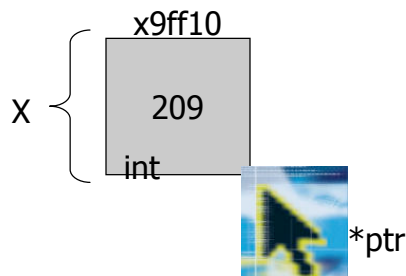


Apuntadores (Punteros)



SESION 7

Definición



Llamados también punteros. **Un Apuntador** es una variable que contiene una dirección de memoria, la cual corresponderá a un dato o a una variable que contiene el dato.

Cada variable que se utiliza en una aplicación ocupa una o varias posiciones de memoria. Estas posiciones de memoria se accesan por medio de una dirección.

Mónica E. García



Operador de dirección y de indirección

El Operador de Dirección (&) regresa la dirección de una variable.

El Operador de Indirección (*), toma la dirección de una variable y regresa el dato que contiene esa dirección.

Mónica E. García



Sintaxis

La declaración de un puntero de manera general es:

Tipo_dato *nombre de apuntador;

Tipo_dato : Especifica el tipo de objeto apuntado y puede ser cualquier tipo (int, float, char, etc).

Nombre de apuntador: Es el identificador (nombre asignado) del apuntador.

Ejemplos de declaración:

```
int    *ptr, cont;
```

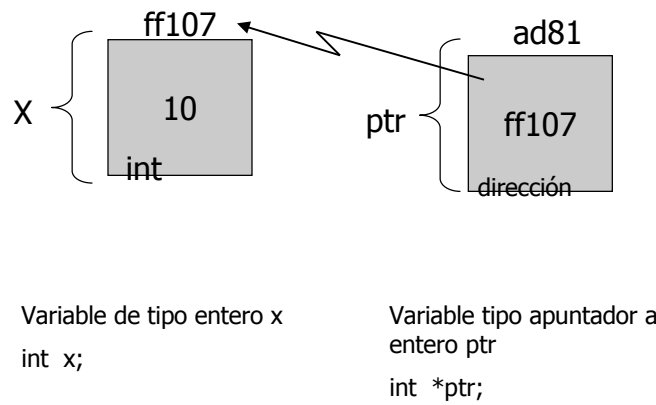
```
float  *res;
```

```
short  *bandera;
```

```
char   *mensaje;
```

Mónica E. García

Representación



Mónica E. García

Ejemplos

Suponer la siguiente declaración:

```
int a=1,b=2,*p;
```

Si se ejecutarán cada una

de las siguientes instrucciones

el resultado sería:

```
p = &a;
```

```
b = *p;
```

```
*p = 0;
```

p apunta a la variable a

ahora b es igual a 1

ahora a es igual a 0

Mónica E. García



Ejemplos

```
// Programa que demuestra el uso basico de los apuntadores
#include "stdio.h"
#include "conio.h"
void main ( )
{
    int var=1, *apun;
    apun = &var;           // Inicialización del apuntador
    printf("\n Acceso directo, var= %d", var);
    printf("\n Acceso indirecto, var= %d", *apun);

    // Se despliega la dirección de la variable de dos maneras

    printf("\n La dirección de var= %d", &var);
    printf("\n LA dirección de var= %d", apun);

    getch( );
}
```

Mónica E. García



Aritmética de Apuntadores

- Las operaciones que se pueden realizar son:

Incremento (suma)	apunta ++;
	apunta = apunta +3;
Decremento (resta)	apunta --;
	apunta = apunta - 7;
Multiplicación	apunta = apunta * 2;
Comparación	if (apunta1 == apunta 2)
	if (*apunta1 == *apunta2)
Asignación	apunta = &variable;
	* apunta = 25.6

Mónica E. García



Aritmética de Apuntadores

- Ejercicio, Probar si las siguientes expresiones son verdaderas o falsas, suponiendo que:

```
char c[11] = "COMPUTACION", *p;  
p = &c;
```

La expresión	es la misma que	y es equivalente a
<code>c = *p++;</code>	<code>c = *(p++);</code>	<code>c = *p; p++;</code>
<code>c = *++p;</code>	<code>c = *(++p);</code>	<code>++p; c = *p;</code>
<code>c = ++*p;</code>	<code>c = ++(*p);</code>	<code>*p+=1; c = *p;</code>
<code>c = (*p)++;</code>		<code>c = *p; (*p)++;</code>

Mónica E. García

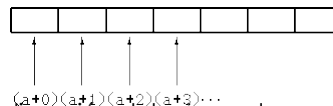


Apuntadores y vectores

- Las versiones con apuntadores en los arreglos son más rápidas que la forma común.
- La declaración `int a[10]; int *pa;`

por lo que `pa=&a[0]` y así se establece que `*pa=a[0]` y `*(pa+1)=a[1]` y así sucesivamente. De esta manera se puede manejar más eficientemente los valores y direcciones de un arreglo Bi o Unidimensional.

`a[n]` equivale exactamente a `*(a+n)`. Por eso empiezan los arreglos con el índice 0.



El nombre del arreglo `a` es la dirección del primer elemento.

Mónica E. García



Apuntadores y Matrices

Considerar:

```
int a[10][10];  
int *b[10];
```

- El uso de a y b puede ser parecido, desde el momento en que a[5][5] y b[5][5] son referencias validas a un int.
- El arreglo a es un arreglo verdadero, existen 100 celdas de memoria asignadas y se efectúa el cálculo de subíndices rectangulares convencional para localizar un elemento dado.
- Sin embargo a b la declaración solo le asigna 10 apuntadores, cada uno de los cuales deberá de apuntar a un arreglo de enteros

Mónica E. García



Diferentes declaraciones

La matriz a puede declararse :

- Como un arreglo de 10 arreglos de tamaño 20
`int a[10][20];`
- Como un arreglo de tamaño 20 de vectores de longitud variable
`int *a[10];`
- Como un apuntador de apuntadores a enteros
`int **a;`
- Como un apuntador a un arreglo de enteros de tamaño 20

```
int (*a)[20];
```

Mónica E. García



Apuntadores y cadenas

Sea la declaración:

```
char * mensaje[4] = {"Hola","Adios","Bye","Saludos"} ;
```

- Cada cadena está almacenada en memoria como una cadena de caracteres terminada en NULL (\0).
- En el arreglo no están colocadas las cadenas, tan solo están almacenados los apuntadores.
- Aunque el arreglo es de tamaño fijo, permite el acceso a cadenas de caracteres de cualquier longitud (por ejemplo la longitud de Bye es mas corta que la de saludos).

Mónica E. García



Ejemplo con cadenas

```
void main()  
{  
    char *esp[10] = { "uno", "dos", "tres" };  
    char frances[5][10] = { "un","deux", "trois" };  
  
    printf("Elemento 3 entrada 2 esp: %c \n",esp[2][3]);  
    printf("Elemento 4 entrada 3 frances: %c \n",frances[3][4]);  
    printf("Elemento 7 entrada 2 esp: %c \n",esp[2][7]);  
    frances[3][4]='A';  
    printf("Elemento 4 entrada 3 frances: %c \n",frances[3][4]);  
    esp[2][3]='A';  
    printf("Elemento 3 entrada 2 esp: %c \n",esp[2][3]);  
    printf("Cadena esp %s \n" ,esp);  
    printf("Cadena frances %s \n",frances[1]);  
    getch();  
}
```

Mónica E. García



Apuntadores a otros apuntadores

- Se puede tener un apuntador a otro apuntador de cualquier tipo. El siguiente código muestra este caso:

```
void main( )
{
    char ch;      /* Un caracter */
    char *pch;    /* Un apuntador a caracter */
    char **ppch; /* Un apuntador a un apuntador a caracter */

    ch = 'A';
    pch = &ch;
    ppch = &pch;
    printf("%c\n", **ppch); /* muestra el valor de ch */
}
```

Mónica E. García



Asignación Dinámica de Memoria

- La asignación dinámica permite obtener la memoria para variables que se precisen en la ejecución del programa.
- Para la asignación dinámica de memoria existe una función llamada malloc que se encuentra en la librería stdlib.h.

void *malloc (tamaño)

Donde tamaño es el tamaño en bytes de la memoria que se requiere asignar o reservar.

Mónica E. García



Asignación de memoria a valores tipo char


```
char texto[50];      // asignación estática de un vector de 50 elementos

char *ptexto;        // declaración de un puntero( no asignamos memoria para
                     // almacenar valores)

int n;

scanf("%d", &n);
ptexto = (char *) malloc(n); // Asigna dinámicamente n bytes
if (ptexto==NULL)
    printf("Error en la asignación de memoria\n");
else
    free(ptexto);    // libera la memoria asignada cuando ya no se necesita
```

Mónica E. García



Asignación de memoria a valores numéricos

- Si se requiere reservar memoria para un tipo de dato que no es char se realiza de la siguiente manera:

tamaño = (número de elementos) * (tamaño del tipo)

El tamaño del tipo se obtiene con la función **sizeof**.

Ejemplos:

//Reserva de memoria para 35 enteros

```
int *apun;
apun = (int *) malloc (sizeof(int));
```

//Reserva de memoria para 50 flotantes

```
float *apun;
apun = (float *) malloc (sizeof(float));
```

Mónica E. García