Escuela Politécnica Superior de Elche

FUNDAMENTOS DE INFORMÁTICA
1º Ingeniería Industrial

PRÁCTICA 5:
Punteros

CURSO 2007-2008

División de Ingeniería de Sistemas y Automática

EJERCICIO 1. PUNTEROS. DECLARACIÓN Y ASIGNACIÓN.

Escribir la salida por pantalla de los siguientes programas:

```
#include <stdio.h>

main ()
{
    int v=3, *pv;
    pv = &v;
    printf("*pv=%d  v=%d\n", *pv, v);
    *pv = 0;
    printf("*pv=%d  v=%d\n", *pv, v);
    system("PAUSE");
}
```

```
#include <stdio.h>

main ()
{
    int u1, u2, v=3;
    int *pv;

    u1 = 2*(v+5);
    pv = &v;
    u2 = 2*(*pv+5);
    printf("u1=%d  u2=%d\n", u1, u2);
    system("PAUSE");
}
```

EJERCICIO 2. PUNTEROS. DECLARACIÓN Y ASIGNACIÓN.

Dados los siguientes programas en C, responder a las cuestiones:

```
char u, v= 'A';
char *pu, *pv = &v;

*pv = v+1;
u = *pv+1;
pu = &u;
```

Suponiendo que cada carácter ocupa un byte de memoria y que a la variable ***u*** se le asigna la celda de memoria con dirección 420 y ***v*** a la 421:

a) ¿Qué valor representa ***&v***?
b) ¿Qué valor es asignado a ***pv***?
c) ¿Qué valor representa ****pv***?
d) ¿Qué valor es asignado a ***u***?
e) ¿Qué valor representa ***&u***?
f) ¿Qué valor es asignado a ***pu***?
g) ¿Qué valor representa ****pu***?

```
int i, j = 25;
int *pi, *pj = &j;

*pj = j+5;
i = *pj+5;
pi = pj;
*pi = i+j;
```

Suponiendo que cada entero ocupa dos bytes de memoria, que la variable ***i*** comienza en la celda con dirección 1156, y ***j*** en la 1158:

h) ¿Qué valores representan ***&i*** y ***&j***?
i) ¿Qué valores son asignados a ***pj***, ****pj*** y ***i***?
j) ¿Qué valor representa ***pi***?
k) ¿Qué valor final es asignado a ****pi***?
l) ¿Qué valor representan las siguientes expresiones: ***(pi+2)***, **(*pi+2)**, ****(pi+2)***?

EJERCICIO 3. PUNTEROS. DECLARACIÓN Y ASIGNACIÓN.

Sean las variables ***n1*** y ***n2*** de tipo *float*, y suponer que ***n1*** se ha inicializado al valor 1.5. Completar el cuadro siguiente con sentencias en C que realicen las siguientes tareas:

a) Declarar la variable ***pf*** como un puntero a *float*
b) Asignar la dirección de memoria de la variable ***n1*** al puntero ***pf***
c) Imprimir el contenido de la posición de memoria a la que apunta ***pf***
d) Asignar el contenido de la posición de memoria a la que apunta ***pf*** a la variable ***n2***
e) Imprimir el valor de ***n2***
f) Imprimir la dirección de memoria de ***n1***
g) Imprimir la dirección de memoria de ***n2***
h) Imprimir el valor que tiene el puntero ***pf***

```
void main (void)
{
  float n1 = 1.5, n2;

  /*a)*/

  /*b)*/

  /*c)*/

  /*d)*/

  /*e)*/

  /*f)*/

  /*g)*/

  /*h)*/
}
```

Compilar y ejecutar el programa completo y responder las siguientes cuestiones:

¿Es el valor impreso en el apartado h) igual al valor impreso en el apartado f)?

¿Y al del apartado g)?

Explicar brevemente las dos respuestas anteriores

EJERCICIO 4. PUNTEROS. DECLARACIÓN Y ASIGNACIÓN.

Completar el cuadro siguiente con los valores que toman las variables después de ejecutar cada una de las sentencias. Si en algún caso la variable tiene un valor indeterminado, escribir una interrogación en la casilla correspondiente (?).

int i,j,*p1,*p2;					
Sentencias		**i**	**j**	***p1**	***p2**
1	p1 = &j;				
2	*p1 = 7;				
3	i = *p1;				
4	p1 = &i;				
5	i++;				
6	*p1 += 4;				
7	p2 = p1;				
8	*p2 *= 3;				
9	*p2 = *p1 * 2;				
10	p2 = &j;				
11	j = 45;				
12	*p2 = *p1 + i;				

EJERCICIO 5. RELACIÓN PUNTEROS-ARRAYS.

Dada la siguiente declaración:

```
int x[8] = {10, 20, 30, 40, 50, 60, 70, 80};
```

Suponiendo que el vector x comienza en la celda de memoria de dirección 186 y que cada entero ocupa dos celdas:

a) ¿Cuál es el valor de ***x***?
b) ¿Cuál es el valor de ***(x+2)***?
c) ¿Cuál es el valor de ****x***, ***(*x+2)*** y ****(x+2)***?
d) Indica formas alternativas de expresar en un programa los anteriores valores.

Dado el siguiente programa, escribir su salida por pantalla:

```
main ()
{
    int x[5] = {10, 11, 12, 13, 14}, i;

    for(i=0; i<5; i++)
    {
        printf("i=%d,  x[i]=%d,  *(x+i)=%d,  ", i, x[i], *(x+i));
        printf("&x[i] = %u,  x+i = %u\n", &x[i], (x+i));
    }

    system("PAUSE");
}
```

Dado el siguiente programa, escribir su salida por pantalla:

```
main ()
{
    char cad[] = "Fund de infor";
    int i;

    for(i=0; *(cad+i) != '\0'; i++)
        printf("%c,  %c,  %u,  %u\n", *(cad+i), cad[i], cad+i,
&cad[i]);

    system("PAUSE");
}
```

EJERCICIO 6. RESERVA DINÁMICA DE MEMORIA.

Se desea escribir un programa que calcule la media de los números que el usuario introduce por teclado. La cantidad de números se pedirá previamente. Sólo se desea reservar espacio en memoria para esta cantidad de números.

El programa necesario es:

```
#include <stdio.h>

main ()
{
    double *numeros, media=0;
    int n, i;

    printf("Cuantos numeros vas a introducir? ");
    scanf("%d", &n);

    numeros = (double *)malloc(n*sizeof(double));
    if(numeros == NULL)
        printf("Error en la asignacion de memoria\n");
    else
    {
        for(i=0; i<n; i++)
        {
            printf("Introduzca el elemento %d:  ", i+1);
            scanf("%lf", numeros+i);
            media += *(numeros+i);
        }
        media /= n;
        printf("\nLa media es %lf\n", media);

        free(numeros);
    }

    system("PAUSE");
}
```

a) Por qué es necesario realizar un reserva dinámica de memoria?
b) Reescribir las sentencias de dentro del bucle *for* utilizando notación vectorial.
c) Reescribir el programa anterior, pero creando una función para calcular la media de los números introducidos.

EJERCICIO 7. RESERVA DINÁMICA DE MEMORIA

El siguiente programa en C contiene varios errores. Se pide corregirlos, escribiendo el programa correcto en el cuadro:

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

void main(void)
{
      int *val;
      val = malloc(2);
      if (val != NULL)
      {
            printf("Introducir el primer elemento de val:");
            scanf("%d", val[0]);
            val[1] = 4;
      }
      printf("val[0] = %i\n", &val[0]);
      printf("val[1] = %i\n", &val[1]);
      free val;

}
```

Programa corregido:

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

void main(void)
{
      int *val;









}
```

EJERCICIO 8. PASO DE VARIABLES POR DIRECCIÓN.

Escribir la salida por pantalla de los siguientes programas:

```
#include <stdio.h>

void func1(int u, int v);
void func2(int *pu, int *pv);

main ()
{
    int u=1;
    int v=3;

    printf("Antes de func1: u=%d, v=%d\n", u, v);
    func1(u, v);
    printf("Despues de func1: u=%d, v=%d\n", u, v);

    printf("Antes de func2: u=%d, v=%d\n", u, v);
    func2(&u, &v);
    printf("Despues de func2: u=%d, v=%d\n", u, v);

    system("PAUSE");
}

void func1(int u, int v)
{
    u=0;
    v=0;
    printf("Dentro de func2: u=%d, v=%d\n", u, v);
}

void func2(int *pu, int *pv)
{
    *pu=0;
    *pv=0;
    printf("Dentro de func2: *pu=%d, *pv=%d\n", *pu, *pv);
}
```

```
#include <stdio.h>

void func1(int a, int *b, int c[]);

main ()
{
    int x=1, y=3, i;
    int z[4] = {0, -1, 2, -3};

    printf("Antes: x=%d, y=%d, z=[", x, y);
    for(i=0; i<4; i++)
        printf("%d, ", z[i]);
    printf("\b\b]\n");
    func1(x, &y, z);
    printf("Despues: x=%d, y=%d, z=[", x, y);
    for(i=0; i<4; i++)
        printf("%d, ", z[i]);
    printf("\b\b]\n");

    system("PAUSE");
}

void func1(int a, int *b, int c[])
{
    int i;

    a = 999;
    *b = 999;
    for(i=0; i<4; i++)
        c[i] = 1000-i;
}
```

En el anterior programa, reescribir la función func1, pero utilizando notación de puntero para la variable c.

EJERCICIO 9. PASO DE VARIABLES POR DIRECCIÓN.

a) Escribir una función que tome dos variables enteras como parámetro e intercambie sus valores en memoria. La función no debe tener un valor de retorno. Escribir una función main para probar el funcionamiento.

b) Escribir una función que tome como parámetros un vector de números reales y una variable entera que indica la cantidad de números que hay en el vector. La función debe calcular y devolver la media, varianza, máximo y mínimo de los números leidos.