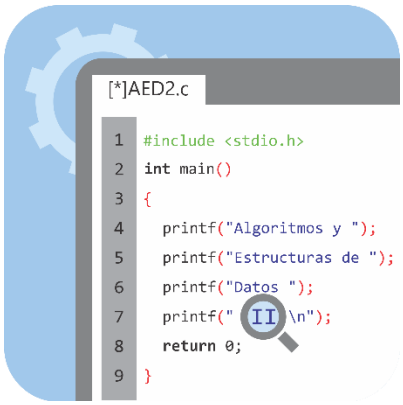


Algoritmos y Estructuras de Datos II

Trabajo Práctico 1
Punteros. Memoria Dinámica.

OBJETIVOS:

- Conocer los conceptos de *punteros*, y *memoria dinámica*.
- Realizar prácticas sobre contenidos de las variables utilizando punteros.

COMPETENCIAS

- Identificar, formular y resolver problemas mediante programación.
- Utilizar de manera efectiva técnicas y herramientas de aplicación para desarrollar software.
- Desempeñarse de manera efectiva en equipos de trabajo.
- Aprender en forma continua, autónoma y de manera colaborativa.

METODOLOGÍA

- El estudiante deberá resolver individualmente los ejercicios propuestos.
- El estudiante deberá codificar las soluciones en el lenguaje de programación C.
- Realizar consultas a través del canal de slack correspondiente a su comisión ó del aula virtual de la asignatura.

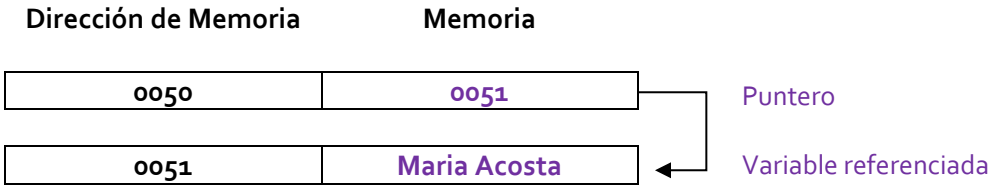
DURACIÓN

Según planificación de la asignatura se deberá utilizar para la resolución de los ejercicios de este práctico, 2 (dos) clases prácticas.

EJERCICIOS PROPUESTOS SOBRE PRÁCTICA DE PUNTEROS

1. Completar el gráfico, en función de cada consiga. Luego escribir un programa, que contenga una función por ítem en la que se declaren cada una de las variables de tipo puntero, se asigne el valor que se indica en cada caso al dato referenciado, y se visualicen tanto las direcciones como el dato referenciado. Puede utilizar asignación estática o dinámica.
   a) Una variable puntero ubicada en la dirección ***0050*** contiene un apuntador a la dirección ***0051***, la cual contiene el dato ***'Maria Acosta'***

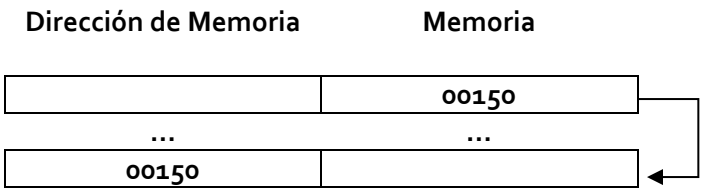
Ejemplo



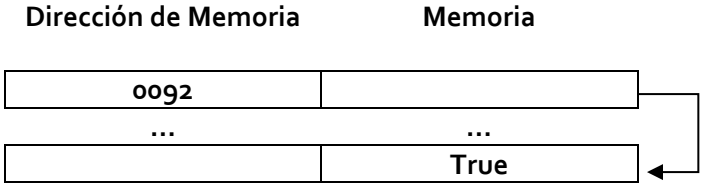
Punteros.c

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

typedef char tstring[25];

void itemA();

tstring * apuntCadena;

int main() {
    itemA();
    return 0;
}

void itemA() {
    /* Asignación dinámica */
    apuntCadena = (tstring *) malloc(sizeof(tstring));
    strcpy ((*apuntCadena), "Maria Acosta");
    printf("\nEl contenido de la variable apuntada es: %s\n", *apuntCadena);
    printf("\nLa direccion de la variable apuntada es: %p\n", apuntCadena);
    printf("\nLa direccion de la variable puntero es: %p\n", &apuntCadena);
}
```

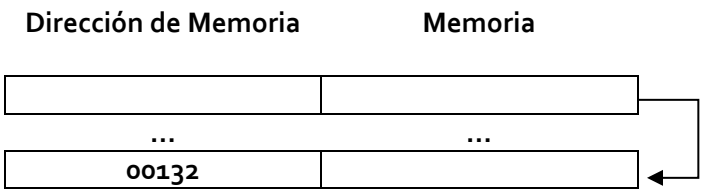
   b) Una variable puntero ubicada en la dirección ***0075*** contiene un apuntador a la dirección ***00150***, la cual contiene el dato ***20.5***



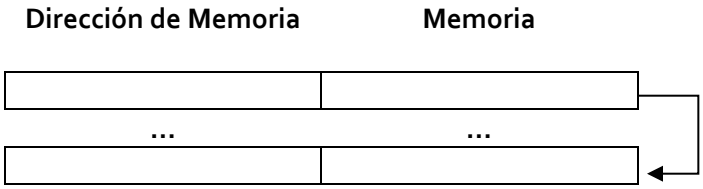
   c) Una variable puntero ubicada en la dirección ***0092*** contiene un apuntador a la dirección ***0099***, la cual contiene el dato ***True***



d) Una variable puntero ubicada en la dirección ***0125*** contiene un apuntador a la dirección ***00132***, la cual contiene el dato ***7350***



e) Una variable puntero ubicada en la dirección ***0023*** contiene un apuntador a la dirección ***0099***, la cual contiene el dato ***'z'***



2. Dadas las siguientes declaraciones:

```
typedef int indice;
typedef indice *apuntIndice;
indice i;
apuntIndice apuntI;
```

a) ¿Qué contiene **apuntI**?

b) Si a continuación de las líneas de código anteriores, ejecutamos lo siguiente:

- ¿Qué contendrá **apuntI**?
- ¿Qué contendrá ***apuntI**?

```
int main() {
        apuntI = malloc(sizeof(int));
        *apuntI = 2;
        i = 4;
        return 0;
}
```

3. Después de ejecutarse el siguiente código:

¿Qué contienen las siguientes variables?

a) apuntC
b) apuntCC
c) c
d) cc
e) *apuntC
f) *apuntCC

```
typedef int cosa;
typedef cosa *apuntadorACosa;
cosa c, cc;
apuntadorACosa apuntC, apuntCC;

int main() {
        apuntC = NULL;
        apuntCC = malloc(sizeof(int));
        return 0;
}
```

4. Suponiendo que:

```
char * eso;
```

   a) Es posible llamar a ***eso = malloc(sizeof(int));** ?
   b) Y llamar a **eso = malloc(sizeof(int));**?
   c) Explíquelo.

5. Suponiendo que:

```
typedef float acertijo;
typedef acertijo * apAcertijo;
apAcertijo a1, a2;
```

   ¿Cuáles de los siguientes enunciados serán posibles?

   a) a1 = 1.1;
   b) a1 = *1.1;
   c) a1 = malloc(sizeof(float));
   d) a1 = NULL;
   e) *a1 = 1.1;
   f) *a1 = malloc(sizeof(int));
   g) a2 = a1;
   h) a2 = *1.1;
   i) a2 = *a1;

6. ¿Qué salida tiene el siguiente programa?

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

typedef char *apuntadorC;
apuntadorC a1, a2;

int main() {
        a1 = malloc(sizeof(char));
        a2 = malloc(sizeof(char));
        *a1 = 'A';
        *a2 = 'B';
        printf("%c \n", *a1);
        printf("%c \n", *a2);
        return 0;
}
```

7. Dadas las siguientes definiciones y declaraciones

```
typedef int * tpEntero;
typedef char * tpCaracter;

tpEntero p1, p2;
tpCaracter q1, q2, q3;
```

¿Cuál será la salida de los siguientes fragmentos de código?

a)
```
p1 = malloc(sizeof(int));
p2 = malloc(sizeof(int));
*p1 = 5;
*p2 = *p1 + 20;
printf("*p1 igual a %d, *p2 igual a %d\n", *p1, *p2);
```

b)
```
p2 = malloc(sizeof(int));
*p2 = 2;
*p2 = pow(*p2,2);
p1 = malloc(sizeof(int));
*p1 = fmod(*p2,3);
printf("*p1 igual a %d, *p2 igual a %d\n", *p1, *p2);
```

c)
```
q1 = malloc(sizeof(char));
q2 = malloc(sizeof(char));
q3 = malloc(sizeof(char));
*q1 = 'Y';
*q2 = (*q1) - 1;
*q3 = (*q1) + 1;
printf("*q1 igual a %c, *q2 igual a %c, q3 igual a %c\n", *q1, *q2, *q3);
```

Verifique sus respuestas codificando este ejercicio en C.