Ejercicios de repaso de C

Informática II – UTN-FRC

Gonzalo F. Perez Paina

Ejercicios

1. Escribir un programa que calcule el promedio de `N` números enteros

   a) Solicitar al usuario la cantidad de números a promediar (`N`)

   b) Solicitar al usuario los `N` valores enteros

   c) Calcular el promedio

   d) Imprimir el resultado del promedio

   La interacción con el usuario a través de la entrada y salida estándar debe ser como se muestra en el Listado 1.

   Listado 1: Ejecución de programa `promedio1`.

```
./promedio1
Ingrese la cantidad de números a promediar: 4
Ingrese el entero 1: 12
Ingrese el entero 2: 23
Ingrese el entero 3: 32
Ingrese el entero 4: 11
El promedio es: 19.500000
```

2. Modificar el programa anterior para que el programa no le solicite al usuario la cantidad de números a promediar, en su lugar el programa solicita los números enteros hasta ingresar el valor 0 (cero).

3. Escribir una función que calcule el promedio de N números enteros en base al siguiente prototipo

```
double promedio(int * , int );
```

   junto al programa que evalúe su correcto funcionamiento.

4. La función *factorial* ($n!$) de un entero positivo se define como el producto de los enteros de 1 hasta n, por ejemplo: $3! = 6$, $4! = 24$. Escribir un programa que calcule los factoriales de los números desde el 1 hasta el 10 e imprima los resultados. El Listado 2 muestra la salida del programa.

   Listado 2: Ejempo de cálculo del factorial.

```
./factorial
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

5. Escribir un programa que muestre en pantalla el valor que deben tomar los índices (i, j) para acceder a los elementos de una matriz (arreglo bidimencional) de N filas y M columnas, o sea $\{(i, j)/i = 0, \cdots, N-1, j = 0, \cdots, M-1\}$. El tamaño del arreglo se define mediante constantes simbólicas (`#define`) dentro del programa: `N` para la cantidad de filas y `M` para la cantidad de columnas. La salida en pantalla del programa debe ser como se muetra en el Listado 3 (para `N=2` y `M=3`). Probar el programa con diferentes valores de `N` y `M`.

Listado 3: Ejecución de programa `matrixidx1`.

```
./matrixidx1
i = 0   j = 0
i = 0   j = 1
i = 0   j = 2
i = 1   j = 0
i = 1   j = 1
i = 1   j = 2
```

6. Modificar el programa anterior para que la salida sea como se muetra en el Listado 4.

Listado 4: Ejecución de programa `matrixidx2`.

```
./matrixidx2
(0, 0) (0, 1) (0, 2)
(1, 0) (1, 1) (1, 2)
```

7. Escribir los programas necesarios para verificar las funciones `pow()`, `sqrt()`, `exp()`, `log()`, `log10()`, `ceil()` y `floor()` de la biblioteca matemática (archivo de cabecera `math.h`). (Ver cap.5 D&D 4ºEd.)

8. Escribir una función que calcule la distancia entre dos puntos, $p_1 = (x_1, y_2)$ y $p_2 = (x_2, y_2)$. El Listado 5 muestra el código fuente del programa al cual hay que agregar la función que realiza el cálculo.

Listado 5: Archivo `distancia.c`.

```c
#include <stdio.h>
#include <math.h> // funciones pow y sqrt

/* Prototipo de la función */

int main(void)
{
  printf("La distancia entre (%g,%g) y (%g,%g) es %g\n",
      1.0, 2.0, 3.0, 4.0, distancia(1.0, 2.0, 3.0, 4.0));
  printf("La distancia entre (%g,%g) y (%g,%g) es %g\n",
      2.0, 3.0, 3.0, 2.0, distancia(2.0, 3.0, 3.0, 2.0));
  printf("La distancia entre (%g,%g) y (%g,%g) es %g\n",
      -1.0, -1.0, 2.0, 2.0, distancia(-1.0, -1.0, 2.0, 2.0));
  return 0;
}

/* Implementación de la función */
```

9. Escribir un programa con la misma funcionalidad que el enunciado en el ejercicio 8 donde los puntos p_1 y p_2 se almacenen como arreglos unidimensionales (vectores) de dimensión 2.

10. Escribir un programa con la misma funcionalidad que el enunciado en el ejercicio 8 donde los puntos p_1 y p_2 se almacenen como estructuras (`struct`) con los campos x e y.

11. Transcribir el código fuente del Listado 6, construir el programa y ejecutarlo. ¿Cuántos bytes ocupa en memoria el arreglo bidimensional de nombre `dato`?

Listado 6: Archivo `dias1.c`.

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

```
2
3  char dato[5][10] = { {76, 117, 110, 101, 115, 0},
4                       {77, 97, 114, 116, 101, 115, 0},
5                       {77, 105, 101, 114, 108, 101, 115, 0},
6                       {74, 117, 101, 118, 101, 115, 0},
7                       {86, 105, 101, 114, 110, 101, 115, 0} };
8
9  int main(void)
10 {
11   int i, j;
12   for(i = 0; i < 5; i++)
13   {
14     for(j = 0; j < 10; j++)
15       printf("%c", dato[i][j]);
16     printf("\n");
17   }
18
19   return 0;
20 }
```

12. Reemplazar la función `main` del código fuente del ejercicio anterior por el código mostrado en el Listado 7, construir y ejecutar el programa. ¿Qué diferencia encuentra en el resultado de ambos programas?. Justificar la respuesta.

Listado 7: Función `main` del archivo `dias2.c`.

```
1  int main(void)
2  {
3    int i;
4    for(i = 0; i < 5; i++)
5      printf("%s\n", dato[i]);
6
7    return 0;
8  }
```

13. Transcribir la definición de la variable `dato` del código fuente del ejercicio 11 en un archivo de texto plano y guardar con el nombre `dias.h`. Luego, realizar las siguientes modificaciones a los programas de los ejercicios 11 y 12:

    a) borrar la definición de la variable `dato`

    b) agregar la línea `#include "dias.h"`

    Construir y ejecutar estos programas (nombrar como `dias1a.c` y `dias2a.c`). ¿Qué puede decir de los archivos de cabecera `.h`?

14. Escribir un programa para obtener el resultado del ejercicio 11 donde los días de la semana se almacenen en un arreglo de punteros tipo `char`. ¿Cuántos bytes se necesitan para almacenar los nombres de los días de la semana?. Compare el resultado con el del ejercicio 11.

15. Escribir un programa que imprima el valor entero ingresado por teclado (en el rango de 0 a 255) en formato decimal, hexadecimal y binario. El programa debe tener la interacción con el usuario que se muestra en el Listado 8.

Listado 8: Ejecución de programa `dec_hex_bin`.

```
Ingrese un entero entre 0 y 255 (-1 para salir): 2
   002 d = 0x02 = 0000 0010 b
Ingrese un entero entre 0 y 255 (-1 para salir): 4
   004 d = 0x04 = 0000 0100 b
Ingrese un entero entre 0 y 255 (-1 para salir): -1
```

16. Escribir un programa que muestre en pantalla la cantidad de bytes que ocupan en memoria las variables de los diferentes tipos de datos disponibles en C, utilizando para ello el operador `sizeof`. Los tipos de datos son: `char` e `int` en su versión `signed`/`unsigned` para ambos y `short`/`long` para `int`; además de `float` y `double`.

17. Escribir un programa que trabaje con un tipo de dato adecuado para almacenar los parámetros de un círculo, a saber: coordenadas del punto central en un sistema cartesiano y radio, utilizando para ello estructuras (`struct`). El programa tiene que:

    a) Declarar un tipo de dato `punto` con una estructura

    b) Declarar un tipo de datos `circulo` que incluya un miembro de tipo `punto` para almacenar el centro del círculo, y un miembro tipo `double` para almacenar el radio del círculo

    La interacción con el usuario a través de la entrada y salida estándar debe ser como se muestra en el Listado 9.

Listado 9: Ejecución del programa `circulo1.c`.

```
Ingrese las coordenadas del círculo (x, y): 1.2 2.3
Ingrese el radio del círculo: 2.5

El círculo ingresado es:
  centro: (1.2, 2.3)
  radio: 2.5
```

18. Agregar al programa del ejercicio 17 el cálculo de la distancia del centro del círculo al origen del sistema de coordendas y del área del círculo. Para ello, implementar las funciones cuyos prototipos son:

```
double dist_origen(struct circulo );
double area_circ(struct circulo );
```

    (Nota: para el cálculo del área utilizar la constante simbólica `M_PI` definida en el archivo de cabecera `math.h`).
    Indicar si los parámetros de las funciones son pasados por valor o por referencia.

19. Escribir um programa que calcule el producto escalar (punto) y vectorial (cruz) de dos vectores enteros de dimensión 3. La interacción con el usuario a través de la entrada y salida estándar debe ser como se muestra en el Listado 10.

Listado 10: Ejecución de programa `vproduct`.

```
./vproduct
Producto escalar y vectorial de vectores de dimensión: 3
Ingrese los elementos del primer vector: 1 2 3
Ingrese los elementos del segundo vector: 4 5 6
Producto punto: 32
Producto vectorial: -3 6 -3
```

20. El archivo de cabecera `limits.h` define constantes simbólicas con los valores máximos y mínimos para los diferentes tipos de datos de C. Escribir un programa que muestre en pantalla el valor de las siguientes constantes simbólicas de dicho archivo: `CHAR_BIT`, `CHAR_MIN`, `CHAR_MAX`, `SCHAR_MIN`, `SCHAR_MAX`, `UCHAR_MAX`, `SHRT_MIN`, `SHRT_MAX`, `USHRT_MAX`, `INT_MIN`, `INT_MAX`, `UINT_MAX`, `LONG_MIN`, `LONG_MAX`, `ULONG_MAX`, `LLONG_MIN`, `LLONG_MAX`, `ULLONG_MAX`. Además:

    a) Investigue qué representa cada valor.

    b) Comprobar los valores obtenidos a partir de la cantidad de bytes que ocupan en memoria los diferentes tipos de datos (ver ejercicio 16).