Fundamentos de Programación
Sesión 5

Actividades a realizar en casa

Actividad: Resolución de problemas.

Resolved los siguientes problemas de la relación 2. Recordad que, **ANTES** del inicio de esta sesión en el aula de ordenadores, hay que subir las soluciones a PRADO.

10 (Divisores)
11 (Mínimo)
14 (Capital)
15 (Pedir mayúscula)
16 (Desgarrable)
18 (Potencia)
21 (Sucursales), excepto la última frase sobre redirección. Añadid diagrama de flujo.

Actividades a realizar en las aulas de ordenadores

Redirección de entradas con cin

Cada vez que se ejecuta cin, se lee un dato desde el periférico por defecto. Hasta ahora, nosotros lo hemos hecho desde el teclado, introduciendo un dato y a continuación un ENTER. Vamos a trabajar otra forma alternativa: redireccionar la entrada de cin para que los datos sean leídos desde un fichero.

Copiad localmente el fichero 5_media09.cpp disponible en PRADO. El programa calcula la media de los números introducidos por el usuario hasta que inserta -1.

```
#include <iostream>

using namespace std;

int main () {

    int valor;
    double media=0;
    int contador=0;

    cout << "\nIntroduce un entero en 0 y 9 (-1 para terminar):" << "\n";
    cin >> valor;
    media += valor;
    contador ++;

    while (valor!=-1){
        cout << "\nIntroduce un entero en 0 y 9 (-1 para terminar):" << "\n";
        cin >> valor;
        if (valor != -1){
            media += valor;
            contador ++;
        }
    }

    media = media / contador;
    cout << "\nLa media es; "  << media << "\n";
}
```

Figura 1. Programa que calcula medias

Observad el código del programa y ejecutadlo. Para comprobar el correcto funcionamiento de nuestro programa, introducid los datos pedidos como lo hemos hecho hasta ahora: utilizando el teclado y pulsando ENTER, despues de cada dato insertado. Por ejemplo, introducid la secuencia 2, 5, 8, 4, 3, 1, 1, 1, 5, 2, 7, -1, cuya salida debe ser 3.54545.

Si quisiéramos comprobar el correcto funcionamiento del código mientras lo vamos desarrollando, es algo arduo insertar dato a dato la secuencia en cada una de las ejecuciones que realicemos. Para no tener que introducir los valores pedidos uno a uno, podemos recurrir a un simple copy-paste. Para comprobarlo, cread un fichero de texto con la secuencia de enteros descrita arriba y llamadlo secuencia.txt. Podéis separar cada entero con algunos espacios en blanco. Seleccionad con el ratón esta secuencia y copiadlos al portapapeles (Click derecho-Copiar). Ejecutad el programa y cuando aparezca la consola del sistema haced click derecho sobre la ventana y seleccionad Editar-Pegar.

Sin embargo, si las secuencias son distintas y algo largas, esto tampoco es todo lo práctico que uno quisiera. La alternativa que se propone en este guión es ejecutar el fichero .exe desde el sistema operativo y redirigir la entrada de datos al fichero que contiene los datos. Para poder leer los datos del fichero, basta con ejecutar el programa .exe desde una consola del sistema[1] y especificar que la entrada de datos será desde un fichero a través del símbolo de redireccionamiento < (no ha de confundirse con el token << que aparece en una instrucción cout).

Para redireccionar, abrimos la carpeta con el programa ejecutable desde el explorador y seleccionamos con el click derecha del ratón Abrir Símbolo del Sistema[2]. Introducimos la instrucción siguiente:

```
5_media09.exe < secuencia.txt
```

En caso de trabajar sobre Linux o MacOS, la instrucción en el Terminal es:

```
./5_media09 < secuencia.txt
```

Cuando ejecutemos el programa, cada ejecución de cin leerá un dato desde el fichero indicado, saltándose todos los espacios en blanco y tabuladores que hubiese previamente. Cuando llegue al final del fichero, cualquier entrada de datos posterior que realicemos dará un fallo, ya que tomará la última entrada leída. De hecho, la sentencia system("pause") no detiene el programa tal y como querríamos.

Redirección de salidas con cout

Al igual que con las entradas, la salida de un programa puede redireccionarse a un fichero, en vez de a la consola (la salida estándar). Por ejemplo, podemos considerar el fichero 5_repeticion.cpp en la carpeta de prácticas de PRADO.

```
1  /***  fichero repeticion.cpp ***/
2
3  #include <iostream>
4  #include <string>
5
6  using namespace std;
7
8  int main(){
9      string cadena;
10
11     cin >> cadena;
12
13     for (int i=0;i<5;i++)
14         cout << cadena << endl;
15 }
```

Figura 2. Programa 5_repeticion.cpp

[1] También pueden leerse datos de un fichero desde dentro del propio código fuente del programa, pero esto se verá en el segundo cuatrimestre

[2] Para poder lanzar una consola desde el explorador de Windows, en nuestra casa, o bien instalamos un programa que permita abrir una consola en el directorio actual, como por ejemplo Open Command Prompt Shell Extension disponible en http://code.kliu.org/cmdopen/ o bien abrimos un símbolo del sistema (Inicio->Ejecutar->cmd) y vamos cambiando de directorio con la orden cd

Este programa muestra como salida cinco repeticiones de una cadena de caracteres (sin separadores) dada como entrada. Descargad el fichero, compilad y comprobad si funciona como se describe en esta memoria.

Para redireccionar la salida, abrid la carpeta con el programa ejecutable desde el explorador y seleccionad con el click derecha del ratón Abrir Símbolo del Sistema. Introducimos la instrucción siguiente:

```
5_repeticion.exe > rep.txt
```

En caso de trabajar sobre Linux o MacOS, la instrucción en el Terminal es:

```
./5_repeticion > rep.txt
```

Ahora, desde teclado, introducid la cadena de caracteres "Pepito". Esto creará un fichero rep.txt (o sobreescribirá en él, si es que existe) como el de la Figura 3.

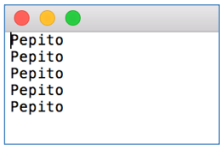

Figura 3. Fichero rep.txt

Si no quisiéramos borrar los datos existentes en el fichero, utilizaríamos el operador >>. Es decir, escribiríamos por consola

```
5_repeticion.exe >> rep.txt
```

Encauzamiento de programas

Por encauzamiento, entendemos que la salida de un programa se pasa directamente como la entrada de otro, sin ficheros intermedios. Considerad ahora el programa 5_cuadrado.cpp en la carpeta de prácticas de PRADO

```cpp
#include <iostream>

using namespace std;

int main(){
    double numero;
    double cuadrado;

    cin >> numero;

    cuadrado = numero * numero;

    cout << cuadrado << endl;
}
```

Figura 4. Programa 5_cuadrado.cpp

Este programa muestra el cuadrado de un número real introducido como entrada. Podemos encauzar la salida de 5_media09.exe con 5_cuadrado.exe utilizando la sintaxis

5_media09.exe | 5_cuadrado.exe

En este caso, la secuencia de entrada 2, 5, 8, 4, 3, 1, 1, 1, 5, 2, 7, -1 daría como resultado 12.5702 (el cuadrado de 3.5454, la salida de 5_media09.exe)

NOTA: La entrada para 5_cuadrado.exe es todo aquello que daba como salida 5_media09.exe, es decir, todo aquello que mostrábamos con cout. En particular, los mensajes para guiar al usuario. Para un correcto funcionamiento del encauzamiento, debemos modificar 5_media09.cpp para que sólo muestre por pantalla la media de los números, puesto que la entrada de 5_cuadrado.exe es solamente un número. Otra opción, si deseamos que siga mostrando los mensajes, es enviar esas cadenas de caracteres a través de la salida de error cerr y reservar cout sólo para el valor de la media. La sintaxis es la misma, cerr << cadenaCaracteres;, sin embargo la salida de errores no se redirecciona con | o >. Por lo que no lo tomará 5_cuadrado.exe como entrada. Por ejemplo, como se muestra en la Figura 5.

```
#include <iostream>

using namespace std;

int main () {

    int valor;
    double media=0;
    int contador=0;

    cerr << "\nIntroduce un entero en 0 y 9 (-1 para terminar):" << endl;
    cin >> valor;
    media += valor;
    contador ++;

    while (valor!=-1){
        cerr << "\nIntroduce un entero en 0 y 9 (-1 para terminar):" << endl;
        cin >> valor;
        if (valor != -1){
            media += valor;
            contador ++;
        }
    }

    media = media / contador;
    cerr << "La media es: ";
    cout << media;
}
```

Figura 5. Programa para calcular medias.

Por último, es posible realizar diversas combinaciones de redirección y encauzamiento. Por ejemplo, podríamos ejecutar

5_media09.exe < secuencia.txt | 5_cuadrado.exe

donde 5_media09.exe toma los datos de secuencia.txt, y la salida es el dato de entrada de 5_cuadrado.exe.

5_media09.exe < secuencia.txt | 5_cuadrado.exe > num.txt

que hace lo mismo de antes, pero la salida la almacena en un fichero num.txt (que creará, si no existe).

Obviamente, no está permitido redireccionar una salida y después encauzarla.

Actividad: Resolución de problemas.

21 (Sucursales), la última frase sobre redirección. Esto no hace falta entregarlo.

Ejercicio a entregar (Transmisión). La NASA ha captado una comunicación extraterrestre compuesta exclusivamente de 1 y 0 que siempre termina con 5 ceros, por ejemplo,

0 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0

Se cree que esta secuencia codifica un número entero. Este número estará descompuesto en sus factores primos y cada factor aparecerá en la secuencia como la longitud de cada secuencia concreta de valores 1. Por ejemplo la secuencia anterior codificaría los números primos 3, 2, 2 y 7, lo que nos da el número 84 = 3*2*2*7. Se pide construir un programa que lea una secuencia de números de este tipo y determine el número oculto.

- Primero, resuélvelo de la manera usual. Los datos se insertan desde teclado y el resultado se muestra por consola.

- Segundo, divide el código en dos programas. Uno que lea una secuencia y devuelva cómo salida (muestre por pantalla) los números primos que forman el número decodificado. Otro que tenga como entrada los factores primos y como salida el producto de ellos (es, decir, el número decodificado). Utiliza redirección y encauzamiento para construir el proceso completo.

- **Ampliación** (Sin redirección. No hace falta entregarlo si no se quiere). ¿Y si la secuencia 0 0 0 0 0 no determina el fin de la transmisión, si no que determina el fin de la transmisión de un número y a continuación se empieza a transmitir otro número (la transmisión termina, por ejemplo, con seis ceros 0 0 0 0 0 0)? Por ejemplo, la secuencia

  0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0

  devolvería como salida la secuencia de números 6 2 30. Implementa dicho decodificador.