INSO - Introducción a la programación I – Curso 2023 - 2024

Ejercicio Evaluable 2

La práctica 2 estará formada por los siguientes 5 ejercicios:

A. Cálculo numérico
B. Suma recursiva
C. Contador de palabras
D. Catán. Probabilidades
E. Clasificador de productos por tamaño

A continuación, encontrarás los detalles de cada enunciado para escribir el código correspondiente. En todos ellos debes realizar el análisis del problema que se plantea, estudiando los datos de entrada que necesitas, plantea a continuación las sentencias de ejecución en base a las operaciones que debes realizar y finalmente muestra el resultado.

Se valorará si se ejecuta correctamente, así como la estructura general, la claridad y la legibilidad. Recuerda siempre utilizar identificadores que sean adecuados y el uso de comentarios. En cada ejercicio se pide crear una función para operaciones específicas, es responsabilidad del alumno implementarlas con la mayor precisión posible.

Ejercicio A. Cálculo numérico

Calcula el resultado de una operación en la que el programa genera de forma aleatoria tanto dos operandos (números enteros) como una operación (suma, resta, multiplicación y división). El usuario formula una solución y hasta que no es correcta el programa le sigue pidiendo la una solución correcta.

Las restricciones son las siguientes:

- Los operandos tienen que estar entre 1 y 9.
- Si la operación es una división el resultado tiene que ser exacto.
- Los resultados no pueden ser mayores que 200 ni menores que 0

Las funciones recomendadas para que se implementen en la realización del programa son las cuatro siguientes:

//Genera un numero aleatorio entre 0 y 9
//Utiliza la función rand (recuerda la generación de la semilla con la función srand
//Devuelve el numero entero y no se le pasan parámetros
int generarNumero();
//Genera una operación pudiendo ser suma, resta, multiplicación y división
//Devuelve un carácter que puede ser '+','-','' y '/'*
//No se le pasa ningún parámetro
//Utiliza la función rand

char generarOperacion();
//Genera el resultado de la operación
//Se le pasa como parámetros los operandos (números enteros) y el signo de la operación (carácter)
//Devuelve el numero entero con el resultado
int generarResultado(int Ope1, int Ope2, char SimboloOperacion);
//Comprueba si la operación es correcta
//Los criterios son:
//la división es exacta y la resta positiva y el resultado es menor que 200
//Se le pasa como parámetros los operandos (números enteros) y el signo de la operación (carácter)
//Devuelve el numero entero a modo de booleano
//Si la operación no es correcta y 1 si la operación es correcta
int comprobarOperacion(int Ope1, int Ope2, char SimboloOperacion, int ResOpera);

Para generar los números y las operaciones de manera aleatoria utiliza las siguientes funciones de las librerías <stdlib> y de la librería <unistd.h>:

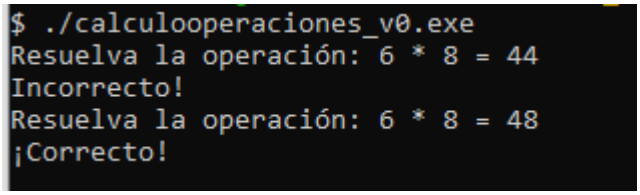
- **int rand(void)** [Returns a pseudo-random number in the range of 0 to RAND_MAX]
- **void srand(unsigned int seed)** [This function seeds the random number generator used by the function rand].
- **pid_t getpid(void))** [To retrieve the Process ID (PID) of the currently running process. PID is a unique identifier assigned to each process by the operating system]

Un ejemplo de uso pude verse a continuación

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

int main() {
    srand((unisigned int)gettpid);
    int numeroAleatorio = rand();
    return 0;
}
```

Un ejemplo de salida por pantalla es:



Ejercicio B. Suma recursiva de las cifras de un número

Sumar las cifras de un número entero positivo usando recursividad. Dado un número entero M, la suma de sus dígitos S(M) puede calcularse como:

$$S(M)\begin{cases} M \ si \ M < 10 \\ (S(M/10) + (M \,\%\, 10) \ si \ M >= 10 \end{cases}$$

Por ejemplo:

```
S(159) = S(15) + 9  = S(1) + 5 + 9 = 15
```

Escribir una función que devuelva la suma de los dígitos de un número entero, cuyo prototipo será

int CalcSuma(int input);

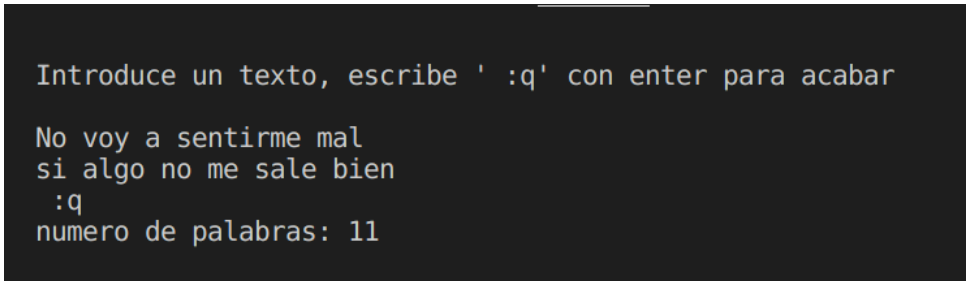
Y que se invocará solo una vez dentro del main para calcular la suma de las cifras del entero 88199, e imprimir el resultado

Ejercicio C Contador de palabras

Realizar un programa que cuente las palabras introducidas por teclado. El programa entenderá como palabra todo lo que esté entre espacios o saltos de línea. Para acabar el usuario deberá escribir la secuencia " :q" con intro a continuación.

- No se permite utilizar librerías ni tipos de datos con los que no se ha trabajado en clase.
- Para leer lo escrito en el terminal, usar la función "getchar()" vista en clase. Esa función devuelve cada carácter escrito por separado.

Ejemplo de la salida por pantalla:



Ejercicio D Probabilidades del Catán

El Catán es un juego de mesa en el que los jugadores obtienen cartas con materias primas en función de los números asignados a sus pueblos y tiradas de dos dados.

Cada pueblo puede tener un máximo de 3 números asociados a 3 materias primas (iguales o diferentes), de manera que cuando sale uno de esos números obtienen una carta de esa materia prima. Un pueblo puede tener varios números iguales, en ese caso cada vez que salga dicho número obtendrá una carta por cada materia situada en ese número.

Así, como ejemplo, los pueblos situados en un 6 tienen mayor probabilidad de obtener una carta que los pueblos situados en un 12.

Crear un programa que calcule las probabilidades de obtener cartas en el catán. Para ello dará dos opciones a elegir:

- Pedirá los números de dos pueblos y dará las probabilidades de cada pueblo de obtener una carta o más en cada tirada. Si un pueblo no tiene los 3 números el usuario introducirá un 0.
- Pedirá los números en los que el jugador tiene una materia prima y dará la probabilidad de obtener la materia en una tirada

Se recomiendan las siguientes funciones a implementar:

- **`void probMateria();`**

`Se encarga de la opción de obtener la probabilidad de una materia`

- **`void probMayor2Pueblos();`**

`Se encarga de obtener las probabilidades de los dos pueblos`

- **`double probNumero(int numero);`**

`Obtiene la probabilidad de obtener un número en una tirada de dos dados. Numero deberá tener un valor entre 2 y 12.`

- **`int casosNumero (int tirada);`**

`devuelve los casos posibles para obtener esa tirada. Por ejemplo, para 12 devoverá 1, sólo es posible obtener un 12 con un 6 + 6.`

- **`void compruebaNumerosRepetidos (int num1, int *num2, int *num3);`**

`comprueba si hay números repetidos en un pueblo y si los hay los modifica con un 0 para no añadir casos repetidos.`

Ejemplos:

```
CATAN.... probabilidades

Para obtener el pueblo con mayor probabilidad pulse 1
Para obtener la probabilidad de una materia prima pulse 2
1
Introduce los 3 numeros de cada pueblos.
Si un pueblo no tiene los 3 numeros, introduce un 0 en su lugar
Introduce los numeros del primer pueblo: 3 4 6
Introduce los numeros del segundo pueblo: 8 4 2
Los numeros son : 3, 4, 6
Los numeros son : 8, 4, 2
La probabilidad de obtener una carta en el primer pueblo es: 27%
La probabilidad de obtener una carta en el segundo pueblo es: 25%
```

```
CATAN.... probabilidades

Para obtener el pueblo con mayor probabilidad pulse 1
Para obtener la probabilidad de una materia prima pulse 2
2
Introduce cuantos pueblos tienes con numeros diferentes en esa materia:3

Introduce los numeros de los 3 pueblos:5 8 9

La probabilidad de obtener esa materia en una tirada es de: 36%
```

Ejercicio E: Clasificador de productos por tamaño

Se pide crear un programa que ayude a clasificar productos según su tamaño, para poder almacenarlos en cajas de un tamaño "ancho x alto x fondo" preestablecido. Se tienen los siguientes tamaños:

- Cajas pequeñas: 10x10x15 cm
- Cajas medianas: 20x15x15 cm
- Cajas grandes: 30x25x20 cm

El programa pedirá al inicio el número de productos que se clasificarán. Por cada producto deberá pedir sus medidas (ancho, alto y fondo) y le asignará una caja ajustada a sus medidas, dando prioridad a las cajas más pequeñas. Ej:

- Un producto de 15x10x5 cm deberá asignarse en una caja "mediana"
- Es responsabilidad del usuario dar las medidas en orden correcto (no hace falta comprobar si los productos pudieran caber haciendo distintas combinaciones de ancho/alto/fondo)
- Si un producto no cabe en ninguna caja, se apunta como "producto no válido"

Una vez acabada petición de medidas de cada producto, el programa mostrará en una única línea:

- El número total de cajas usadas
- Número de cajas usadas por tipo (número de cajas pequeñas, medianas, grandes)
- El número de productos marcados como "no válidos"

Se pide crear las siguientes funciones para poder realizar el ejercicio:

- **`void pideMedidas(int* ancho, int* alto, int* grosor);`**

  Esta función pide tres números al usuario, y los devuelve usando variables modificables por cabecera.

- **`int calculaTipoCaja(int ancho, int alto, int grosor);`**

  Esta función recibe tres números por parámetros, calculará el tipo de caja según las condiciones del enunciado, y devolverá un único número como resultado. Las opciones que puede devolver son las siguientes:
  - 1 : En caso de haber calculado una caja pequeña
  - 2 : En caso de haber calculado una caja mediana
  - 3 : En caso de haber calculado una caja grande