Relación de prácticas de la asignatura
METODOLOGÍA DE LA PROGRAMACIÓN
Segundo Cuatrimestre
Curso 2021-2022

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 programas hay que hacerlos con funciones.
- Todos los programas se resolverán utilizando un fichero *.h* con los prototipos de las funciones y dos ficheros *.c*, uno que contenga el *main* y otro que contenga las funciones desarrolladas para resolver el problema.

Temporización

2 sesiones de prácticas.

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;*
   *char g, *h, **i;*

2. Explica el significado de cada una de las siguientes declaraciones, así como el valor inicial que toman las variables en cada caso:

   *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, después de ejecutar todas las sentencias:

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 la variable *a* empieza en la dirección *1130* (hexadecimal) , el valor asignado a la variable *b* empieza en 1134, y el valor asignado a la variable *c* empieza en *1138*, entonces, tras ejecutar todas las sentencias:

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

NOTA IMPORTANTE: Lo deseable es que, para facilitar la reutilización del código, una función realice una única tarea, devolviendo un solo valor. En este apartado en algunos ejercicios las funciones realizan más de una tarea con el objetivo docente de practicar el 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 ***minimo***
      1. Recibe dos números *num1* y *num2* de tipo *int* pasados por valor.
      2. Devuelve como resultado el mínimo de los números *num1* y *num2* pasados como parámetros.
   b) Segunda versión: función denominada ***minimo_referencia***
      1. Recibe dos números *num1* y *num2* de tipo *int* pasados por valor.
      2. Recibe otro parámetro denominado ***resultado*** de tipo *int* pero pasado por ***referencia***.
      3. La función debe asignar a ***resultado*** el valor del mínimo de *num1* y *num2*.
   c) Codifica un programa que permita comprobar el funcionamiento de las dos funciones anteriores.

7. Codifica una función denominada ***estadisticasVector*** que reciba un vector de enteros y calcule, ***devolviendo mediante paso de parámetros por referencia***, la media, la varianza y la desviación típica de los datos del vector. Implementa un programa que mediante funciones lea un vector de enteros, lo imprima por pantalla, calcule las estadísticas y finalmente muestre los resultados.

8. Una cadena de caracteres en C es un vector de caracteres terminado en un carácter especial ('\0'). Este carácter marca la terminación de la cadena y es utilizado por las funciones de cadenas. Teniendo esto en cuenta, codifica una función denominada ***estadisticasCadena*** que reciba una cadena de caracteres y calcule, ***devolviendo mediante paso de parámetros por referencia***, el número de dígitos, mayúsculas, minúsculas y espacios que hay en la cadena. Implementa un programa para probar la función anterior. Utiliza funciones de la librería <*ctype.h*> para determinar si un carácter es dígito, mayúscula, etc.

9. Un monomio ($5x^3$) puede codificarse en C mediante una estructura con dos campos enteros *coeficiente* y *grado* (ej. coeficiente=5, grado=3). Por su parte, un polinomio es una sucesión de monomios de diferente grado ($5x^3 + 7x^2 + 4x^5$).
   a) Implementa una función denominada ***leerMonomio***, que reciba una ***estructura pasada por referencia*** y permita leer los datos de un monomio.
   b) Implementa una función denominada ***imprimirMonomio***, que reciba una estructura por valor y muestre el coeficiente y el grado del monomio.
   c) Utilizando las funciones anteriores, en el programa principal lee y escribe un polinomio (vector de monomios).
   d) Crea una función que, ***usando paso de parámetros por referencia***, devuelva el monomio de mayor grado y el de menor grado. Utilízala en tu programa.
   e) Crea un función que evalúe un polinomio en un punto *X*. Utilízala en tu programa.

Punteros y arrays

10. Codifica un programa que usando funciones y **aritmética de punteros**:
    - Lea un vector de elementos tipo *double* (*leeVector*).
    - Imprima los datos del vector por pantalla (*escribeVector*).
    - Sume los elementos mayores que cero del vector (*sumaPositivos*).

11. Implementa una programa que permita determinar si una cadena es prefijo o sufijo de otra. Utiliza la función *strstr* de la biblioteca <*string.h*> para implementar:
    - 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.
    - Una función que responda al siguiente prototipo: *int es_sufijo(char *cadena, char *sufijo),* que compruebe si una cadena es sufijo de otra. La función devolverá 1 si es sufijo y 0 en otro caso.

12. Un proyecto informático requiere trabajar con vectores de números enteros generados de manera aleatoria. Para hacer más rápido el acceso, han decidido crear dos índices que les permitan recorrer solamente los divisibles por 3 o los elementos impares. Un **índice**, es un vector de punteros, que almacena la dirección de los elementos que se quieren recorrer. Escribe un programa que:
    - Rellene un vector de enteros con números aleatorios (en un rango fijado por el usuario).
    - Imprima el vector completo.
    - Cree un índice con las direcciones de los elementos impares del vector original.
    - Cree un índice con las direcciones de los divisibles por 3 que hay en el vector original.
    - Imprima, **usando los índices**, los elementos impares y los divisibles por 3.

Implementa, al menos, las funciones: *rellenarVectorAleatorio*, *imprimirVector*, *crearndiceImpares*, *crearIndicesDivisibles*, *imprimeIndice*.