

PROGRAMACIÓN

Unidad 6: Punteros

Lic. Mariela A. Velázquez

Temas a tratar

P
A
R
T
E

1

Definición de punteros

Conceptos básicos

Declaración de punteros

Operaciones de punteros

Punteros y Arreglos

P
A
R
T
E

2

P
A
R
T
E

Punteros y Funciones

Pasaje de Parámetros

3

Recordemos un poco...

Puntero



Es una variable que contiene la dirección de memoria de otra variable

Declaración de Punteros: **tipo *nombre-vble-puntero**

***** Operador de indirección

& Operador de dirección

Iniciación: `punt_num = №`

Pasaje de Parámetros

El lenguaje C transfiere variables entre funciones usando 2 mecanismos diferentes. El método a emplear depende de si se quiera modificar o no los valores de las variables transferidas, como así también de la cantidad de valores “devueltos” por la función.

Revisemos algunos conceptos

- **Parámetros Formales:** son aquellos especificados en la cabecera de la función. Al declarar un parámetro formal hay que especificar su tipo de dato. Los parámetros formales sólo se conocen dentro de la función.
- **Parámetros Actuales:** son las expresiones pasadas como argumentos en la llamada a una función.
- **Función Emisora:** Función que emite valores para la transferencia de información.
- **Función Receptora:** Función que recepciona valores transferidos.

Pasaje de Parámetros

- Pasaje por valor
- Pasaje por Referencia

Pasaje por Valor

Los argumentos que se pasan por valor a una función, significa que la función que se invoca recibe los valores de sus argumentos en variables temporales y no en las originales.

La función que se invoca no puede alterar directamente una variable de la función que hace la llamada, solo puede modificar su copia temporal.

Pasaje por Valor

Ejemplo: Función para calcular el peso de una persona en la luna.

```
FUNCION pesoenlaluna;  
ARGUMENTO: Peso. real >0;  
SALIDA: real ≥ 0;  
    pesoenlaluna ← peso/6;
```

Función emisora: transfiere el peso en la tierra.

Función receptora: recibe el valor del peso en la tierra, y calcula y devuelve el valor en la luna, sin modificar el valor del peso en la tierra.

Pasaje por Valor

- En el momento de la invocación, la función receptora toma, en forma temporal, un espacio de almacenamiento de características idénticas a las del P. Actual.
- Si se hacen cambios dentro de la función en el valor de la variable transferida, los cambios se reflejan en esa localidad de memoria temporal.
- Cuando la función termina su ejecución, el espacio de almacenamiento temporal que se usó para “la copia” queda liberado, perdiéndose los valores que había allí almacenado.

Pasaje por Valor

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

float pesoEnLuna(float PesoTierra);
int main(void)
{
    setvbuf ( stdout , NULL , _IONBF , 0);
    float PesoTierra, PesoLuna;
    PesoTierra = 95.8;
    PesoLuna = pesoEnLuna(PesoTierra);

    printf("El peso en la luna de Homero es: %.2f\n", PesoLuna);
    printf("El peso en la tierra de Homero es: %.2f", PesoTierra);

    return 0;
}

float pesoEnLuna(float PesoTierra){
    float aux;
    aux= PesoTierra/6;
    return (aux);
}
```

Calcular el peso de Homero Simpson en la luna.



Pasaje por Valor

- **Una desventaja fuerte de este mecanismo:**

Las funciones que lo usan pueden devolver a lo sumo un resultado asociado a la proposición return.

Puede ocurrir que la función no devuelva ningún valor, en este caso tenemos una función de tipo void.

- **Ventaja:** Preserva los valores de las variables transferidas.

Pasaje por Referencia

Ejemplo: cambios de precios de varios productos comerciales.

El precio de un producto “pro” se calcula como la suma de los precios de sus insumos M1 y M2 más una constante. Los precios de M1 y M2 se modifican periódicamente, por lo cual el precio de nuestro producto “pro” también se modifica. Calcular los incrementos de M1, M2 y pro según los porcentajes de cambio indicados.

Porcentajes de Cambios: M1 se incrementa en un 3% y M2 en un 5%.

$$M1 \leftarrow M1 * 1.03$$

$$M2 \leftarrow M2 * 1.05$$

$$Pro \leftarrow M1 + M2 + cte$$

Pasaje por Referencia

- **Función emisora:** transfiere los precios de los productos (generalmente main).
- **Función receptora:** recibe los precios de los productos, los modifica y los devuelve modificados.

Pasaje por Referencia

Ejemplo: Cambios de precios de productos comerciales.

El precio de un producto **prod**, se calcula como la suma de los precios de los insumos **ins1**, **ins2**. Los precios **ins1**, **ins2** se modifican periódicamente, por lo cual el precio de nuestro producto **prod** también se modifica. Calcular los incrementos de **ins1**, **ins2** y **prod** según los porcentajes de cambios indicados.

Porcentajes de cambios:

- **ins1 se incrementa un 3%**
- **Ins2 se incrementa un 5%**

```
float modificarPrecio (float *ins1, float *ins2)
{
    float precioProd;
    *ins1 = (*ins1) * 1.03;
    *ins2 = (*ins2) * 1.05;
    precioProd = (*ins1)+ (*ins2);
    return(precioProd);
}
```

Pasaje por Referencia

- La función receptora habrá de tener definidos como p. Formales variables de tipo puntero.
- En el momento de la invocación, la función receptora recibe como parámetros actuales las direcciones de memoria de variables de la función emisora.
- En la función receptora se llevan a cabo modificaciones sobre las variables referenciadas. Estas transacciones modifican los valores originales de las variables.
- Los cambios realizados adquieren el carácter de permanentes, es decir, mientras dure la ejecución del programa. Cuando la función termina su ejecución, todos los cambios realizados en esa locación de memoria quedan allí, por esto su carácter de permanentes.

Pasaje por Referencia

- Ventajas:
 - Bajo costo en términos de espacio de almacenamiento, No se realizan duplicaciones o copias de variables.
 - La función puede “regresar” más de un valor.
- Desventaja:
 - Usar con precaución, las variables quedan muy vulnerables.

Pasaje por Referencia

Cada vez que se transfiere una variable por dirección, si la función receptora modifica la variable, ésta es modificada también en la función invocante.

Pasaje de Parámetros

- Considérese la siguiente función para permutar el valor de sus dos argumentos **x** e **y**:

```
#include <stdio.h>
void permutar(double a, double b);
int main(void)
{
    setvbuf ( stdout , NULL , _IONBF , 0);
    double a=1.0, b=2.0;

    printf("ANTES DE PERMUTAR: a = %lf, b = %lf\n", a, b);
    permutar(a, b);
    printf("DESPUES DE PERMUTAR: a = %lf, b = %lf\n", a, b);

    return 0;
}
```

Pasaje de Parámetros

```
void permutar(double x, double y)
{
    double temp;
    temp = x;
    x = y;
    y = temp;
}
```

Compilando y ejecutando este programa se ve que **a** y **b** siguen teniendo los mismos valores antes y después de la llamada a **permutar()**, a pesar de que en el interior de la función los valores sí se han permutado, la razón está en que *se han permutado los valores de las copias* de **a** y **b**, pero no los valores de las propias variables.

Pasaje de Parámetros

Una versión correcta de la función **permutar()** que pasa direcciones en vez de valores sería como sigue:

```
#include <stdio.h>
void permutar(double *, double *);
int main(void)
{
    setvbuf ( stdout , NULL , _IONBF , 0);
    double a=1.0, b=2.0;
    printf("ANTES DE PERMUTAR: a = %lf, b = %lf\n", a, b);
    permutar(&a, &b);
    printf("DESPUES DE PERMUTAR: a = %lf, b = %lf\n", a, b);
    return 0;
}
```

Pasaje de Parámetros

```
void permutar (double *x, double *y)
{
    double temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
```

El utilizar *punteros como valor de retorno* permite superar la limitación de devolver un único valor de retorno. Puede devolverse un puntero al primer elemento de un vector o a la dirección base de una matriz, lo que equivale a devolver múltiples valores.

Pasaje de Parámetros

- ¿Cómo elegir que mecanismo usar?
 - No modificar los valores de las variables usadas como parámetros actuales: POR VALOR
 - Modificar valores de las v. usadas como parámetro actuales y obtener mas de un valor de retorno: POR REFERENCIA

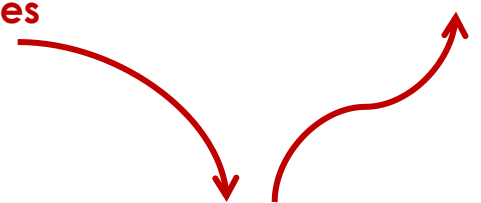
Pasaje por Referencia - Arreglos

Cuando el nombre de un arreglo se emplea como argumento, el valor que se pasa a la función es la dirección de memoria de la 1era componente del arreglo, se trabaja sobre la dirección original.

Pasaje por Referencia - Arreglos

**Ingresa con sus
valores iniciales**

Sale modificado



```
Void Inicializar(int arre[] , int n)
{
    int i;
    For(i=0; i<n; i++)
    {
        arre[i]=0;
    }
}
```

Cuando se llama la función de nombre “inicializar” se le envían 2 parámetros para que pueda proceder. Un entero que corresponde al tamaño del arreglo, (pasaje por valor), y también se envía un arreglo.

Cuando enviamos el nombre del arreglo, lo que enviamos en realidad es la dirección de memoria del primer elemento del arreglo. Es por este motivo que decimos que al pasar un arreglo a una función utilizamos pasaje por referencia.


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

```
int main()
```

```
{
```

```
    printf("Hasta la próxima clase!!\n");
```

```
    return 0;
```

```
}
```