tecnun
CAMPUS TECNOLÓGICO DE LA UNIVERSIDAD DE NAVARRA
NAFARROAKO UNIBERTSITATEKO CAMPUS TEKNOLOGIKOA
Escuela Superior de Ingenieros · Ingeniarien Goi Mailako Eskola

***Carnet*: _____** ***Nombre*: _____**

Practica Calificada de C++

Curso 18-19

Informática II

Dr. Paul Bustamante / Dr. Ibón Elosegui

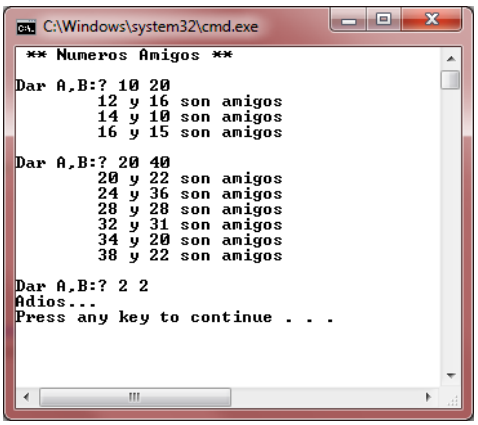
ÍNDICE

1. Introducción.

Recuerde que si no entiende bien el enunciado de los ejercicios o no está seguro de lo que se le pide, puede preguntar.

1.1 Números amigos (2pts).

En este ejercicio se pide implementar un programa que permita encontrar los números **amigos** entre un intervalo A y B.



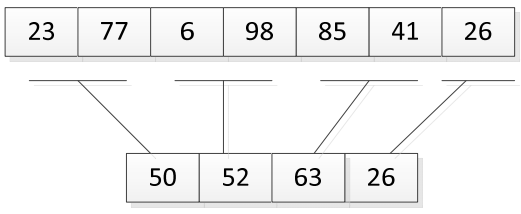
Un número **"x"** es amigo de otro **"y"** si la suma de los divisores (incluyendo el 1) de "x" da como resultado "y" (p.e 12 y 16, donde la suma de los divisores del 12 (1+2+3+4+6) es igual a16).

El programa deberá pedir los límites A y B, tras lo cual imprimirá por pantalla los números amigos. El criterio de terminación del programa será la introducción de dos números iguales (en el ej. 2 y 2).

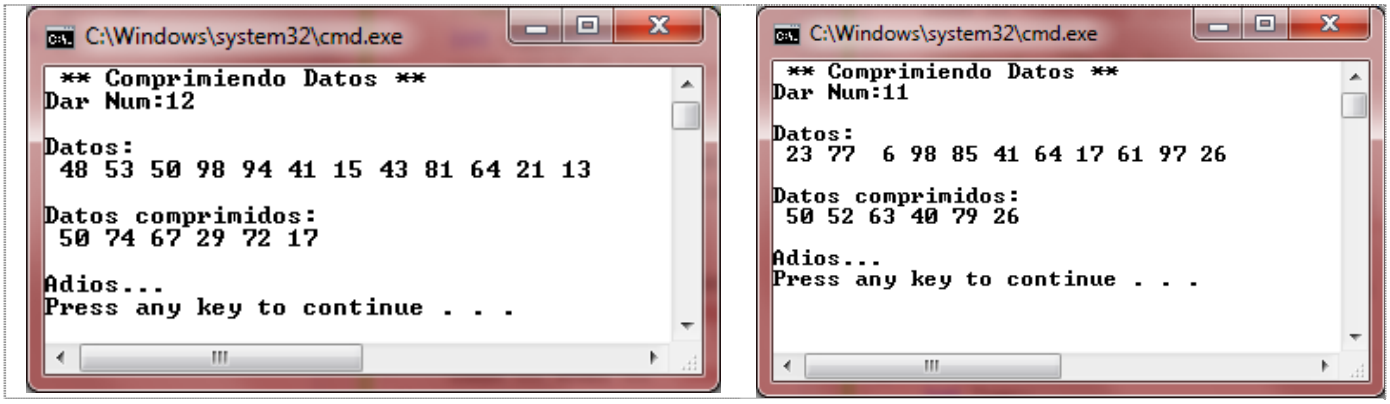
Se deberá implementar la función **bool amigo(int x, int y)**, la cual tendrá el algoritmo que compruebe si los dos números son amigos. Esta función será llamada desde **main**.

1.2 Comprimiendo información (4pts).

Este ejercicio consiste en hacer un programa que permita **comprimir la información** de un vector de datos de **números enteros**, con el fin de reducir su tamaño. Para esto, básicamente tiene que hacer la media de dos números adyacentes, con lo cual ya reduce el vector a la mitad. Como podrá ver, si el tamaño del vector es **impar**, el último elemento se copia directamente en el nuevo.



En las siguientes pantallas se muestra el funcionamiento del programa:



Requisitos del programa:

- Los vectores deberán ser **dinámicos** (tanto el orignal como el comprimido)
- Deberá pedir sólo una vez el número de elementos (**Num** en la pantalla)
- El vector Inicial se deberá llenar con número enteros aleatorios (entre 0 y 100).
- El programa deberá contar con las siguientes funciones (las cuales serán llamadas desde **main**):
  - `int *generaData( int &num);` esta funcion pide el número de elemntos (**Num)**, crea el vector dinámico y genera los numeros aleatorios (entre 0 y 100). Deberá devolver el vector Inicial creado.
  - `int *comprimeData( int *vec, int num, int &nnum);` A esta funcion se le pasa el vector original (vec y num) y devuelve un vuevo vector dinámico, con la informacion comprimida y el nuevo numero (nnum). Tener en cuenta que si **num** es impar, el último elemento lo deberá copiar directamente.
  - `void printData(int *vec, int num);` Por último, deberá crear esta funcion para imptrimir cualquier vector (el original o el comprimido)

Nota: si NO crea estas **funciones** y no usa reserva **dinámica** de memoria, el ejercicio no se puntuará.

1.3 Control de gastos con estructuras, en C++ (4pts).

Este ejercicio consiste en realizar un programa que permita llevar un control adecuado de los gastos, tales como comida, transporte, luz, agua, teléfono, etc. durante el mes, haciendo uso de las estructuras del C++.

El programa debe contar con un **menú** de 3 opciones:

1) **Cargar Gastos**: Al seleccionar esta opción deberá aparecer en la pantalla un mensaje que permita al usuario introducir el nombre del Gasto (alimentación, bus, móvil, alquiler, etc.). Luego, el programa le pedirá que introduzca el valor del gasto y el día del mes. Una vez introducidos los datos, debe volver a aparecer el menú para permitir seguir cargando los gastos.
2) **Ver Gastos**: Esta opción es la que tiene más peso en el programa, ya que debe permitir sacar un reporte de los gastos que se han introducido y el total de los mismos.
3) **Salir**. Termina exitosamente el programa.

Para la programación, **deberá** crear la siguiente estructura y en **main** deberá crear un array o vector de gastos y un menú con las opciones descritas anteriormente:

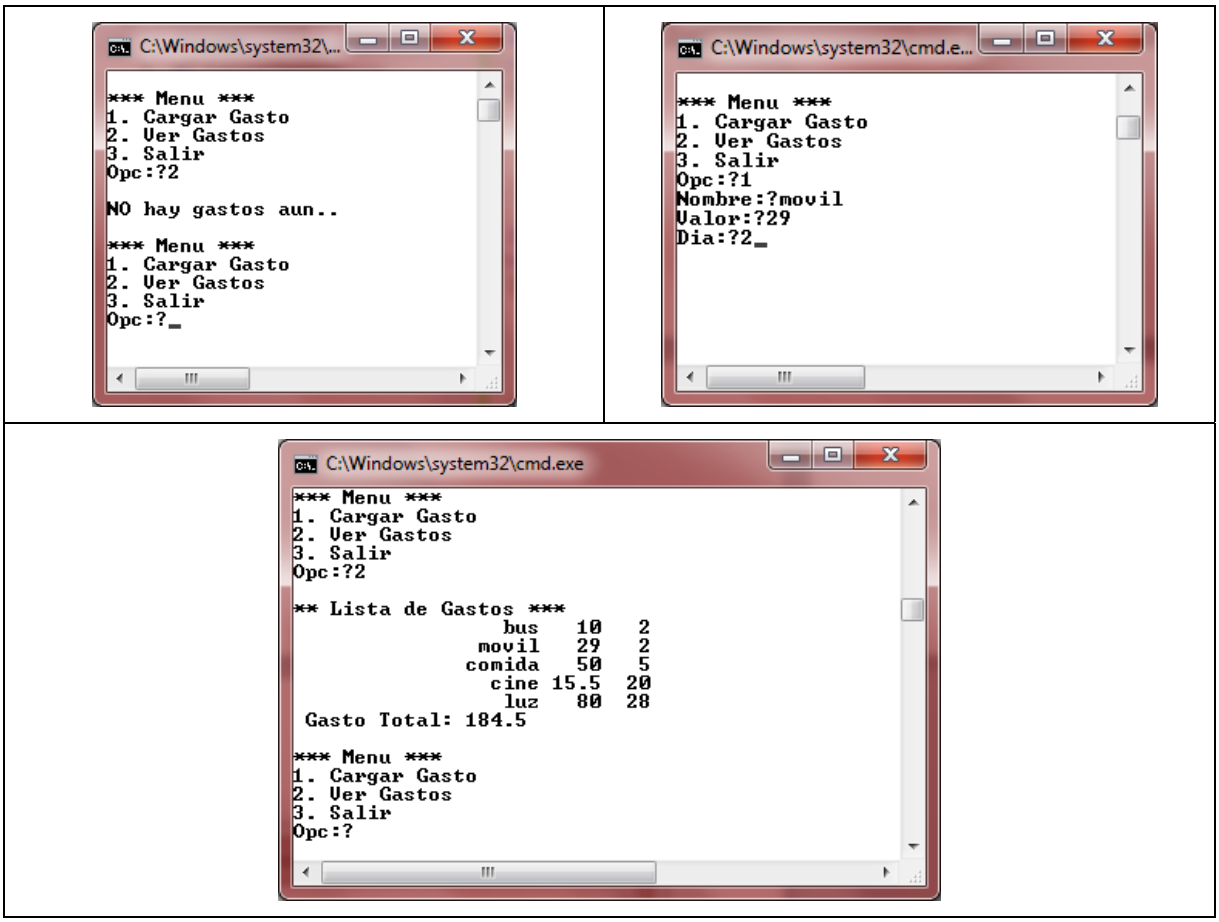
```
struct Gasto
{
	char nombre[40];	//bus, comida, agua, movil, etc.
	double valor;		//valor en euros
	int dia;		//día del mes

};
```

También deberá crear **02 funciones**, las cuales deberán ser llamadas desde **main**. Los argumentos de entrada deberán ser el ***array*** de estructuras y el número de elementos que hay:

- **cargarGasto**(…)
- **verGastos**(…)

Funcionamiento del programa:



Buena suerte a todos!!!