TALLER 3 DE ESTRUCTURAS DE DATOS

APUNTADORES Y MANEJO DE MEMORIA EN C

Propósito

Desarrollar la capacidad del estudiante para interpretar y utilizar apuntadores en lenguaje C mediante el análisis de:

- direcciones de memoria con &;
- desreferenciación con *;
- apuntadores simples;
- apuntadores a apuntadores **;
- operaciones básicas con apuntadores;
- recorrido de vectores mediante apuntadores;
- modificación de datos mediante apuntadores;
- identificación de errores frecuentes;
- representación gráfica del almacenamiento en memoria.

Metodología

No se debe ejecutar el programa antes de realizar el análisis.

Para cada ejercicio, el estudiante debe:

1. Dibujar las variables en memoria.
2. Asignar una dirección ficticia a cada variable.
3. Indicar qué dirección contiene cada apuntador.
4. Determinar el valor de cada expresión.
5. Predecir la salida del programa.
6. Explicar el resultado.
7. Cuando exista un error, identificarlo y explicar qué ocurre.

EJERCICIO 1. MODIFICACIÓN MEDIANTE *p

Analice:

```c
#include <stdio.h>

int main()
{
```

```c
    int numero = 10;

    int *p = &numero;

    *p = 50;

    printf("numero = %d\n", numero);
    printf("*p = %d\n", *p);

    return 0;
}
```

Actividades

Realice dos dibujos.

Estado 1

Antes de:

```c
*p = 50;
```

Estado 2

Después de:

```c
*p = 50;
```

Explique por qué la modificación realizada mediante `*p` también modifica `numero`.

EJERCICIO 2. DOS VARIABLES Y DOS APUNTADORES

Analice cuidadosamente:

```c
#include <stdio.h>

int main()
{
    int a = 15;
    int b = 30;

    int *p = &a;
    int *q = &b;

    printf("%d\n", *p);
    printf("%d\n", *q);
```

```
    *p = 100;
    *q = 200;

    printf("%d\n", a);
    printf("%d\n", b);

    return 0;
}
```

Actividades

Suponga:

```
a → 0x1000
b → 0x2000
```

Construya el siguiente mapa:

VARIABLE	DIRECCIÓN	VALOR
a	_____	_____
b	_____	_____
p	_____	_____
q	_____	_____

Después responda:

1. ¿A qué variable apunta p?
2. ¿A qué variable apunta q?
3. ¿Qué ocurre cuando se ejecuta:

```
*p = 100;
```

4. ¿Qué ocurre cuando se ejecuta:

```
*q = 200;
```

5. ¿Cuál es la salida final?

EJERCICIO 3. APUNTADOR A APUNTADOR **

Ahora aparece un segundo nivel de direccionamiento.

Analice:

```c
#include <stdio.h>

int main()
{
    int numero = 25;

    int *p = &numero;
    int **q = &p;

    printf("A = %d\n", numero);
    printf("B = %d\n", *p);
    printf("C = %d\n", **q);

    return 0;
}
```

Suponga:

```
numero → 0x1000
p      → 0x2000
q      → 0x3000
```

dibuje las asignaciones y estados de cada línea paso por paso; a manera de fotograma de cambio de cada línea donde intervengan apuntadores

Pregunta fundamental

¿Por qué:

`*p`

y:

`**q`

terminan produciendo el mismo valor?

EJERCICIO 4. RECORRIDO DE UN VECTOR CON APUNTADOR

Analice:

```c
#include <stdio.h>

int main()
```

```c
{
    int v[5] = {10, 20, 30, 40, 50};

    int *p = v;

    for (int i = 0; i < 5; i++)
    {
        printf("%d\n", *p);
        p++;
    }

    return 0;
}
```

Actividades

Considere:

`v[0] → dirección 1000`

y suponga que cada entero ocupa 4 bytes.

Dibuje:

Elemento	Dirección	Valor
v[0]	_____	10
v[1]	_____	20
v[2]	_____	30
v[3]	_____	40
v[4]	_____	50

Luego indique el contenido de p en cada iteración:

Iteración	p	*p
1	_____	_____
2	_____	_____
3	_____	_____
4	_____	_____
5	_____	_____

Pregunta de análisis

¿Por qué p++ permite que *p acceda al siguiente elemento del vector?

EJERCICIO 5. DETECTAR Y CORREGIR ERRORES DE APUNTADORES

Analice cada fragmento.

Para cada uno:

1. Identifique el error.
2. Explique por qué ocurre.
3. Indique cómo corregirlo.
4. Señale si el error puede producir un comportamiento indefinido.

Caso A

```c
int *p;

*p = 20;
```

Pregunta:

> ¿A qué posición de memoria está apuntando p?

Explique el problema.

Caso B

```c
int numero = 10;

int *p = &numero;

printf("%d", p);
```

Pregunta:

> ¿Qué problema existe con el formato utilizado para mostrar p?

¿Cómo debería imprimirse una dirección?

Caso C

```c
int numero = 10;
```

```c
int *p = &numero;

printf("%d", *numero);
```

Pregunta:

¿Por qué esta expresión es incorrecta?

Caso D

```c
int numero = 10;

int *p = &numero;

p = 50;
```

Pregunta:

¿Qué está intentando almacenar el programa en p?

¿Es correcto?

Caso E

```c
int v[5] = {10, 20, 30, 40, 50};

int *p = v;

printf("%d", *(p + 5));
```

Pregunta:

¿Qué posición está intentando acceder el programa?

¿Cuál es el último elemento válido?

Caso F

```c
int numero = 10;

int **q;

*q = &numero;
```

Pregunta:

¿Está correctamente inicializado q?

Explique el problema.

No se calificará únicamente la respuesta final.

Cada ejercicio debe contener el **dibujo de memoria, el seguimiento del apuntador y la justificación de la respuesta**.

Un estudiante que indique correctamente la salida, pero no pueda explicar mediante un dibujo cómo llegó a ella no demuestra completamente el dominio del concepto.

Rubrica trabajo: Cada punto tiene un valor de 20 puntos

Trabajo:40%

video sustentación con cámara y todos las miembros de grupo interviniendo:60%.

Nota aclaratoria: tiempo de video: 15 minutos. penalización de 20 puntos si se pasa del tiempo o no entrega el documento en formato pdf. si por motivos no se logra realizar la entrega por correo será penalizada de 20 puntos por cada 5 minutos retrasados sobre el total de la nota.

cantidad de integrantes: max=4 personas