DESARROLLO DE SISTEMAS DISTRIBUIDOS

Proyecto 3

Elaborado por: Ukranio Coronilla

PROGRAMADOR PRINCIPIANTE

Elaborar un programa en C++ que grafique un histograma en la pantalla de la computadora, (https://es.wikipedia.org/wiki/Histograma) con distribución normal. El programa debe recibir como parámetro el número de clases(barras) que va a graficar, el límite inferior y el límite superior.

Para generar los valores con distribución aleatoria normal puede ocupar cualquier algoritmo, como los mostrados en la dirección:

https://es.wikipedia.org/wiki/Distribuci%C3%B3n_normal

Para los gráficos se debe utilizar la biblioteca simple para gráficos X11 GFX:

https://www3.nd.edu/~dthain/courses/cse20211/fall2013/gfx/

Si no compila el programa `gfx.c` en Linux probablemente requiera instalar las librerías X11 con:

```
sudo apt-get install libx11-dev
```

El programa debe estar completamente en código C++ y orientado a objetos. Esto significa que debe crear las clases y métodos que considere necesarios así como un programa principal que los utilice. Por esta ocasión debe dejar las implementaciones, la interfaz y el código principal dentro de un mismo archivo. Debe utilizar las librerías GFX y no necesita modificarlas de modo alguno para que yo pueda compilar su programa.

El programa solo recibirá como único parámetro en la línea de comandos el entero n que indica el número de asteroides que se van a mantener coexistiendo en la pantalla.

Para utilizar las librerías GFX que se encuentran en lenguaje C dentro de nuestro código en C++ es necesario agregar al inicio del archivo gfx.h las siguientes líneas:

```
#if defined(__cplusplus)
extern "C" {
#endif
```

Y al final del archivo gfx.h las siguientes líneas:

```
#if defined(__cplusplus)
}
```

```
#endif
```

Finalmente para ejemplificar la forma de compilar escriba el siguiente programa con nombre de archivo `animación.cpp`

```
#include "gfx.h"
#include <unistd.h>

using namespace std;

int main()
{
   int t;

   gfx_open(800, 600, "Ejemplo Micro Animacion GFX");
   gfx_color(0,200,100);

   for(t = 0; t < 100; t++){
      gfx_clear();
      gfx_line( t*1+80, t*2+40, t*2+40, t*3+80 );
      gfx_line(t*5+80, t*3+40, t*3+40, t*5+80);
      gfx_flush();
      usleep(41666);  //24 por segundo
   }
   return 0;
}
```

Ahora después de haber descargado los archivos `gfx.h` y `gfx.c` solo debe ejecutar en la línea de comandos los siguientes:

```
gcc gfx.c -c
g++ animacion.cpp -c
g++ gfx.o animacion.o -o animacion -lX11
```

PROGRAMADOR "NORMAL"

Elabore un programa en C++ que analice los n archivos de texto que se encuentren almacenados en una carpeta, e imprime las 500 palabras más frecuentes que aparecen en los textos, ordenadas de mayor frecuencia a menor frecuencia, tal y como aparecen en la página web:

http://www.lavanguardia.com/vangdata/20150724/54434113592/cuales-son-las-500-palabras-mas-frecuentes-en-espanol.html

Se deja una liga para descargar un conjunto de textos y usarlos como pruebas.

Con intención de mejorar los tiempos de procesamiento, el programa debe recibir como parámetro el número de hilos que trabajarán de manera concurrente en el procesamiento, y puede ser desde 1 hasta 8. El programa también debe usar la clase Archivo desarrollada en clase, y recibir como segundo parámetro la carpeta donde se guardan los archivos.

En UNIX para abrir un directorio y leerlo se utilizan las funciones `opendir()` y `readdir()`. A continuación se muestra un ejemplo de su uso, tomado de internet.

```
/* openreaddir.c by mind [mind metalshell com]
 *
 * Example on using opendir, closedir, and readdir to open a directory
 * stream and read in and print file names.
 *
 * 06/04/03
 *
 * http://www.metalshell.com/
 *
 */

#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <errno.h>

int main(int argc, char *argv[])
{
        DIR             *dip;
        struct dirent   *dit;

        int             i = 0;

        /* check to see if user entered a directory name */
        if (argc < 2)
        {
                printf("Usage: %s <directory>\n", argv[0]);
                return 0;
        }

        /* DIR *opendir(const char *name);
         *
         * Open a directory stream to argv[1] and make sure
         * it's a readable and valid (directory) */
        if ((dip = opendir(argv[1])) == NULL)
        {
                perror("opendir");
                return 0;
        }

        printf("Directory stream is now open\n");

        /*  struct dirent *readdir(DIR *dir);
         *
         * Read in the files from argv[1] and print */
        while ((dit = readdir(dip)) != NULL)
        {
                i++;
                printf("\n%s", dit->d_name);
        }

        printf("\n\nreaddir() found a total of %i files\n", i);

        /* int closedir(DIR *dir);
         *
         * Close the stream to argv[1]. And check for errors. */
        if (closedir(dip) == -1)
        {
                perror("closedir");
                return 0;
        }
```

```
        printf("\nDirectory stream is now closed\n");
        return 1;
}
```

IMPORTANTE:

Es obligatorio hacer el proyecto “NORMAL” en este caso si el proyecto 2 lo elegiste “NORMAL”.

Para subir los proyectos a MOODLE: Los códigos de los programas elaborados deberán concatenarse en un solo archivo de texto plano, dejando una línea de asteriscos “*” entre cada ejercicio. En caso de que tenga que incluir el programa principal, la interfaz, la implementación, y archivo Makefile separar cada uno de estos archivos con una línea de guiones bajos "_".

El nombre del archivo debe ser el nombre del alumno separado con guion bajo, materia (DSD), grupo, numero de proyecto y extensión txt. El no cumplir con estos requisitos provocará la disminución de la calificación.

Ejemplo:
Antonio_Orlando_Rodriguez_DSD_4CM2_2.txt

Advertencia: Evite copiar programas y que le sean copiados, cualquier acto de plagio se castigará para plagiario y plagiado con cero