.1.- Modifica el archivo **`f2.c`** para que ahora contenga el siguiente código:

```
/* --------- f2.c -------------*/
#include <stdlib.h>
int obten_sig_random(int x)
{
      return (rand() % x);
}
```

5.2.- Usa el comando `tcc -c f2.c` para convertir a objeto el nuevo programa fuente `f2.c`
Tamaño nuevo `f2.obj`: _____

5.3.- Modificación de la librería
Para actualizar la biblioteca `varios.lib` con la nueva función `obten_sig_random` contenida en el nuevo `f2.c`, debemos teclear el siguiente comando:

```
tlib varios.lib -+ f2
```

Que elimina de `varios.lib` el anterior `f2.obj` e inserta en su lugar el nuevo `f2.obj`, dejando actualizada la librería.

5.4.- Usando la nueva función `obten_sig_random()`
Para probar la nueva función `obten_sig_random()`, sólo es necesario repetir el paso del ligado de mi programa con sus librerías. Para esto tecleamos:

```
tcc -Ic:\tc\include -Lc:\tc\lib usolib.obj varios.lib
```

generando un nuevo `usolib.exe`.

5.5.- Ejecuta el nuevo `usolib.exe` y comenta sobre los resultados de su ejecución:

_____
_____
_____

6.- Comentarios y conclusiones

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____