Relación de prácticas de la asignatura
METODOLOGÍA DE LA PROGRAMACIÓN
Segundo Cuatrimestre
Curso 2014-2015.

1º Grado en Informática

Práctica 1: Punteros

Objetivos

Se hará hincapié en la aritmética de punteros y en el paso de parámetros por referencia tanto de tipos de datos simples como de estructuras.

Recomendaciones:

- Se recomienda utilizar el depurador cuando den fallos de segmentación de memoria.
- Todos los ejercicios que involucren vectores o cadenas deberán implementarse utilizando aritmética de punteros.
- Todas las funciones que se escriban deben tener un comentario de cabecera que contenga, al menos, la siguiente información:
  - Nombre de la función
  - Objetivo
  - Descripción de la solución (salvo que se deduzca de forma inmediata)
  - Significado de los parámetros de entrada.
  - Significado del resultado que devuelve.
  - Funciones auxiliares a las que llama.

Temporización:

2 sesiones de prácticas.

¿Qué hay que entregar?

El análisis y diseño del ejercicio 7 y de la función es_prefijo (ejercicio 9)
El análisis consistirá en el estudio del problema que plantea el ejercicio.

- Qué datos de entrada necesita y de qué tipo son
- Cómo van a llegar esos datos
- Qué resultado se va a obtener y de qué tipo es
- Cómo se obtiene el resultado a partir de los datos de entrada
- Cómo se va a presentar al usuario el resultado final
- Ejemplo de que la solución propuesta funciona, utilizando los nombres dados a los datos

El diseño incluirá un algoritmo en pseudocódigo o diagrama de flujo que resuelva el problema y que servirá como base para la posterior codificación. Recordad que el diseño es independiente del lenguaje de programación utilizado

¿Cuándo hay que entregar el análisis y el diseño?

Grupo	P3, P5, P6 y P9	P4, P8	P1, P2 y P7
Fecha	17/02/2015	18/02/2015	20/02/2015

Cuestiones sobre punteros

1. Codifica un programa que utilice la sentencia printf para escribir el tamaño de los tipos de las siguientes variables:
   int a, *b, **c;
   double d, *e, **f;

2. Explica el significado de cada una de las siguientes declaraciones:
   a) int *px;
   b) float a, b;
   c) float *pa, *pb;
   d) float a=-0.167;
   e) float *pa =&a;
   f) char c1, c2, c3;
   g) char *pcl, *pc2, *pc3 =&cl

3. Un programa de C contiene las siguientes sentencias

```
int i, j = 25;
int *pi, *pj = &i:
*pj = j + 5;
i = *pj + 5:
pi = pj;
*pi = i + j;
```

Si el valor asignado a i empieza en la dirección F9C (hexadecimal) y el valor asignado a j empieza en FE9 entonces:
   a) ¿Qué valor es representado por *&i*?
   b) ¿Qué valor es representado por *&j*?
   c) ¿Qué valor es asignado a *pj*?
   d) ¿Qué valor es asignado a **pj*?
   e) ¿Qué valor es asignado a *i*?
   f) ¿Qué valor es representado por *pi*?
   g) ¿Qué valor es asignado a **pi*?
   h) ¿Qué valor es representado por la expresión *(*pi + 2)*?

4. Un programa de C contiene las siguientes sentencias

```
float a = 0.001;
float b = 0.003;
float c, *pa, *pb
pa = &a;
*pa = 2 * a;
pb = &b;
c = 3 * (*pb # *pa );
```

Si el valor asignado a "a" empieza en la dirección 1130 (hexadecimal) y el valor asignado a "b" empieza en 1134 y el valor asignado a "c" empieza en 1138, entonces:
   a) ¿Qué valor es representado por *&a*?
   b) ¿Qué valor es representado por *&b*?
   c) ¿Qué valor es representado por *&c*?
   d) ¿Qué valor es asignado a *pa*?

e) ¿Qué valor es representado por **pa*?
f) ¿Qué valor es representado por *&(*pa)*?
g) ¿Qué valor es asignado a *pb*?
h) ¿Qué valor es representado por **pb*?
i) ¿Qué valor es asignado a c?

5. Un programa en C contiene la siguiente declaración:
   int x[8] = {10, 20, 30, 40, 50, 60, 70, 80};
   a) ¿Cuál es el significado de x?
   b) ¿Cuál es el significado de (x + 2) ?
   c) ¿Cuál es el valor de *x?
   d) ¿Cuál es el valor de (*x+2) ?
   e) ¿Cuál es el valor de *(x+2)?

Paso de parámetros por referencia

6. Se desea mostrar la equivalencia entre funciones que devuelven un resultado y funciones que utilizan parámetros por referencia.
   a) Primera versión: función denominada ***media***
      1. Recibe dos números *x* e *y* de tipo *double* pasados por valor.
      2. Devuelve como resultado la media aritmética de los números *x* e *y* pasados como parámetros.
   b) Segunda versión: función denominada ***media_referencia***
      1. Recibe dos números x e y de tipo *double* pasados por valor
      2. Recibe otro parámetro denominado ***resultado*** de tipo *double* pero pasado por **referencia**.
      3. La función debe asignar a ***resultado*** el valor de la media aritmética de *x* e *y*.
   c) Codifica un programa, denominado ***media.c***, que permita comprobar el funcionamiento de las dos funciones anteriores.

7. En un curso se han realizado dos exámenes diferentes, A y B, entre sus 50 alumnos (alumnos pares, examen A; alumnos impares, examen B). Implementa una función que calcule la nota media, máxima y mínima de cada examen.

8. Se desea codificar un programa que permita gestionar los datos de personas:
   - `nombre: array de N caracteres`
   - `apellidos: array de M caracteres`
   - `edad: entero`
   - `sexo: carácter`

   a) Implementa una función denominada leer_persona, que reciba una estructura pasada por referencia y permita leer los datos de una persona.
   b) Implementa una función denominada escribir_persona, que reciba una estructura y escriba los datos de una persona.
   c) Utilizando las funciones anteriores, en el programa principal lee y escribe un vector de personas.
   d) Crea una función que usando paso de parámetros por referencia, devuelva los datos de la persona con mayor edad y los de la persona con menor edad. Utilízala en tu programa

e) Crea un función que calcule la edad media de las personas. Utilízala en tu programa.

Punteros y arrays

9. Implementa una función que responda al siguiente prototipo: *int es_prefijo(char *cadena, char *prefijo,* que compruebe si una cadena es prefijo de otra. La función devolverá 1 si es prefijo y 0 en otro caso. Utiliza la función *strstr* de la bibliotecas <*string.h*> .

10. Escribe una función que responda al siguiente prototipo: *void replace (char *s, char nuevo, char viejo)* y reemplace en la cadena *s* todas las apariciones del caracter *viejo* por el carácter *nuevo*. Utiliza para ello la función `char *strchr(char *s, int c)` de <string.h> que devuelve un puntero a la primera aparición de *c* en la cadena apuntada por *s*. Llámala repetidas veces, pasando cada vez como cadena *s* la dirección de la última aparición.

11. Codifica un programa denominado ***vector.c*** que:
    a) Defina una constante denominada N que indique el número de elementos máximo de un vector.
    b) Declare una variable denominada ***valores*** que sea un vector (array unidimensional) de tipo *double* con N componentes
    c) Invoque a las siguientes funciones:
        1. Lectura: lee los datos del vector
        2. Escritura: escribe los datos del vector
        3. Media_aritmetica: calcula la media aritmética del vector

12. Sin usar matrices, diseña un programa que almacene los valores de una matriz de *nfil* y *ncol* en un vector de dimensión *nfil*ncol*. De cara al usuario, el programa debe comportarse como si los valores se almacenarán en una matriz. Para ello implementa las siguientes dos funciones:
    - void leerMatriz (int * m, int nfil, int ncol);
    - void escribirMatriz (int * m, int nfil, int ncol);

13. En una clase, se quieren crear parejas de alumnos para hacer un trabajo. Para representar a cada alumno se utiliza la estructura siguiente:

```
struct alumno{
   char nombre[50];
   char apellidos[150];
   int dni;
   struct alumno * companyero;
}
```

donde el campo *companyero* almacena la dirección de memoria del compañero que le toca. Escribe un programa que:

- Rellene un vector estático de *M* (valor preguntado al usuario) alumnos (el campo del compañero se inicializará a NULL). El vector tendrá un tamaño máximo de 100.
- Asigne parejas de alumnos de manera aleatoria.
- Recorra el vector e imprima las parejas creadas.