# C++

## PRACTICA Nº4


# Informática II
## Fundamentos de Programación
## Curso 2001-2002

Paul Bustamante

*San Sebastián, Febrero 2002*

# ÍNDICE



## 1. Introducción.

Como ya lo hemos venido haciendo, el primer ejercicio de esta práctica debes realizarlo tú solo, con el fin de que puedas ganar más experiencia en la programación en C++.

En algunos ejercicios te daré el código, lo que no significa que sólo tienes que escribirlo en el ordenador. Debes tratar de realizarlos por tu cuenta.

Recuerda borrar los ficheros que estén en los subdirectorios ***debug*** y ***release*** del proyecto, para que liberes espacio en tu disco ***G***\.

### *1.1 Ejercicio 1: Trabajando con complejos.*

En este ejercicio vas a utilizar las ***estructuras***, ya vistas en clase. Para ello haz de crear una estructura ***Complejo*** para ***sumar, restar y multiplicar*** dos números complejos. El resultado del programa debe ser el ***complejo resultante***, ***su módulo y su fase***. Debes utilizar las funciones matemáticas ( ***sqrt()***, ***atan()*** ) que se encuentran definidas en el fichero ***math.h***.

El programa debe pedir al usuario que selecciones de un menú la operación que desea realizar: sumar, restar o multiplicar dos complejos. Puedes usar la función ***getch()*** para capturar la tecla presionada o usar ***cin*** para pedir que elija la opción el usuario.

La estructura debe tener esta forma:

```
struct Complejo{
   float real;
   float imag;
};
```

El algoritmo para multiplicar dos complejos, por si alguien no lo recuerda es:

***(a+bi)*(c+di) = a*c – b*d, a*d+b*c i***

Recuerda que debes crear un proyecto nuevo ***Ejer1*** y en él el fichero ***Complejo.cpp*** en el que escribirás el código.

### *1.2 Ejercicio 2: Reserva dinámica de memoria para vectores.*

En este ejercicio vamos a hacer uso de la reserva dinámica de memoria para almacenar un número determinado de valores (obtenidos de forma aleatoria, entre 0 y 100) y ordenarlos de mayor a menor.

El código para obtener los números y para la reserva dinámica de memoria te la voy a dar, el resto debes tratar de implementarlo tú solo.

Debes crear el proyecto ***Ejer2*** y el fichero ***ordena.cpp*** para introducir el código.

```
// fichero ordena.cpp
// ordena usando memoria dinamica
#include <iostream.h>
#include <stdlib.h>        //para rand()
void main(void)
{
    int Num;      //Numero de datos
    int *datos;      //puntero a int

    cout << "Cuantos numeros deseas generar:";
    cin >> Num;
    //asignacion de memoria
    datos = new int[Num];
    if (datos == NULL) cout << "Error";
    //Llenar el vector
    for (int i=0;i<Num;i++){
        datos[i] = rand() * 100 / RAND_MAX;
    }
    //ordena los datos -> Hacer aquí el algoritmo
    . . .
    . . .
    //imprime los datos ordenados
    for (i=0;i<Num;i++) cout << i << ":" << datos[i] <<endl;
    //liberar memoria
    delete [] datos;
}
```

## 1.3 Ejercicio 3: Introducción a la estadística

En este ejercicio vas a tener una pequeña introducción a la estadística (realmente ya la has tenido y quizá no lo has notado).

El ejercicio consiste en pedir una serie de números al usuario y hallar el máximo, el mínimo y la media aritmética de ellos. Debes crear una variable puntero tipo ***float*** y pedir al usuario que introduzca el número de datos, luego debe introducir todos los datos a calcular.

Recuerda que debes reservar memoria de forma dinámica para almacenar el vector de datos.

Debes crear el proyecto ***Ejer3*** y el fichero ***estadística.cpp*** para escribir el código en él.

La salida del programa debe ser algo así (es un ejemplo, no tienen que coincidir):

```
Numero de datos: 10
Máximo:  25
Minimo: 4
Media Aritmética: 14.6
```

## 1.4 Ejercicio 4: Matrices dinámicas.

Ha llegado el momento de hacer una introducción a la reserva de memoria dinámica para matrices.

Los pasos para hacer la *reserva dinámica de memoria* para matrices son:

a. Crear el puntero a punteros del tipo de datos: ***float **datos;***

b. Reservar memoria para el array de filas: ***datos = new float*[fil];***

c. Hacer un bucle para reservar memoria para ***col*** columnas de cada fila:

```
for (int i=0;i<fil;i++) datos[i] = new float[col];
```

d. Ya tenemos creada la matriz, podemos trabajar con ella con los índices, por medio de los corchetes [].

e. Finalmente, debemos hacer otro bucle para liberar la memoria de cada fila y luego debemos liberar la memoria asignada al vector de filas.

En el siguiente ejemplo verás cómo asignar memoria para una matriz de ***fil$_x$col***:

```
// fichero matriz.cpp
// crear matrices usando memoria dinamica
#include <iostream.h>
#include <string.h>        //para strlen
#include <stdlib.h>        //para atoi()
void main(void)
{
    int fil = 8; //numero de filas
    int col = 5; //numero de columnas
    float **datos;

    datos = new float*[fil];   //vector filas
    //reserva memoria para las columnas de cada fila
    for (int i=0;i<fil;i++)
        datos[i] = new float[col];

    //ya puedes usar la matriz
    for (int f=0;f<fil;f++)
        for (int c=0;c<col;c++) datos[f][c] = (float)rand()/RAND_MAX;

    //imprime los valores
    for ( f=0;f<fil;f++){
        for (int c=0;c<col;c++) {
            cout << f << "," << c << ":"<<datos[f][c]<< " ";
        }
        cout <<endl;
    }

    //liberar memoria
    for (f=0;f<fil;f++) delete datos[f]; //libera filas
    delete [] datos;        //libera vector
}
```

Debes crear un proyecto ***Ejer4*** y el fichero ***matriz.cpp*** para escribir el código.

## *1.5 Ejercicio 5: Producto de matriz por matriz.*

Basándote en el ejercicio anterior, vamos a realizar la ***versión 2.0*** del ejercicio 5 de la práctica Nº3: producto de matrices con asignación dinámica de memoria.

Crea un proyecto llamado ***Ejer5*** que contenga un programa que multiplique dos matrices **MatA** y **MatB** de cualquier tamaño; llámalo ***matxmat2.cpp***.

Pasos a seguir:

1- Debes crear dos punteros a punteros.

2- Pedir por teclado las dimensiones de la matriz **MatA** (filas y columnas).

3- Pedir por teclado sólo las columnas de la matriz **MatB**, ya que las filas tienen que ser igual a las columnas de **MatA** para la multiplicación.

4- Realizar la reserva dinámica de memoria, usando el operador ***new.***

5- Pedir los datos de las dos matrices **MatA** y **MatB**.

6- Realizar la multiplicación, como en el ejercicio de la práctica anterior, en dos bucles ***for*** anidados.

7- Finalmente debes sacar por consola :

   a. Las dimensiones de la matriz resultante.

   b. Los datos de la matriz resultante.

8- Liberar la memoria asignada, utilizando el operador delete.

Recuerda la fórmula para multiplicar dos matrices:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \qquad i = 1...n, j = 1,...,n$$

## 1.6 Ejercicio 6: Reserva dinámica de memoria con frases.

Este sexto y último ejercicio consiste en crear un programa capaz de leer desde el teclado un conjunto de frases, almacenando dichas frases en un "***vector de punteros a caracteres***". Se recuerda que cada letra (***char***) es "tratado como si fuera un número" (una casilla) y por lo tanto almacenar una frase se asemeja a almacenar una fila de una matriz. Se necesita por lo tanto un doble puntero (***char **pfrases***) para poder almacenar un conjunto de frases o palabras. En esta "matriz de letras" el número de columnas puede variar para cada fila (es ***strlen(frase)+1***). Utilizando una única llamada a la función ***strcpy()*** es posible rellenar una fila completa de dicha ***matriz de caracteres***. Para utilizar dicha función es necesario incluir el fichero ***string.h***.

Crea un proyecto llamado ***Ejer6*** y el fichero ***frases.cpp***, donde vas a escribir el siguiente código:

```
// fichero frases.cpp
#include <iostream.h>
#include <string.h>
#include <stdio.h>      //para gets()
void main(void)
{
    char frase[120];
    char** pfrases;
    int NumFrases;

   cout << "Cuantas frases desea almacenar:";
   cin >> NumFrases;

    pfrases = new char*[NumFrases]; //Espacio para NumFrases
   //pedir datos
   for (int i=0;i<NumFrases;i++) {
      cout << "Escriba la frase " << i+1 << ":" << flush;
      gets(frase);

      // +1 porque strlen no guarda para el '\0'
      pfrases[i] = new char[strlen(frase)+1];
      strcpy(pfrases[i], frase);
    }
   //sacar datos por consola
   for (i=0;i<NumFrases;i++)
      cout << "Frase " << i+1 << ":" << pfrases[i] << endl;
   //liberar memoria
   for (i=0;i<NumFrases;i++) delete pfrases[i];
   delete [] pfrases;
}
```