**UNIVERSIDAD TECNOLOGICA DE PANAMA**

**FACULTAD DE INGENIERIA DE SISTEMAS COMPUTACIONALES**

**GUIA DEL ESTUDIANTE**



**Nombre de la Asignatura:** Herramienta de Programación I

**Capítulo:** IV. Punteros

Título de la Guía: **Trabajo individual**

Número de Guía: 4.2

**Docente responsable:** Ludia B. Gómez. De Meléndez

**Fecha:** 25 de octubre de 2018

**OBJETIVO**.

Que el estudiante sea capaz debe:

* Identificar las operaciones que con punteros pueden realizarse en el lenguaje de programación C.
* Manejar la estructura de datos punteros con estructuras de datos simples y compuestas en el lenguaje C.
* Resolver problemas aplicando los conceptos de punteros, en sentencias simples y compuestas en el lenguaje C.

# ENUNCIADOS

1. **CONCEPTOS BASICOS SOBRE PUNTEROS.**

A. Los punteros son inicializados cuando se declara

SI\_\_\_\_ NO\_\_\_\_\_

B. Las siguientes sentencias que hacen, son equivalentes SI \_\_\_\_ NO\_\_\_

a) double \*ptrsd b) \*ptrsd = 67.89; Constituyen :

a. Declaración de una variable puntero

b. asignar un valor a una dirección de memoria

C. Esta sentencia que hace double gana; ptrsd = &gana;

\_\_declara variable gana, a ptrsd le asigna el dirección de gana\_\_\_\_\_

D. double \*ptrsd = 1234.65;

\_\_\_\_\_error, puntero solo almacena dirección \_\_\_\_\_\_\_

E. El uso de punteros para obtener el valor al que apunta, se le denomina:

\_\_\_dirección de memoria\_\_, para ello utiliza el operador :\_\_\_\_&\_\_\_\_.

F. printf (“%lf”,\*ptrsd); Imprime \_\_valor de dirección de memoria\_\_\_ y printf (“%p”,ptrsd); : \_\_\_contenido de ptrsd\_\_\_\_ .

**AHORA RESUELVA CADA UNO DE LOS SIGUIENTES ENUNCIADOS.**

Para los siguientes fragmentos de código, explique que realiza cada sentencia de código.

* + 1. Punteros y tipos
       1. **int** x **=** **1,** y **=** **2; Declaran dos variable**
       2. **int** **\***ip**; declara puntero**
       3. ip **=** **&**x**; a ip asignale la dirrecion de x**
       4. **\***ip **=** **0; valor de dirección de memoria ip = 0 (x=0)**
    2. Operadores
       1. **++\***p**; a valor de dirección de memoria ip suma 1**
       2. **(\***ip**)++; a valor de dirección de memoria ip suma 1**
       3. **\***ip**++; a contenido de ip suma 1(desplazar posiciones de memoria)**
    3. ¿Qué realizan las siguientes sentencias en el segmento?
       1. **int** a**[7]; declara un arreglo**
       2. **int** **\***pa**; declara un puntero**
       3. pa **=** **&**a**[0]; a pa le asigna el dirección de primer valor de arreglo a**
       4. pa **=** a**; a pa le asigna valor a**
    4. Operaciones con punteros, son válidas o no y para cada uno de las correctas indique que hacen.
       1. pa **+** i**; no valido, \*pa + i**
       2. **\*(**pa**+**i**); no valido, \*pa +i**
       3. **\*(**pa**+**i**); no valido, \*pa +i**
       4. **\*(**a**+**i**); no valido, \*a +i**

Explique de manera gráfica que ocurre en este programa:

#include <stdio.h>

Imprimio valor de x y dirección de x

Imprimio dirección de p1 y p2

p1 le asignó la dirección de x y imprime contenido de p1

p2 le asigna dirección de p1 y imprime contenido de p2

imprime los valores de direcciones de p1 y p2

valor de dirección de p1 =15

int main(){

int \*p1, \*p2;

int x= 8;

printf(" X=%i\n X%p\n",x,&x);

printf(" %x \n%p\n",&p1,&p2);

p1 = &x;

printf("se asigna direccion: %p \n",p1);

p2=p1;

printf("P2: %p\n",p2);

printf(" %d %d\n",\*p1,\*p2);

\*p1 = 15;

p2++;

Desplazar posición de p2

Si valor de dirección de memoria p2 es x imprime valor x

sino, imprime no apunta a A

if (\*p2 == x)

printf("\n %d\n",x);

else

printf(" \n NO APUNTA A A");

return 0; }

Del siguiente fragmento de código, ejecútelo y explique la salida obtenida.

* + - * 1. **#include** <stdio.h>

Al inicio r vale 0, después fue modificado a través de puntero u, con una función llamado cambio que le sumó 3 a valor de dirección de u que era 0, ahora r o \*u vale 3.

* + - * 1. **void** cambio**(int** **\*** v**){**
        2. **\***v**+=3;**
        3. **}**
        4. **int** main**()**
        5. **{**
        6. **int** r**=0;**
        7. **int** **\***u**;**
        8. u**=&**r**;**
        9. printf**(**"VAL: %d**\n**"**,**r**);**
        10. cambio**(**u**);**
        11. printf**(**"VAL: %d**\n**"**,**r**);**
        12. return 0;
        13. **}**

En el siguiente programa, ¿se mostrará alguna salida o mostrará algún error? Porque sucede lo que sucede, expliquelo.

No muestra error tampoco salida. Porque no le asigno dirección al puntero.

* + - * 1. **#include** <stdio.h>
        2. **int** main**()**
        3. **{**
        4. **char** **\***puntero**;**
        5. **\***puntero**=**'a'**;**
        6. printf**(**"Valor del puntero %c**\n**"**,\***puntero**);**

     return 0;    **}**

1. **PROBLEMAS Y SENTENCIAS.**

Resuelva y explique que hace cada una de las siguientes operaciones. Con las siguientes declaraciones.

#include <stdio.h>

Declaró arreglo de tipo char ch con valor 00110, le asigno a contenido de puntero pcodigo =ch, declaro peso y a puntero ppeso le asigna la dirección de peso.

int main (){

char ch[] = "00110";

char \*pcodigo = ch;

float peso=567.89,\*ppeso= &peso;

int i=0, totcondu[5];

while(\*pcodigo){

printf("\n entrada %i pcodigo = %c", i,\*pcodigo);

pcodigo++;

}

for(i=0; i<=4; i++){

\*(totcondu+i) = i+2;

printf("\n%d", \*(totcondu+i));

}

\*ppeso = 0.97;

ppeso = ppeso+1;

printf("\n QUE CAMBIO %f",\*ppeso);

return 0; }

Porque las instrucciones con ppeso dan cero y no 0.97 en peso, si estamos utilizando el operador de indirección para acceder a peso:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_porque al final la dirección de ppeso se cambió \_\_\_\_\_\_.

1. **RESUELVA LOS SIGUIENTES PROBLEMAS DE PUNTEROS.**

¿Cuál sería la salida del siguiente programa? (Explique lo sucedido)

* + - * 1. **#include** <stdio.h>

Manda dirección de o a func1, func1 hace que \*x vale 36 y le manda a func2, func2 recibe la dirección de x y divide el valor de x entre 2. Valor retornado es 18

* + - * 1. **void** func2**(int** **\***u**){**
        2. **\***u**=\***u**/2;}**
        3. **void** func1**(int** **\***x**){**
        4. **\***x**=\***x**\*(\***x**);**
        5. func2**(**x**);}**
        6. **int** main**(){**
        7. **int** o**=6;**
        8. func1**(&**o**);**
        9. printf**(**"Valor retornado %d**\n**"**,**o**);**

**}**

int \*ptr; //4 byte

long \*ptrl; // 8 byte

float \*ptrf; //4 byte

Para estos problemas esquematice la memoria necesaria de cada uno de los datos a los cuales apuntaran los punteros.

8. Explique se sucedió en este programa:

Declaró unvariable c tipo char, un puntero ptrsc de tipo char. Le asigna a puntero la dirección de c, un for desde A hasta Z con paso 1, imprime el valor de dirección de memoria de puntero ptrsc.

#include <stdio.h>

int main (){

char c;

char \*ptrsc;

ptrsc = &c;

for (c='A'; c<='Z';c++)

printf ("%c", \*ptrsc);

return 0;

}