# Programación en C/C++

## (Manual FV)

# ÍNDICE

# Introducción

- **Manual F.V.**
  Significa "manual **práctico** de informática", pero realmente realmente PRÁCTICO.

- En primer lugar deberías decidir en qué carpeta de tu ordenador quieres grabar tus programas. Personalmente me referiré continuamente a la carpeta **C:***TuCarpeta*, es decir, cuando encuentres en el manual *TuCarpeta*, deberás sustuirla por el nombre de tu carpeta de verdad.

- En segundo lugar deberías decidir el compilador de C/C++ que deseas usar.
  La mayoría de programas del libro han sido probados en tres compiladores:
  - Para el entorno MS/DOS: **DJGPP 2.1**
  - Para el entorno Windows 3.x: **Turbo C++ 3.1 de Borland**
  - Para el entorno Windows 95/98: **Visual C++ 6.0 de Microsoft**

  En los apéndices A, B y C encontrarás las instrucciones básicas del funcionamiento de estos tres compiladores

  Si no dispones de ningún compilador de C/C++, bájate de Internet el **DJGPP 2.1**, encontrarás las instrucciones correspondientes en el apéndice B.

  En el caso concreto del "Visual C/C++", hay unos pocos programas que no te funcionarán: consulta el apéndice C.

- Cómo aprovechar al máximo este curso:
  - Escribe los programas, utilizando tu compilador de C/C++
  - Grábalos utilizando el nombre que aparece en la primera línea del programa en *TuCarpeta*.
  - Ejecútalos un par o tres de veces, observando detenidamente el resultado del programa, el programa fuente y las explicaciones que te indico en cada programa
  - Subraya, corrige, tacha y añade todo lo que consideres importante en tu manual (o hazte tu propio manual con tus apuntes personales en una libreta).
  - Es muy importante que hagas los ejercicios de **autoevaluación**, que aparecen al final de cada capítulo. Deberás guiarte por los programas que aparecen en cada tema y su realización demostrará que has asimilado los conceptos y procedimientos del capítulo.

- Por último y antes de empezar:

Debes tener en cuenta que en ningún caso "programaremos" utilizando las características propias del "entorno", quiero decir: aunque utilices un compilador de C en entorno Windows por ejemplo, todos los programas que harás utilizando este manual, en principio funcionan igual en el entorno MS/DOS o UNIX, por citar dos muy conocidos.
Seria materia de otro curso el estudio del "entorno de desarrollo" e "interface gráfico" de tu compilador de C/C++
Un caso aparte es la introducción al "Visual C/C++", del apéndice C, que es conveniente realizar después del capítulo 8 (¡el último!), para conocer las tendencias de la moderna "programación orientada a objetos".

# I.- Introducción a la Programación en C

## a) Introducción Teórica

### Creador:

Dennis Ritchie (Laboratorios Bell) el 1972, cuando trabajaba junto con Ken Thompson en el diseño del sistema operativo UNIX.
El 'C' se creó como herramienta para programadores, en consecuencia su principal objetivo es ser un lenguaje útil.

### Características:

El "C" es un lenguaje de programación de "alto nivel" (alto nivel quiere decir "próximo al lenguaje humano"), pero con características de "bajo nivel" (bajo nivel= próximo al lenguaje máquina).

Es de ALTO NIVEL porque es racional, estructurado y fácil de aprender.

Es de BAJO NIVEL porque permite trabajar con "bits", registros de la C.P.U. y posiciones de memoria.

### ¿Porqué el "C"?

El lenguaje 'C' es **poderoso y flexible**: la mayor parte del sistema operativo UNIX está escrito en 'C'. Incluso están escritos en 'C' los compiladores e intérpretes de otros lenguajes, como FORTRAN, APL, PASCAL, LISP, LOGO y BASIC.

El lenguaje 'C' es **"amistoso"** porque es lo suficientemente **esrtucturado** para ejercer buenos hábitos de programación. Es el lenguaje de programación más utilizado por el programador de sistemas.

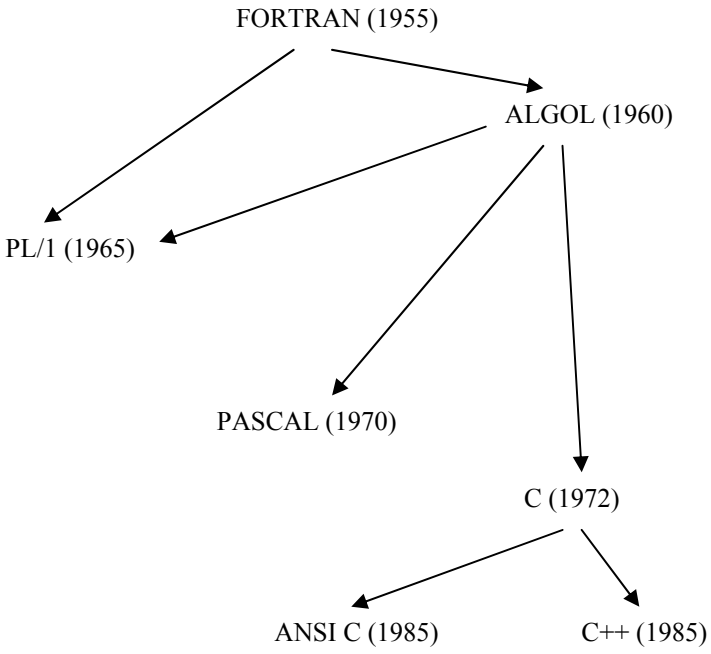
### Estructura de un programa en "C":

El 'C' es un lenguaje compilado, vamos a ver que es esto gráficamente:


CÓDIGO FUENTE: es el programa que nosotros escribimos, se graba con la extensión **CPP**
CÓDIGO OBJETO: es el programa fuente pero traducido a lenguaje máquina (sucesión de ceros y unos), se graba con la extensión **OBJ**
PROGRAMA EJECUTABLE: es el programa objeto más las "librerías del C", se graba con la extensión **EXE.** Y no necesita el programa que hemos utilizado para crearlo, para poder ejecutarlo.

El código Objeto que genera un **compilador** de "C", es casi tan eficiente (rápido) como si lo hubiéramos escrito en lenguaje ENSAMBLADOR (lenguaje de programación más próximo al lenguaje máquina).

## Relaciones del "C" con otros lenguajes de programación:



# b) **Los Primeros Programas en 'C'**

- Prog001.cpp

```
/* Prog001.cpp */

#include <stdio.h>

void main()
{
      printf("Curso de Programación en C");
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog001.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

/* */
Todo lo que escribimos entre los símbolos anteriores son "comentarios" para el programador, que no influyen para nada en la ejecución del programa.

**#include <stdio.h>**
Ordena al compilador que incluya en el programa, las funciones de entrada – salida (en nuestro caso hay una: **printf**).
**stdio.h** no es más que un fichero "biblioteca", que contiene una serie de funciones (instrucciones), en nuestro caso de "entrada-salida por pantalla".
Si en nuestro programa necesitamos una raíz cuadrada por ejemplo, deberíamos incluir (**include**) otra "biblioteca" (tranquilo, ya las iremos estudiando poco a poco).

**void main()**
Función o programa principal (main), que no devuelve nada (void)

{ Inicio

**printf(" ")** Función que escribe en pantalla

} Fin del programa

- Prog002.cpp

```
/* Prog002.cpp */

#include <stdio.h>

void main()
{
printf("\n");     /* Línea en blanco */
```

```
printf("Segunda Linea");
/*********************************
      COMENTARIO
*********************************/
printf(" continuamos en la 2ª línea\n");
printf("\nAhora estamos en la cuarta línea");
/*
           Hasta luego
.................................*/
}
```

/* */ Todo lo que escribimos entre los símbolos anteriores, el compilador no hace caso.

**\n** Es equivalente a un [Return]. Se le llama **manipulador de formato**.

## c) Datos en "C"

- Prog003.cpp

```
/* Prog003.cpp */

#include <stdio.h>

void main()
{
char nom[20];
printf("\nEscribe tu nombre: ");
scanf("%s",nom);
printf("\nHola %s",nom);
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog003.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

**Estudio del PROG003**:

- En el lenguaje "C" hay dos tipos de datos: **Constantes y Variables**
  En un programa, una **constante** es un dato que no puede ser variado durante su ejecución.
  El dato **"Escribe tu nombre:"** del PROG4 es una **constante**, ya que cada vez que ejecutamos el programa, aparece la misma frase: no hay posibilidad de cambiarla.
  En un programa, una **variable** es un dato que sí puede cambiar mientras se ejecuta un programa.
  El dato **"nom"** del PROG4 es una variable, ya que cada vez que ejecutamos el programa adquiere el valor correspondiente a lo que escribimos ante la orden: **"Escribe tu nombre:"**

- La sentencia: **char nom[20];**
  Define (declara) la variable **"nom"**, cuyo contenido serán caracteres (de ahí la palabra **char** que abre la línea), en un número no superior a 20, porque así lo especifica dicha cifra entre corchetes. En otras palabras, la línea en cuestión indica al compilador que, para la ejecución del programa debe reservar espacio en memoria para almacenar 20 datos del tipo **char**, esto es, caracteres, y que en el resto del programa dicha zona de memoria será designada como **"nom"**
  La definición (declaración) de variables en un programa "C" es imprescindible: si no definimos una variable antes de usarla, el programa no funcionará ya que al llegar al nombre de la variable, el programa no sabrá que hacer.

- La sentencia: **printf("\nHola %s ",nom);**
  El **printf** en cuestión, escribe en pantalla dos datos: Una constante **"Hola"** y el valor de la variable **nom**.
  El símbolo **"%s"** es un **indicador de formato**, que sirve para decir al **printf**, cómo queremos que nos muestre el valor de la variable **nom**. La **"s"** del indicador proviene de la palabra **string** ("cadena" en inglés).

  En definitiva: **printf("\nHola %s ", nom);**

  Escribe en pantalla:
  Una línea en blanco, debida a **\n**

  **Hola** (**%s**) → Se substituye por el valor de la variable **nom**

Para que el lenguaje "C" nos escriba en pantalla (printf), el valor de una variable tipo **char**, es necesario utilizar **el indicador de formato: %s**

Variable **char** → (**%s**) ← Será substituido por el valor de la variable

- La sentencia: **scanf("%s", nom);**
  Inmovilizará la ejecución del programa, hasta que nosotros escribamos alguna cosa.
  Lo que escribamos se guardará en la variable de nombre **nom**
  Hemos de introducir el **indicador de formato**: **%s**, correspondiente a datos "tipo carácter" para que lo que escribamos sea interpretado como una cadena de caracteres.
  **Scanf** igual que **printf** es una función del fichero de cabecera: **stdio.h**

## d) **Tipos de Datos**

- Prog004.cpp

```
/* Prog004.cpp */

#include <stdio.h>

void main()
```

```
{
float a,b,c;
printf("\nPrimer sumando: ");
scanf("%f",&a);
printf("\nSegundo sumando: ");
scanf("%f",&b);
c=a+b;
printf("\n\nLa suma de %f y %f es %f",a,b,c);
}
```

- La sentencia: **float a, b, c ;**
  Define (declara) 3 variables (a,b y c) **numéricas** tipo **float** (números reales).

| **Tipos de Variables más importantes:** | |
|---|---|
| **INT** | **número entero** |
| **LONG** | **número entero de doble precisión** |
| **FLOAT** | **número real** |
| **DOUBLE** | **número real de doble precisión** |

  El rango de valores que puede tener cada tipo de variable (número de dígitos) depende del compilador de "C".
  En otro ejercicio veremos la forma de descubrirlo.

- **El indicador %f**
  La utilización de variables **float**, determina que el indicador de formato sea **%f**

| Variable: | **CHAR** | Indicador de formato: | **%s** |
|---|---|---|---|
| | **FLOAT** | | **%f** |
| | **INT** | | **%d** |
| | **DOUBLE** | | **%lf** |

- El símbolo **&**
  Una variable **simple** debe aparecer en el **scanf** precedida del símbolo **&**: scanf("%f",&b)
  En el programa anterior **Prog003**: scanf("%s",nom)
  No estaba precedida de **&**, porque la variable **nom**, no era una variable **simple** (un solo valor), sino **compuesta** (llamada también array o vector), porque tomaba un máximo de **20** valores (caracteres).
  Ya veremos más adelante, exactamente lo que indica &

- La sentencia: **c=a+b**
  Es la forma de asignar el valor de una variable, a partir del valor de otras.

# e) Asignación de variables

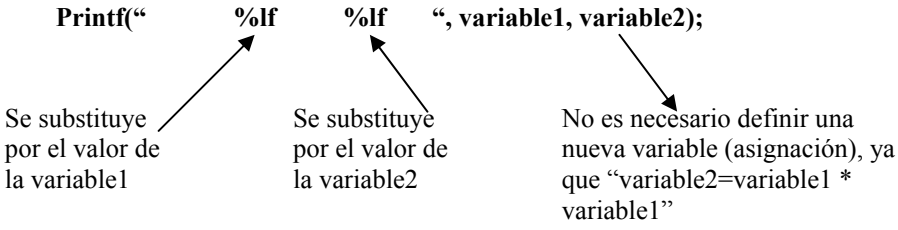
- Prog005.cpp

```
/* Prog005.cpp */

#include <stdio.h>

void main()
{
double num;
printf("\nEscribe un número= ");
scanf("%lf",&num);
printf("\nEl cuadrado de %lf es %lf",num,num*num);
printf("\nEl cubo de %lf es %lf",num,num*num*num);
}
```

Observa la sintáxis completa de **printf**:




- Prog006.cpp

```
/* Prog006.cpp */

#include <stdio.h>

void main()
{
int valor;
valor=15;
printf("Valor= %d",valor);
valor=27;
printf("\nValor= %d",valor);
valor=valor+5;
printf("\nValor= %d",valor);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog006.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Observa cómo podemos cambiar el valor de una variable, durante la ejecución del programa sin ningún tipo de problema.

La sentencia **valor=valor+5;** indica que el **nuevo** valor es igual al antiguo valor más 5 unidades.

# f) Tipo de datos "long"

- Prog007.cpp

```
/* Prog007.cpp */

#include <stdio.h>

void main()
{
int inta,intb;
long longa,longb;
inta=32500;
longa=-2000342527;
printf("El valor de inta es = %d",inta);
printf("\n Escribe un entero negativo menor de 32.768 : ");
scanf("%d",&intb);
printf("\n El valor de intb es = %d",intb);
printf("\n\n");
printf("\n El valor de longa es = %ld",longa);
printf("\n Escribe un entero positivo menor de 2.100.000.000 = ");
scanf("%ld",&longb);
printf("\n El valor de longb es = %ld",longb);
}
```

**long** (abreviatura de **signed long int** = entero largo con signo) corresponde a un número entero que puede ser mayor (o menor) a un **int**.

Observa el formato correspondiente: **%ld**, a diferencia del **%d** que correspondía a **int**.

# g) Rango de Datos

- Prog008.cpp

```
/* Prog008.cpp */


#include <stdio.h>
#include <values.h>

void main()
{
printf("\n Escribe el máximo número entero: %d",MAXINT);
printf("\n Escribe el máximo número entero largo: %ld",MAXLONG);
printf("\n Escribe el mínimo número float: %e",MINFLOAT);
printf("\n Escribe el máximo número float: %e",MAXFLOAT);
```

```
printf("\n Escribe el mínimo número double: %e",MINDOUBLE);
printf("\n Escribe el máximo número double: %e",MAXDOUBLE);
}
```

**MAXINT, MAXLONG, MINFLOAT, MAXFLOAT, MINDOUBLE, MAXDOUBLE** son constantes que están definidas en el fichero **values.h**, por lo tanto para que funcione el programa necesitamos la línea: **#include <values.h>**

El código de formato **%e** indica que queremos visualizar el número en notación exponencial (científica).

La constante **MAXLONG** nos da el máximo número entero (entero largo = long).

Si utilizas el compilador **DJGPP** observarás que **MAXINT = MAXLONG**, es decir en este compilador de 'C', **int** es equivalente a **long**

## h) Cadenas de caracteres: "puts – gets"

- Prog009.cpp

```
/* Prog009.cpp */

#include <stdio.h>

void main()
{
char texto[65];
printf("Escribe tu nombre y dos apellidos: ");
scanf("%s",texto);
printf("\nAcabas de escribir : %s",texto);
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog009.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Observa que el valor de la variable "texto" es todo lo que hemos escrito antes del primer espacio.

- Prog010.cpp

```
/* Prog010.cpp */

#include <stdio.h>

void main()
```

```
{
char texto[65];
puts("Escribe tu nombre y dos apellidos: ");
gets(texto);
puts("Acabas de escribir : ");
puts(texto);
}
```

Observa que ahora sí, escribe todo lo que hemos escrito ante la pregunta.

La función **gets(texto)** es equivalente a **scanf**, pero con dos diferencias fundamentales:
- Sólo funciona para cadenas de caracteres
- Abarca todo el texto que escribamos hasta que pulsamos la tecla [Return]. Por lo tanto es la que nos interesa si hay espacios en blanco.

**puts** es equivalente a **printf**, pero con dos diferencias fundamentales:
- Sólo funciona para cadenas de caracteres.
- No podemos utilizar códigos de formato para "imprimir" variables.

# i) Mejoremos la salida por pantalla

- Prog011.cpp

```
/* Prog011.cpp */

/* Programa que despliega un mensaje de
        bienvenida en la pantalla */

#include<stdio.h>
#include<conio.h>

void main()
{
  clrscr();
  /* La función anterior borra la pantalla. Está en <conio.h>
     solo tiene sentido si utilizas el DJGPP
  */

  printf("\n\n\n\n\n\n"); /* 6 lineas en blanco */
  printf("\t\t");         /* 2 tabuladores */

  printf("BIENVENIDO AL CURSO DE C ESTANDAR");

  printf("\n\n\n\n\n\n\n\n"); printf("\t\t\t\t");

  printf("Pulsa cualquier tecla para terminar...\n");

  getch();
  /* La función anterior inmoviliza la ejecución del programa
     hasta que pulsamos una tecla */
  clrscr();
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog011.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Observa detenidamente los comentarios que aparecen en el programa.

El **"manipulador"** \n determinaba una línea en blanco, es decir es equivalente a pulsar la tecla [Return]. El manipulador **\t** es equivalente a pulsar la tecla de tabulación.

Observa que podemos colocar varias sentencias de C en una misma línea de programa, siempre y cuando separemos cada sentencia con un punto y coma.

- Prog012.cpp

```
/* Prog012.cpp */

/*Programa que calcula el número de dias vividos*/

  #include<stdio.h>
  #include<conio.h>

  void main()
  {
    char nombre[50];
    int edad;
    clrscr();

    printf("\n\n\t\t\t¿Cómo te llamas? ");
    scanf("%s",nombre);
    printf("\n\n\t\t\t¿Cuántos años tienes? ");
    scanf("%d",&edad);

    edad=edad*365;

    printf("\n\n\n\t\t%s, has vivido %d dias",nombre,edad);

    printf("\n\n\n\n\t\t\t\t\tPulsa cualquier tecla para
                                          terminar...\n");
    getch();

  }
```

Supongo que estamos de acuerdo en que es muy fácil mejorar la salida por pantalla del programa.

- Prog013.cpp

```
/* Prog013.cpp */
```

```
/*Programa que saca el promedio de 3 números*/

  #include<stdio.h>
  #include<conio.h>


  void main()
  {
    float numero;
    float promedio=0;

    printf("Dame el primer número: ");
    scanf("%f",&numero);
    promedio+=numero;
    /* La expresión anterior es equivalente a
       promedio=promedio+numero */

    printf("Dame el segundo número: ");
    scanf("%f",&numero);
    promedio+=numero;
    printf("Dame el tercer número: ");
    scanf("%f",&numero);
    promedio+=numero;
    promedio/=3;
    /* La expresión anterior es equivalente a
       promedio=promedio/3 */
    clrscr();printf("\n\n\n\n\n\t\t\t");
    printf("El promedio es %f",promedio);
    printf("\n\n\t");
    printf("Presione cualquier tecla para terminar...\n");
    getch();
  }
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog013.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Toma nota del significado de: **promedio+=numero** o **promedio/=3**
Es decir en la primera expresión: "el nuevo valor de **promedio** es igual al valor antiguo de **promedio** + el valor de **numero**".
En la segunda expresión: "el nuevo valor de **promedio** es igual al antiguo valor de **promedio** partido por 3".

# AUTOEVALUACIÓN 1

1) El siguiente programa tiene errores. Escribelo (grábalo con el nombre **EVAL1A** en *TuCarpeta*) y corrígelo para que funcione:

```
#include <stdio.h>
void main()
{
    float radio;
    pi=3.141592
    printf("Radio=  ");
    scanf("%f",radio);
    printf("\n\nLongitud = %f",2*pi*radio);
    printf("\n\nÁrea = %f ",pi*radio*radio);
}
```

2) Haz un programa que funcione de la siguiente forma:

- El programa nos pregunta en qué población vivimos.
- El programa nos pide que introduzcamos la base de un triángulo.
- El programa nos pide que introduzcamos la altura de un triángulo.
- El programa nos da el resultado del área del triángulo correspondiente.
- El programa nos despide con la frase: **"Adiós habitante de "** y a continuación nos escribe la población que hemos escrito al principio.

- Graba el programa con el nombre **EVAL1B** en *TuCarpeta*.

3) Haz un programa que sirva para calcular un determinante de 2º orden.
Recuerda:

| a b |
| c d | es igual a: a*d-c*b

Graba el programa con el nombre **EVAL1C** en *TuCarpeta*.

4) Haz un programa que sirva para calcular el área y el perímetro de un rectángulo.
(Área= base x altura y Perimetro= 2veces la base + 2veces la altura)

Graba el programa con el nombre **EVAL1D** en *TuCarpeta*.

5) Haz un programa que nos pide nuestro nombre y teléfono. Como resultado el programa nos escribe una ficha aproximadamente igual a la siguiente:

```
================================================

  NOMBRE: (el nombre que hemos introducido)

  TELÉFONO: (el teléfono que hemos introducido)

================================================
```

Graba el programa con el nombre **EVAL1E** en *TuCarpeta*.

6) Haz un programa que nos pida un número entero y a continuación el programa nos escribe los 2 enteros siguientes al que hemos introducido.

Graba el programa con el nombre **EVAL1F**, en *TuCarpeta*.

7) ¿Qué diferencia hay entre un programa con extensión **CPP** o **OBJ**

8) Cita un programa de ordenador muy popular que esté escrito con el lenguaje de programación **"C"**.

9) ¿Qué quiere decir un lenguaje de programación de **ALTO NIVEL** y **BAJO NIVEL**?

10) ¿Porqué el "C" es un lenguaje de programación con características de **ALTO NIVEL** y de **BAJO NIVEL**?

11) Relaciona el "C" con otros lenguajes de programación.

12) Vuelve a hacer el programa **EVAL1E**, pero en lugar del nombre que aparezca el nombre y apellidos

Grábalo con el nombre **EVAL1G** en *TuCarpeta.*

13) Repite el **EVAL1C** pero utilizando \n y \t, de forma que quede bonito.

Grábalo con el nombre **EVAL1H** en *TuCarpeta.*

14) Repite el **EVAL1D** pero que quede "bonito".

Grábalo con el nombre **EVAL1I** en *TuCarpeta.*

15) Idem con el **EVAL1F**

Grábalo con el nombre **EVAL1J** en *TuCarpeta.*

# II.- Estructuras de Programación

## Operadores Lógicos y Relacionales

| | |
|---|---|
| >, >=, <, <= | |
| == | igualdad |
| != | diferente |
| && | y |
| \|\| | o |
| ! | no |

## La Estructura IF – ELSE

- Prog014.cpp

```
/* Prog014.cpp */

#include <stdio.h>

void main()
{
float a, b;
printf("\nEscribe el primer número a= ");
scanf("%f",&a);
printf("\nEscribe el segundo número b= ");
scanf("%f",&b);

if (a==b) printf ("\nLos dos números son iguales\n");

if (a!=b) printf("\nLos dos números son distintos\n");

if (a>b) printf("\nEl número %f es mayor que %f\n ",a,b);
else printf("\nEl número %f no es mayor que %f\n",a,b);

if ((a>b) && (100>a))
      {
      printf("\nEl número %f es mayor que %f ",a,b);
      printf("\nAdemás los dos son menores de 100");
      }
else
      {printf("\nEl número %f no es mayor que %f ",a,b);
      printf(" O uno de los dos números es mayor de 100");
      }
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog014.cpp** en *TuCarpeta*.

- Ejecútalo un par o tres de veces

Ejecútalo en los siguientes casos, observando detenidamente lo que aparece:

1) Si a= 70 y b= 2
2) Si a= 50 y b=30
3) Si a= 7 y b= 11
4) Si a= 100 y b= 50
5) Si a= 50 y b= 100

Estudio del programa **PROG014**:

- Observa el uso de los **operadores relacionales y lógicos:**
  (a == b) "a" igual a "b"
  (a != b) "a" diferente a "b"
  ((a>b) && (100>a))
  "a" es mayor que "b" y además "100 es mayor que "a"

  Observa también que cada relación hemos de encerrarla entre paréntesis.

- **Estructura de programación IF-ELSE**
  Observa en primer lugar la sintáxis diferente entre los 4 "IF" del programa:

  Los 2 primeros son iguales y más simples: **IF (condición) sentencia;**
  Que quiere decir: **Si se cumple la "condición" se ejecutará la sentencia.**

  El tercer "IF":

```
IF (condición) sentencia1 ;
else sentencia2 ;
```

  Que quiere decir: **Si se cumple la (condición) se ejecutará la sentencia1, en caso contrario se ejecutará la sentencia2.**

  El cuarto "IF" es el más completo:

```
IF (condición)
        {
        sentencia1;
        sentencia2;
        }
else
        {
        sentencia3;
        sentencia4;
        }
```

  Que quiere decir: **Si se cumple la "condición" se ejcutará la sentencia1 y la sentencia2, en caso contrario se ejecutará la 3 y la 4.**

## La Estructura WHILE

- Prog015

```
/* Prog015.cpp */
```

```
#include <stdio.h>

void main()
{
int x;
x=0;
while (x<6)
      {
      printf("\nEl valor de x es %d ",x);
      x=x+1;
      }
}
```

- **Estructura de programación WHILE:**

```
While (condición)
      {
      sentencia1;
      sentencia2;
      sentencia3;
      ----------;
      ----------;
      }
```

Que quiere decir: mientras se cumpla la "condición", ejecuta las sentencias que hay entre llaves.

- Observemos nuestro programa:
  1) Definimos una variable entera (int) de nombre: **x**
  2) Inicializamos el valor de la variable **"x"** a **0**
  3)

```
while (x<6)
{
printf("\nEl valor de x es %d ",x);
x=x+1;
}
```

Es decir: **Mientras el valor de la variable "x" sea inferior al número 6**
**Escribe en pantalla el valor de "x"**
**El valor de "x" es igual al anterior valor de "x" pero + 1**

Veamos:
Inicialmente el valor de x=0, por lo tanto se cumple la condición del "While":
En pantalla aparecerá **0**
Y el nuevo valor de "x" será **1**.
Cómo el valor actual de "x" es 1, se cumple la condición:
En pantalla aparecerá **1**
Y el nuevo valor de "x" será 2.
Cómo el valor actual de "x" es 2, se cumple la condición:
En pantalla aparecerá **2**
Y el nuevo valor de "x" será 3.
Cómo el valor actual de "x" es 3, se cumple la condición:
En pantalla aparecerá **3**
Y el nuevo valor de "x" será 4.
Cómo el valor actual de "x" es 4, se cumple la condición:
En pantalla aparecerá **4**
Y el nuevo valor de "x" será 5.
Cómo el valor actual de "x" es 5, se cumple la condición:

En pantalla aparecerá **5**
Y el nuevo valor de "x" será 6.

Cómo el valor actual de "x" es 6, ya no cumplirá la condición. Es decir "saldremos del While" y se acaba nuestro programa.

- El **contador**:

  La sentencia **x=x+1** se llama "contador" y es muy usada en programación como irás viendo.

## Contadores:

- Prog016

```
/* Prog016.cpp */

#include <stdio.h>
void main()
{
int i,j;
i=2;
j=7;
while (i<j)
      {
      printf("\n i= %d --- j= %d ",i,j);
      i++;
      j--;
      }
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog016.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Estudio del **Prog016**:

- **i++**
  Es equivalente a escribir **i=i+1**

- **j—**
  Es equivalente a escribir **j=j-1**

Vamos a hacer un programa que repita 10 veces nuestro nombre. Está claro que la solución al problema es el uso de un "contador" dentro de un "While".

- Prog017

```
/* Prog017.cpp */

#include <stdio.h>

void main()
{
int x;
char nom[20];
x=1;
printf("\nEscribe tu nombre: ");
scanf("%s",nom);
while (x<=10)
      {
      printf("%d - ",x);
      printf("%s \n",nom);
      x++;
      }
}
```

- Prog018

```
/* Prog018.cpp */

#include <stdio.h>

void main()
{
double num,sum,pro;
sum=0;
pro=1;
printf("\nEscribe un número diferente de 0 = ");
scanf("%lf",&num);
while (num != 0)
{
sum=sum+num;
pro=pro*num;
printf("\nNúmero= %lf  Suma= %lf  Producto= %lf ",num,sum,pro);
printf("\nNuevo número (para acabar escribe 0)= ");
scanf("%lf",&num);
}
printf("\n\n\nSuma Total= %lf  Producto Total= %lf",sum,pro);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog018.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Observa:

- **Contador SUMA:**

  **sum = sum + num**

  Es la forma que conseguimos acumular en la suma (variable **sum**) los diferentes valores de **num**.

- **Contador PRODUCTO:**
  **pro = pro * num**
  Es la forma que conseguimos acumular en el producto (variable **pro**) los valores de **num**.
- Observa también que la **suma** hemos de inicializarla por **cero** y el **producto** por **uno**.

## La Estructura de Programación FOR

- Prog019

```
/* Prog019.cpp */

#include <stdio.h>

void main()
{
int x;
for(x=1;x<=10;x=x+1)
{
printf("\nEstructura FOR    ");
printf("Ahora x= %d",x);
}
}
```

Estudio del **PROG019:**

- **La Estructura FOR:**

```
for(valor inicial,valor final, paso)
{
sentencia 1;
sentencia 2;
-------------;
-------------;
}
```

  Que quiere decir: **Repite la ejecución de las sentencias de programa que hay encerradas entre llaves, tantas veces como te indico en "valor inicial, valor final, paso".**

- **for(x=1;x<=10;x=x+1)**
  Las líneas de programa de nuestro "for" se repetiran: desde x=1 hasta x=10 de 1 en 1, es decir 10 veces.
  Si escribiéramos: **for(x=23;x<=76;x=x+4)**, querría decir: repite las líneas de programa desde x=23 hasta x=76 de 4 en 4.
  Podríamos hacerlo más sofisticado: **for(i=-0.23;i>=-67.78;i=i-1.567)**. Que traducido dice: repite las líneas de programa que hay dentro del "for" desde 1=0.23 hasta i=-67.78 de –1.567 en –1.567

- Prog020

```
/* Prog020.cpp */
```

```
#include <stdio.h>

void main()
{
int edad,indice;
char nom[25];
printf("\Escribe tu nombre: ");
scanf("%s",nom);
printf("\nEscribe tu edad: ");
scanf("%d",&edad);
for(indice=1;indice<=edad;indice++)
      printf("\n%s",nom);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog020.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Observa:

- Si es una sola la sentencia a repetir por el "for", no se encierra entre llaves (en este aspecto funciona igual que la estructura IF-ELSE).
- Recuerda que **indice++** es equivalente a **indice=indice+1**
- Podemos introducir en "los límites del for" una variable (en nuestro caso **edad**)

- Prog021

```
/* Prog021.cpp */

#include <stdio.h>

void main()
{
int mult;
float total;
total=0;
for(mult=11;mult<3000;mult=mult+11)
      {
      printf("%d-",mult);
      total=total+mult;
      }
printf("\n\nSuma= %f",total);
}
```

Ejecútalo varias veces hasta descubrir lo que hace el programa.

Vamos a hacer un programa que sirva para calcular el **factorial** de un número. Recuerda que el factorial de un número **x** es 1*2*3*4....*x

- Prog022

```
/* Prog022.cpp */

#include <stdio.h>

void main()
{
int i,num;
double fa;
printf("\nCálculo del factorial del número: ");
scanf("%d",&num);
fa=1;
for(i=1;i<=num;i++) fa=fa*i;
printf("\nEl Factorial de %d es %lf",num,fa);
}
```

Observa la línea que contiene el **for**: como sólo contiene una instrucción, puede escribirse todo el **"ciclo for"** en una sóla línea de programa.

Vamos a hacer un programa que calcule la suma de los 10 primeros múltiplos del número que queramos.

- Prog023

```
/* Prog023.cpp */

#include <stdio.h>

void main()
{
int num,i;
double sum,mult;
printf("\nMúltiplos de qué número: ");
scanf("%d",&num);
sum=0;
for(i=1;i<=10;i++)
      {
      mult=num*i;
      printf("\nMúltiplo= %lf",mult);
      sum=sum+mult;
      printf("\nSuma Parcial= %lf",sum);
      }
printf("\nSuma Total= %lf",sum);
}
```

Vamos a hacer un programa que calcule **la tabla de valores** de una función dada.

- Prog024

```
/* Prog024.cpp */

#include <stdio.h>
```

```
void main()
{
float x1,x2,paso;
float y,i;
printf("\nTabla de valores para la función Y=X*X-5*X+10");
printf("\n\nIntroduce el valor menor de X: ");
scanf("%f",&x1);
printf("\nIntroduce el valor mayor de X: ");
scanf("%f",&x2);
printf("\nIntroduce el incremento del valor de X: ");
scanf("%f",&paso);
for(i=x1;i<=x2;i=i+paso)
      {
      y=i*i-5*i+10;
      printf("\nX= %15f Y= %15f",i,y);
      }
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog024.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

La notación **%15f** indica que la variable correspondiente ocupará **15 espacios** en pantalla.

Al ejecutar el programa anterior, nos podemos encontrar con una serie de problemas, por ejemplo si introducimos en el valor menor de **x** (x1), un valor que sea mayor que el introducido en la variable **x2**, o también puede suceder que en la variable **paso** escribamos un número negativo.

Escribe a partir del programa **Prog024:**

- Prog025

```
/* Prog025.cpp */

#include <stdio.h>

void main()
{
float x1,x2,paso;
float y,i;
printf("\nTabla de valores para la función Y=X*X-5*X+10");
printf("\n\nIntroduce el valor menor de X: ");
scanf("%f",&x1);
printf("\nIntroduce el valor mayor de X: ");
scanf("%f",&x2);
if (x1>x2)
      {
      printf("\nLo siento, vuelve a intentarlo");
      return;
        }
printf("\nIntroduce el incremento del valor de X: ");
scanf("%f",&paso);

if(paso<=0)
      {
```

```
        printf("\nLo siento, vuelve a intentarlo");
        return;
          }

for(i=x1;i<=x2;i=i+paso)
        {
        y=i*i-5*i+10;
        printf("\nX= %15f Y= %15f",i,y);
        }
}
```

Ejecútalo varias veces, probando los casos "conflictivos": x1>x2 o paso=0 o paso negativo.

Está claro que la sentencia **return**, sirve para acabar el programa.

## F) La biblioteca <math.h>

- Prog026

```
/* Prog026.cpp */

#include <stdio.h>
#include <math.h>

void main()
{
double catet1,catet2,hipot;
printf("\nCálculo de la hipotenusa de un T.R.");
printf("\n==================================");
printf("\n\nIntroduce el valor de un cateto: ");
scanf("%lf",&catet1);
printf("\nIntroduce el valor del otro cateto: ");
scanf("%lf",&catet2);
hipot=sqrt(catet1*catet1+catet2*catet2);
printf("\n\nHIPOTENUSA: %lf",hipot);
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog026.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Estudio del **PROG026**:
- En el programa utilizamos la función **sqrt**, que calcula la raíz cuadrada.
- Para poder utilizar la función matemática **sqrt**, necesitamos "incluir" la biblioteca "C", que contiene las funciones matemáticas. En nuestro caso es la biblioteca: **math.h**, por esta razón necesitamos la sentencia **#include <math.h>**

Nos gustaría la posibilidad de calcular 300 hipotenusas. Vamos a mejorar el programa anterior para que sea más cómodo calcularlas.

- Prog027

```
/* Prog027.cpp */

#include <stdio.h>
#include <math.h>

void main()
{
double catet1,catet2,hipot;
char pregunta[1];
pregunta[0]='s';
while ((pregunta[0]=='s')||(pregunta[0]=='S'))
     {
     printf("\nCálculo de la hipotenusa de un T.R.");
     printf("\n==================================");
     printf("\n\nIntroduce el valor de un cateto: ");
     scanf("%lf",&catet1);
     printf("\nIntroduce el valor del otro cateto: ");
     scanf("%lf",&catet2);
     hipot=sqrt(catet1*catet1+catet2*catet2);
     printf("\n\nHIPOTENUSA: %lf",hipot);
     printf("\n\n\nSi quieres calcular otra hipotenusa pulsa");
     printf(" la tecla [s] y a continuación [Return], en caso");
     printf(" contrario pulsa cualquier otra ");
     scanf("%s",pregunta);
     }
}
```

- Prog028

```
/* Prog028.cpp */

#include <stdio.h>
#include <math.h>

/* Programa que calcula el tipo de soluciones */
/* de una ecuación polinómica de 2° grado     */

void main()
{
double a,b,c,discr;
/* a,b,c representan los coeficientes de la */
/* ecuación                                 */
/*                                          */
/* discr= es el discriminante de la ecuación*/
/* discr= b*b - 4*a*c                       */
printf("\nEscribe el coeficiente del término de 2° grado: ");
scanf("%lf",&a);
if(a==0)
     {
     printf("\nEste programa sólo sirve para ecuaciones de 2°
grado");
     return;
     }
```

```
printf("\nEscribe el coeficiente del término de 1r. grado: ");
scanf("%lf",&b);
printf("\nEscribe el termino independiente: ");
scanf("%lf",&c);
discr=b*b-4*a*c;
printf("\nDiscriminante de la ecuación= %lf",discr);
if(discr==0) printf("\n\nSolución Doble");
if(discr<0) printf("\n\nSoluciones Imaginarias");
if(discr>0) printf("\n\nSoluciones Reales y diferentes");
}
```

- Pruébalo varias veces, por ejemplo para:
  a=1, b=1, c=1
  a=0
  a=2, b=-6, c=-20
  a=1, b=4, c=4

- Prog029

```
/* Prog029.cpp */

#include <stdio.h>
#include <math.h>

void main()
{
double A,B,C,x1,x2,Disc;
printf("Resolución de la ecuación Ax^2+Bx+C=0 \n");
printf("\nEscribe los valores de A, B y C\n");
printf("A=");scanf("%lf",&A);
printf("B=");scanf("%lf",&B);
printf("C=");scanf("%lf",&C);

if(A==0)
      {
      printf("\nError!, esto es una ecuación de 1r.
grado");
      return;
      }
else
      {
      Disc=B*B-4*A*C;
      if(Disc>0)
            {
            x1=(-B+sqrt(Disc))/(2.0*A);
            x2=(-B-sqrt(Disc))/(2.0*A);
            printf("\nSolución x1= %lf",x1);
            printf("\nSolución x2= %lf",x2);
            }
      else
      if(Disc==0)
            {
            x1=(-B)/(2.0*A);
            printf("\nSolución Doble= %lf",x1);
            }
```

```
            else printf("\nError: Raices Imaginarias");
            }
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog029.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Observa el **"anidamiento" de if – else**.

- Prog030

```
/*  Prog030.cpp  */

#include <stdio.h>
#include <math.h>

void main()
{
double gra,rad;
double pi=3.141592;
printf("   Grados   Radianes  Seno   Coseno  Tangente");
for(gra=0;gra<=270;gra=gra+15)
{
rad=pi*gra/180;
printf("\n%11lf %11lf %11lf %11lf
%11lf",gra,rad,sin(rad),cos(rad),tan(rad));
}
}
```

Observa:
- **sin(), cos(), tan()** son funciones que contiene la biblioteca **math.h**
- Como puedes observar las fórmulas anteriores funcionan en **radianes** (y **double**).

- Prog031

```
/* Prog031.cpp  */

#include <stdio.h>
#include <math.h>

void main()
{
double num;
char volver[1];
volver[0]='s';
while ((volver[0]=='s')||(volver[0]=='S'))
{
printf("\nEscribe un número: ");
scanf("%lf",&num);
if (num<=0)
```

```
{
printf("\nLos logaritmos de este número no existen");
return;
}
printf("\n\n\nEl Logaritmo Neperiano de %lf es %lf
                                    ",num,log(num));
printf("\nEl Logaritmo Decimal de %lf es %lf
                                   ",num,log10(num));
printf("\n\n¿Quieres volver a comenzar (S/N)? ");
scanf("%s",volver);
}
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog031.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Que quede claro que las funciones **log(), log10()** están en la biblioteca **math.h**. Es decir, cuando necesites utilizarlas deberás incluir en tu programa: **#include <math.h>**

## G) Las sentencias BREAK y CONTINUE

- Prog032

```
/* Prog032.cpp */

#include <stdio.h>

void main()
{
int x;
for(x=5;x<15;x++)
{
if(x==8) break;
printf("\n x= %d",x);
}
printf("\n\n");
for(x=5;x<15;x++)
{
if(x==8) continue;
printf("\n x=%d ",x);
}
}
```

Estudio del **PROG032:**

- La sentencia **break** nos obliga a salir del ciclo **for**. Por esta razón el primer "for" del programa, sólo escribe del 5 hasta el 7.

- La sentencia **continue** salta al final del ciclo **for** y continua ejecutando el ciclo. Por esta razón el segundo ciclo **for** escribe todos los números del 5 al 14 exceptuando el 8.

- Prog033

```
/* Prog033.cpp */

#include <stdio.h>

void main()
{
int i;
char sn[1];
char sn2[1];
printf("2");
for(i=4;i<=1000;i=i+2)
{
printf("\nQuieres continuar (S/N)? ");
scanf("%s",sn);
if ((sn[0]=='n')||(sn[0]=='N')) break;
printf("\nQuieres que me salte el próximo número par (S/N)?
                                                        ");
scanf("%s",sn2);
if ((sn2[0]=='s')||(sn2[0]=='S')) continue;
printf("\n%d",i);
}
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog033.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

## H) La estructura de programación SWITCH

- Prog034

```
/* Prog034.cpp */

#include <stdio.h>
#include <math.h>

void main()
{
int x;
double num;
char sn[1];
```

```
sn[0]='s';
while ((sn[0]=='s')||(sn[0]=='S'))
{
printf("\n          (1) EL TRIPLE");
printf("\n          (2) EL CUADRADO");
printf("\n          (3) LOGARITMO NATURAL");
printf("\n          (4) LOGARITMO DECIMAL");
printf("\n          (5) SENO");
printf("\n          (6) COSENO");
printf("\n\n\n Escribe el número de la opción que desees:
");
scanf("%d",&x);

switch(x)
{
case 1:
printf("\n\nEscribe el número: ");
scanf("%lf",&num);
printf("\nEl triple de %lf es %lf",num,3*num);
break;
case 2:
printf("\n\nEscribe el número: ");
scanf("%lf",&num);
printf("\nEl cuadrado de %lf es %lf",num,num*num);
break;
case 3:
printf("\n\nEscribe el número: ");
scanf("%lf",&num);
printf("\nEl logaritmo neperiano de %lf es
%lf",num,log(num));
break;
case 4:
printf("\n\nEscribe el número: ");
scanf("%lf",&num);
printf("\nEl logaritmo decimal de %lf es
%lf",num,log10(num));
break;
case 5:
printf("\n\nEscribe el número: ");
scanf("%lf",&num);
printf("\nEl seno de %lf es %lf",num,sin(num));
break;
case 6:
printf("\n\nEscribe el número: ");
scanf("%lf",&num);
printf("\nEl coseno de %lf es %lf",num,cos(num));
break;

default:
printf("\n\nEsto no es ninguna opción ");
break;
}
printf("\n\n\nQuieres volver a calcular (S/N)? ");
scanf("%s",sn);
}

}
```

- **Estructura de programación SWITCH:**

```
switch(x)
   {
   case valor1:
   línea de programa1;
   línea de programa2;
   ----------------------;
   ----------------------;
   break;

   case valor2:
   línea de programa3;
   línea de programa4;
   ----------------------;
   ----------------------;
   break;

   default:
   línea de programa1;
   línea de programa2;
   ----------------------;
   ----------------------;
   break;
   }
```

Según el valor que tome la variable **x**, se ejecutarán las líneas de programa del **case** correspondiente. Observa que cada **case** termina con **break**. Si la variable **x** no toma el valor de ningún **case**, se ejecutarán las líneas correspondientes al **"default"**, que termina con el correspondiente **break**.

## I) Otro código de formato: %o

- Prog035

```
/* Prog035.cpp */

/* Programa que convierte un número decimal entero a octal */

  #include<stdio.h>
  #include<conio.h> /* Porque utilizo: getch() */

  void main()
  {
    int numero;
    printf("Numero entero en decimal: ");
    scanf("%d", &numero);
    printf("\n\nSu representacion en octal es %o");
    /* Observa que no es necesario escribir el nombre de la
       variable. Toma nota del código de formato de un
       número en octal: %o  */
```

```
    printf("\nPresione cualquier tecla para terminar...\n");
    getch();
  }
```

Un número entero en decimal quiere decir en base 10, un número en **octal** quiere decir en base 8

Si tubiéramos de pasar un número en base 10 a base 8 "a mano", deberíamos hacer lo siguiente, por ejemplo con el número 85:

85: 8 = 10, resto=5
10:8 = 1, resto = 2

Iríamos dividiendo el número y sus cocientes sucesivos por 8, hasta que no pudiéramos más.
El número correspondiente en base 8, serían los diferentes restos y el último cociente, escritos al revés, en nuestro ejemplo: 85 en base 10 = 125 en base 8. Como puedes comprobar si ejecutas el **Prog035.cpp** y ante la pregunta **"Número entero en decimal:"** escribes **85**

## J) Más funciones de <math.h>

- Prog036

```
/* Prog036.cpp */

/* Uso de las funciones: pow y fabs de <math.h> */

/* Programa que encuentra las raíces de una ecuación
   cuadrada, de acuerdo a las fórmulas:
   (-b±raiz(b^2-4ac))/2a  */

#include <stdio.h>
#include <math.h>
#include <conio.h>

void main()
 {
    float a,b,c,x1,x2;
    printf("Este programa encuentra las raíces  reales\n");
    printf("de la ecuación de la forma\n");
    printf(" ax^2+bx+c=0 \n");
    printf("A continuación, escriba los  coeficientes\n");
    printf("de la ecuación:\n");
    printf("a= ");
    scanf("%f",&a);
    printf("b= ");
    scanf("%f",&b);
    printf("c= ");
    scanf("%f",&c);
    x1=pow(b,2);
    /* La función "double pow(double base, double exp)"
       es una función que se encuentra en math.h, que
         devuelve "base" elevado a "exp". */

    x1=x1-(4*a*c);
    x1=-b+sqrt(fabs(x1));
    /* La función "double fabs(double num)" es una función
       que se encuentra en math.h, que devuelve el valor
```

```
            absoluto de "num" */

         x1=x1/(2*a);
         x2=pow(b,2);
         x2=x2-(4*a*c);
         x2=-b-sqrt(fabs(x2));
         x2=x2/(2*a);
         printf("x1= %f\n",x1);
         printf("x2= %f\n",x2);
         getch();
      }
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog036.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

## K) La función toupper

- Prog037

```
/* Prog037.cpp */

#include <ctype.h>  /* función toupper              */
#include <stdio.h>  /* funciones printf() y scanf() */
#include <math.h>   /* función fabs()               */
#include <conio.h>  /* función getch()              */
main()
{
   char respuesta;
   respuesta='\0';
   while (respuesta!='N')
   {
      float x;
      printf("Dame un número: ");
      scanf("%f",&x);
      x=fabs(x);
      printf("Su valor absoluto es: %f\n",x);
      printf("Presione 'N' para salir...\n");
      respuesta=toupper(getch());
      /* La función "int toupper(int ch)" que está
         en el fichero "ctype.h", devuelve la
         mayúscula de "ch" siempre y cuando
         sea una letra. */
    }
}
```

- Prog038

```
/* Prog038.cpp */

/* Este programa elabora el ticket de entrada y los
   resumenes de recaudación de un espectaculo.
```

```
   El precio del Ticket depende de la edad del espectador
   (niño, joven, adulto o jubilado). */

#include <stdio.h>
#include <conio.h>
#include <ctype.h>

void main()
{
const int precio=800;   /* Precio de la butaca */
int edad,tarifa,butacas,totalPtas;
char opcion,tecla;
totalPtas=0;butacas=0;opcion=' ';

while (opcion != 'F')
      {
      printf("\nOpción (Ticket, Resumen o Fin) ? \n");
      opcion=getche();
      opcion=toupper(opcion);
      /* la función 'toupper' está en <ctype.h> y devuelve
         la letra mayúscula. */

      printf("\n\n");
      if (opcion == 'T')
            {
            tecla=' ';
            while (tecla != 'N')
                  {
                  printf("\n    Edad= ? ");scanf("%d",&edad);
                  butacas=butacas+1;
                  printf("\n=====================================");
                  printf("\n          TICKET DE ENTRADA");
                  if (edad<6)
                  {printf("\n             |  Niño     |");tarifa=0;};
                  if ((edad>=6) && (edad<18))
                  {printf("\n             |  Joven
                                          |");tarifa=precio/2;};
                  if ((edad>=18) && (edad<65))
                  {printf("\n             |  Adulto
                                          |");tarifa=precio;};
                  if (edad>=65)
                  {printf("\n             |  Jubilado
                                          |");tarifa=precio/4;};
                  totalPtas=totalPtas+tarifa;
                  printf("\n\n           Precio = %5d",tarifa);
                  printf("\n\n=====================================");
                  printf("\n\n");
                  printf("Otro Ticket (S/N) ? \n");
                  tecla=toupper(getche());
                  };
            };
      if (opcion=='R')
            {
            printf("\n\n        RESUMEN DE VENTAS");
            printf("\n=====================================");
            printf("\n\n           %d Butacas",butacas);
            printf("\n\n       Total recaudado = %d",totalPtas);
            printf("\n\n");
            }
```

```
      }
}
```

## L) La estructura de programación "do – while"

- Prog039

```
/* Prog039.cpp */

/* El mismo Prog037, pero utilizando la estructura
   do {  } while (  )
   Observa:
   do {
        sentencia 1;
        sentencia 2;
        ------------
     } while(condición);  */

#include <ctype.h>
#include<conio.h>
#include<stdio.h>
#include <math.h>

void main()
{
     char respuesta;

     do{
         float x;
         printf("Dame un número: ");
         scanf("%f",&x);
         x=fabs(x);
         printf("Su valor absoluto es: %f\n",x);
         printf("Continuar...\n");
         respuesta=toupper(getch());
      }while(respuesta!='N');
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog039.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

## M) Cálculo en forma ITERATIVA

- Prog040

```
/* Prog040.cpp */

/* Sucesión de FIBONACCI (forma iterativa)
   se trata de la siguiente: 0,1,1,2,3,5,8,13,...
   Es decir: cada término es igual a la suma de
   los dos anteriores */

#include <stdio.h>

void main()
{
int anterior,ultimo,aux;
anterior=0;ultimo=1;
printf("         0");
printf("\n         1");
while (ultimo<=25000)
      {
      aux=anterior+ultimo;
      anterior=ultimo;
      ultimo=aux;
      if (ultimo>0) printf("\n%10d",ultimo);
      }
}
```

- Prog041

```
/* Prog041.cpp */

/* Factorial de un número sin recursividad (forma iterativa)

   El factorial de 12 es 479.001.600
   El factorial de 13 es 6.227.020.800
   El máximo valor que puede tomar un "long" es 2.147.483.647
   Por lo tanto éste programa sólo puede calcular hasta el
   factorial de 12. */

#include <stdio.h>

void main()
{
int i,num,fin;
long fa;
long pepps;
num=2;
printf("\n\nFactoriales hasta el número: "); scanf("%d",&fin);
if (fin>=13)
     {
     printf("\n No puedo calcular tanto");
     return;
     }
while(num<=fin)
     {
```

```
            fa=1;
            for(i=1;i<=num;i++)
                  {
                  fa=fa*i;

                        }


            printf("\n El factorial de %d es \t\t %ld",num,fa);
            /* Recuerda que el manipulador"\t" fuerza un tabulador */
               num++;
            }

      }
```

## N) Las funciones getche y putchar

- Prog042

```
/* Prog042.cpp */

/* La función 'getche()' lee un caracter. Espera hasta que se
   pulsa una tecla y entonces devuelve su valor. El eco de la
   tecla pulsada aparece automáticamente en la pantalla.
   La función 'putchar()' imprime un carácter en pantalla.
   Estas dos funciones están en <conio.h> */

#include <stdio.h>
#include <conio.h>

void main()
{
char c;
printf("Escribe una letra: \n");
c=getche();
if (c=='A') printf("\nHas escrito una 'A'");
else
printf("\nNo has escrito una 'A'");
printf("\nHas escrito: ");
putchar(c);
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog042.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

- Prog043

```
/* Prog043.cpp */
```

```
/* Se trata de investigar si el uso de 'getche'nos sirve
   para analizar cada una de las letras que escribimos
   en una frase.  */


#include <stdio.h>
#include <conio.h>

void main()
{
char frase[75];
char c;
int i;
printf("\n\nEscribe una frase máx 74 caracteres ");
printf("y pulsa un '.' para acabar\n\n");
for(i=0;i<=74;i++)
      {
      frase[i]=getche();
      if(frase[i]=='.') break;
      }
printf("\n\n\nLa frase que has escrito es: ");
printf("\n\n");
for(i=0;i<=74;i++)
      {
      if (frase[i]=='.') break;
        else
            putchar(frase[i]);
      };
printf("%c",frase[i]);
}
```

- Prog044

```
/* Prog044.cpp */

/* Al escribir un texto que contiene "algo" entre parentesis
   me interesa que el programa dé como resultado el texto
   original pero sin el texto entre paréntesis */

#include <stdio.h>
#include <conio.h>

void main()
{
char frase[75],frasefin[75];
char c;
int i,j;
printf("\n\nEscribe una frase máx 74 caracteres ");
printf("y pulsa un '.' para acabar\n\n");
for(i=0;i<=74;i++)
      {
      frase[i]=getche();
      if(frase[i]=='.') break;
      }
printf("\n\n");

/* En los espacios en blanco escribo el carácter 'ç' */
```

```
for(i=0;i<=74;i++)
      if (frase[i]==' ')
            {
            frase[i]='ç';
            };

/* Todo lo que hay entre un '(' y un ')' escribo caracteres
   en blanco. */
for(i=0;i<=74;i++)
      {
      if (frase[i]=='.') break;
      if (frase[i]=='(')
            {
            for(j=i;j<=74;j++)
                  {
                  if (frase[j] != ')') frase[j]=' ';
                  else
                      {
                      frase[j]=' ';
                            break;
                      };
                  };
            };
      };


/* Elimino todos los espacios en blanco */
j=0;
for(i=0;i<=74;i++)
      {
      if (frase[i] != ' ')
            {
            frasefin[j]=frase[i];
                j=j+1;
            };
      };


/* Los caracteres 'ç' vuelvo a transformarlos en espacios en blanco */
for(i=0;i<=74;i++)
      {
      if (frasefin[i]=='ç') frasefin[i]=' ';
      };

/* Escribo la cadena resultante. */
printf("\n");
for(i=0;i<=74;i++)
      {
      if (frasefin[i]=='.') break;
        else
            putchar(frasefin[i]);
      };
printf("%c",frasefin[i]);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog044.cpp** en *TuCarpeta*.

- Ejecútalo un par o tres de veces

- Prog045

```
/* Prog045.cpp */

/* El programa anterior pero en el caso de paréntesis
   anidados. Es decir si escribimos un texto entre
   paréntesis, y en su interior hay otro texto
   entre paréntesis */

#include <stdio.h>
#include <conio.h>

void main()
{
char frase[75],frasefin[75];
char c;
int i,j;
printf("\n\nEscribe una frase máx 74 caracteres ");
printf("y pulsa un '.' para acabar\n\n");
for(i=0;i<=74;i++)
      {
      frase[i]=getche();
      if(frase[i]=='.') break;
      }
printf("\n\n");

/* En los espacios en blanco escribo el carácter 'ç' */
for(i=0;i<=74;i++)
      if (frase[i]==' ')
            {
            frase[i]='ç';
            };

/* Todo lo que hay entre un '(' y un ')' escribo caracteres
   en blanco. */
for(i=0;i<=74;i++)
      {
      if (frase[i]=='.') break;
      if (frase[i]=='(')
            {
            for(j=i;j<=74;j++)
                  {
                  if (frase[j] != ')') frase[j]=' ';
                  else
                      {
                      frase[j]=' ';
                            break;
                      };
                  };
            };
      };
```

```
/* Todo lo que hay entre un ' ' y un ')' escribo caracteres
   en blanco. */
for(i=0;i<=74;i++)
      {
      if (frase[i]=='.') break;
      if (frase[i]==')')

            {

            for(j=i;j>=0;j=j-1)
                  {

                  if (frase[j] != ' ') frase[j]=' ';
                  else
                      {
                      break;
                      };
                  };
            };
      };



/* Elimino todos los espacios en blanco */
j=0;
for(i=0;i<=74;i++)
      {
      if (frase[i] != ' ')
            {
            frasefin[j]=frase[i];
                j=j+1;
            };
      };


/* Los caracteres 'ç' vuelvo a transformarlos en espacios en blanco */
for(i=0;i<=74;i++)
      {
      if (frasefin[i]=='ç') frasefin[i]=' ';
      };

/* Escribo la cadena resultante. */
printf("\n");
for(i=0;i<=74;i++)
      {
      if (frasefin[i]=='.') break;
        else
            putchar(frasefin[i]);
      };
printf("%c",frasefin[i]);
}
```

- Prog046

```
/* Prog046.cpp */

/*********************************************************
```

```
   Programa que analiza un texto terminado en un punto y
   contabiliza los siguientes aspectos:
   número de caracteres.
   número de vocales
   número de 'a' o 'A'
   número de 'e' o 'E'
   número de 'i' o 'I'
   número de 'o' o 'O'
   número de 'u' o 'U'
*******************************************************/

#include <stdio.h>
#include <conio.h>
#include <ctype.h>

void main()
{
char c;
int n,v,a,e,i,o,u;
printf("\nEscribe un texto, y acaba con un punto\n\n");
c=' ';n=0;v=0;a=0;e=0;i=0;o=0;u=0;
while(c != '.')
 {
 c=toupper(getche());
 if (c=='A')
       {
       v=v+1;
       a=a+1;
       }
 if (c=='E')
       {
       v=v+1;
       e=e+1;
       }
 if (c=='I')
       {
       v=v+1;
       i=i+1;
       }
 if (c=='O')
       {
       v=v+1;
       o=o+1;
       }
 if (c=='U')
       {
       v=v+1;
       u=u+1;
       }
 n=n+1;
 }
printf("\n Caracteres = %d      Vocales = %d",n,v);
printf("\nA = %d  E = %d  I = %d  O = %d  U = %d ",a,e,i,o,u);
}
```

## O) Variable auxiliar

- Prog047

```
/* Prog047.cpp */

/*************************************************************
   Programa que lee la longitud de los 3 lados de un triángulo
   y analiza qué tipo de triángulo es: no es triángulo,
   equilátero, isósceles, escaleno, rectángulo.
*************************************************************/
#include <stdio.h>

void main()
{
float lado1,lado2,lado3,auxiliar;
printf("\nPrimer lado : ");scanf("%f",&lado1);
printf("\nSegundo lado : ");scanf("%f",&lado2);
printf("\nTercer lado : ");scanf("%f",&lado3);

/* Ordenación de los tres valores
toma nota del uso de una variable auxiliar */
if (lado1>lado2)
      {
      auxiliar=lado1;
      lado1=lado2;
      lado2=auxiliar;
      }
if (lado3<lado1)
      {
      auxiliar=lado3;
      lado3=lado2;
      lado2=lado1;
      lado1=auxiliar;
      };
if (lado3<lado2)
      {
      auxiliar=lado2;
      lado2=lado3;
      lado3=auxiliar;
      }


/* Comprobemos la ordenación: */
printf("\n lado 1: %f",lado1);
printf("\n lado 2: %f",lado2);
printf("\n lado 3: %f",lado3);

/* Clasificación del triángulo */
if (lado3>=lado1+lado2) printf("\nEsto no es un triángulo");
else if ((lado1==lado2) && (lado2==lado3)) printf("\nTriángulo
Equilátero");
else if ((lado1==lado2)||(lado1==lado3)||(lado2==lado3))
printf("\nTriángulo Isósceles");
else printf("\nTriángulo Escaleno");
if ((lado3<lado1+lado2)&&(lado3*lado3==lado1*lado1+lado2*lado2))
printf("\nY además RECTANGULO");
```

```
}
```

En el **Prog040** ya utilizábamos una **variable auxiliar**.

## P) Dibujando con 'C'

- Prog048

```
/* Prog048.cpp */

/***********************************************
            Programa: Triángulo.

   Este programa imprime el borde de un triángulo
   usando asteriscos.
***********************************************/
#include <stdio.h>

void main()
{
const int n=7;    /* Altura del triángulo */
int j,k;    /* Contadores */

for (k=1;k<=n-1;k++) printf(" ");
printf("*");
printf("\n");
for (k=2;k<=n-1;k++)
      {
      for (j=1;j<=n-k;j++) printf(" ");
      printf("*");
      for (j=1;j<=2*k-3;j++) printf(" ");
      printf("*");
      printf("\n");
      }
printf("*");
for (k=1;k<=n-1;k++)
      {
      printf(" *");
      };
}
```

- Prog049

```
/* Prog049.cpp */

/***********************************************
        Programa: Triángulo Bis
   Este programa imprime el borde de un triángulo
   usando asteriscos. La altura del triángulo,
   en lineas de escritura se lee como dato.
***********************************************/
```

```
#include <stdio.h>

void main()
{
int n,j,k;
printf("Altura: ");scanf("%d",&n);
printf("\n");
if (n>0)
      {
      for (k=1;k<=n-1;k++) printf(" ");
      printf("*");
      printf("\n");
      }
for (k=2;k<=n-1;k++)
      {
      for (j=1;j<=n-k;j++) printf(" ");
      printf("*");
      for (j=1;j<=2*k-3;j++) printf(" ");
      printf("*");
      printf("\n");
      }
if (n>1)
      {
      printf("*");
      for (k=1;k<=n-1;k++)
            {
            printf(" ");
            printf("*");
            }
      printf("\n");
      };
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog049.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

- Prog050

```
/* Prog050.cpp */

/************************************************
        Programa: Rombo de asteriscos
   Dibuja un rombo simétrico de asteriscos.
   Tomando como dato
   el número de asteriscos que tiene el lado.
************************************************/

#include <stdio.h>

void main()
{
int fila,j,n;
```

```
printf("\nLado? ");scanf("%d",&n);
printf("\n\n");
for (fila=1;fila<=n;fila++)
   {
   for (j=1;j<=n-fila;j++) printf(" ");
   for (j=1;j<=fila;j++) printf("* ");
   printf("\n");
   }
   ;
for (fila=1;fila<=n-1;fila++)
   {
   for (j=1;j<=fila;j++) printf(" ");
   for (j=1;j<=n-fila;j++) printf("* ");
   printf("\n");
   };
}
```

# AUTOEVALUACIÓN 2

1) El siguiente programa tiene errores. Escríbelo (grábalo con el nombre **EVAL2A** en *TuCarpeta*) y corrígelo para que funcione:

```
/* eval2a */
#include <stdio.h>

/* Conversión de Temperaturas
void main()
{
int cuenta;
int farenheit;
int celsius;
printf("Temperaturas Farenheit y Celsius /n\n");
for(cuenta=-2;cuenta<=12;cuenta=cuenta+1)
        {
        celsius=10*cuenta;
        farenheit=32+(celsius*9)/5;
        printf("  C= %4d    F= %4f",celsius,farenheit);
        if(celsius==0) printf("     Punto de congelación del AGUA");
        if(celsius==100) printf("   Punto de ebullición del AGUA");
        printf("\n");
        }
}
```

2) Haz un programa que funcione de la siguiente manera:

- El programa nos pide que escribamos dos números positivos menores de 57
- El programa nos da como resultado el producto de los dos números.

- Si los números no son positivos o son mayores o iguales a 57, el programa nos lo dice y se acaba la ejecución del mismo.
- El programa nos pregunta al final si queremos volver a empezar.

Graba el programa con el nombre **EVAL2B** en *TuCarpeta*.

3) Escribe un programa que nos vaya pidiendo números. Si escribimos el número **9999** se acaba; por último el programa nos da como resultado el número de números introducidos, exceptuando el **9999**.

Graba el programa con el nombre **EVAL2C**

4) Haz un programa que haga lo mismo que el anterior, pero además nos dé la suma de todos los números introducidos, exceptuando el 9999.

Graba el programa con el nombre **EVAL2D**.

5) Haz un programa que haga lo mismo que el anterior, pero además que nos dé el producto de los números introducidos, exceptuando el 9999.

Graba el programa con el nombre **EVAL2E**.

6) Haz un programa que escriba todos los múltiplos de 23 inferiores a 1000 y por último nos dé la suma de todos ellos.

Graba el programa con el nombre **EVAL2F**, en *TuCarpeta*.

7) Haz un programa que sirva para hacer una tabla de valores de la función: Y=sen(7X-5)
- El programa nos pide los dos valores de "x" (valores máximo y mínimo)
- El programa nos pide el incremento (variación) de la **X**.

Graba el programa con el nombre **EVAL2G**, en *TuCarpeta*.

8) Haz un programa que sirva para calcular un cateto de un triángulo rectángulo a partir del otro cateto y la hipotenusa, de la siguiente forma:
- El programa nos pide el valor de la hipotenusa.
- El programa nos pide el valor de un cateto.
- Si el cateto es mayor que la hipotenusa, el programa nos da un mensaje de error y se acaba
- El programa nos da como resultado el valor del otro cateto y nos pregunta si queremos volver a empezar.

Graba el programa con el nombre **EVAL2H**, en *TuCarpeta*.

9) Haz un programa que sirva para resolver ecuaciones de 2º grado del tipo $ax^2 + bx = 0$

Graba el programa con el nombre **EVAL2I**

10) Haz un programa que sirva para resolver sistemas de ecuaciones del tipo:
    ax + by = c
    a'x + b'y=c'

Graba el programa con el nombre **EVAL2J**

11) Haz un programa con la posibilidad de hacer el **EVAL2I** o el **EVAL2J** (debes utilizar la estructura SWITCH)

    Graba el programa con el nombre **EVAL2K**, en *TuCarpeta*.

12) Haz un programa que escriba la tabla de valores de la función $y=ax^2+bx+c$, el programa nos pide los valores de a, b, c entre los valores "-v" y "v" (el programa nos pide el valor del numero natural "v").

    Graba el programa con el nombre **EVAL2L**, en *TuCarpeta*.

13) Haz un programa que escriba los 15 primeros múltiplos de 7, su suma y su producto. El programa ha de tener la posibilidad de volver a empezar.

    Graba el programa con el nombre **EVAL2M**, en *TuCarpeta*.

14) El siguiente programa tiene errores, indícalos y explica detalladamente lo que hace el programa:

```
#include <stdio.h>
void main()
{
int I; I=0;
while(I<5);
  {
  printf("\nN=%f",I);
  I++;
  }
}
```

    Graba el programa con el nombre **EVAL2N**, en *TuCarpeta*.

15) Haz un programa que sirva para calcular el área de un triángulo **o** el área de un rectángulo **o** el área de un círculo. El programa ha de tener la posibilidad de volver a empezar.

    Graba el programa con el nombre **EVAL2O**, en *TuCarpeta*.

16) Haz un programa tal que: dados 2 vectores del espacio, calcule su producto escalar, producto vectorial y además nos dé el módulo de los dos vectores y también el módulo del producto vectorial.

    Graba el programa con el nombre **EVAL2P**, en *TuCarpeta*.

17) Haz un programa que "dibuje" un rectangulo de asteriscos a partir de la base y la altura.

    Graba el programa con el nombre **EVAL2Q**, en *TuCarpeta*.

18) Haz un programa que dibuje un cuadrado, con el carácter que quieras, a partir del lado.

Graba el programa con el nombre **EVAL2R**, en *TuCarpeta*.

19) Haz un programa que "analice" las vocales, consonantes y espacios en blanco de un texto.

Graba el programa con el nombre **EVAL2S**, en *TuCarpeta*.

20) Haz un programa que nos pida un número y dé como resultado la tabla de multiplicar del número introducido.

Graba el programa con el nombre **EVAL2T**, en *TuCarpeta*.

21) Haz un programa que calcule el número "e" mediante el desarrollo en serie: e = 1 + 1/(1!) + 1/(2!) + 1/(3!) + 1/(4!) + ...

Graba el programa con el nombre **EVAL2U**, en *TuCarpeta*.

22) Haz un programa que calcule la anualidad, a partir del capital, el tanto por ciento anual, y los años de amortización de un crédito:

$$\text{Anualidad} = \frac{\text{Cap}*(1+i/100)^{\text{años}} * i/100}{(1+i/100)^{\text{años}} - 1}$$

El programa también debe calcular para todos los años, la parte de la anualidad dedicada al pago de intereses y la parte dedicada a la amortización de la deuda.

Graba el programa con el nombre **EVAL2V**, en *TuCarpeta*.

# III.- Funciones

## Matrices. Vectores. Arrays.

- Prog051

```
/* Prog051.cpp */

#include <stdio.h>

void main()
{
int i;
char nom[6];
printf("\nEscribe una palabra de exactamente 6 letras: ");
scanf("%s",nom);
for(i=0;i<6;i++)
{
printf("\nnom[%d]= ",i);
printf("%c",nom[i]);
}
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog051.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Estudio del **PROG051**
- La variable **nom**, es una variable compuesta (por esta razón en el **scanf** no hemos de poner el símbolo **&** antes del nombre de la variable).
- A este tipo de variables se les llama también vectores, matrices o arrays, porque una sóla variable (nom), contiene diferentes valores (en nuestro caso 6 caracteres).
- Observa la diferencia con los **identificadores de formato** (string – char)
  %s : si queremos mostrar la palabra (todos los valores de la matriz).
  %c : si queremos mostrar uno de los valores de la matriz.
- Observa también la forma de acceder a **uno** de los valores de la matriz: **nom[2]** indica el 3r. valor, ya que empieza por 0 (el llamado índice de la matriz).
  Es decir:
  char nom[6]: define una matriz de 6 caracteres.
  nom[0]: indica el 1º
  nom[1]: indica el 2º
  ----------
  nom[5]: indica el último (6º).

El array del programa anterior era una matriz de caracteres, veamos que también podemos construir "arrays" de números:

- Prog052

```
/* Prog052.cpp */

#include <stdio.h>

void main()
{
int valors[12];
int i;
for(i=0;i<12;i++) valors[i]=2*(i+4);
for(i=0;i<12;i++)
printf("\n El valor de valors[%d] es %d",i,valors[i]);
}
```

Estudio del **PROG052**:

- **int valors[12]**
  Definimos una variable (matriz, vector o array) de 12 valores enteros.
- Asignamos valores a la matriz anterior:
  i = 0 : valors[0] = 2*(0+4) = 8
  i = 1 : valors[1] = 2*(1+4) = 10
  ……
  ……
  i = 11 : valors[11] = 2*(11+4) = 30
- Observa que en una matriz entera (int) no hay problema en cuanto al identificador: siempre es **%d**.

- Prog053

```
/* Prog053.cpp */

#include <stdio.h>

void main()
{
float vector1[3],vector2[3];
int i;
float proesc;

/* Introducción del primer vector */
printf("\n Introduce las componentes del primer vector\n");
scanf("%f %f %f",&vector1[0],&vector1[1],&vector1[2]);

/* Introducción del segundo vector */
printf("\n Introduce las componentes del segundo vector\n");
scanf("%f %f %f",&vector2[0],&vector2[1],&vector2[2]);

printf("\ Vector 1= ( %f, %f, %f )",vector1[0],vector1[1],vector1[2]);
printf("\ Vector 2= ( %f, %f, %f )",vector2[0],vector2[1],vector2[2]);

/* Cálculo del producto escalar */
for(i=0;i<3;i++) proesc=proesc+vector1[i]*vector2[i];

/* Resultado */
```

```
printf("\n\n                El producto escalar es = %f",proesc);
}
```

La utilidad de los arrays está clara: observa de que forma más fácil calculamos **el producto escalar**.

- Prog054

```
/* Prog054.cpp */

#include <stdio.h>

void main()
{
float f1[3],f2[3],f3[3];
int i;
float det;

/* Introducción de la primera fila del determinante */
printf("\n Introduce los elementos de la primera fila\n");
scanf("%f %f %f",&f1[0],&f1[1],&f1[2]);

/* Introducción de la segunda fila del determinante */
printf("\n Introduce los elementos de la segunda fila\n");
scanf("%f %f %f",&f2[0],&f2[1],&f2[2]);

/* Introducción de la tercera fila del determinante */
printf("\n Introduce los elementos de la tercera fila\n");
scanf("%f %f %f",&f3[0],&f3[1],&f3[2]);

printf("\n          | %11f   %11f   %11f |",f1[0],f1[1],f1[2]);
printf("\n          | %11f   %11f   %11f |",f2[0],f2[1],f2[2]);
printf("\n          | %11f   %11f   %11f |",f3[0],f3[1],f3[2]);

/* Cálculo del determinante */
det=f1[0]*f2[1]*f3[2]+f2[0]*f3[1]*f1[2]+f1[1]*f2[2]*f3[0]-
f1[2]*f2[1]*f3[0]-f2[0]*f1[1]*f3[2]-f3[1]*f2[2]*f1[0];

/* Resultado */
printf("\n\n                El Determinante es = %f",det);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog054.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

# Funciones

- Prog055

```
/* Prog055.cpp */

#include <stdio.h>
#include <math.h>

void mediageo();

void main()
{
printf("\n Programa Principal que llama a una ");
printf("\n\n función de nombre mediageo");
mediageo();
printf("\n\n\n Se acabó lo que se daba");
}

void mediageo()
{
float a,b;
printf("\n Introduce los dos números: \n");
scanf("%f %f",&a,&b);
printf("\n\n La Media Geométrica de %f y %f es %f",a,b,sqrt(a*b));
}
```

Estudio del **PROG055:**

- La sentencia **void mediageo();** indica al compilador, que el programa contiene una función definida por nosotros de nombre **mediageo** (llamado también subrutina o subprograma).
- El programa principal (void main()), llama al subprograma en la línea: **mediageo();**
- Observa de qué forma construimos el subprograma: **void mediageo()** sin punto y coma, y a continuación, entre llaves, las líneas o sentencias del programa.
- Observa también que cuando se acaba la ejecución del subprograma, el control pasa al programa principal, en la línea siguiente a la llamada del subprograma. En nuestro caso se ejecuta la línea: **printf("\n\n\n Se acabó lo que se daba");**

- Prog056

```
/* Prog056.cpp */

#include <stdio.h>
#include <math.h>

void suma();
void raiz();
void logaritmo();
void ayuda();


void main()
{
char c[1];
```

```
c[1]='x';
while(c[0] != 'T')
      {
        printf("\n ================================");
      printf("\n Pulsa S, para SUMAR");
      printf("\n Pulsa R, para RAIZ CUADRADA");
      printf("\n Pulsa L, para LOGARITMO");
      printf("\n Pulsa A, para AYUDA");
      printf("\n Pulsa T, para TERMINAR \n");
      scanf("%s",c);
        if (c[0]=='S') suma();
      if (c[0]=='R') raiz();
      if (c[0]=='L') logaritmo();
      if (c[0]=='A') ayuda();
      }
}


void suma()
{
float a,b;
printf("\n Sumandos: \n");
scanf("%f %f",&a,&b);
printf("\n %f + %f = %f",a,b,a+b);
}

void raiz()
{
float x;
printf("\n Radicando: \n");
scanf("%f",&x);
printf("\n Raiz cuadrada de %f = %f",x,sqrt(x));
}

void logaritmo()
{
float l;
printf("\n Logaritmo de : \n");
scanf("%f",&l);
printf("\n El logaritmo neperiano de %f es %f",l,log(l));
}

void ayuda()
{
printf("\n Es bastante tonto que me pidas ayuda,");
printf("\n pero aquí la tienes:");
printf("\n\n\n Pulsa S, para SUMAR");
printf("\n Pulsa R, para RAIZ CUADRADA");
printf("\n Pulsa L, para LOGARITMO");
printf("\n Pulsa A, para AYUDA");
printf("\n Pulsa T, para TERMINAR \n");
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog056.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

# Funciones que retornan parámetros

- Prog057

```
/* Prog057.cpp */

#include <stdio.h>

int cuenta(char nom[25]);

void main()
{
int longit;
char palabra[25];
printf("\n Escribe una palabra de 25 caracteres como máximo: ");
scanf("%s",palabra);
longit=cuenta(palabra);
printf("\n\n Longitud de %s = %d",palabra,longit);
}

int cuenta(char nom[25])
{
int i;
i=0;
while(nom[i] != '\0') i++;
return i;
}
```

Estudio del **PROG057**:

- La línea de programa **int cuenta(char nom[25]);** indica al compilador que además del programa principal (void main), hay una función de nombre **cuenta()**, que utilizará una variable tipo **string** con un máximo de 25 caracteres y además dicho subprograma retornará un valor tipo **int**, a diferencia de las funciones anteriores que no retornaban nada (void).
- La línea de programa: **longit= cuenta(palabra);**, llama al subprograma **cuenta()** y en número entero que **retorna**, se asigna a la variable entera **longit**
- Observa de qué forma contamos los caracteres ocupados por la variable **nom**: nom[i] != '\0'
- Observa que la variable utilizada por la función **cuenta()**, no tiene porqué ser la misma: En el programa principal es **palabra** y en la función es **nom**. De todas formas es imprescindible que sea del mismo tipo: en los dos casos es **char de 25 caracteres**.
- El valor retornado por la función, no tiene porqué corresponder a la misma variable: en el programa principal es **longit** y en la función es **i**. De todas formas es imprescindible que sean del mismo tipo: en los dos casos es **int**.

- Prog058

```
/* Prog058.cpp */

#include <stdio.h>
#include <math.h>

void trigon(double inicial,int num,double paso);

void main()
{
```

```
int num;
double ini,paso;
printf("\n Valor inicial,número de valores,paso ?");
scanf("%lf %d %lf",&ini,&num,&paso);
trigon(ini,num,paso);
}

void trigon(double inicial,int num,double paso)
{
int i;
double v,s,c;
for(i=0;i<num;i++)
      {
      v=inicial+i*paso;
      s=sin(v);
      c=cos(v);
      printf("\n%lf %lf %lf",v,s,c);
      }
}
```

- Estudio del **PROG058**:
  - La línea: **void trigon(double inicial, int num, double paso);** indica al compilador que el programa contiene una función (trigon), que no retorna ningún valor (void), pero que trabaja con tres variables: la primera y última **double** y la 2ª **int**.
  - La llamada a la función por parte del programa principal (main), se encuentra en la línea: **trigon(ini, num, paso);**
    En efecto: **ini** y **paso** son variables **double** y **num** es **int**.
  - En la función **trigon**, utilizamos los valores entrantes: ini, num y paso y definimos otros: **i, v, s, c**.

Vamos a "programar" otra función con retorno de parámetro:

- Prog059

```
/* Prog059.cpp */

#include <stdio.h>

int diasmes(int mes);

void main()
{
printf("\n Enero      tiene %d días",diasmes(1));
printf("\n Febrero    tiene %d días",diasmes(2));
printf("\n Marzo      tiene %d días",diasmes(3));
printf("\n Abril      tiene %d días",diasmes(4));
printf("\n Mayo       tiene %d días",diasmes(5));
printf("\n Junio      tiene %d días",diasmes(6));
printf("\n Julio      tiene %d días",diasmes(7));
printf("\n Agosto     tiene %d días",diasmes(8));
printf("\n Septiembre tiene %d días",diasmes(9));
printf("\n Octubre    tiene %d días",diasmes(10));
printf("\n Noviembre  tiene %d días",diasmes(11));
printf("\n Diciembre  tiene %d días",diasmes(12));
printf("\n El mes 13  tiene %d días, porque no existe",diasmes(13));
}
```

```
int diasmes(int mes)
{
int dias;
switch(mes)
      {
      case 2:
            dias=28;
            break;
      case 1:
      case 3:
      case 5:
      case 7:
      case 8:
      case 10:
      case 12:
            dias=31;
            break;
      case 4:
      case 6:
      case 9:
      case 11:
            dias=30;
            break;
      default:
            dias=0;
            break;
      }
return dias;
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog059.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Estudio del **PROG059:**

- **int diames( int mes);**
  Indicamos al compilador que nuestro programa contiene una función que necesita un valor **int** (mes) y devolverá un valor **int**.
- El programa principal, con la sentencia **diames(1)** llama a la función **diames** y ésta devuelve el número **31**.
- Observa la estructura **switch**: si el case 1, 3, 5, 7, 8, 10 no contienen nada, se ejecuta el siguiente **case** que sí contiene algo: en nuestro caso el **case 12**, que da a la variable **dias** el valor **31**.

## El operador "resto de la división entera"

- Prog060

```
/* Prog060.cpp */

#include <stdio.h>
```

```
void main()
{
int n,i,resto;
printf("\Escribe un número: "); scanf("%d",&n);
for(i=2;i<n-1;i++)
      {
      resto=n % i;
      if ((resto==0) && (n != 2))
            {
            printf("\n No es primo");
            return;
            }
      }
printf("\n %d es un número primo",n);
}
```

Estudio del **PROG060:**

**n % i**

Nos da el resto de la división entera entre **n** y **i** (% es un operador como lo es la suma o el producto).

Vamos a hacer el mismo programa que el anterior pero a través de una función: **int primo(int num)**

- Prog061

```
/* Prog061.cpp */

#include <stdio.h>

int primo(int num);

void main()
{
int n;
printf("\n Escribe un número: ");scanf("%d",&n);
if (primo(n)==1) printf("\n\n Es primo");
else printf("\n\n No es primo");
}

int primo(int num)
{
int resto,i,result;
for(i=2;i<num-1;i++)
      {
      resto=num % i;
      if((resto==0) && (num != 2)) return 0;
      }
return 1;
}
```

- Prog062

```
/* Prog062.cpp */

/* Programa que dado un número 'N' escribe todos los
```

```
   números comprendidos entre 1 y 10.000, que cumplen
   las siguientes reglas:
   - La suma de sus cifras debe ser un divisor de N
   - El producto de sus cifras debe ser un múltiplo de 'N' */

#include <stdio.h>

void main()
{
int n,i,c1,c2,c3,c4,suma,producto;
printf("\nEscribe un número: ");scanf("%d",&n);
for (i=1;i<=9999;i++)
   {
   c1=int(i/1000);
   c2=int((i % 1000)/100);
   c3=int(((i % 1000) % 100)/10);
   c4=(((i % 1000) % 100) % 10);
   suma=c1+c2+c3+c4;
   producto=c1*c2*c3*c4;
   if ((n % suma == 0) && (producto % n == 0) &&
         (suma <= n) && (producto >= n))
               {
               printf("%5d",i);
               printf("\n");
               }
   }
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog062.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

## Bibliotecas de programas.

Crea, utilizando tu editor favorito un fichero de nombre **Fich001** y extensión **c** y grábalo como siempre en *TuCarpeta*:

- Fich001.c

```
   /* Fich001.c */

   /* Determinación de un número primo */

   int primo(int num)
   {
   int resto,i,result;
   for(i=2;i<num-1;i++)
         {
         resto=num % i;
         if ((resto==0) && (num != 2)) return 0;
```

```
        }
return 1;
}
```

Acabamos de crear un fichero biblioteca de nombre **Fich001.c** que contiene de momento una sóla función, no lo ejecutes porque entre otras cosas no te funcionará.

- Prog063

```
/* Prog063.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich001.c"

void main()
{
int n;
printf("\n Escribe un número: ");scanf("%d",&n);
if (primo(n)==1) printf("\n\n Es primo");
else printf("\n\n No es primo");
}
```

**Recuerda:**
- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog063.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Observa de qué forma decimos al compilador que utilice las funciones que tenemos en **Fich001.c**:
**#include "c:\TuCarpeta\Fich001.c"**

Vamos a hacer un programa que escribe el listado de números primos inferiores a uno dado (utiliza la función **primo()** de **Fich001.c**

- Prog064

```
/* Prog064.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich001.c"

void main()
{
int n,i;
int j,panta;
char c;
printf("\nHasta qué número quieres la lista de números primos: ");
scanf("%d",&n);
printf("\n");
c=getchar();
printf("\n1\n");
j=0;panta=1;
for(i=2;i<n;i++)
 if(primo(i)==1)
 {
```

```
 printf("%10d",i);
 j++;
 if(j==160*panta)
       {
       printf("\n       Pulsa [Return] para continuar \n");
       c=getchar();
                panta++;
       }
 }
if(primo(n)==1) printf("\n%10d",n);
}
```

Descomposición de un número en factores primos (utiliza la función **primo() de Fich001.c**)

- Prog065

```
/* Prog065.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich001.c"

void main()
{
int n,i;
int k;
printf("\n Escribe el número a descomponer: \n");
scanf("%d",&n);
for(i=2;i<n;i++)
      {
        k=1;
      while ((primo(i)==1) && (n % (k*i) ==0))
            {
            printf("%d-",i);
            k=k*i;
            }
      }
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog065.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Cálculo del máximo común divisor MCD por el método de Euclides.

- Prog066

```
/* Prog066.cpp */

/* Cálculo del M.C.D de dos números */
```

```
#include <stdio.h>

void main()
{
int x,y;
int aux;
int resto;

printf("\n Escribe un número x: "); scanf("%d",&x);
printf("\n Escribe otro número y: "); scanf("%d",&y);
if(x<y)
      {
      aux=x;
      x=y;
      y=aux;
      }
if((x % y) == 0) resto=y;
while ((x % y) != 0)
      {
      resto=x-y*(x/y);
        x=y;
      y=resto;
      }
printf("\n El M.C.D es: %d",resto);
}
```

El programa anterior pero utilizando una función

- Prog067

```
/* Prog067.cpp */

#include <stdio.h>

int MCD(int,int);
void main()
{
int a,b;
int mcd;
printf("\n Número: "); scanf("%d",&a);
printf("\n Número: "); scanf("%d",&b);
mcd=MCD(a,b);
printf("\n\n El MCD de %d y %d es %d",a,b,mcd);
}

int MCD(int x,int y)
{
int aux;
int resto;
if(x<y)
      {
      aux=x;
      x=y;
      y=aux;
      }
if((x % y) == 0) resto=y;
while ((x % y) != 0)
```

```
        {
        resto=x-y*(x/y);
          x=y;
        y=resto;
        }
return resto;
}
```

A partir del **Fich001.c** escribe el siguiente fichero, que deberás grabar donde siempre

- Fich002.c

```
/* Fich002.c */

/* Determinación de un número primo */
int primo(int num)
{
int resto,i,result;
for(i=2;i<num-1;i++)
        {
        resto=num % i;
        if ((resto==0) && (num != 2)) return 0;
        }
return 1;
}

/* Determinación del M.C.D de dos números */
int MCD(int x,int y)
{
int aux;
int resto;
if (x<y)
        {
        aux=x;
        x=y;
        y=aux;
        }
if ((x % y)==0) resto=y;
while ((x % y) != 0)
        {
        resto=x-y*(x/y);
        x=y;
        y=resto;
        }
return resto;
}
```

No compiles el fichero anterior ya que se trata de un fichero biblioteca

Vamos a probar la función **MCD**, que acabamos de incluir en **Fich002.c**

- Prog068

```
/* Prog068.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich002.c"

void main()
{
int a,b;
int mcd;
char otra[1];
otra[0]='s';
while (otra[0]=='s')
        {
      printf("\n Número: "); scanf("%d",&a);
      printf("\n Número: "); scanf("%d",&b);
      mcd=MCD(a,b);
      printf("\n\n El MCD de %d y %d es %d",a,b,mcd);
      printf("\n\n Quieres otro MCD (s/n)? ");
      scanf("%s",otra);
      }
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog068.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Vamos a hacer el mismo programa anterior pero que calcule también el mínimo común múltiplo.

- Prog069

```
/* Prog069.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich002.c"

void main()
{
int a,b;
int mcd;
int prod;
char otra[1];
otra[0]='s';
while (otra[0]=='s')
        {
      printf("\n Número: "); scanf("%d",&a);
      printf("\n Número: "); scanf("%d",&b);
        prod=a*b;
      mcd=MCD(a,b);
      printf("\n\n El MCD de %d y %d es %d",a,b,mcd);
      printf("\n\n El mcm de %d y %d es %d",a,b,prod/mcd);
      printf("\n\n Quieres otro cálculo (s/n)? ");
      scanf("%s",otra);
      }
}
```

Programa que simplifica una fracción, utilizando la función **MCD**, que tenemos en **Fich002.c**

- Prog070

```
/* Prog070.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich002.c"

void main()
{
int a,b;
int div;
printf("\n Escribe el numerador: ");
scanf("%d",&a);
printf("\n Escribe el denominador: ");
scanf("%d",&b);
div=MCD(a,b);
printf("\n Fracción simplificada: %d / %d",a/div,b/div);
}
```

Vamos a hacer el programa anterior pero con arrays.

- Prog071

```
/* Prog071.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich002.c"

void main()
{
int f[2],fs[2];
int i,div;
printf("\n Escribe el numerador: ");
scanf("%d",&f[0]);
printf("\n Escribe el denominador: ");
scanf("%d",&f[1]);
div=MCD(f[0],f[1]);
for(i=0;i<2;i++) fs[i]=f[i]/div;
printf("\n Fracción simplificada: %d / %d",fs[0],fs[1]);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog071.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Vamos a hacer el mismo programa pero a través de una función: **simplifica**

- Prog072

```
/* Prog072.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich002.c"

void simplifica(int ,int );


void main()
{
int f[1];
int i;
printf("\n Escribe el numerador: ");
scanf("%d",&f[0]);
printf("\n Escribe el denominador: ");
scanf("%d",&f[1]);
simplifica(f[0],f[1]);
}


void simplifica(int a,int b)
{
int div;
div=MCD(a,b);
printf("\n Fracción Simplificada: %d / %d",a/div,b/div);
}
```

A partir del **Fich002.c** escribe el siguiente:

- Fich003.c

```
/* Fich003.c */

/* Determinación de un número primo */
int primo(int num)
{
int resto,i,result;
for(i=2;i<num-1;i++)
      {
      resto=num % i;
      if ((resto==0) && (num != 2)) return 0;
      }
return 1;
}

/* Determinación del M.C.D de dos números */
int MCD(int x,int y)
{
int aux;
int resto;
if (x<y)
```

```
        {
        aux=x;
        x=y;
        y=aux;
        }
if ((x % y)==0) resto=y;
while ((x % y) != 0)
        {
        resto=x-y*(x/y);
        x=y;
        y=resto;
        }
return resto;
}

/* Simplifica una fracción */
void simplifica(int a,int b)
{
int div;
div=MCD(a,b);
printf("\nFracción Simplificada: %d / %d",a/div,b/div);
}
```

Vamos a probar la función **simplifica** que tenemos en **Fich003.c**

- Prog073

```
/* Prog073.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich003.c"


void main()
{
int f[1];
int i;
printf("\n Escribe el numerador: ");
scanf("%d",&f[0]);
printf("\n Escribe el denominador: ");
scanf("%d",&f[1]);
simplifica(f[0],f[1]);
}
```

Vamos a hacer un programa que sume dos fracciones.

- Prog074

```
/* Prog074.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich003.c"
```

```
void main()
{
int f1[2],f2[2];
int mcd,num;
printf("\Escribe la 1ª fracción: \n"); scanf("%d %d",&f1[0],&f1[1]);
printf("\Escribe la 2ª fracción: \n"); scanf("%d %d",&f2[0],&f2[1]);
mcd=(f1[1]*f2[1])/MCD(f1[1],f2[1]);
num=(mcd/f1[1])*f1[0]+(mcd/f2[1])*f2[0];
printf("\n\n Suma de %d/%d   +   %d/%d = ",f1[0],f1[1],f2[0],f2[1]);
printf("  %d / %d",num,mcd);
printf("\n\n");
simplifica(num,mcd);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog074.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

## Macros

- Prog075

```
/* Prog075.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich003.c"

#define mcm(x,y) ((x*y)/MCD(x,y))

void main()
{
int f1[2],f2[2];
int mcd,num;
printf("\Escribe la 1ª fracción: \n"); scanf("%d %d",&f1[0],&f1[1]);
printf("\Escribe la 2ª fracción: \n"); scanf("%d %d",&f2[0],&f2[1]);
mcd=mcm(f1[1],f2[1]);
num=(mcd/f1[1])*f1[0]+(mcd/f2[1])*f2[0];
printf("\n\n Suma de %d/%d   +   %d/%d = ",f1[0],f1[1],f2[0],f2[1]);
printf("  %d / %d",num,mcd);
printf("\n\n");
simplifica(num,mcd);
}
```

Estudio del **PROG075**:

- El programa suma dos funciones pero utiliza **una macro** para el cálculo del m.c.m.
- La **macro** está definida en la línea:

  **#define mcm(x,y) ((x*y)/MCD(x,y))**

  Y no es más que una función, pero resumida, es decir no es necesario definir el tipo de variables.

- Observa que la **macro** contiene una llamada a la función **MCD** que tenemos en **Fich003.c**, que funciona sin ningún problema siempre y cuando tengamos en el programa la línea: **#include "c:\TuCarpeta\Fich003.c"**

Escribe a partir del **Fich003** el siguiente:

- Fich004.c

```
/* Fich004.c */

/* Determinación de un número primo */
int primo(int num)
{
int resto,i,result;
for(i=2;i<num-1;i++)
      {
      resto=num % i;
      if ((resto==0) && (num != 2)) return 0;
      }
return 1;
}

/* Determinación del M.C.D de dos números */
int MCD(int x,int y)
{
int aux;
int resto;
if (x<y)
      {
      aux=x;
      x=y;
      y=aux;
      }
if ((x % y)==0) resto=y;
while ((x % y) != 0)
      {
      resto=x-y*(x/y);
      x=y;
      y=resto;
      }
return resto;
}

/* Simplifica una fracción */
void simplifica(int a,int b)
{
int div;
div=MCD(a,b);
printf("\nFracción Simplificada: %d / %d",a/div,b/div);
}

/* Macro que calcula el m.c.m. */
#define mcm(x,y) ((x*y)/MCD(x,y))
```

Vamos a hacer el mismo programa **PROG075**, pero con la macro en el fichero **Fich004.c**

- Prog076

```
/* Prog076.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich004.c"

void main()
{
int f1[2],f2[2];
int mcd,num;
printf("\Escribe la 1ª fracción: \n"); scanf("%d %d",&f1[0],&f1[1]);
printf("\Escribe la 2ª fracción: \n"); scanf("%d %d",&f2[0],&f2[1]);
mcd=mcm(f1[1],f2[1]);
num=(mcd/f1[1])*f1[0]+(mcd/f2[1])*f2[0];
printf("\n\n Suma de %d/%d   +   %d/%d = ",f1[0],f1[1],f2[0],f2[1]);
printf("  %d / %d",num,mcd);
printf("\n\n");
simplifica(num,mcd);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog076.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Crea a partir del **Fich004.c** el siguiente fichero:

- Fich005.c

```
/* Fich005.c */

/* Determinación de un número primo */
int primo(int num)
{
int resto,i,result;
for(i=2;i<num-1;i++)
    {
    resto=num % i;
    if ((resto==0) && (num != 2)) return 0;
    }
return 1;
}

/* Determinación del M.C.D de dos números */
int MCD(int x,int y)
{
int aux;
int resto;
if (x<y)
    {
    aux=x;
    x=y;
    y=aux;
    }
```

```
  if ((x % y)==0) resto=y;
  while ((x % y) != 0)
      {
      resto=x-y*(x/y);
      x=y;
      y=resto;
      }
  return resto;
  }

  /* Simplifica una fracción */
  void simplifica(int a,int b)
  {
  int div;
  div=MCD(a,b);
  printf("\nFracción Simplificada: %d / %d",a/div,b/div);
  }

  /* Macro que calcula el m.c.m. */
  #define mcm(x,y) ((x*y)/MCD(x,y))

  /* Macro que calcula el numerador de la suma de dos
     fracciones */
  #define SumaFracNum(n1,d1,n2,d2)
                       (((mcm(d1,d2)/d1)*n1+(mcm(d1,d2)/d2)*n2))
```

- Prog077

```
/* Prog077.cpp */

#include <stdio.h>
#include "c:\TuCarpeta\Fich005.c"



void main()
{
int f1[2],f2[2];
int den,num;
printf("\Escribe la 1ª fracción: \n"); scanf("%d %d",&f1[0],&f1[1]);
printf("\Escribe la 2ª fracción: \n"); scanf("%d %d",&f2[0],&f2[1]);
num=SumaFracNum(f1[0],f1[1],f2[0],f2[1]);
den=mcm(f1[1],f2[1]);
printf("\n\n Suma de %d/%d   +   %d/%d = ",f1[0],f1[1],f2[0],f2[1]);
printf("  %d / %d",num,den);
printf("\n\n");
simplifica(num,den);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog077.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Está claro lo que hemos conseguido en el **Fich005.c**, una biblioteca de funciones y macros que podemos utilizar en nuestros programas sin más que incluir: **#include "c:\TuCarpeta.Fich005.c"**.
Todo programador, según el tipo de programas que debe hacer dispone de su propia **biblioteca de funciones y macros**, de todas formas hay muchas "bibliotecas de funciones y macros" que son de dominio público y que puedes bajarte de Internet.

## Bases de numeración

- Prog078

```
  /* Prog078.cpp */

  /* Programa que convierte un número hexadecimal a
decimal*/

  #include <stdio.h>
  #include<math.h>
  #include<string.h>
  #include<conio.h>

  float hex_dec(char cadena[]);

  void main()
  {
      char hexa[10];
      float numero;

      clrscr();
      printf("Numero hexadecimal (mayúsculas): ");
      gets(hexa);
      printf("\nCadena= %s",hexa);
      numero=hex_dec(hexa);
      printf("\nEn decimal es : %.0f",numero);
   }

  float hex_dec(char cadena[])
  {
     int i,j;
     char letra;
     float decimal=0;
     float temp;
     temp=0;
     i=strlen(cadena);
     /* La función "strlen", que se encuentra en <string.h>
devuelve la longitud de la cadena. */

      for (j=0;i>0;j++,i--)
     {
        letra=cadena[i-1];
        printf("\n Letra= %c",letra);
          switch(letra){
                    case  '1':temp=(1*pow(16,j));
                              break;
```

```
                                case  '2':temp=(2*pow(16,j));
                                          break;
                                case  '3':temp=(3*pow(16,j));
                                          break;
                                case  '4':temp=(4*pow(16,j));
                                      break;
                                case  '5':temp=(5*pow(16,j));
                                          break;
                                case  '6':temp=(6*pow(16,j));
                                          break;
                                case  '7':temp=(7*pow(16,j));
                                          break;
                                case  '8':temp=(8*pow(16,j));
                                          break;
                                case  '9':temp=(9*pow(16,j));
                                          break;
                                case  '0':temp=(0*pow(16,j));
                                          break;
                                case  'A':temp=(10*pow(16,j));
                                        break;
                                case  'B':temp=(11*pow(16,j));
                                        break;
                                case  'C':temp=(12*pow(16,j));
                                            break;
                                case  'D':temp=(13*pow(16,j));
                                            break;
                                case  'E':temp=(14*pow(16,j));
                                            break;
                           case  'F':temp=(15*pow(16,j));
                                            break;
                         }

                        decimal+=temp;
                    }
                    return(decimal);
                }
```

- Prog079

```
/* Prog079.cpp */

/* Función que cuenta las cifras en binario de un número decimal */

#include <stdio.h>

int numerocifras(int n);

void main()
{
int n;
printf("\nEscribe un número natural: ");scanf("%d",&n);
printf("\n\nEl número de cifras en binario es: %d",numerocifras(n));
}

int numerocifras(int n)
{
int i;
i=1;
```

```
while (n>=2)
      {
      n=int(n/2);
      i=i+1;
      }
return i;
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog079.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

- Prog080

```
/* Prog080.cpp */

/* Programa que escribe un número decimal en binario */

#include <stdio.h>
#include <conio.h>

int numerocifras(int n);

void main()
{
int numci,n,i,cifra;

printf("\nEscribe un número natural: ");scanf("%d",&n);
printf("\n\nEl número en binario es:");
numci=numerocifras(n);
i=0;
while (n>=2)
      {
      cifra=n%2;
      gotoxy(numci-i,8);
      printf("%d",cifra);
      n=int(n/2);
      i++;
      }
gotoxy(numci-i,8);
printf("%d",n);
}

int numerocifras(int n)
{
int i;
i=1;
while (n>=2)
      {
      n=int(n/2);
      i=i+1;
      }
return i;
}
```

La función **gotoxy(a,b)** que se encuentra en el fichero **conio.h**, sitúa el cursor en la columna **a**, fila **b**. De forma que si escribimos:

**gotoxy(3,9);printf("Hola");**

aparecerá en la columna 3, fila 9 la palabra **Hola**.

- Prog081

```
/* Prog081.cpp */

/* Programa que escribe un número decimal en base 3 */

#include <stdio.h>
#include <conio.h>

int numerocifras(int n);

void main()
{
int numci,n,i,cifra;

printf("\nEscribe un número natural: ");scanf("%d",&n);
// printf("\n\nEl número en base 3 es:");
numci=numerocifras(n);
i=0;
while (n>=3)
      {
      cifra=n%3;
      gotoxy(numci-i,8);
      printf("%d",cifra);
      n=int(n/3);
      i++;
      }
gotoxy(numci-i,8);
printf("%d",n);
}

int numerocifras(int n)
{
int i;
i=1;
while (n>=3)
      {
      n=int(n/3);
      i=i+1;
      }
return i;
}
```

- Prog082

```
/* Prog082.cpp */

/* Programa que escribe un número decimal en base 4 */

#include <stdio.h>
```

```
#include <conio.h>

int numerocifras(int n);

void main()
{
int numci,n,i,cifra;

printf("\nEscribe un número natural: ");scanf("%d",&n);
printf("\n\nEl número en base 4 es:");
numci=numerocifras(n);
i=0;
while (n>=4)
      {
      cifra=n%4;
      gotoxy(numci-i,8);
      printf("%d",cifra);
      n=int(n/4);
      i++;
      }
gotoxy(numci-i,8);
printf("%d",n);
}

int numerocifras(int n)
{
int i;
i=1;
while (n>=4)
      {
      n=int(n/4);
      i=i+1;
      }
return i;
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog082.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

## Coprimos

Dos números enteros se dicen **coprimos** si y sólo si el **M.C.D.** entre ellos es 1. Por ejemplo: m.c.d(17,91) = 1 luego 17 y 91 son coprimos

- Prog083

```
/* Prog083.cpp */

/* Determinar los dos 'int' más grandes que
   sean 'coprimos' */
```

```
#include <stdio.h>
#include <values.h>

int MCD(int x,int y);

void main()
{
int max,num1,num2;
int i=1;
max=MAXINT;
while (MCD(max,MAXINT-i) != 1) i++;
num1=MAXINT;
num2=MAXINT-i;
printf("\n Los 2 números mayores coprimos tipo 'int' son %d y
%d",num1,num2);
printf("\n\n C O M P R O B A C I O N:");
printf("\nMáximo INT= %d",MAXINT);
printf("\n\nMCD de %d y %d es %d",num1,num2,MCD(num1,num2));
}

int MCD(int x,int y)
{
int aux,resto;
if (x<y)
   {
   aux=x;
   x=y;
   y=aux;
   }
if ((x % y)==0) resto=y;
while ((x % y) != 0)
   {
   resto=x-y*(x/y);
   x=y;
   y=resto;
   }
return resto;
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog083.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

- Prog084

```
/* Prog084.cpp */

/* Matriz tal que Aij=1 si i,j son coprimos
                Aij=0 si i,j no son coprimos */

#include <stdio.h>


int MCD(int x,int y);
```

```
void main()
{
int n; int i,j;
int matr[20][20];
printf("\nOrden de la matriz: "); scanf("%d",&n);
printf("\n\n");
for(i=0;i<n;i++)
      {
      for(j=0;j<n;j++)
                  {
            if(MCD(i+1,j+1)==1)
                  matr[i][j]=1;
            else
            matr[i][j]=0;
            printf("%d ",matr[i][j]);
            };
      printf("\n");
      }
}


int MCD(int x,int y)
{
int aux,resto;
if (x<y)
      {
      aux=x;
      x=y;
      y=aux;
      }
if ((x % y)==0) resto=y;
while ((x % y) != 0)
      {
      resto=x-y*(x/y);
      x=y;
      y=resto;
      }
return resto;
}
```

# AUTOEVALUACIÓN 3

1) Haz un programa que calcule el producto vectorial de 2 vectores del espacio.

Graba el programa con el nombre **EVAL3A**.

2) Haz un programa que calcule el módulo de un vector en el espacio, utilizando una función con retorno de parámetro (guíate por el **PROG057**)

Graba el programa con el nombre **EVAL3B**.

3) Haz un programa que calcule el área de un triángulo en el espacio, dadas las coordenadas de los 3 vértices (utiliza la función del programa **EVAL3B**)

Graba el programa con el nombre **EVAL3C**.

4) Haz un programa que funcione de la siguiente forma:
- El programa nos pide 10 valores (serie estadística).
- El programa calcula la media aritmética (función con retorno de parámetro).
- El programa calcula las desviaciones respecto a la media.
- El programa calcula la desviación media (llamada a la misma función de antes).
- El programa calcula la desviación típica (llamada a la misma función de antes)

Graba el programa con el nombre **EVAL3D**

5) Escribe el siguiente programa:

```
/* eval3e */

#include <stdio.h>
#include <math.h>

float modul(float v[3]);


void main()
{
int i;
float a[3],b[3],c[3];
float v1[3],v2[3];
float provect[3];
```

```
float modu,area;
printf("\n Introduce las coordenadas del punto A: \n");
for(i=0;i<2;i++) scanf("%f",&a[i]);
a[2]=0;
printf("\n Introduce las coordenadas del punto B: \n");
for(i=0;i<2;i++) scanf("%f",&b[i]);
b[2]=0;
printf("\n Introduce las coordenadas del punto C: \n");
for(i=0;i<2;i++) scanf("%f",&c[i]);
c[2]=0;
printf("\n\n Punto A = (%f,%f,%f)",a[0],a[1],a[2]);
printf("\n\n Punto B = (%f,%f,%f)",b[0],b[1],b[2]);
printf("\n\n Punto C = (%f,%f,%f)",c[0],c[1],c[2]);
for(i=0;i<3;i++)
        {
        v1[i]=b[i]-a[i];
        v2[i]=c[i]-a[i];
        }
printf("\n\n Vector AB = (%f,%f,%f)",v1[0],v1[1],v1[2]);
printf("\n\n Vector AC = (%f,%f,%f)",v2[0],v2[1],v2[2]);
provect[0]=v1[1]*v2[2]-v2[1]*v1[2];
provect[1]=v2[0]*v1[2]-v1[0]*v2[2];
provect[2]=v1[0]*v2[1]-v2[0]*v1[1];

printf("\n\n Producto Vectorial AB^AC = (%f,%f,%f)",provect[0],provect[1],provect[2]);
modu=modul(provect);
printf("\n\n\n Módulo del producto Vectorial = %f ",modu);
area=0.5*modu;
printf("\n\n Área del triangulo = %f",area);
}


float modul(float v[3])
{
float modu;
modu=sqrt(v[0]*v[0]+v[1]*v[1]+v[2]*v[2]);
return modu;
}
```

- Grábalo con el nombre **EVAL3E**

- Ejecútalo varias veces.

- Explica qué es lo que hace. Compáralo con el **EVAL3C**.

6) Haz un programa de nombre **EVAL3F** que sirva para elevar al cubo un número, pero utilizando una función.

7) Haz un programa de nombre **EVAL3G** que haga lo mismo que el anterior pero no a través de una función sino de una macro.

8) Haz un programa de nombre **EVAL3H** que transforme un número en base 10 a base 4 y 5

9) Haz un programa de nombre **EVAL3I** que haga lo mismo que el **EVAL3A** pero utiliza la función **gotoxy** para que la salida por pantalla sea más estética.

10) Haz un programa de nombre **EVAL3J** que construya el triángulo de **Tartaglia o de Pascal**.

# IV.- Punteros y Arrays

## a) Constantes Simbólicas

- Prog085

```
/* Prog085 */

#include <stdio.h>
#include <math.h>

void sencos(float inicial,float paso);

void main()
{
float inicial,paso;
printf("\n Valor inicial y incremento ?\n");
scanf("%f %f",&inicial,&paso);
sencos(inicial,paso);
}

void sencos(float inicial,float paso)
{
int i;
double v[50],s[50],c[50];
for(i=0;i<50;i++)
     {
     v[i]=(double)(inicial+i*paso);
     s[i]=sin(v[i]);
     c[i]=cos(v[i]);
     }
for(i=0;i<50;i++) printf("\n%10lf %10lf %10lf",v[i],s[i],c[i]);
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog085.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Estudio del **PROG085:**

Es importante que tengas el **PROG058** a la vista, para poder compararlo con el **PROG085**

- En el PROG058 el número de valores de la tabla de senos y cosenos correspondía a una variable (num), en nuestro caso (PROG085) el número de valores es fijo: 50.
- En el PROG058 las variables **inicial** y **paso** eran **double**, encambio en el PROG085 las declaramos **float**.
- El argumento de las funciones sin() y cos() (funciones que hay en la librería math.h) debe ser double. Por esta razón en el PROG085 hemos de transformar el argumento que es float a double. Y esto lo conseguimos de la siguiente forma: **v[i]=(double)(inicial+i*paso);** Es decir, basta escribir double entre parentesis antes de la expresión que es float, también entre paréntesis: (inicial+i*paso)

- En el PROG058 las variables v, s, c son simples, encambio en el PROG085 utilizamos arrays: v[50], s[50] y c[50]

Nos gustaría modificar el PROG085 de forma que en lugar de 50 valores, sea otro número, pero que esta modificación sea fácil de hacer: vamos a utilizar las llamadas **constantes simbólicas**…

Escribe, a partir del PROG085, el siguiente programa:

- Prog086

```
/* Prog086.cpp */

#include <stdio.h>
#include <math.h>

#define tam 50

void sencos(float inicial,float paso);

void main()
{
float inicial,paso;
printf("\n Valor inicial y incremento ?\n");
scanf("%f %f",&inicial,&paso);
sencos(inicial,paso);
}

void sencos(float inicial,float paso)
{
int i;
double v[tam],s[tam],c[tam];
for(i=0;i<tam;i++)
 {
 v[i]=(double)(inicial+i*paso);
 s[i]=sin(v[i]);
 c[i]=cos(v[i]);
 }
for(i=0;i<tam;i++) printf("\n%10lf %10lf %10lf",v[i],s[i],c[i]);
}
```

Estudio del **PROG086**:

- **#define tam 50**, definimos una **constante simbólica** de nombre **tam** y valor igual a **50**. En el resto del programa, donde aparece **tam**, el compilador entiende **50**.
- La utilidad de una constante simbólica está en que basta cambiar el número **50**, que hay al final de la línea **define**, por el número que queramos, para que el programa funcione exactamente igual pero con un valor de **tam** igual al nuevo valor introducido. Pruébalo.

## b) La función rand()

- Prog087

```
/* Prog087.cpp */

#include <stdio.h>
#include <stdlib.h>

#define num 20

void main()
{
int vect[num];
int i;
char nada[1];
printf("\n Listado de números aleatorios entre 0 y 9 \n");
for(i=0;i<num;i++)
     {
     vect[i]=rand() % 10;
     printf("\n %d",vect[i]);
     }
printf("\n\n        Continuo (s/n) ?   "); scanf("%s",&nada);
if (nada[0]=='n') return;

printf("\n Listado de números aleatorios entre 0 y 10 \n");
for(i=0;i<num;i++)
     {
     vect[i]=rand() % 11;
     printf("\n %d",vect[i]);
     }
printf("\n\n        Continuo (s/n) ?   "); scanf("%s",&nada);
if (nada[0]=='n') return;

printf("\n Listado de números aleatorios entre 0 y 99 \n");
for(i=0;i<num;i++)
     {
     vect[i]=rand() % 100;
     printf("\n %d",vect[i]);
     }
printf("\n\n        Continuo (s/n) ?   "); scanf("%s",&nada);
if (nada[0]=='n') return;

printf("\n Listado de números aleatorios entre 1 y 6 \n");
for(i=0;i<num;i++)
     {
     vect[i]=(rand() % 6)+1;
     printf("\n %d",vect[i]);
     }
}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog087.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Estudio del **PROG087:**

- En este programa utilizamos una nueva función: **rand()**. Dicha función se encuentra en la biblioteca **stdlib.h**, por esta razón necesitamos escribir la línea: **#include <stdlib.h>**
- Funcionamiento de la función **rand():**
  rand() % 10: determina un número aleatorio entre 0 y 9 (incluidos los dos):
  rand() % 11: " " " " 0 y 10 "
  rand() % 100: : " " " " 0 y 99 "
  (rand() % 6)+1: " " " 1 y 6 "

- En realidad la función **rand()** genera números pseudoaleatorios, es decir cada vez que ejecutemos el programa, la sucesión de números "rand" será la misma.
- Si necesitamos "series de **rand()**" distintas, basta anteponer una línea de contenido **srand(semilla)**, función que también está en **stdlib.h**, con "semilla" un número entero. Es decir si escribimos **srand(1)** la serie de números "rand" será distinta si escribimos **srand(2)**. Dicho de otra forma: según el valor de "semilla" obtendremos una serie de números pseudoaleatorios u otra.

Vamos a hacer un programa que ordene un listado de 20 números.

- Prog088

```
/* Prog088.cpp */

#include <stdio.h>
#include <stdlib.h>

void main()
{
int vect[20];
int i,j;
int aux;
char nada[1];

printf("\n Listado de números entre 0 y 100 \n");
for(i=0;i<20;i++)
      {
      vect[i]=rand() % 100;
      printf("\n El número %d es %d",i,vect[i]);
      }
printf("\n\n                Continuo (s/n) ?  "); scanf("%s",&nada);
if (nada[0]=='n') return;
printf("\n Vamos a ordenar los números");
for(i=0;i<20;i++)
      {
      printf("\n Determinando el número que debe ocupar el lugar
%d",i);
      for(j=i+1;j<20;j++)
            {
            printf(".");

            if(vect[j]<vect[i])
```

```
                    {
                    aux=vect[j];
                    vect[j]=vect[i];
                    vect[i]=aux;
                    }
              }
       }
printf("\n\n                  Continuo (s/n) ?  "); scanf("%s",&nada);
if (nada[0]=='n') return;
printf("\n\n La lista ordenada es: \n\n");
for(i=0;i<20;i++) printf("El número %d es %d \n",i,vect[i]);

}
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog088.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

Estudio del **PROG088:**

- El problema central es: "dadas 2 variables vect[j], vect[i] cómo intercambiamos sus valores". Es decir, nos interesa asignar a vect[j] el valor de vect[i] y viceversa.

Observa la solución:

```
aux=vect[j];
vect[j]=vect[i];
vect[i]=aux;
```

siendo **aux** una variable auxiliar que hace de puente.

- Está claro que con:

```
if(vect[j]<vect[i])
      {
      aux=vect[j];
      vect[j]=vect[i];
      vect[i]=aux;
      }
```

Conseguimos colocar los números menores delante. En definitiva, lo que conseguimos es **ordenar** los valores del array **vect[20]**

- Prog089

```
/* Prog089.cpp */

/* Programa: Multiplicaciones
   El programa nos pide '¿Cuántas multiplicaciones?' queremos
   El programa nos las pregunta aleatoriamente.
   Al final, el programa nos da la nota. */

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

void main()
{
int n,a,b,result;
```

```
int i;
int bien,mal;
bien=0;mal=0;
printf("\n¿Cuántas Multiplicaciones? ");scanf("%d",&n);
for (i=1;i<=n;i++)
 {
 clrscr();
 a=rand() % 10;
 b=rand() % 10;
 printf("%d  x  %d = ",a,b);scanf("%d",&result);
 if (result==a*b)
       {
       printf("\n MUY BIEN");
       bien=bien+1;
       printf("\n\nPulsa una tecla para continuar ...");
       getche();
                clrscr();
       }
 else
       {
       printf("\n NO ES CORRECTO ");
       printf("\n %d  x  %d  =  %d ",a,b,a*b);
       mal=mal+1;
       printf("\n\n Pulsa una tecla para continuar ...");
       getche();
                clrscr();
       };
 }
clrscr();
printf("\n Bien = %d       Mal = %d",bien,mal);
printf("\n\n            NOTA = %d",int(bien*10/n));
}
```

## c) Arrays Multidimensionales

Hasta ahora los arrays que hemos utilizado eran de una dimensión, vamos a ver ahora un array bidimensional (tabla de doble entrada).

- Prog090

```
/* Prog090.cpp */

#include <stdio.h>
#include <stdlib.h>

void main()
{
int matriz[20][5];
int i,j;
int aux;
char nada[1];
```

```
printf("Vamos a generar un listado de notas \n\n");
for(i=0;i<20;i++)
 {
 printf("\n Fila %3d:       ",i);
 for(j=1;j<5;j++)
       {
       matriz[i][j]=rand() % 10;
       printf("%d",matriz[i][j]);
       printf("   ");
       }
 }
printf("\n\n            Continuo (s/n) ?  "); scanf("%s",&nada);
if (nada[0]=='n') return;
printf("\n\n Vamos a calcular la media \n\n");
for(i=0;i<20;i++)
 {
 printf("\n La media de la fila %2d es: ",i);
 aux=0;
 for(j=1;j<5;j++) aux=aux+matriz[i][j];
 matriz[i][0]=aux/4;
 printf("%5d",matriz[i][0]);
 }
}
```

Estudio del **PROG090:**

- El programa utiliza un **array** bidimensional (matriz[20][5]) de 20 filas y 5 columnas, que simula las notas de 20 alumnos en 4 asignaturas. En la columna 0 del array, el programa coloca la media de las notas.
- Observa la línea: **matriz[i][0]=aux/4;**
  La división entre números enteros (aux y 4), se trunca automáticamente para que sea otro entero (matriz[i][0])

- Prog091

```
/* Prog091.cpp */

#include <stdio.h>

void main()
{
int m1[2][2];
int m2[2][2];
int prod[2][2];
int i,j;
int aux;

/* Introducción de las matrices */
printf("\n Primera matriz \n");
for(i=0;i<2;i++)
     {
     printf("\n Escribe los elementos de la fila %d:\n",i+1);
     for(j=0;j<2;j++) scanf("%d",&m1[i][j]);
     }
printf("\n Segunda matriz \n");
for(i=0;i<2;i++)
```

```
        {
        printf("\n Escribe los elementos de la fila %d:\n",i+1);
        for(j=0;j<2;j++) scanf("%d",&m2[i][j]);
        }
    /* Escritura de las dos matrices */
    printf("\n\n Tenemos:       \n\n");
    for(i=0;i<2;i++)
        {
           printf(" | ");
        for(j=0;j<2;j++)
              {
              printf("%3d",m1[i][j]);
              }
        printf(" |        | ");
        for(j=0;j<2;j++)
              {
              printf("%3d",m2[i][j]);
              }
        printf(" | \n");
        }
    /* Cálculo del producto */
    prod[0][0]=m1[0][0]*m2[0][0]+m1[0][1]*m2[1][0];
    prod[1][0]=m1[1][0]*m2[0][0]+m1[1][1]*m2[1][0];
    prod[0][1]=m1[0][0]*m2[0][1]+m1[0][1]*m2[1][1];
    prod[1][1]=m1[1][0]*m2[0][1]+m1[1][1]*m2[1][1];

    /* Escritura del resultado */
    printf("\n\n El producto de las dos matrices es: \n");
    for(i=0;i<2;i++)
        {
        printf(" | ");
        for(j=0;j<2;j++)
              {
              printf("%5d",prod[i][j]);
              }
        printf(" | \n");
        }
    }
```

**Recuerda:**

- Escribe el programa anterior utilizando tu compilador.
- Grábalo con el nombre **Prog091.cpp** en *TuCarpeta*.
- Ejecútalo un par o tres de veces

- Prog092

```
/* Prog092.cpp */

/* Programa que lee un sistema de 3 ecuaciones con 3 incógnitas
   y escribe la matriz ampliada */

#include <stdio.h>

void main()
{
```

```
float a[3][4];
int j;

/* Introducción de datos */
printf("\n Primera ecuación\n ");
for (j=0;j<=3;j++)
      {
      if (j==3)
            {
            printf("\tTérmino independiente: \t");
            scanf("%f",&a[0][3]);
            break;
            };
      printf("\tCoeficiente de x%d: \t",j+1);
      scanf("%f",&a[0][j]);
      }
printf("\n");
printf("\n Segunda ecuación\n ");
for (j=0;j<=3;j++)
      {
      if (j==3)
            {
            printf("\tTérmino independiente: \t");
            scanf("%f",&a[1][3]);
            break;
            };
      printf("\tCoeficiente de x%d: \t",j+1);
      scanf("%f",&a[1][j]);
      }
printf("\n");
printf("\n Tercera ecuación\n ");
for (j=0;j<=3;j++)
      {
      if (j==3)
            {
            printf("\tTérmino independiente: \t");
            scanf("%f",&a[2][3]);
            break;
            };
      printf("\tCoeficiente de x%d: \t",j+1);
      scanf("%f",&a[2][j]);
      }
printf("\n");

/* Escritura de la matriz ampliada */
printf("\n\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[0][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[1][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[2][j]);
printf("\n");

}
```

- Prog093

```
/* Prog093.cpp */

/* Programa que lee un sistema de 3 ecuaciones con 3 incógnitas
   y resuelve el sistema por el método de GAUSS
   Supongo la existencia de coeficientes distintos de cero
   en las incógnitas
   Supongo también que el sistema es Compatible Determinado. */

#include <stdio.h>

void main()
{
float a[3][4];
int j,i,k;
float b;
float x[3],aux[3][4];

/* Introducción de datos */
printf("\n Primera ecuación\n ");
for (j=0;j<=3;j++)
      {
      if (j==3)
            {
            printf("\tTérmino independiente: \t");
            scanf("%f",&a[0][3]);
            break;
            };
      printf("\tCoeficiente de x%d: \t",j+1);
      scanf("%f",&a[0][j]);
      }
printf("\n");
printf("\n Segunda ecuación\n ");
for (j=0;j<=3;j++)
      {
      if (j==3)
            {
            printf("\tTérmino independiente: \t");
            scanf("%f",&a[1][3]);
            break;
            };
      printf("\tCoeficiente de x%d: \t",j+1);
      scanf("%f",&a[1][j]);
      }
printf("\n");
printf("\n Tercera ecuación\n ");
for (j=0;j<=3;j++)
      {
      if (j==3)
            {
            printf("\tTérmino independiente: \t");
            scanf("%f",&a[2][3]);
            break;
            };
      printf("\tCoeficiente de x%d: \t",j+1);
      scanf("%f",&a[2][j]);
      }
```

```
printf("\n");

/* Escritura de la matriz ampliada */
printf("\n\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[0][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[1][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[2][j]);
printf("\n");



/* Método de GAUSS: hemos de conseguir una matriz
   triangular superior. */
for (i=0;i<=2;i++)
      {
      b=a[i][i];
        // Observa que estoy suponiendo a[i][i] distinto de 0
      for (j=i;j<=3;j++) a[i][j]=a[i][j]/b;
      for (j=i+1;j<=2;j++)
            {
            b=a[j][i];
            for (k=i+1;k<=3;k++) a[j][k]=a[j][k]-a[i][k]*b;
            }
      }



for (k=0;k<=2;k++)
      {
        i=2-k;
      b=0;
      for (j=i+1;j<=2;j++) b=b+a[i][j]*x[j];
      x[i]=a[i][3]-b;
      }
a[1][0]=0;
a[2][0]=0;
a[2][1]=0;

/* Escribe la matriz resultante, Observa que es
   triangular superior. */
printf("\n\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[0][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[1][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[2][j]);
printf("\n");

printf("\n\nX1= %f    X2= %f     X3= %f ",x[0],x[1],x[2]);
}
```

Prueba el **prog093** para los sistemas:

3x +2y –5z = -8
-2x –4y +z = -7
5x –4y +2z = 3
Si todo funciona correctamente nos sale: x= 1 ; y=2 ; z=3

0x –5y +3z =-25
3x +0y –5z = 22
2x –7y +0z = -16
En este caso no funcionará porque a[0][0]= 0

1x +2y +3z = 4
2x +5y +3z = 6
2x +4y +5z = 3
Solución: x=-37 ; y=13 ; z=5

- Prog094

```
/* Prog094.cpp */

/* Programa que lee un sistema de 3 ecuaciones con 3 incógnitas
   y resuelve el sistema por el método de GAUSS */

#include <stdio.h>

void main()
{
float a[3][4];
int j,i,k;
float b,aux;
float x[3];

/* Introducción de datos */
printf("\n Primera ecuación\n ");
for (j=0;j<=3;j++)
      {
      if (j==3)
            {
            printf("\tTérmino independiente: \t");
            scanf("%f",&a[0][3]);
            break;
            };
      printf("\tCoeficiente de x%d: \t",j+1);
      scanf("%f",&a[0][j]);
      }
printf("\n");
printf("\n Segunda ecuación\n ");
for (j=0;j<=3;j++)
      {
      if (j==3)
            {
            printf("\tTérmino independiente: \t");
            scanf("%f",&a[1][3]);
            break;
```

```
            };
      printf("\tCoeficiente de x%d: \t",j+1);
      scanf("%f",&a[1][j]);
      }
printf("\n");
printf("\n Tercera ecuación\n ");
for (j=0;j<=3;j++)
      {
      if (j==3)
            {
            printf("\tTérmino independiente: \t");
            scanf("%f",&a[2][3]);
            break;
            };
      printf("\tCoeficiente de x%d: \t",j+1);
      scanf("%f",&a[2][j]);
      }
printf("\n");

/* Escritura de la matriz ampliada */
printf("\n\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[0][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[1][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[2][j]);
printf("\n");



/* Método de GAUSS: hemos de conseguir una matriz
   triangular superior. */
for (i=0;i<=2;i++)
      {
      if (a[i][i]==0)
            {
            j=i+1;
            while (a[j][i]==0) j=j+1;
            if (j==3)
                  {
                  printf("\n    ES UN SISTEMA INCOMPATIBLE");
                  return;
                  }
            for (k=0;k<=3;k++)
                  {
                  aux=a[i][k];
                  a[i][k]=a[j][k];
                  a[j][k]=aux;
                  }
            };

      b=a[i][i];

      for (j=i;j<=3;j++) a[i][j]=a[i][j]/b;
      for (j=i+1;j<=2;j++)
            {
```

```
            b=a[j][i];
            for (k=i+1;k<=3;k++) a[j][k]=a[j][k]-a[i][k]*b;
            }
      }



for (k=0;k<=2;k++)
      {
        i=2-k;
      b=0;
      for (j=i+1;j<=2;j++) b=b+a[i][j]*x[j];
      x[i]=a[i][3]-b;
      }
a[1][0]=0;
a[2][0]=0;
a[2][1]=0;

/* Escribe la matriz resultante, Observa que es
   triangular superior. */
printf("\n\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[0][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[1][j]);
printf("\n");
printf("  |");
for (j=0;j<=3;j++) printf("\t%7.2f",a[2][j]);
printf("\n");

printf("\n\nX1= %f    X2= %f     X3= %f ",x[0],x[1],x[2]);
}
```

Prueba el **prog094** para los sistemas:

3x +2y –5z = -8
-2x –4y +z = -7
5x –4y +2z = 3
Si todo funciona correctamente nos sale: x= 1 ; y=2 ; z=3

0x –5y +3z =-25
3x +0y –5z = 22
2x –7y +0z = -16
Solución: x=-1 ; y=2 ; z= -5

1x +2y +3z = 4
2x +5y +3z = 6
2x +4y +5z = 3
Solución: x=-37 ; y=13 ; z=5

x + y + z = 3
2x –5y –z = 5
3x –4y +0z = 8
Es compatible indeterminado, una de x=2 ; y=-0,5 ; z= 1,5

x +y +z = 2
x –y +z = -3
2x +0y +2z = 1
Es un sistema incompatible.

- Prog095

```
/* Prog095.cpp */

/* Programa que transforma una matriz en otra
   equivalente, pero triangular superior, por
   el método de GAUSS */

#include <stdio.h>

void main()
{
float a[3][3];
int j,i,k;
float b,aux;

/* Introducción de datos */
printf("\n Primera fila de la matriz\n ");
for (j=0;j<=2;j++)
      {
      printf("\tColumna %d: \t",j+1);
      scanf("%f",&a[0][j]);
      }
printf("\n");
printf("\n Segunda fila de la matriz\n ");
for (j=0;j<=2;j++)
      {
      printf("\tColumna %d: \t",j+1);
      scanf("%f",&a[1][j]);
      }
printf("\n");
printf("\n Tercera fila de la matriz\n ");
for (j=0;j<=2;j++)
      {
      printf("\tColumna %d: \t",j+1);
      scanf("%f",&a[2][j]);
      }
printf("\n");

/* Escritura de la matriz */
printf("\n\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[0][j]);
printf("\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[1][j]);
printf("\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[2][j]);
printf("\n");
```

```
/* Método de GAUSS: hemos de conseguir una matriz
   triangular superior. */
for (i=0;i<=2;i++)
      {
      if (a[i][i]==0)
            {
            j=i+1;
            while (a[j][i]==0) j=j+1;
            if (j==2)
                  {
                  printf("\nNo sé hacerlo porque hay muchos ceros");
                  return;
                  }
            for (k=0;k<=2;k++)
                  {
                  aux=a[i][k];
                  a[i][k]=a[j][k];
                  a[j][k]=aux;
                  }
            };

      b=a[i][i];

      for (j=i+1;j<=2;j++)
            {
            b=a[j][i]/a[i][i];
            for (k=i+1;k<=2;k++) a[j][k]=a[j][k]-a[i][k]*b;
            }
      }


a[1][0]=0;
a[2][0]=0;
a[2][1]=0;

/* Escribe la matriz triangular superior */
printf("\n\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[0][j]);
printf("\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[1][j]);
printf("\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[2][j]);
printf("\n");
}
```

Matemáticamente no es del todo correcto, ya que en el caso a[i][i]=0, permutamos las filas de la matriz

Pruébalo para la matriz de las incógnitas de los sistemas del programa anterior.

- Prog096

```
/* Prog096.cpp */

/* Programa que calcula el determinante
   por el método de GAUSS */

#include <stdio.h>

void main()
{
float a[3][3];
int j,i,k;
float b,aux,deter;
deter=1;
/* Introducción de datos */
printf("\n Primera fila de la matriz\n ");
for (j=0;j<=2;j++)
      {
      printf("\tColumna %d: \t",j+1);
      scanf("%f",&a[0][j]);
      }
printf("\n");
printf("\n Segunda fila de la matriz\n ");
for (j=0;j<=2;j++)
      {
      printf("\tColumna %d: \t",j+1);
      scanf("%f",&a[1][j]);
      }
printf("\n");
printf("\n Tercera fila de la matriz\n ");
for (j=0;j<=2;j++)
      {
      printf("\tColumna %d: \t",j+1);
      scanf("%f",&a[2][j]);
      }
printf("\n");

/* Escritura del determinante */
printf("\n\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[0][j]);
printf("\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[1][j]);
printf("\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[2][j]);
printf("\n");

/* Método de GAUSS: hemos de conseguir un determinante
   triangular superior. */
for (i=0;i<=2;i++)
      {
      if (a[i][i]==0)
            {
            j=i+1;
                deter=-1;
            while (a[j][i]==0) j=j+1;
            if (j==2)
                  {
```

```
                printf("\nNo sé hacerlo porque hay muchos ceros");
                return;
                }
          for (k=0;k<=2;k++)
                {
                aux=a[i][k];
                a[i][k]=a[j][k];
                a[j][k]=aux;
                }
          };

    b=a[i][i];

    for (j=i+1;j<=2;j++)
          {
          b=a[j][i]/a[i][i];
          for (k=i+1;k<=2;k++) a[j][k]=a[j][k]-a[i][k]*b;
          }
    }


a[1][0]=0;
a[2][0]=0;
a[2][1]=0;

/* Escribe la matriz triangular superior */
printf("\n\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[0][j]);
printf("\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[1][j]);
printf("\n");
printf("  |");
for (j=0;j<=2;j++) printf("\t%7.2f",a[2][j]);
printf("\n");

deter=deter*a[0][0]*a[1][1]*a[2][2];

printf("\nEl determinante es %f",deter);
}
```

Prueba els programa para los determinantes:

| | | |
|---|---|---|
| 3 | 2 | -5 |
| -2 | -4 | 1 |
| 5 | -4 | 2 |

Solución: -134

| | | |
|---|---|---|
| 0 | -5 | 3 |
| 3 | 0 | -5 |
| 2 | -7 | 0 |

Solución: -13

1 2 3
2 5 3
2 4 5

Solución: -1

1 1 1
2 -5 -1
3 -4 0

Solución: 0

1 1 1
1 -1 1
2 0 2

Solución: 0

- Prog097

```
/* Prog097.cpp */

/* Programa que calcula un determinante
   de orden 'n' (n<=10) */

#include <stdio.h>

void main()
{
float a[10][10];
int j,i,k,n,s;
float b,aux,deter;
deter=1;
printf("\nCálculo de un determinante de orden ? ");
scanf("%d",&n);
s=1;
/* Introducción de datos */
while (s<=n)
{
printf("\n Fila  %d del determinante\n ",s);
for (j=0;j<=n-1;j++)
      {
      printf("\tColumna %d: \t",j+1);
      scanf("%f",&a[s-1][j]);
      }
printf("\n");
s++;
};

/* Escritura del determinante */
printf("\n\n");
for (s=0;s<=n-1;s++)
{
printf("  |");
for (j=0;j<=n-1;j++) printf("\t%7.2f",a[s][j]);
printf("\n");
}
```

```
/* Método de GAUSS: hemos de conseguir un determinante
   triangular superior. */
for (i=0;i<=n-1;i++)
      {
      if (a[i][i]==0)
            {
            j=i+1;
                deter=-1;
            while (a[j][i]==0) j=j+1;
            if (j==n-1)
                  {
                  printf("\nNo sé hacerlo porque hay muchos ceros");
                  return;
                  }
            for (k=0;k<=n-1;k++)
                  {
                  aux=a[i][k];
                  a[i][k]=a[j][k];
                  a[j][k]=aux;
                  }
            };

      b=a[i][i];

      for (j=i+1;j<=n-1;j++)
            {
            b=a[j][i]/a[i][i];
            for (k=i+1;k<=n-1;k++) a[j][k]=a[j][k]-a[i][k]*b;
            }
      }



for (i=1;i<=n-1;i++)
      {
      for (k=0;k<=n-1;k++)
            {
            j=i-1-k;
            a[i][j]=0;
            };
      }


/* Escribe la matriz triangular superior */
printf("\n\n");
for (s=0;s<=n-1;s++)
{
printf("  |");
for (j=0;j<=n-1;j++) printf("\t%7.2f",a[s][j]);
printf("\n");
}

for (i=0;i<=n-1;i++) deter=deter*a[i][i];


printf("\nEl determinante es %f",deter);

}
```

Pruébalo para los determinantes:

| | | | | |
|---|---|---|---|---|
| 5 | 2 | -3 | 5 | |
| -1 | 1 | 2 | 6 | |
| 1 | 0 | -1 | 7 | |
| 3 | 1 | 0 | 8 | |
| | | | | Solución: -112 |

| | | | | |
|---|---|---|---|---|
| -3 | 0 | 1 | 3 | |
| 2 | -1 | 2 | -2 | |
| 1 | 2 | 3 | 4 | |
| -1 | 5 | -4 | 5 | |
| | | | | Solución: -31 |

| | | | | |
|---|---|---|---|---|
| 0 | -1 | 0 | -1 | 0 |
| 1 | 2 | 0 | -1 | 1 |
| 2 | 1 | -3 | -2 | 2 |
| -1 | 1 | 3 | 2 | 3 |
| 5 | 1 | 4 | -3 | 4 |
| | | | | Solución: 183 |

## d) Direcciones de Memoria

Imaginemos que la memoria del ordenador es como una fila de casillas en las que podemos almacenar datos. En cada una de ellas podemos almacenar una cantidad fija de información: **1 byte** (8 bits = 8 ceros y unos que representan un carácter).
A cada una de estas casillas de nuestro ordenador, le corresponde **una dirección de memoria**.
De esta forma, cuando en nuestro programa usamos una variable, internamente lo que estamos haciendo es referirnos a unas posiciones de memoria (es decir un lugar de la memoria), que han sido reservadas cuando se produjo la declaración de ésta.

- Prog098

```
/* Prog098.cpp */

#include <stdio.h>

void main()
{
int num;
num=78;
printf("\n La dirección de memoria de la variable num es %p",&num);
printf("\n El valor de la variable num es %d",num);
}
```

Graba el programa **PROG098**, en *TuCarpeta*.

Ejecútalo varias veces.

Estudio del **PROG098**:

- La dirección de memoria de la variable **num**, no es más que un numero en sistema de numeración hexadecimal (por eso aparecen letras), que indica el lugar de la memoria donde se guarda el valor de la variable.
- Debe quedar clara la diferencia entre **valor de una variable y su dirección de memoria**.
- **Identificador de formato:**
  Para el **valor** de una variable **int** es %d
  Para la **dirección de memoria** es %p
- Para acceder a una variable:
  Nombre de la variable: accedemos a su valor
  &nombre de la variable: accedemos a su dirección de memoria.

## e) Sistema de numeración hexadecimal:

Una dirección de memoria viene dada por un número en sistema hexadecimal. Veamos como funciona esto…

**Sistema Binario:**

El sistema de numeración que nosotros conocemos es el **decimal** (sistema de numeración en base 10). Se llama decimal porque cualquier número lo podemos expresar utilizando únicamente los 10 dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

El sistema binario (sistema de numeración en base 2) utiliza sólo dos dígitos: 0, 1

El problema es cómo pasar del sistema binario al decimal, para poder entendernos:

Sea **1011** un número binario, para hallar su equivalente en decimal hemos de hacer lo siguiente:

$$1011 = 1x2^0 + 1x2^1 + 0x2^2 + 1x2^3 = 1 + 2 + 8 = 11$$

El número **1011** en base 2, es el número **11** en base 10.

1 **Bit** es un 1 o un 0, que representa en informática, un circuito electrico por el que pasa corriente (1) o no (0).

1 **Byte** = 8 **Bits** por esta razón a 1 byte se le llama también **octeto** (8).

1 **Byte** representa un carácter, se alfabético, numérico o especial.
Por ejemplo:
El Byte: 01000100 = 4 + 64 = 68 (en decimal) corresponde al carácter **D** (código ASCII= 68)
El Byte: 00101101 = 45 (en decimal) corresponde al carácter **–** (signo menos) y se corresponde con el código ASCII= 45
El byte: 00111001 = 57 corresponde al carácter 9 (dígito 9) y se corresponde con el código ASCII = 57

**Sistema hexadecimal:**

El sistema hexadecimal utiliza 16 dígitos, que son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

El número **4A0** en sistema hexadecimal será **1184** en sistema decimal, porque:

$4A0 = 0x16^0 + Ax16^1 + 4x16^2 = 0x1 + 10x16 + 4x256 = 1184$

La utilidad del sistema hexadecimal está en que utilizamos menos dígitos para expresar un número mayor: El número 1184 en decimal ocupa 4 dígitos, encambio el mismo número en hexadecimal (4A0) ocupa 3.

Volviendo a nuestro lenguaje de programación: en una variable hemos de distinguir entre su valor (contenido de la celdilla o celdillas de memoria) y su dirección de memoria (lugar de la celdilla o celdillas = número en hexadecimal).

## f) Punteros (pointers)

- Prog099

```
/* Prog099.cpp */

#include <stdio.h>

void main()
{
int *pint;
int *pint2;

printf("\n Dirección de memoria de pint: %p \n",pint);
printf("\n Dirección de memoria de pint2: %p \n",pint2);

*pint=10;
*pint2=25;

printf("\n Valor de pint: %d \n",*pint);
printf("\n Valor de pint2: %d \n",*pint2);

pint2=pint;
printf("\n Atención acabo de igualar los dos punteros \n");

printf("\n Dirección de memoria de pint: %p \n",pint);
printf("\n Dirección de memoria de pint2: %p \n",pint2);

printf("\n Valor de pint: %d \n",*pint);
printf("\n Valor de pint2: %d \n",*pint2);
}
```

Estudio del **PROG099:**

- Un puntero (pointer) es una variable cuyo contenido no es un valor, sino una dirección de memoria.
- La forma de declarar un puntero es igual que para declarar una variable normal y corriente pero anteponiendo un asterisco.

De esta forma: **int *pint;**
**Int *pint2;**
declaramos 2 punteros, es decir 2 direcciones de memoria que contendran datos enteros (int).

- **printf("\n Dirección de memoria de pint: %p \n",pint);**
  Debido a que **pint** es un puntero, en la instrucción anterior no hemos de anteponer **&**.
  Aunque sí utilizar el identificador de formato correspondiente (%p).
  Está claro que:
  **int pepe;**
  **printf("\n Dirección de memoria de pepe: %p \n", &pepe);**
  sería equivalente.
- Para acceder al dato correspondiente a un puntero, hemos de anteponer un asterisco. De esta forma: ***pint=10;** sirve para asignar a la variable puntero (pint) el valor entero **10**.
- Si observas detenidamente el resultado del programa **PROG099**, conseguimos "eliminar" la dirección de memoria de la variable **pint2**.

Vamos a intentar hacer un programa que haga lo mismo que el anterior, pero sin utilizar **punteros**.

- Prog100

```
/* Prog100.cpp */

#include <stdio.h>

void main()
{
int pint;
int pint2;

printf("\n Dirección de memoria de pint: %p \n",&pint);
printf("\n Dirección de memoria de pint2: %p \n",&pint2);

pint=10;
pint2=25;

printf("\n Valor de pint: %d \n",pint);
printf("\n Valor de pint2: %d \n",pint2);

pint2=pint;
printf("\n Atención acabo de igualar las dos variables \n");

printf("\n Dirección de memoria de pint: %p \n",&pint);
printf("\n Dirección de memoria de pint2: %p \n",&pint2);

printf("\n Valor de pint: %d \n",pint);
printf("\n Valor de pint2: %d \n",pint2);
}
```

Grábalo con el nombre **PROG100**

Ejecútalo varias veces y compáralo con el **PROG099**

Estudio del **PROG100:**

- Básicamente el PROG100 es equivalente al PROG099, pero con una diferencia fundamental: En el PROG099 igualamos el valor de dos variables, pero eliminando físicamente una dirección de memoria.
- En el PROG100 igualamos 2 variables pero continuamos con dos variables de memoria.
- Podríamos concluir: "el uso de punteros permite ahorrar memoria". Próximamente veremos que el uso de punteros (característico del lenguaje C), sirve para bastantes más cosas.

- Al escribir: **int peps;** el compilador reserva una posición de memoria para albergar la dirección de una variable "int" y de nombre **"***peps". Es decir: "peps" guardará la dirección de memoria de la variable entera "*peps", se dice que "peps" apunta a la variable entera "*peps".

- Al escribir:

```
int variab,*pint;
pint=&variab;
```

  diremos que "pint" apunta a la variable "variab".
  Ante la orden **"int variab"** el compilador "reserva" un grupo de bits (los correspondientes a un número entero), podríamos definir **una variable como un conjunto de bits**. Encambio "pint" es una **dirección de memoria**, en nuestro ejemplo: la dirección de memoria de la variable "variab" (pint=&variab).

- Si consideras el siguiente programa:

```
/* Prog100b.cpp */
#include <stdio.h>
void main()
{
char var1;
char *pchar;
pchar=&var1;
for (var1='a';var1<='z';var1++) printf("%c",*pchar);
}
```

  Evidentemente seria equivalente en lugar de **"printf("%c",*pchar)"** escribir **"printf("%c",var1)"**
  Observa que en la línea: **pchar=&var1** inicializamos el puntero (asignamos a **pchar** la dirección de **var1**) y automáticamente inicializamos "*pchar" (asignamos a "*pchar" la variable "var1"), aunque el valor de **var1** no lo asignamos hasta **for(var1='a'; ...)**

## g) Funciones Recursivas

- Prog101

```
/* Prog101.cpp */

#include <stdio.h>

long fact(int n);

void main()
{
int num;
printf("\n Introduce un número entero: ");
scanf("%d",&num);
printf("\n El factorial de %d es %ld", num,fact(num));
}

long fact(int n)
{
if (n<=1) return 1;
      else return n*fact(n-1);
}
```

Graba el programa con el nombre **PROG101**, en *TuCarpeta*.

Ejecútalo varias veces. Probablemente el programa no funcionará para valores mayores que 16 (piensa que es una función que crece muy aprisa).

Estudio del **PROG101:**
Una función recursiva es una función que se llama a sí misma: observa que en el "interior" de la función **fact()**, hay una llamada a sí misma: **n*fact(n-1)**

- Prog102

```
/* Prog102.cpp */

/* Sucesión de Fibonacci (forma recursiva)
   0,1,1,2,3,5,8,13,....  */

#include <stdio.h>

int fibona(int v);

void main()
{
int r,valor;
printf("0");printf("\n1");printf("\n1");
for(valor=1;valor<=20;valor++)
      {
      r=fibona(valor);
```

```
        printf("\n%d",r);
        }
}

int fibona(int v)
{
if((v==0)||(v==-1)) return 1;
else
return fibona(v-1)+fibona(v-2);
}
```

## h) Punteros y Funciones

- Prog103

```
/* Prog103.cpp */

#include <stdio.h>

void main()
{
int v1,v2,aux;
printf("\n Primer  valor ? "); scanf("%d",&v1);
printf("\n Segundo valor ? "); scanf("%d",&v2);

aux=v1;
v1=v2;
v2=aux;

printf("\n\n Nuevo primer  valor: %d",v1);
printf("\n\n Nuevo segundo valor: %d",v2);
}
```

Estudio del **PROG103:**

- El programa lo que hace es simplemente intercambiar los valores de 2 variables
- Recuerda (ya lo hicimos en el PROG088), que el intercambio de valores se consigue gracias al uso de una variable auxiliar:

  aux=v1;
  v1=v2;
  v2=aux;

  v1 adquirirá el valor de v2 y v2 tomará el valor de v1.
- El problema del "intercambio" se encuentra en el caso de una función, como veremos a continuación.

Se trata de hacer el programa anterior, pero el proceso de "intercambio" lo pondremos en una función.

- Prog104

```
/* Prog104.cpp */

#include <stdio.h>

void cambiar(int v1,int v2);

void main()
{
int v1,v2,aux;
printf("\n Primer  valor ? "); scanf("%d",&v1);
printf("\n Segundo valor ? "); scanf("%d",&v2);
cambiar(v1,v2);
printf("\n\n Nuevo primer  valor: %d",v1);
printf("\n\n Nuevo segundo valor: %d",v2);
}

void cambiar(int v1,int v2)
{
int aux;
aux=v1;
v1=v2;
v2=aux;
}
```

Si todo funciona correctamente verás que el programa **PROG104** no funciona, es decir: "no se intercambian los dos valores"..

Vamos a solucionar el problema, se trata de hacer el mismo programa **PROG104**, pero trabajando con punteros.

- Prog105

```
/* Prog105.cpp */

#include <stdio.h>

void cambiar(int *pv1,int *pv2);

void main()
{
int v1,v2,aux;
printf("\n Primer  valor ? "); scanf("%d",&v1);
printf("\n Segundo valor ? "); scanf("%d",&v2);
cambiar(&v1,&v2);
printf("\n\n Nuevo primer  valor: %d",v1);
printf("\n\n Nuevo segundo valor: %d",v2);
}

void cambiar(int *pv1,int *pv2)
{
int aux;
aux=*pv1;
*pv1=*pv2;
*pv2=aux;
}
```

Graba el programa con el nombre **PROG105**, en *TuCarpeta*.

Ejecútalo varias veces.

Observa que el uso de punteros no sirve sólo para ahorrar memoria.

## i) Punteros y Arrays

- Prog106

```
/* Prog106.cpp */

#include <stdio.h>

void main()
{
int i,t[5],*pun;
for(i=0;i<5;i++) t[i]=i;
printf("\n Listado del array: \n");
for(i=0;i<5;i++) printf("\n%d",t[i]);
printf("\n Listado del array, pero a través de punteros: \n");
for(i=0;i<5;i++)
 {
 pun=t+i;
 printf("%d\n",*pun);
 }
}
```

Estudio del **PROG106:**

- El nombre de un array (t, en nuestro caso) es sinónimo de la dirección de memoria del primer byte de sus elementos. Por lo tanto: **pun=t+i** al variar **i= 0, 1, 2, 3, 4** no es más que las direcciones de memoria de los 5 valores del array.

- Prog107

```
/* Prog107.cpp */

#include <stdio.h>

void main()
{
int i,t[5],*pun;
for(i=0;i<5;i++) t[i]=i;
```

```
printf("\n Listado del array: \n");
for(i=0;i<5;i++) printf("\n%d",t[i]);
pun=&t[4];
printf("\n Array en orden inverso: \n");
for(i=0;i<5;i++)
      {
      printf("%d\n",*pun);
      pun=pun-1;
      }
}
```

- Prog108

```
/* Prog108.cpp */

#include <stdio.h>

void main()
{
int i,j,mat[5][5];
for(i=0;i<5;i++)
      for(j=0;j<5;j++)
            mat[i][j]=5*i+j;
printf("\n Listado de la matriz:\n");
for(i=0;i<5;i++)
      for(j=0;j<5;j++)
            printf("%d-",mat[i][j]);
printf("\n Listado de la matriz pero utilizando punteros:
\n");
for(i=0;i<5;i++)
        {
      for(j=0;j<5;j++)
            printf("%d-",*(*(mat+i)+j));
      printf("\n");
        }
}
```

Compara el PROG106 y el PROG108, parecen complicados ¿verdad?. Ciertamente que sí, pero debes tener en cuenta las siguientes reglas:

1) El nombre de un array unidimensional es un sinónimo de la dirección de memoria de su primer byte.
2) El contenido del elemento "i" de un array unidimensional de nombre "pepe" se obtiene como ***(pepe+i)**
3) Una tabla bidimensional no es sino una matriz unidimensional cuyos elementos son, a su vez, arrays unidimensionales.

- Prog109

```
/* Prog109.cpp */
```

```
#include <stdio.h>

float invertir(float *numero);

void main()
{
float i;
printf("\n Escribe un número:  "); scanf("%f",&i);
if (invertir(&i) != 0) printf("\n El inverso es %f",i);
      else printf("\n No tiene inverso");
}

float invertir(float *numero)
{
if(*numero==0) return 0;
      else
      {
      *numero=1/(*numero);
      return *numero;
      }
}
```

- Prog110

```
  /* Prog110.cpp */

  /* Ejemplo de funciones por valor y por referencia (dirección de
memoria) */
  /* Calcula dos veces el procentaje de gastos, la primera vez
  utilizando una función por valor y la segunda por referencia
(punteros) */

  #include<stdio.h>
  #include <conio.h>

  void porcentaje_xvalor(float ingreso, float egreso);
  void porcentaje_xref(float *ingreso,float *egreso);


  void main()
  {
      float entrada,salida;
     clrscr();
      printf("Entradas: ");
     scanf("%f",&entrada);
      printf("Salidas: ");
     scanf("%f",&salida);
      porcentaje_xvalor(entrada,salida); /*Llamada a la  función
porcentaje
                        utilizando paso de parámetros  por valor*/
     printf("\n\n");
      porcentaje_xref(&entrada,&salida); /*Utilización de la función
porcentaje
                         con paso de parámetros por referencia*/
      getch();
  }
```

```
void porcentaje_xvalor(float ingreso, float egreso)
{
    egreso=((egreso/ingreso)*100);
   printf("Usted gasta el %.2f por ciento de lo  que gana",egreso);
}

void porcentaje_xref(float *ingreso,float *egreso)
{
   *egreso=(((*egreso)/(*ingreso))*100);
   printf("Usted gasta el %.2f por ciento de lo  que gana",*egreso);
}
```

## j) Variables dinámicas

Hasta ahora teníamos una serie de variables que declaramos al principio del programa o de cada función. Estas variables que reciben el nombre de **estáticas**, tienen un tamaño asignado desde el momento en que se crea el programa.
Pensemos en la "programación de una agenda": tenemos una serie de fichas y nos interesa añadir más. Si pensamos en **variables estáticas**, prepararíamos la agenda para 1000 fichas aunque creamos que no vamos a pasar de 300. Está claro que esto es desperdiciar memória.

El uso de **variables dinámicas** se basa en reservar un poco de memoria para cada nuevo elemento que nos haga falta y enlazarlo a los que ya teníamos. Cuando queramos borrar un elemento, enlazamos el anterior a él con el posterior a él y liberamos la memoria que estaba ocupando.

Así que para seguir, necesitamos saber cómo reservar memoria y cómo liberarla.

Y entre otras cosas necesitamos **"los punteros"**….

Recordemos:

```
int num;        // "num" es un número entero
int *pos;       // "pos" es un puntero a entero = dirección de memoria en la que podemos guardar
                // un entero.
num=1;          // ahora "num" vale 1
pos=1000;       // "pos" ahora es la dirección 1000. Es peligroso ya que en la dirección de memoria
                // que contiene "pos" no sabemoa lo que hay y podemos provocar una catástrofe.
*pos=25;        // en la dirección de memoria de "pos" guardamos un 25
pos=&num;       // la variable "pos" contiene la dirección de memoria de la variable "num".
```

En la práctica "pedimos" al compilador que nos reserve un poco de memoria donde él crea adecuado, utilizando la función **"malloc"**, que se encuentra en el fichero **stdlib.h**
Una vez hemos utilizado esta memoria es conveniente liberarla, utilizando la función **"free"**

Veamos:

- Prog111

```
/* Prog111.cpp */
```

```
#include <stdio.h>
#include <stdlib.h>

int num;
int *pos;

void main()
{
printf("num vale: %d (arbitrario)\n",num);
printf("La dirección pos es: %p (arbitrario)\n",pos);
num=1;
printf("num vale: %d (fijado)\n",num);

pos=(int *) malloc(sizeof(int)); /* reservamos espacio */
printf("La dirección pos es: %p (asignado)\n",pos);
printf("El contenido de pos es: %d (arbitrario)\n",*pos);
*pos=25;
printf("El contenido de pos es: %d (fijado)\n",*pos);
free(pos);
pos=&num;
printf("Y ahora el contenido de pos es: %d (valor de num)\n",*pos);
}
```

free(pos);
libera la memoria que ocupaba "pos".

pos=(int *) malloc(sizeof(int));
sizeof(int): espacio a reservar, como el tamaño debe corresponder a un entero. sizeof(int) corresponde al tamaño de un entero.

"malloc" nos devuelve un puntero sin tipo (void *), como queremos guardar un dato entero, hemos de hacer una **conversión de tipos**: de "puntero sin tipo (void *)" a "puntero a entero (int *)".

Una vez ejecutado el programa, observemos:

- Inicialmente "num" vale 0 (no podemos fiarnos de que siempre sea así, ya que no lo hemos inicializado nosotros)
- Inicialmente "pos" es 0. Es un puntero nulo, para el cual no se ha asignado un espacio en memoria.

- Prog112

```
/* Prog112.cpp */

/* Creación de un array dinámicamente */

#include <stdio.h>
#include <stdlib.h>

int *num;  /* puntero a numero entero */
int *temporal;   /* temporal, para recorrer el array */
int i;           /* para bucles */

void main()
{
/* Reservamos espacio para 10 números (array dinámico) */
num=(int *) malloc(10*sizeof(int));
```

```
for(i=0;i<10;i++)      /* Recorremos el array */
     num[i]=i*2; /* Dando valores */
printf("La dirección de comienzo del array es: %p\n",num);
printf("Valores del array: ");
for(i=0;i<10;i++)           /* Recorremos el array */
     printf(" %d ",num[i]);    /* mostrando los valores */
printf("\n Valores del array (como puntero): ");
temporal=num;
for(i=0;i<10;i++)           /* Recorremos como puntero */
     printf(" %d ",*temporal++); /* mostrando los valores y
                                 aumentando */
free(num); /* Liberamos lo reservado */
}
```

Como se ve, en "C" hay muy poca diferencia entre arrays y punteros: hemos declarado "num" como un puntero, pero hemos reservado espacio para más de un dato, y hemos podido recorrerlo como si hubiésemos definido: **int num[10];**

## f) Cadenas de texto

Una cadena de texto en "C" es un array de caracteres
Como a todo array, se le puede reservar espacio estáticamente o dinámicamente:

Estáticamente: char texto[80];
Dinámicamente: char *texto; reservando espacio con "malloc" cuando nos haga falta.

De todas formas una cadena de caracteres siempre terminará con un carácter nulo (\0)
Es decir:
Si necesitamos 7 letras para un teléfono: char telefono[8];
7 letras del teléfono + \0

Para copiar el valor de una cadena de texto en otra, no podemos hacer: **texto1 = texto2**, porque estaríamos igualando dos punteros. En vez de eso, debemos usar una función de biblioteca: **strcpy** que se encuentra en **string.h**

Strcpy(destino, origen);

El problema está en que en la cadena destino haya suficiente espacio reservado para copiar lo que queremos:

- Si queremos copiar sólo los primeros "n" bytes de origen, usamos:
  strncpy(destino,origen,n);

- Para añadir una cadena al final de otra (concatenarla), usamos: **strcat(origen,destino);**

- Para comparar dos cadenas alfabéticamente (para ver si son iguales o para ordenarlas, por ejemplo), usamos:
  **strcmp(cad1,cad2);**
  = 0 si las cadenas son iguales.
  <0 si cad1<cad2
  >0 si cad1>cad2

- Según el compilador tenemos: **strupr(cadena);** que convierte la cadena a mayúsculas

- Prog113

```
/* Prog113.cpp */

/* Cadenas de texto */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char texto1[80]="Hola";  /* Cadena Estática */
char *texto2;            /* Cadena Dinámica */

void main()
{
/* Reservo espacio para la cadena dinámica */
texto2=(char *) malloc (70*sizeof(char));
strcpy(texto2,"Adios");  /* Le doy un valor */
puts(texto1);
puts(texto2);            /* Escribo las dos */
strncpy(texto1,texto2,3);      /* copio las 3
                          primeras
                          letras */
puts(texto1);
strcat(texto1,texto2);   /* añado text2 al final */
puts(texto1);
printf("Si las comparamos obtenemos: %d",strcmp(texto1,texto2));
printf("(Numero negativo: texto1 es menor) \n");
printf("La longitud de la primera es %d \n", strlen(texto1));
printf("En mayúsculas es %s \n",strupr(texto1));
free(texto2);
}
```

- Prog114

```
/* Prog114.cpp */

/* Programa que analiza lo que escribimos: dígitos numéricos
   o caracteres alfabéticos. */


   #include <stdio.h>
   #include<string.h>
   #include <stdlib.h>
   #include <conio.h>


   int menu();
   /* función que hace aparecer un menú con dos opciones:
      1: sólo números, 2: sólo letras. Devuelve 1 o 2 */

   int numeros();
   /* función que lee dígitos numéricos en forma de caracteres
      y devuelve el valor en entero. */

   void captura(char palabra[]);
```

```
   /* función que lee en forma de caracteres y analiza si
      son letras */


   void main()
   {
       int cifra;
      char word[20];
      clrscr();

     switch(menu())
     {
        case 1: cifra=numeros();
        printf("\n%d",cifra);
        break;
        case 2: captura(word);
        printf("\n%s",word);
        break;
        default: exit(0);
     }
    getch();
   }



   void captura(char palabra[])
   {
     char *letra;
     char
alfabeto[]="ABCDEFGHIJKLMNÑOPQRSTUVWXYZabcdefghijklmnñopqrstuvwxyz";
     int i;

     palabra[0]='\0';
     clrscr();
     do
     {
          *letra=getch();
          for (i=0;i<=53;i++)
          {
              if (alfabeto[i]==*letra)
             {
                    printf("%c",*letra);
                strcat(palabra,letra);
                /* la función "strcat" añade "letra" a "palabra" y
añade
                   al final el caracter nulo. Se encuentra en
<string.h> */

                    break;
             }
           }
     }while((*letra!=13) && (strlen(palabra)<20));
   }



   int numeros()
   {
      char cadena[10];
```

```
      char car='\0';
      int i=0;
      int cantidad;

      do
      {
          car=getch();
          switch(car)
          {
              case'0': cadena[i]=car;
                          printf("%c",car);
                          break;
              case'1': cadena[i]=car;
                       printf("%c",car);
                          break;
              case'2': cadena[i]=car;
                          printf("%c",car);
                          break;
              case'3': cadena[i]=car;
                       printf("%c",car);
                          break;
              case'4': cadena[i]=car;
                          printf("%c",car);
                          break;
              case'5': cadena[i]=car;
                          printf("%c",car);
                          break;
              case'6': cadena[i]=car;
                          printf("%c",car);
                          break;
              case'7': cadena[i]=car;
                          printf("%c",car);
                          break;
              case'8': cadena[i]=car;
                          printf("%c",car);
                          break;
              case'9': cadena[i]=car;
                          printf("%c",car);
                          break;
             default: i--;
                       break;
          }
       i++;
      }while((car!=13) && (i<5));
          cantidad=atoi(cadena);
          /* la función "atoi" devuelve el número entero
correspondiente
               a "cadena". Se encuentra en <stdlib.h> */

            return(cantidad);
      }



   int menu()

   {
        int numero;
         printf("Escoge una opción:\n");
```

```
        printf("1. Escribir sólo   números\n");
        printf("2. Escribir únicamente letras\n");
        printf("Opción: ");
        scanf(" %d",&numero);
        return(numero);
}
```

- Prog115

```
/* Prog115.cpp */


/* Programa que demuestra el procedimiento copia el cual,
   copia una cadena en  otra*/

#include<string.h>
#include <conio.h>
#include<stdio.h>


char *copia(char *cad1,char *cad2);

void main()
{
    char palabra1[10];
    char palabra2[10];
    char palabra3[20];
   printf("palabra1= ");
   scanf("%s",palabra1);
   printf("palabra2= ");
   scanf("%s",palabra2);
   copia(palabra2,palabra1);
   printf("palabra1+palabra2= %s",palabra1);
   getch();
}


char *copia(char *cad1, char *cad2)
{
   char *inicio;
   int i;
   inicio=cad2;
   while(*cad2!='\0')
   cad2++;
   while(*cad1!='\0')
    {
        *cad2=*cad1;

 cad2++;
        cad1++;
    }
  *cad2='\0';
  cad2=inicio;
}
```

## g) Estructuras (o registros)

Una estructura es un nuevo tipo de dato, que consta de una agrupación de datos (como los arrays), pero de distinto tipo (a diferencia de los array).

- Prog116

```
/* Prog116.cpp */

/* Uso de "estructuras". */

#include <stdio.h>

struct {
      char inicial;
      int edad;
      float nota;
      } persona;


void main()
{
persona.edad=20;
printf("La edad es %d",persona.edad);
}
```

# AUTOEVALUACIÓN 4

1) Haz un programa para calcular el área de un círculo, utilizando una constante simbólica para el número PI. Graba el programa con el nombre **EVAL4A**

2) Haz un programa que "simula una tirada aleatoria de 3 dados de parchís" utilizano la función **rand()**. Graba el programa con el nombre **EVAL4B**.

3) Utilizando la función **rand()** haz un programa que simule una jugada de los dados de póker. Graba el programa con el nombre **EVAL4C**.

4) Utilizando matrices bidimensionales haz un programa que resuelva por el método de Cramer un sistema de 2 ecuaciones con 2 incógnitas. Graba el programa con el nombre **EVAL4D**.

5) Utilizando matrices bidimensionales haz un programa que calcule la media de las temperaturas máxima y mínima de una semana para un "lugar determinado". Graba el programa con el nombre **EVAL4E**.

6) Haz un programa que nos pida los elementos de una matriz de orden 4 y por último la escribe en pantalla (bien escrita). Graba el programa con el nombre **EVAL4F**.

7) Haz un programa que al escribir un número en base 2 o base 3 lo escribe en decimal. Graba el programa con el nombre **EVAL4G**.

8) Haz un programa que multiplique 3 números y nos muestre la dirección de memoria de cada uno de los 3 números y también del producto. Graba el programa con el nombre **EVAL4H**.

9) Haz un programa igual que el anterior pero utilizando punteros (**EVAL4I**).

10) Haz un programa que calcule los 20 primeros términos de la sucesión de término general:

$$\frac{3n+1}{2n-1}$$

utilizando una función **recursiva** (**EVAL4J**).

11) Haz un programa igual que el **EVAL4D** pero utilizando una función con punteros que calcule los determinantes de orden 2 (**EVAL4K**).

12) Haz un programa que escriba el vector (1, 2, 6, 24, 120, ...) de 7 componentes utilizando punteros (**EVAL4L**).

13) Haz un programa que escriba la matriz:

$$\begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}$$

utilizando punteros (**EVAL4M**).

14) Haz un programa que calcule el cubo de un número, utilizando una función de argumento un puntero (**EVAL4N**).

15) Haz un programa que calcule la raiz cuarta de un número, utilizando una función con argumento un puntero (**EVAL4O**).

**Hasta aquí, la versión no registrada del manual.**

**Si deseas la parte que falta, es decir:**



**Debes adquirir la versión registrada, es decir entera.**
**Es muy fácil, has de hacer lo siguiente:**

1) **Rellena el siguiente formulario con tus datos:**

**Nombre y Apellidos:**

**Dirección:**

**Código Postal:** **Población:**

**Versión completa del "C/C++ (Manual FV)"**

2) **Envíame el formulario anterior por correo ordinario junto con un "billete" (lo que consideres justo por un disquete, gastos de manipulación, envío y "estímulo por mi parte para que continue colgando en Internet, mis manuales").**

   **A mi dirección que es:**

   **F.V.**
   **c) Valencia 21-25, 2º , 4ª**
   **08915 – Badalona (Barcelona)**
   **España**

3) **A vuelta de correo recibirás en tu dirección, un disquete con la versión completa del manual "C/C++ (Manual FV)".**