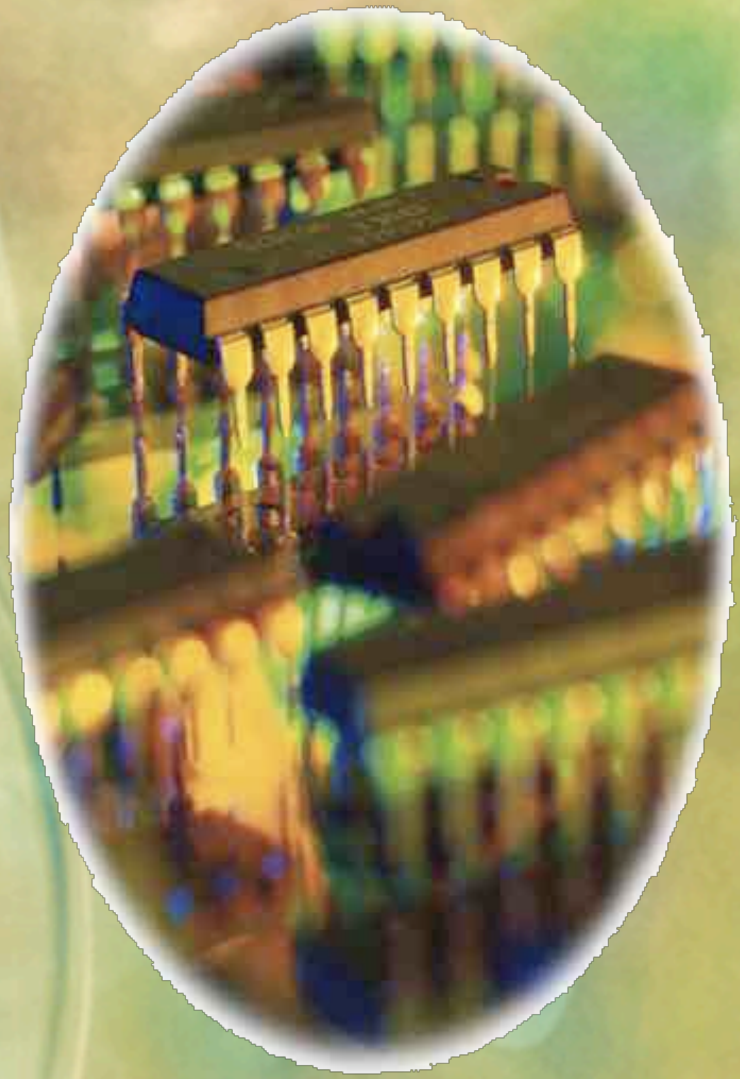
TEMA

6

Gestión Dinámica de Memoria



Dept. Ciencias de la Computación e I.A.

Universidad de Granada

Definición de Puntero

Un puntero es un dato que contiene una dirección de memoria

Nota: Existe una dirección de memoria especial que se representa por la constante **NULL** (definida en *stdlib.h*) y se emplea cuando queremos indicar que un puntero no apuntan a ninguna dirección.

Declaración

<tipo> *<identificador>;

<*tipo*> indica el tipo de dato del objeto referenciado por el puntero.

<identificador> identificador de la variable de tipo puntero.

Ejemplos

```
double *a;
int *b;
char *c;
```

Consideraciones

Cuando se declara un puntero se reserva memoria para albergar una dirección de memoria, pero NO PARA ALMACENAR EL DATO AL QUE APUNTA EL PUNTERO.

El espacio de memoria reservado para almacenar un puntero es el mismo independientemente del tipo de dato al que apunte: el espacio que ocupa una dirección de memoria.

Operaciones Básicas (1)

Dirección: Operador &

&<id> devuelve la dirección de memoria donde comienza la variable *<id>*

El operador & se utiliza para asignar valores a datos de tipo puntero.

Ejemplo

```
init i;
int *ptr;
…
ptr=&i;
```

Operaciones Básicas (2)

Indirección: Operador *

*<ptr> devuelve el contenido del objeto referenciado por el puntero <ptr>

El operador * se utiliza para acceder a los objetos a los que apunta un puntero.

Ejemplo

```
char c;
char *ptr;
…
ptr=&i;
*ptr= 'A'; /*equivale a c='A';*/
```

Asignación

A un puntero se le puede asignar:

- **una dirección de memoria concreta**
- **la dirección de una variable**
- **el contenido de otro puntero**

Ejemplo

```
char c;
char *ptr1=NULL;
char *ptr2=NULL;
…
ptr1 = &c;
ptr2 = ptr1;
ptr1=0x1F3CE00A;
```

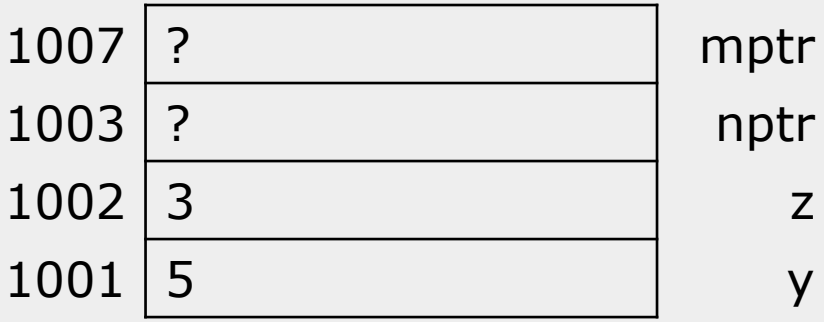
Como cualquier variable, los punteros también debe ser inicializados para que tenga un valor conocido. Es una buena costrumbre inicializarlos con **NULL**.

Ejemplo variables Referencia

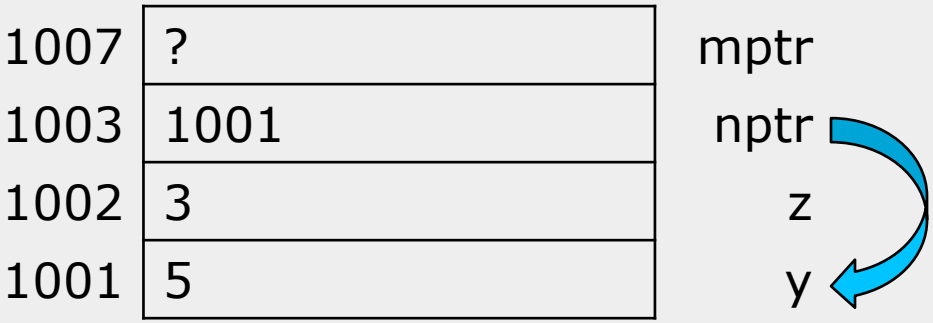
```
/*Intercambio de valores con punteros. Variables por
  referencia*/
#include <stdio.h>
void cambia (int *a, int *b){
  int aux;
  aux = *a;
  *a = *b;
  *b = aux;
}
int main(){
  int x=0,y=1;
  …
  cambia(&x,&y);
  …
  return 0;
}
```

Un Ejemplo (1)

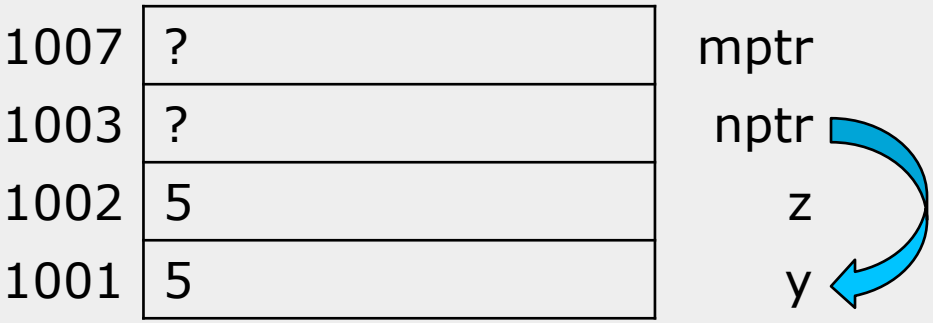
```
int main(){
   int y=5;
   int z=3;
   int *nptr;
   int *mptr;
```




```
   nptr=&y;
```



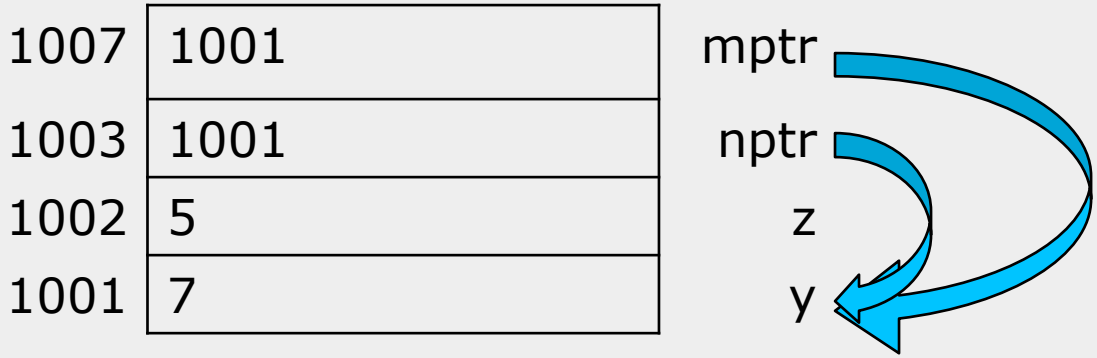

```
   z=*nptr;
```



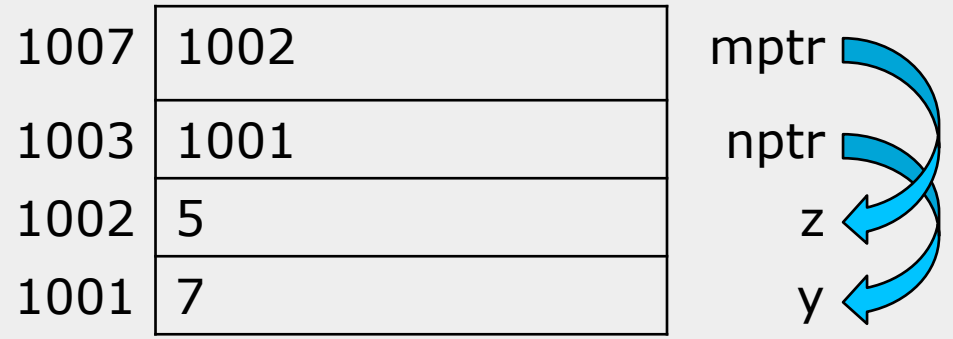
Un Ejemplo (2)

```
*nptr= 7;
```

```
mptr=nptr;
```

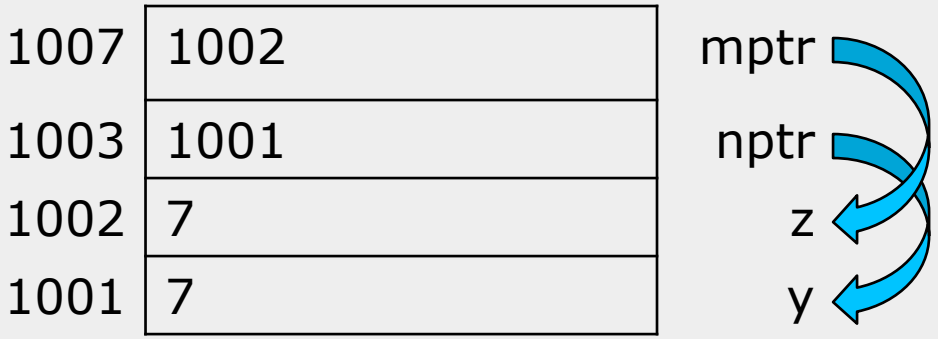



```
mptr=*z;
```

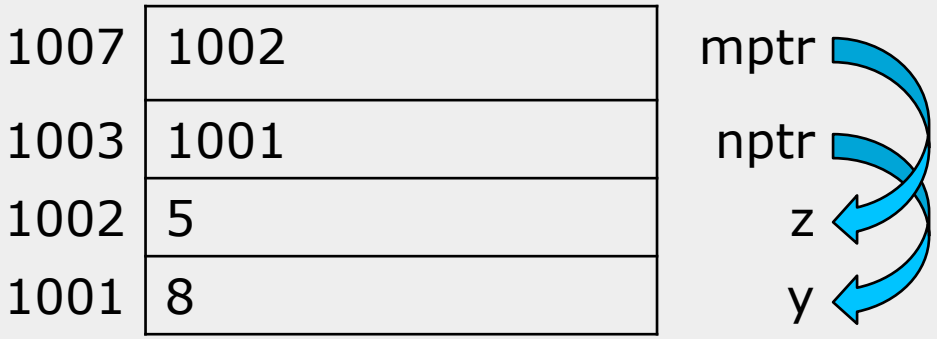


Un Ejemplo (y 3)

```
    *mptr= *nptr;
```




```
    y=(*nptr)+1;
```




```
    return 0;
}
```

Errores Comunes (1)

- Asignar punteros de distinto tipo

```
int a=10;
int *ptri=NULL;
double x=;
doubles *ptrf=NULL;
…
ptri= &a;
ptrf= &x;
ptrf=ptri; /*ERROR*/
```

- Utilizar punteros no inicializados

```
char *ptr;
*ptr='a'; /*ERROR*/
```

Errores Comunes (y 2)

- Asignar valores a un puntero y no a la variables a la que apunta

```
int n;
int *ptr=&n;
ptr= 9; /*ERROR*/
```

- Intentar asignarle un valor al dato apuntado por un puntero NULL

```
int *ptr=NULL;
*ptr= 9; /*ERROR*/
```

Correspondencia Punteros y Vectores

Cuando declaramos un vector:

<tipo> <identificador>[dim];

en realidad:

- Reservamos memoria para almacenar ***<dim>*** elementos de ***<tipo>***
- Creamos el puntero ***<identificador>*** que apunta a la primera posición de memoria reservada para almacenar componentes de un vector

Por tanto, el identificador del vector es un puntero.

Memoria Dinámica

Reserva de memoria: función *malloc*

Para reservar memoria usaremos la función malloc que nos devuelve un puntero a la zona de memoria reservada cuyo tamaño se le indica como parámetro. Al reservar memoria puede que no quede espacio suficiente, en cuyo caso la sirección devuelta es **NULL**

Liberación de memoria: función free

Tras utilizar la memoria reservada dinámicamente, hay que liberar el espacio reservado con la función free, al que se le pasa el puntero con la zona de memoria reservada.

free y ***malloc*** pertenecen a ***stdlib.h***

Ejemplo: vector tamaño dinámico

```
#include <stdio.h>
#include <stdlib.h>
…
int main(){
 int i, n;
 double *serie;
 /*Leemos el tamaño del vector*/
 printf("Número de elemntos del vector:");
 scanf("%d",&n);
 /*Creación del vector*/
 serie=malloc(n*sizeof(double));
 /*Manejo del vector*/
 for (i=0; i<n; i++)
   serie[i]= i;
 printf("Media del vector=%lf\n",media(serie,n));
 /*Liberación de memoria*/
 free(serie);
 return 0;
}
```