Práctica 3: Estructuras de control condicionales (2 sesiones)

Programación 1. Grado en Ingeniería Informática

26 de septiembre de 2016

Objetivos:
- Revisar el concepto de algoritmo y entender la necesidad del diseño de algoritmos en el estudio y resolución de programas.
- Comprender y diferenciar perfectamente los distintos tipos de instrucciones utilizadas en el diseño de algoritmos.
- Conocer y manejar con habilidad las sentencias de control condicionales.

1.

Diseña un programa que pida un número por teclado y nos diga si es múltiplo de 5, es decir, si es divisible por cinco o no o no.

2.

Modifica el programa 5 de la práctica 2 para que además de calcular el IMC dado el peso y la altura informe al paciente de su estado de salud de acuerdo a la siguiente tabla.

IMC	**Salud**
Menor que 16	Falto de Peso
De 16 a 25	Normal
Mayor que 25	Sobrepeso

3.

Implementa un algoritmo que lea un carácter e indique si es o no una letra. Además, en caso de ser letra, debe indicarse si es vocal o consonante y si está en minúsculas o mayúsculas. Ten en cuenta que en lenguaje C la comparación de caracteres utiliza el código ASCII asociado a cada carácter. Sugerencia: buscar en Internet o en un libro "tabla códigos ASCII".

4.

Tres empresas de acceso a Internet han lanzado diferentes ofertas de tarifa plana parcial mensual para esta temporada de verano. Dichas empresas proponen una tarifa plana de lunes a viernes entre las 15:00 y las 8:00, que también se aplica para el fin de semana. Para el resto, es decir, de lunes a viernes del 8:00 a 15:00, ofrecen una tarificación adicional por minuto. Por tanto, el coste total mensual será la suma de la tarifa plana más los minutos utilizados con el horario adicional. Los importes de las diferentes compañías son los siguientes:

Empresa/Tarifa	**Tarifa plana**	**Tarifa minuto**
PapaFon / C1	16 euros	0.03 euros
Movestor / C2	15 euros	0.06 euros
Tevés / C3	11 euros	0.05 euros

Escribe un programa que permita determinar qué oferta de esas tres nos resultaría más económica en función de los minutos de conexión del usuario fuera del horario de la tarifa plana. Para ello, el programa deberá solicitar el número de minutos que el usuario accede al servicio de lunes a viernes entre las 8:00 y las 15:00 durante un mes. Como salida el programa debe proporcionar el coste mensual de cada compañía, el coste mensual más económico y el nombre de la compañía que lo ofrece.

5.

Se pretende gestionar el cobro por fracciones horarias en un parking. Se cobra por fracciones de 30 minutos, donde cada una de ellas cuesta 0.35 euros.

Por ejemplo, para un tiempo de permanencia en el parking de 17 minutos, la cantidad a abonar sería de 0.35 euros. Observa que si se ha estado menos de 30 minutos, se abona la fracción completa. De modo similar para un tiempo de 91 minutos habría que abonar el importe correspondiente a 4 fracciones, es decir, 1.40 euros.

Realiza dos versiones del programa:

- A partir de la fecha, hora y minuto de entrada y la fecha, hora y minuto de salida indique la cantidad a pagar. (Se debe comprobar que la hora de salida es superior a la hora de entrada. Puede permanecer en el parking más de un día)

- A partir de la fecha, hora y minuto de entrada indique la cantidad a pagar. (La hora y fecha de salida se debe extraer del sistema)

6.

Escribe un programa en C que dibuje un cuadrado o un rectángulo en función de la elección del usuario. Para ello, el programa preguntará al usuario si desea dibujar un cuadrado o un rectángulo. En función de la respuesta solicitará el tamaño de un lado (para el cuadrado) o de dos lados (para el rectángulo).

ANEXO I (extracción fecha y hora del sistema)

```
#include<iostream>
#include<ctime>
using namespace std;

int main() {
     time_t fecha_sistema, hora_local;
     fecha_sistema = time(NULL);
     hora_local = time (NULL);

     struct tm *tiempo = localtime(&fecha_sistema);
     struct tm *horario = localtime(&hora_local);

     int dia, mes, anio;
     int hora,min;

     hora = horario->tm_hour;
     min = horario->tm_min;

     anio=tiempo->tm_year + 1900;
     mes=tiempo->tm_mon + 1;
     dia=tiempo->tm_mday;

     cout<< dia << "/" << mes << "/" << anio <<endl;
   cout << hora << ":" << min << endl;

     return 0;
}
```

ANEXO II (Librería Gráfica para C)

C no tiene instrucciones estándar para dibujar en pantalla. Por tanto, vamos a usar una librería gráfica adicional. Puedes mirar en el apéndice de la práctica los comandos para dibujar en pantalla que ofrece la librería. En lugar de esos, que usaremos en otros ejercicios, aquí emplearemos lo que se llaman "gráficos de tortuga". Consisten en que dibujamos moviendo un cursor gráfico situado en un punto de la pantalla (que en la versión original era una tortuga, ya que los gráficos de este tipo nacieron para enseñar a niños). Podemos hacer las siguientes operaciones:

- Dibujar avanzando en línea recta en la dirección en que está mirando ahora la tortuga: `draw(longitud)`. Inicialmente, la tortuga está mirando hacia la derecha (0 grados).
- Cambiar la orientación de la tortuga haciendo que gire sobre sí misma un cierto número de grados: `turn(grados)`. Para girar a la izquierda hay que hacerlo una cantidad negativa, y a la derecha, positiva.
- Colocar la tortuga en la posición inicial `(x,y)` que queramos: `setTurtlePos(x,y)`

Así por ejemplo podríamos dibujar los cuatro lados de un cuadrado repitiendo cuatro veces las siguientes órdenes

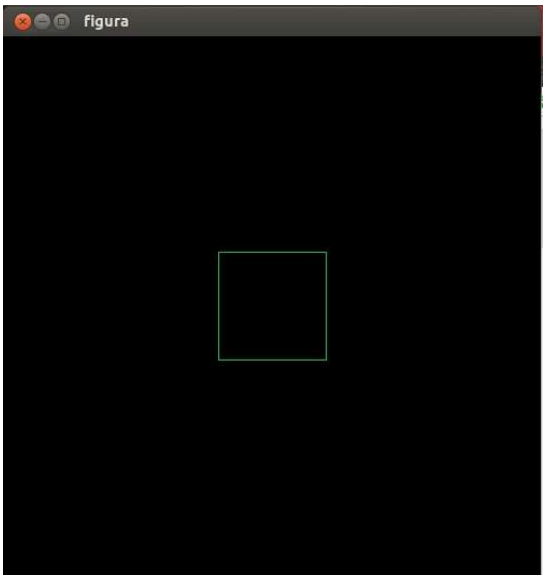
```
//dibujar en línea recta 100 pixeles
draw(100);
//girar 90 grados hacia la derecha (positivo es a derecha, negativo a
izquierda)
turn(90);
```

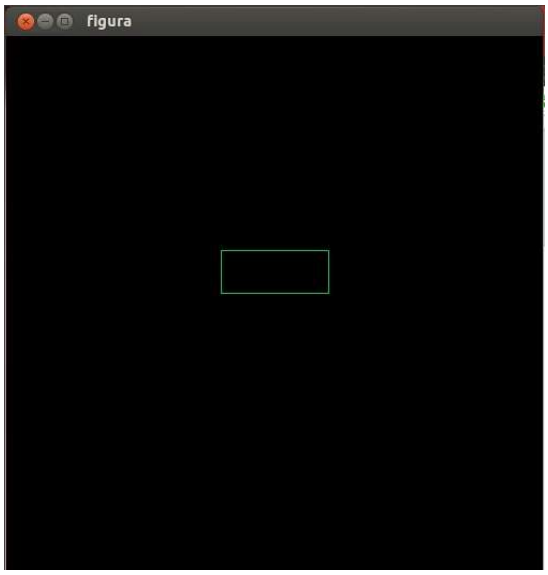
Cuidado, para compilar los ejemplos gráficos necesitas una orden distinta a la habitual, ya que estamos usando una librería adicional

```
g++ -o figura figura.c gfx.c -lX11
```

Deberás descargarte también los ficheros `gfx.c` y `gfx.h` desde Moodle.
Además del código para resolver el ejercicio planteado deberás escribir las siguientes líneas de código:

```
#include "gfx.h"  // fichero con las funciones gráficas

...

int main() {

char c;
...
//instrucción para abrir una pantalla para dibujar
gfx_open(500, 500, "figura");
// instrucción para indicar el color del dibujo
gfx_color(0,200,100);
// situación del punto de inicio del dibujo
setTurtlePos(200,200);
...

...

//finalizar la ejecución y cerrar la pantalla de dibujo
c = gfx_wait();
}
```





Cómo dibujar en pantalla con gfx.c

Para dibujar en pantalla dispones de una librería muy simple llamada gfx.c que además de los gráficos de tortuga que ya hemos visto en los ejercicios nos permite hacer una serie de operaciones básicas:

- Abrir una ventana donde dibujar: `gfx_open(ancho_ventana, alto_ventana, titulo_ventana)`. El sistema de coordenadas es como el de ejes cartesianos, la x es horizontal, la y vertical. El origen de coordenadas (x=0,y=0) está en la esquina superior izquierda de la ventana.
- Cambiar el color en que se va a dibujar como combinación de rojo, verde y azul: `gfx_color(valor_rojo, valor_verde, valor_azul)`. Por ejemplo `gfx_color(255,0,0) es rojo.`
- Dibujar una línea sabiendo su (x,y) inicial y final: `gfx_line(x1,y1,x2,y2)`. Para dibujar un punto simplemente puedes usar la misma x e y como coordenada inicial y final.
- Las instrucciones de dibujar en pantalla no tienen efecto hasta que no se hace `gfx_flush()`. Es decir, si quieres que aparezca un objeto formado por varias líneas dibújalas todas y luego haz `gfx_flush()` para que aparezca.
- Borrar la pantalla: `gfx_clear()`.

Instalar la librería gráfica

Este apartado solo se aplica en caso de que estés trabajando en tu propio ordenador. Los ordenadores del laboratorio ya tienen hecha la instalación necesaria

Es posible que no tengas instalado en tu Ubuntu lo necesario para compilar estos programas gráficos. Necesitas el paquete `'xorg-dev'`. Teclea en una terminal

```
sudo apt-get install xorg-dev
```

Te pedirá la contraseña de administrador. En la máquina virtual que usamos para la asignatura es "p1" (sin las comillas).

Compilar programas

Para compilar los programas que hagan uso de la librería en linux necesitarás añadir un par de parámetros al final de la orden que usamos siempre:

```
g++ -o cuadrado cuadrado.c gfx.c -lX11
```