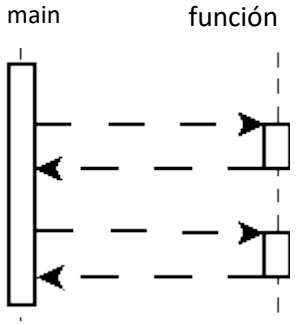
Universidad de Morón
Escuela Superior de Ingeniería, Informática y Cs. Agroalimentarias

Asignatura:

(701) Programación II

CLASE 10

Funciones
Parámetros por dirección



Prof. Lic. Sonia Zugna de Jausoro

Tips: Color de pantalla y de texto

```
#include <stdlib.h>

system("color cc");      // system en libreria stdlib
```

Código representa el **color de fondo** y **Código** representa el **color del texto**.

Códigos:

a: verde
b: celeste
c: rojo
d: fucsia
e: amarillo
f: blanco

Códigos:

0 = Negro
1 = Azul
2 = Verde
3 = de Aqua
4 = Red
5 = púrpura
6 = Amarillo
7 = Blanco
8 = Gris
9 = Light Blue

Ejemplos:

system(“color **12**”); fondo azul y texto verde.

System(“color **a1**”); fondo verde y letras azul

```cpp
// Prueba de cambio de COLOR de fondo y de texto

#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <iomanip>
#include <stdlib.h>   // para usar función system - color
using namespace std;


main()
{
 int x = 10;
 int y = 20;

 system("color f2");      // cambia el color: Consola ( f: blanco) - Texto (2: verde)

 cout << endl << endl;

 cout << "    Hola Alumnos: " << endl << endl;

 cout << "    Feliz Primavera.!!!" << endl << endl ;


}
```

Tipo de dato: **Puntero** *(pointer, en inglés)*

- ✓ es un **apuntador**, es un **tipo de dato** provisto por el lenguaje para "apuntar" a algo.
- ✓ Lo "apuntado" es una dirección de memoria, donde puede almacenarse un valor de un tipo determinado.
  - *Ejemplo:* un apuntador a enteros es la variable puntero que apunta a una posición donde se almacena un entero, etc.

- ❖ *Un puntero es una variable que contiene la dirección de memoria de otra variable.*

- ✓ Una variable de tipo puntero contiene la dirección (también podemos decir la referencia) de un lugar en la memoria.

- ✓ El tipo de datos puntero es considerado un tipo dinámico.

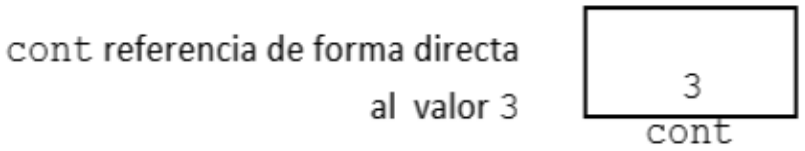
- ➢ *Se usan para:*

1) pasar información entre una función y sus puntos de llamada.

2) armar en memoria estructuras dinámicas de datos (pilas, colas, etc.)

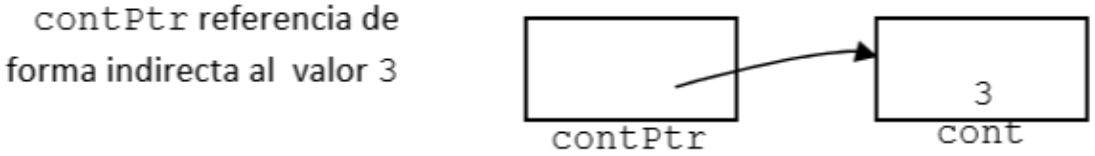
La declaración int **cont** = 3;

hará que el compilador reserve lugar en la memoria, en donde se almacenará el valor 3 de la variable **cont**.

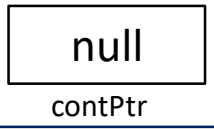


Una variable de **tipo puntero** tomará valores que son referencias (direcciones o apuntadores) a:

1) posibles valores de un tipo dado o



2) un valor particular llamado **puntero nulo**, representado en C por la constante **NULL**, definida en la biblioteca estándar ***stdio.h***, la cual indica que la variable puntero "no apunta a nada".



***Ejemplo*:**

Declaración de una variable "**punte**" **de tipo "puntero a entero",** *inicializada en NULL*:

```
int *punte = NULL;
```

→ Esto hará que el *compilador* reserve un lugar en memoria para la variable "punte", la cual contendrá el valor NULL.

---

1) Se trata de un **puntero a entero**. También podríamos inicializarlo con la dirección de una variable entera, por ejemplo, con la dirección de la variable aux. (definida: int aux;)

2) El ***operador de dirección*** **&**, permite obtener la dirección de una variable:

```
int *punte = &aux;
```

Pasaje de Parámetros

- ✓ Los parámetros *pueden ser pasados* a la función de las formas:

1) **por valor**
2) **por dirección o referencia**

- ✓ En el **pasaje por valor**, se pasa a la función una **copia del dato**.
  Todo cambio que la función haga dentro de ella, **no** se verá reflejado en el parámetro real (el original).

- ✓ En el **pasaje por dirección o referencia** sería equivalente a pasar la variable propiamente dicha donde se encuentra el dato *y no una copia* del dato.
  Se pasa la **dirección** del parámetro real.
  → *Por ello todo cambio que se haga sobre el parámetro formal se verá reflejado en el parámetro real (parámetro actual).*
  → el correspondiente parámetro actual ***debe ser siempre una variable***.

Ejemplo: "pasarle" un apunte de una clase a un compañero:
si le doy una fotocopia (éste es el pasaje por valor),
o le doy el cuaderno mismo (pasaje por dirección).

Ejemplo Intercambia → fallido…

```
void intercambia(int  a, int b)
{
int aux;
aux= a;
 a = b;
 b = aux;
}
int main()
{
 int  x = 10;
 int y = 20;
cout << "x= " << x  << "  y= " << y ;
 intercambia (x,y) ;
cout << "x= " << x  << "  y= " << y ;
}
```

→ La Función NO logra intercambiar los valores de x e y. Por qué???

Ejemplo Intercambia → OK

```
// Función que intercambia 2 variables
// pasaje de parámetros por REFERENCIA
#include<iostream.h>
#include<conio.h>
#include<stdio.h>

void intercambia(int  &a, int  &b)
{
int aux;
aux= a;
 a = b;
 b = aux;
}

int main() {
 int x = 10;
 int y = 20;
cout << "x= " << x  << "  y= " << y << endl ;
 intercambia (x , y) ;
cout << "x= " << x  << "  y= " << y ;

getch();
return(0);
}
```

→ La Función Sí logra intercambiar los valores de x e y.

Ejemplo Intercambia → OK

```
// Función que intercambia 2 variables
// pasaje de parámetros por DIRECCIÓN
#include<iostream.h>
#include<conio.h>
#include<stdio.h>

void intercambia(int  *a, int  *b)
{
int aux;
aux= *a;
 *a = *b;
 *b = aux;
}

int main() {
 int x = 10;
 int y = 20;
cout << "x= " << x  << "  y= " << y << endl ;
 intercambia (&x , &y) ;
cout << "x= " << x  << "  y= " << y ;

getch();
return(0);
}
```

→ La Función Sí logra intercambiar los valores de x e y.

Ejercicio de ingenio

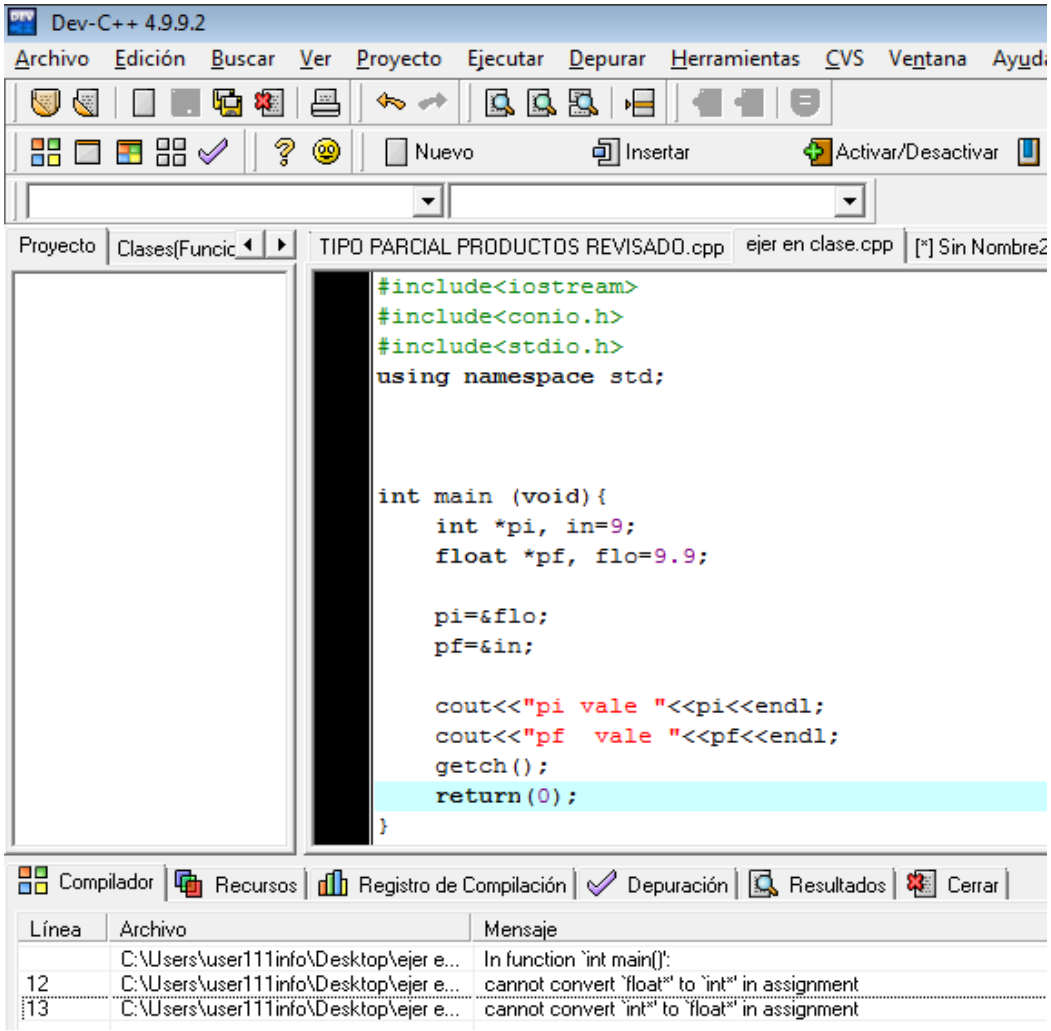
```
#include <iostream>
#include <conio.h>
using namespace std;

int fUno (int &y)
{
        y = y*2;
        return y;
}

int fDos (int x)
{
        x = x*x;
        return x;
}

int main ()  {
    int nro, x1;
    cout << "Ingrese un numero del 1 al 10" << endl;
    cin >> nro;
    x1 = fUno(nro);
    cout << "El doble del número ingresado es: " << x1 << endl;
    x1 = fDos(nro);
    cout << x1 << endl;
    cout << "El cuadrado del número ingresado  es: " << x1 << endl;
    getch();
    return 0;
}
```

→ Hay algún error?



Da error de COMPILACIÓN:

- asignar a una variable entera, un apuntador a flotante
- asignar a una variable flotante, un apuntador a entero

Uso de los parámetros:

- ✓ Entrada
- ✓ Salida
- ✓ Entrada/Salida

✓ **Entrada:**

Ej.: Función que dado tres valores , devuelva el mayor de ambos.

int mayor (int n1, int n2, int n3)

✓ **Salida**: *Ej.: Idem función*

Opc.1: A través del nombre de la función: int mayor (int n1, int n2, int n3)

Opc.2: A través de un parámetro de salida: void mayor (int n1, int n2, int n3, int &max)

(un 4to. Parámetro de uso de salida estrictamente y función de tipo void)

✓ **Entrada/Salida**:

Ej.: Función que dado un valor, lo devuelva incrementado en un 50 %.

void increm50porc (float &nro) (*Modifica el contenido original.)*

Pregunta:

- Un parámetro **de uso estrictamente de Entrada**, puede ser un parámetro pasado a la función, por referencia o por dirección?

- La respuesta es…

Funciones que devuelven ***más*** *de un resultado*.

Ej.: Función que dado 3 valores enteros, devuelve <u>3 valores</u> el mayor, el menor y la suma.

<u>Opción 1:</u>

int *fMayMenSum (int n1, int n2, int n3, int &min, int &max)*

Devuelve:

- *1 valor a través del nombre de la función (por ejemplo la suma)*
- *Y los otros 2 resultados los devuelve a través de 2 parámetros estrictamente de salida.*

<u>Opción 2:</u>

void *fMayMenSum (int n1, int n2, int n3, int &min, int &max, int &sum)*

Devuelve:

- *Nada a través del nombre de la función (void).*
- *Y los 3 resultados los devuelve a través de 3 parámetros estrictamente de salida.*

Qué pueden ser los parámetros ACTUALES ?

El parámetro ACTUAL de un parámetro FORMAL **por valor o por copia**, puede ser:

- ✓ Una constante
- ✓ Una expresión
- ✓ Una variable
- ✓ Una llamada a otra función del mismo tipo de dato que el parámetro.

Ejemplo:

Para la cabecera de función: Int suma (int a, int b)

En las llamadas, son parámetros actuales VÁLIDOS:

resu = suma (7, 8); *// dos constantes*

resu = suma (nro, 6); *// una variable y una constante*

resu = suma (nro+2, nro-3); *// expresiones aritméticas del mismo tipo de dato que el parámetro*

resu = suma (suma(n1, n2), suma(n3, n4)); *// llamadas a funciones del mismo tipo de dato que el parámetro*

El parámetro ACTUAL de un parámetro FORMAL **por referencia o por dirección**, **SIEMPRE DEBE ser**:

- ✓ Una variable

Funciones *printf* y *scanf* de entrada/salida en C

Incluyen una **cadena de texto** para indicar diferentes tipos y opciones de formato y justificación.

Los **modificadores** más utilizados son:

%c Un único caracter
%d Un entero con signo, en base decimal
%u Un entero sin signo, en base decimal
%o Un entero en base octal
%x Un entero en base hexadecimal
%e Un número real en coma flotante, con exponente
%f Un número real en coma flotante, sin exponente
%s Una cadena de caracteres
%p Un puntero o dirección de memoria

El formato completo de los modificadores es el siguiente:

% [signo] [longitud] [.precisión] modificador

***Signo*: indicamos si el valor se **ajustará** a la izquierda, en cuyo caso utilizaremos el signo menos, o **a la derecha (por defecto)**.

***Longitud*: especifica la longitud máxima del valor que aparece por pantalla. Si la longitud es menor que el número de dígitos del valor, éste aparecerá ajustado a la izquierda.

***Precisión*: indicamos el número máximo de decimales que tendrá el valor.

Ejercicio para investigar. (1 de 3)

```
// FUNCIONES - Pasaje de Parámetros
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
```

// Función f_uno --> 3 parámetros (x valor, x dirección y x referencia)

```
void f_uno(char e, char *f, char & d)
{
  *f = e;
  d = '9';
  *f = d;
  cout <<endl<<"***  Funcion 1"<<endl<<endl;
 printf("posicion de e (parametro por valor o copia): %p\n", &e);
 cout<<endl<<"contenido de e: "<< e <<endl<<endl;
 printf("posicion de d (parametro por referencia): %p\n", &d);
 cout<<endl<<"contenido de d: "<< d <<endl<<endl;
 printf("posicion de f (parametro por direccion): %p\n", &f);
 printf("posicion de f (parametro por direccion - a donde apunta)es: %p\n", f);
 cout<<"contenido de a donde apunta f: "<< *f;
 getch();
}
```

Ejercicio (2 de 3)

// Función f_dos --> 2 parámetros (x valor y x referencia)

```
void f_dos (char x, char &y)
{
 char w;

 x = 'g';
 w = y;
 y = x;

 cout <<endl<<"***   Funcion 2"<<endl<<endl;
 printf("posicion de y (parametro por referencia): %p\n", &y);
 cout<<endl<<"contenido de y: "<< y<<endl<<endl;
 printf("posicion de x (parametro por valor o copia): %p\n", &x);
 cout<<endl<<"contenido de x: "<< x <<endl<<endl;
 printf("posicion de w (variable local): %p\n", &w);
 cout<<endl<<"contenido de w: "<< w <<endl<<endl;
 getch();

}
```

Ejercicio (3 de 3)

```
int main (void)
{
  char a, b,c;

  a= '3';
  b = 'z';
  c = b;

  cout <<endl<<"main"<<endl;
  printf ("posicion de a: %p\n", &a);
  printf ("posicion de b: %p\n", &b);
  printf ("posicion de c: %p\n", &c);
  cout<<endl<<endl;

      f_dos ( a, b);

      f_uno( '8', &a, c);

  cout<<endl<<c<<"  " << b <<"  " << a <<"  a";
  getch();

}
```