PRACTICA DEL MODULO 4

Ejercicios

Breve descripción

Ejercicios de programación para el módulo 4.

Versión 1.3

Martin Jerman
Martin.jerman@inspt.utn.edu.ar

Ejercicios con funciones

IMPORTANTE: Lea atentamente el enunciado antes de empezar a resolver cada ejercicio.

1) Ejecuta el siguiente programa y responde:

```
main()
{
printf("%lf", sqrt(floor(fabs(-16.8))));
getch();
return 0;
}
```

   a. ¿Qué realiza cada función predefinida?
   b. ¿Cuántos parámetros necesita cada una de esas funciones?

2) El siguiente programa indica si un número leído desde el teclado es par:

```
main ()
{
  int numero;
  scanf ("%d", &numero);
  if (numero%2 == 0)
    printf ("Es un número par");
  else
    printf ("Es un número impar");
}
```

   a. Convierte el programa en una función `void esPar(int unNumero)`.
   b. Luego convierte el programa en una función `int esPar(int unNumero)` que devuelve "verdadero" si es par. ¿Tendría más sentido?
   c. Piensen cual sería su principal diferencia para el usuario programador.

3) Desarrolle la función "`mayorDeDos`" que reciba dos enteros por parámetro y muestre por pantalla al mayor de ellos. Luego modifíquela a "`getMayorDeDos`" para que devuelva el mayor como resultado.

4) Desarrolle la función "`potencia`" que reciba por parámetros un entero X y una potencia Y; y devuelva por resultado X^Y.

5) Desarrolle la función "`tablaDeMultiplicar`" que reciba un entero por parámetro y muestre por pantalla su tabla de multiplicar de 0 a 10.

6) Desarrolle la función "`sumaIntervalo`" que reciba dos enteros por parámetro y devuelva por resultado la suma de todos los números enteros entre dichos valores (inclusive).

7) Desarrolle la función "`menu`" que muestre por pantalla 4 opciones, pida ingresar una de esas opciones y devuelva por resultado la opción elegida. La función debe validar que la opción ingresada sea valida 8si hay 4 opciones, no debo poder elegir la opción 6).

8) Desarrolle la función "`esPrimo`" que devuelva verdadero si el numero pasado por parámetro es un numero primo.

9) Desarrolle la función "`múltiplo`" que recibe dos valores enteros y emite "verdadero" si el primero es múltiplo del segundo.

10) Desarrolle la función "`areaRectangulo`" que reciba 3 parámetros (base, altura y área) devolviendo el área en los parámetros.

11) Realiza una función de nombre Siguiente tal que, recibiendo un número primo mayor que uno, devuelva el número primo inmediatamente siguiente y superior a dicho número primo. Por ejemplo, si se invoca `siguiente(7)`, la función devolverá el número 11.

12) Desarrolle la función "`esBisiesto`" que reciba un año por parámetro y devuelva "verdadero" si el año es bisiesto (Un año es bisiesto si es divisible por 400, o bien si es divisible por 4 pero no por 100).

13) Desarrolle la función "`fechaValida`" que reciba por parámetro un día, un mes y un año y devuelva por resultado "verdadero" si la fecha es válida (tener en cuenta años bisiestos).

14) Desarrolle la función `void maxmin (int x1, int x2, int* max, int* min);` que recibiendo por parámetros x1 y x2, devuelva el menor de ambos en `min` y el mayor en `max`.

15) Desarrolle la función `void datoValidado (int *dato, int min, int max)` que reciba un mínimo y un máximo por parámetro; que pida por teclado el ingreso de un valor, valide que el valor este entre min y max y lo devuelva en *dato.

16) Desarrollar una función de encabezado `int ordenarMayor(int* v1, int* v2, int*v3)` en la que la función ponga en V1 el menor valor de las tres variables, en v2 el del medio y en v3 el mayor. Noten el encabezado y el nombre de la función (que haga lo que el nombre de la función dice que hace).

17) Luego, análogo al punto anterior, desarrollen un función `int ordenarMenor(int* v1, int* v2, int*v3).`

18) Con las funciones de los dos puntos anteriores, queda la pregunta: ¿sería posible hacer una única función que pudiera ordenar las tres variables de menor a mayor o de mayor a menor según se requiera? Plantéenlo, desarróllenlo y hagan un programa que las utilice. Si pudieron hacerlo, dieron un gran paso hacia la anidación de funciones.

19) Desarrolle la función `void acumulador(int* acumCat1, int* acumCat2, char categoría)` y que acumule en acumCat1 si la categoría ingresada es 'A' o en acumCat2 si es 'B'. Utilice la función en un programa con un ciclo para que acumule.

20) Escribe una función que reciba como parámetro de entrada un número entero y devuelva como resultado el número de cifras del número.

21) Escribe una función que reciba como parámetros de entrada un valor entero y compruebe si se encuentra comprendido entre dos valores constantes MIN y MAX definidos dentro de la propia función.

22) Escribe una función que reciba como parámetros de entrada tres números enteros que representan las longitudes de tres segmentos rectilíneos, y devuelva como resultado un valor de tipo lógico que indique si dichos segmentos pueden formar o no un triángulo (la condición necesaria pero no suficiente es que ninguno de los segmentos tenga una longitud superior a la suma de los otros dos).

23) RECOMENDADO: Escribe un programa teniendo en cuenta las siguientes funciones:
    - `leeOpcion` lee la opción deseada y comprueba su validez
    - menú muestra el menú en la pantalla
    - cuadrado, circulo, rectángulo, trapecio, triángulo calculan la superficie correspondiente.

El menú por mostrar sería algo como lo que sigue:

```
**********************************
****** CÁLCULO DE SUPERFICIES ******
1. Cuadrado (lado*lado)
2. Círculo (pi*radio*radio)
3. Rectángulo (base*altura)
4. Trapecio (base1+base2)*altura/2)
5. Triángulo (base*altura)/2)
0. Salir del programa
**********************************
```

24) Realiza un programa que lea un número de hasta 8 cifras y emita por pantalla la cifra resultante de aplicar el siguiente proceso:
   a) Sumar el valor absoluto de todas las cifras del número.
   b) Si el valor resultante tiene más de una cifra, volver a sumar todas sus cifras sucesivamente hasta obtener un valor de una única cifra.

      Ejemplos: Valor introducido por el usuario: 68543210 -> 6+8+5+4+3+2+1+0 = 29; 29 -> 2+9 = 11; 11 -> 1+1 = 2: Valor a mostrar: 2

25) Construir una función que permita procesar los datos de productos vendidos. La función pedirá por teclado:
   a) El id de producto (número entero > 0 y menor a 1000). Si es invalido, saltearlo; si es válido, pedir lo demás:
   b) Precio de costo, ej: 5.30
   c) Precio de venta, ej: 9.50
   d) Cantidad vendida, ej: 25

   La función debe:

   a- Mostrar por pantalla la ganancia de cada id de producto.
   b- Calcular las ganancias totales de todos los id de productos.
   c- Calcular la cantidad de id de productos procesados.

   Desarrollar un programa que utilice la función anterior y que emita el total de ids de productos procesados y las ganancias que se obtuvieron por todas las ventas. Ej: se procesaron 43 productos, las ganancias obtenidas fueron $2398.

Ejercicios con funciones recursivas

26) Desarrolle una función que reciba un numero entero y recursivamente y devuelva la suma de sus cifras.

27) Desarrolle la función multiplicar que reciba dos números enteros y los multiplique recursivamente (recuerde que la multiplicación es una serie de sumas sucesivas).

28) Escriba una función recursiva que implemente:
   a) Mostrar los números del 1 al N en orden creciente.
   b) Mostrar los números del 1 al N en orden decreciente.

29) Escriba el planteo recursivo e implemente en C los siguientes ejercicios teniendo en cuenta las restricciones impuestas para cada caso.
   a) Una función recursiva resto: `N x N→N` que obtenga el resto (módulo) de la división entera utilizando como única operación aritmética la resta (no puede usarse `div`). Ej.: `resto(5,2) = 1, resto(8,2) = 0, resto(1,2) = 1`.
   b) Una función recursiva `divEntera`: `N x N→N` que obtenga el cociente (resultado) de la división entera utilizando como únicas operaciones aritméticas la suma y la resta.
   c) Una función recursiva cuadrado: `N→N` que obtenga el cuadrado de un número natural distinto de cero utilizando exclusivamente el siguiente método: el cuadrado(k) es igual a la suma de los k primeros números impares. Por ejemplo, el cuadrado de `4` es `1+3+5+7=16`.

30) Escriba un planteo recursivo e implemente en C los siguientes ejercicios:
   a) Una función recursiva que determine si un dígito D no pertenece a un número entero positivo N. Ej.: si `N=1323` y `D=5` el resultado es Verdadero, y si `D=1` el resultado es `Falso`.
   b) Una función que cuente la cantidad de dígitos pares en un número entero. Ej.: si el número es 22005 el resultado es 4, y si fuera 35 el resultado es 0.

31) Escriba un planteo recursivo e implemente en C los siguientes ejercicios:
   a) Una función recursiva que determine si un número natural es potencia de 2. Ej.: `espot2(33) = false`, `espot2(64)=true`.
   b) Una función recursiva que determine si dígito D está ubicado en la posición más significativa de un número natural. Ej.: `pmasS (2,2345) = true, pmasS (6,5604) = false, pmasS (7,945) = false`.

32) Escriba el planteo recursivo e implemente en C una función recursiva que calcule la suma de los dígitos que ocupan posiciones impares para un número natural. Se considera que la posición 1 es la posición del dígito menos significativo (lugar de la unidad), la posición 2 es la posición de la decena, etc. Por ejemplo, si se considera el natural 587, el 7 está en la posición 1, el 8 en la posición 2 y el 5 en la posición 3. En el ejemplo, la función debería retornar 12 (7+5).

33) Dado un número natural, definiremos como su número promedio al número que se obtiene de sumar sus dígitos impares y restar sus dígitos pares. Por ej.: el número promedio de `318547` es `4` esto es, `numeroPromedio(318547) = numeroPromedio(31854)+7 = numeroPromedio(3185) - 4 + 7 =` … Escriba el planteo recursivo e implemente en C una función que obtenga su número promedio. Vale inspirarse en `strtok`.

34) Escriba un planteo y una función recursiva para imprimir una media pirámide de dígitos como se muestra en la siguiente figura. Utilice un procedimiento recursivo para generar cada fila de la media pirámide.

```
1
21
321
```

```
4321
54321
```

35) Dada una secuencia de números enteros positivos finalizada en -1 (el cual no se considera parte de esta), escribir un planteo recursivo y la correspondiente implementación para:
   a) Sumar todos los enteros de dichas secuencia. Ej.: Para la secuencia 2 5 3 6 12 3 -1 el resultado es 31.
   b) Mostrar por pantalla todos los valores de la secuencia que sean divisibles por el último valor de esta.
   c) Calcular el promedio de los valores de la secuencia.
   d) Determinar el k-ésimo elemento de la secuencia comenzando desde adelante. El valor k debe ser proporcionado por el usuario. Ej.: Para la secuencia 2 5 3 6 12 3 -1 y k = 4 el resultado es 6.

Ejercicios avanzados

36) El INSPT requiere de un programa para procesar notas de alumnos de un solo curso. Para ello se pide un programa que:
   a) En main se pida ingresar el número de curso, cantidad de clases totales y los legajos de alumnos a procesar
   b) Una función que reciba **como mínimo** un legajo por parámetro y las clases totales. Luego debe pedir por teclado la cantidad de clases asistidas y las 3 notas de sus parciales. La función devuelve el porcentaje de asistencia.
   c) Otra función que reciba las notas y devuelva el promedio.
   d) Al final del programa, se debe emitir: la cantidad total de alumnos, el legajo con el mejor promedio (mostrar legajo y promedio) y el legajo con mejor asistencia (mostrar legajo y asistencia).

37) El radar de artillería la Armada Argentina es capaz de detectar objetos hasta 20km de distancia a los cuales hacer blanco.
   a) Se pide un programa que implemente la funcionalidad `distancia(x,y)` que dada una coordenada GPS, nos devuelva la distancia al objeto.
   b) Otra función que pida coordenadas y pueda evaluar la peligrosidad del objetivo. Si esta entre 0 y 5 km es de peligrosidad ALTA, si esta de 5 a 10 km es de peligrosidad MEDIA y si esta entre 10 y 20 km es de peligrosidad BAJA.
   c) Además, el programa debe contabilizar la cantidad total de objetivos, la cantidad de objetivos por cada nivel de peligrosidad y la cantidad de objetivos fuera de alcance.

38) El videojuego Mortal Kombat es un juego de lucha entre dos jugadores. Desarrollar el juego de lucha entre dos jugadores que:
   a) Al iniciar el programa, pida los datos de cada jugador: vida (`int`), ataque (`int`), defensa (`int`).
   b) Implemente la función `atacar()` que debe recibir los parámetros necesarios para modelar el ataque de un jugador a otro.
   c) Al iniciar la pelea, el jugador1 inicia atacando al jugador2. Si el jugador2 muere, se termina el juego. Si no, el jugador2 devuelve el ataque. Si el jugador1 muere, termina el juego.
   d) Al finalizar la pelea, se debe mostrar qué jugador ganó y con cuánta vida quedó.

39) La rotisería Morfi requiere de un sistema de cómputo de pedidos. El sistema es sencillo: cada pedido tiene un código numérico de `3` dígitos y un precio. El código numérico tiene el formato `234`, donde:
   `2`: id de comida: 1) pollo 2) pizza 3) empanadas 4) ensaladas
   `3`: id de tipo de preparación: 1) a la parrilla 2) al horno 3) frito 4) frio
   `4`: cantidad (de 1 a 9)
   Se requiere de un programa permita:
   a) Mostrar un menú de opciones: 1- Ingresar pedidos, 2- mostrar cómputos, 3- salir.
   b) Ingresar pedidos debe pedir el código numérico y su precio para luego computarlo
   c) Implementar una función `computarDatos()` que reciba los datos necesarios para computar:
      i) Cantidad total de pedidos por id de comida
      ii) El código numérico del pedido más grande (de mayor cantidad)
      iii) El código numérico del pedido mas caro (de mayor precio).
   d) Mostrar por pantalla los datos computados hasta el momento.
   e) Salir del programa mostrando el mensaje "`Gracias por utilizar el sistema`".