13. Punteros.

Las variables tienen direcciones de memoria. Si deseamos conocer dicha dirección en lenguaje C se usa el operador **&** de dirección.

Un puntero es una simple variable que tiene como objetivo almacenar o apuntar a la dirección de memoria de otra. Y si apunta a la dirección de memoria es como si obtuviera su valor.

Declaración de punteros

int *p;

- Un tipo de dato: el puntero solo podrá almacenar direcciones de memoria de variables del tipo especificado. Se pueden definir punteros de cualquier tipo:

  *float *pf;*

  *char *pc;*

- Un identificador que siempre va antecedido del operador *.

  *int *pt;*

Si queremos que un puntero apunte a la dirección de memoria de una variable tenemos que igualarlo a ella y además poner el signo “&”delante.

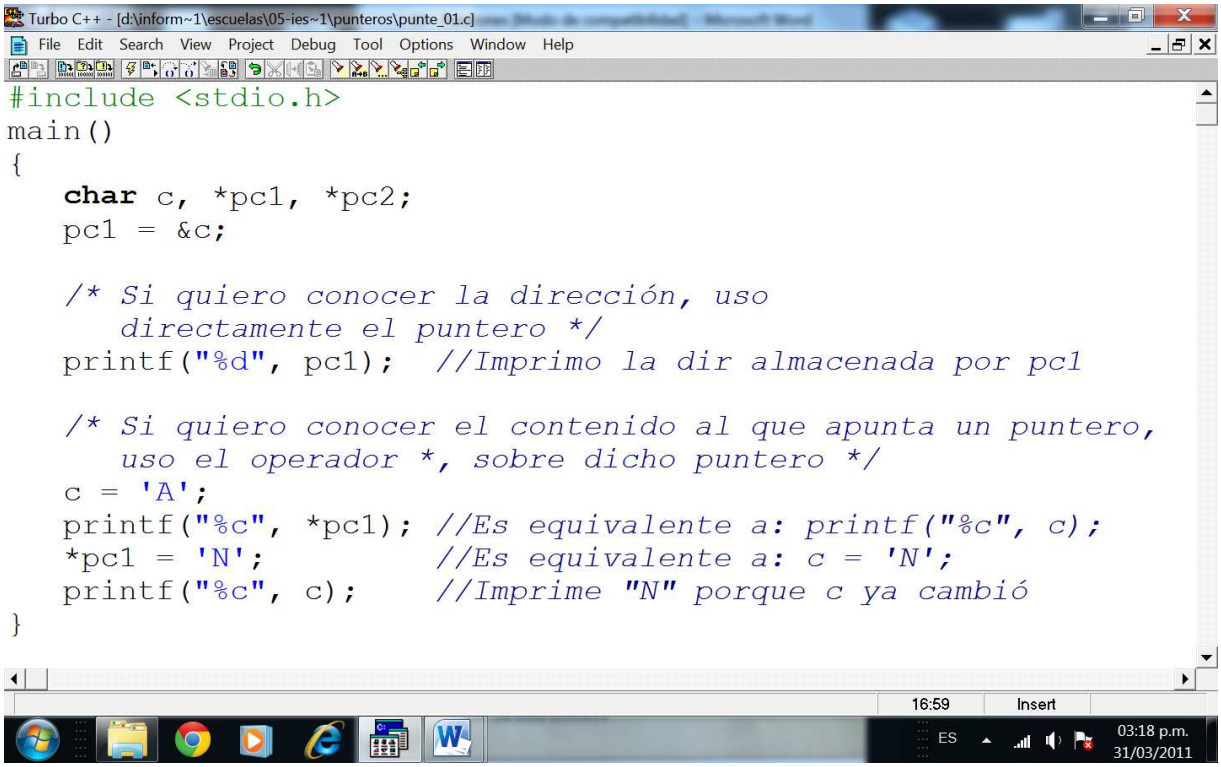
int x;

*pt = &x; (*pt almacena la dirección de x, se dice que ***pt apunta a x***)

Realmente el **ampersand** ha servido todo este tiempo para representar la dirección de memoria de todas las variables con las que hemos trabajado.

Entonces, si un puntero apunta a una variable, a través del puntero se puede llegar a conocer todo sobre la variable.

Ejemplo:


Turbo C++ - [d:\inform~1\escuelas\05-ies~1\punteros\punte_01.c]
File Edit Search View Project Debug Tool Options Window Help
#include <stdio.h>
main()
{
char c, *pc1, *pc2;
pc1 = &c;
/* Si quiero conocer la dirección, uso
directamente el puntero */
printf("%d", pc1); //Imprimo la dir almacenada por pc1
/* Si quiero conocer el contenido al que apunta un puntero,
uso el operador *, sobre dicho puntero */
c = 'A';
printf("%c", *pc1); //Es equivalente a: printf("%c", c);
*pc1 = 'N'; //Es equivalente a: c = 'N';
printf("%c", c); //Imprime "N" porque c ya cambió
}
16:59
Insert
ES
03:18 p.m.
31/03/2011

14. Paso de Parámetros.

Generalmente la forma de regresar los resultados se hace a través de datos compartidos con los procedimientos que lo llamaron, a esta forma de pasar argumentos se le llama *paso de parámetros por referencia*. Esta forma de paso de parámetros se hace cuando es necesario que después de ejecutada la función las variables registren la actualización.

Generalmente también es necesario enviar ciertos datos que serán requeridos por una función pero que no interesan al momento de finalizar dicho procedimiento, a esta forma de enviar argumentos se le llama paso de parámetros por valor. En el paso de parámetros por valor el procedimiento o la función realizan una copia del valor, de tal forma que el valor en la variable original permanecerá inalterado. Es decir, no se generará ningún cambio después de ejecutar el procedimiento.

Podemos modelar los procedimientos como si fueran un sastre. El sastre es el procedimiento que procesa la tela y la transforma, pero es necesario darle elementos al sastre para que pueda trabajar, un elemento es la tela, la cual él va a procesar y la cual va a regresar pero convertida en un traje, por lo tanto la tela es un argumento que debemos pasar por referencia o variable ya que tiene que regresar transformada. Pero no solo eso, también necesitamos darle al sastre las medidas, ya que sin las medidas el sastre no sabrá como transformar la tela. Así, pasaremos al sastre las medidas. El sastre hará una copia de las medidas en un papelito las cuales no es importante que regrese.

A la especificación de las variables que se reciben como argumentos se le llama especificación de los parámetros formales. A los argumentos enviados al invocar la función o el procedimiento se les llama parámetros actuales.

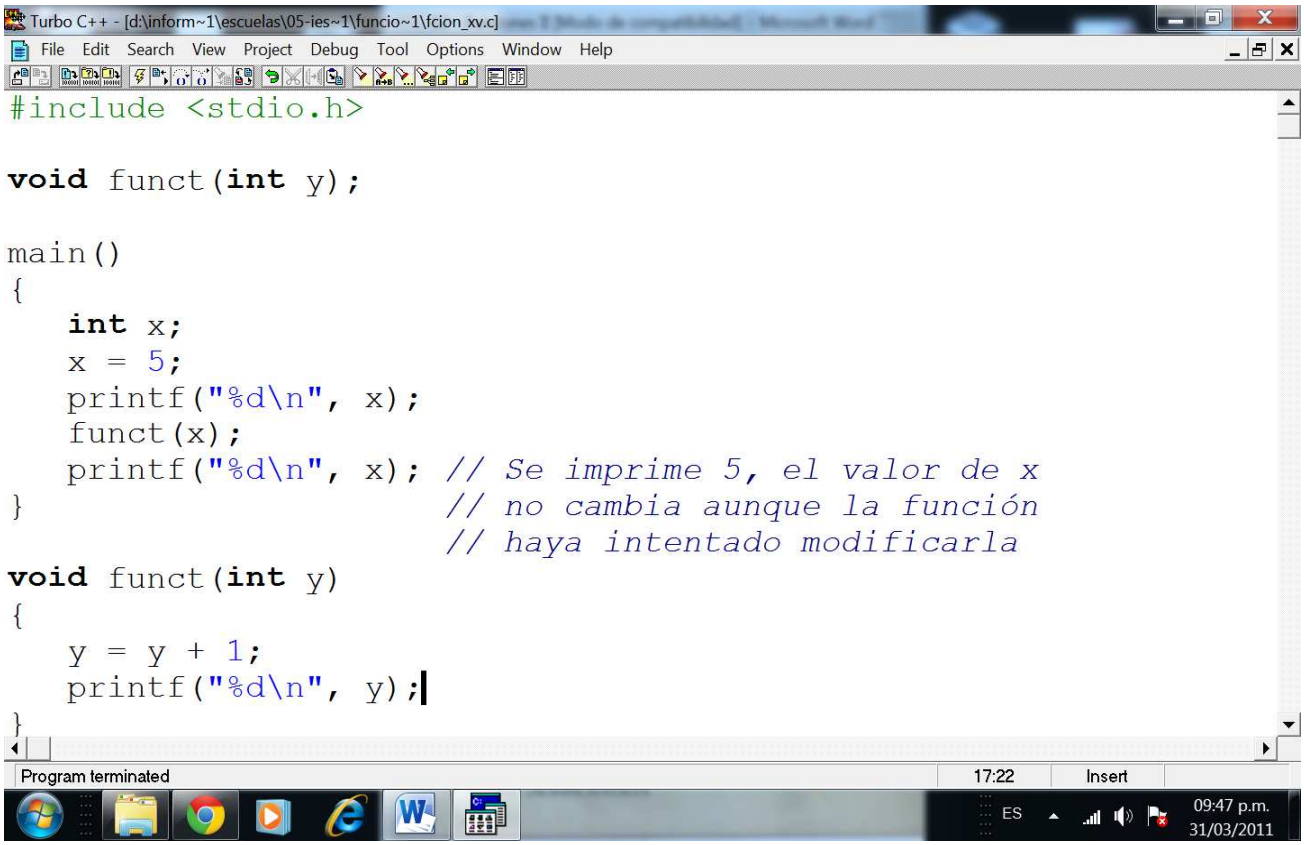
Paso por Valor

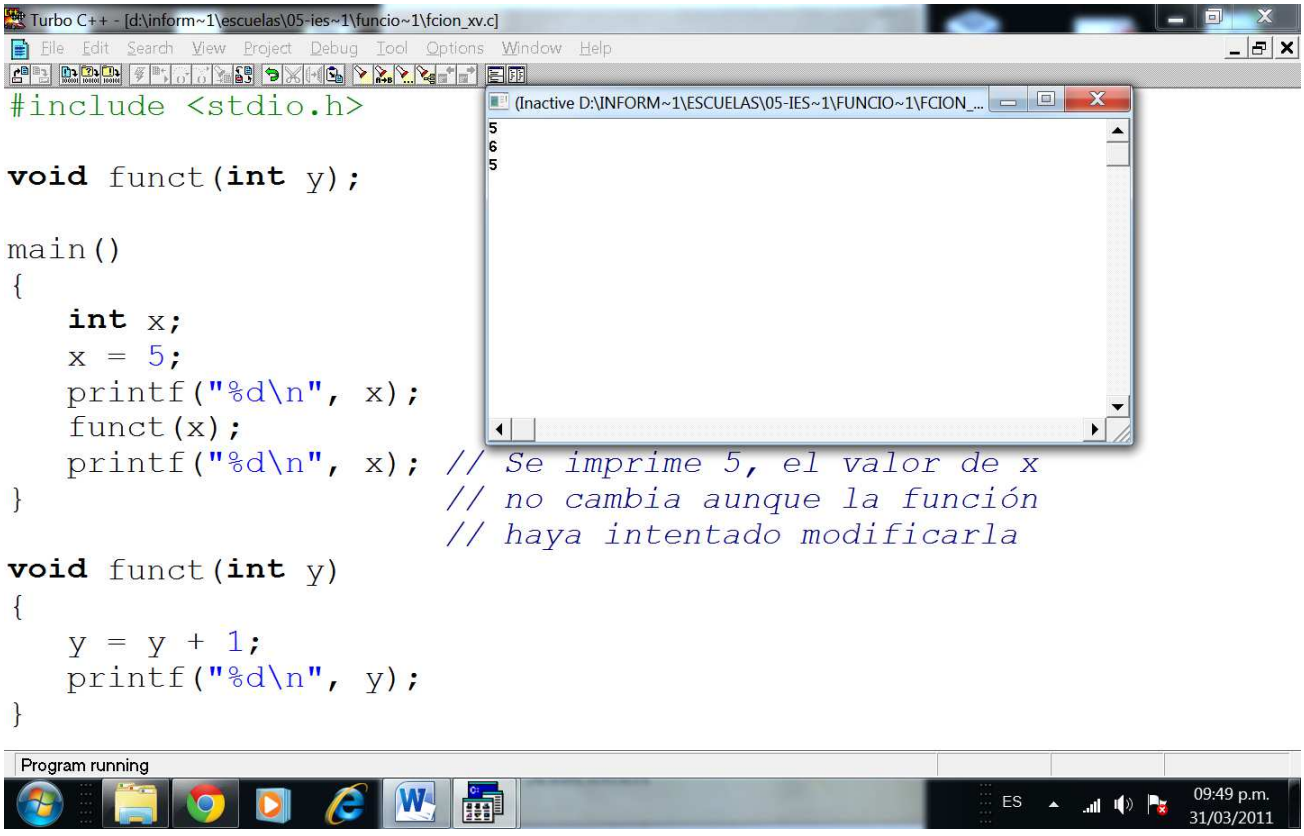
La función no recibe la variable enviada:

- ❖ Recibe una copia
- ❖ Similar a cuando va a hacer algún trámite y le piden el documento
  - ∗ No entrega el D. N. I. verdadero
  - ∗ Entrega una copia
  - ∗ El verdadero estará seguro, aunque quemen y destruyan la copia

EJEMPLO

A continuación se muestra un ejemplo muy sencillo del paso de parámetro a una función por valor.





Paso por Referencia

Aquí la función recibe exactamente la variable enviada:

- ❖ No hay copias
- ❖ Para usar parámetros por referencia se puede, en la llamada a la función, anteponer el operador & a las variables actuales que serán pasadas como por referencia.
- ❖ En la declaración formal de la función se usan punteros.
  - ∗ La función trabaja con un puntero a la variable enviada
  - ∗ Sabe todo sobre esa variable y se puede acceder a través de “*”

EJEMPLO