Fundamentos de Programación

Grado en Ing. Informática

Guion práctico nº 3

Tema 4.- Tipos de datos estructurados

1. Diseñe un programa que declare un vector de 10 elementos (enteros o reales), lo rellene completamente con valores leídos desde teclado y calcule y muestre por pantalla el mínimo y el máximo de todos ellos.

2. Diseñe un programa que declare un vector de 10 elementos enteros, lo rellene completamente con valores leídos desde teclado, pida por teclado un valor X a buscar y muestre por pantalla un mensaje de texto indicando si el valor X se encuentra o no en el vector.

3. Diseñe un programa que declare una tabla de 10 filas y 15 columnas (10x15) de elementos enteros, la rellene completamente con valores aleatorios en el intervalo **[0,100]**, pida por teclado un valor **X** a buscar y muestre por pantalla un mensaje de texto indicando si el valor **X** se encuentra o no en la tabla.

   Para generar números aleatorios serán de utilidad las siguientes funciones:
   - **`rand()`**. Se encuentra en la librería **`cstdlib`** y obtiene un número aleatorio entre `0` y **`RAND_MAX`** (constante con un valor muy elevado). Esta función siempre genera la misma secuencia de números aleatorios, para cambiar este comportamiento se debe utilizar la función **`srand`**.
   - **`srand()`**. Establece una nueva secuencia de números aleatorios a partir de un valor proporcionado denominado semilla. Para conseguir diferentes secuencias de números aleatorios hay que llamar a esta función con distintas semillas al principio del programa. Para ello la función **`srand`** se utiliza conjuntamente con función **`time(0)`**, que ofrece la hora actual y que se encuentra en la librería **`ctime`**. Así pues, **`srand()`** se usaría en nuestro código del siguiente modo **`srand(time(0))`**.

4. Diseñe un programa que declare una tabla de 3 filas y 4 columnas de elementos tipo cadena denominada **`diccionario`**. El tipo cadena es definido de la forma:

```
typedef char cadena[30];
```

   El programa debe rellenar toda la tabla con palabras introducidas desde teclado. A continuación se debe pedir una nueva palabra **S** (definida también como **`cadena`**) para ser buscada en la tabla. El programa debe mostrar un mensaje que indique si la palabra **S** se encuentra o no en la tabla **`diccionario`**.

   Nota: Se recuerda que para manejar cadenas se debe hacer mediante las operaciones definidas en la librería `cstring`.

5. Diseñe un programa que declare una tabla de 2 filas y 4 columnas (2x4) de tipo persona definido de la siguiente manera:

```
typedef char cadena[30];

struct persona {
  int dni;         //DNI de la persona sin letra
  cadena nombre;   //Nombre de la persona
};
```

El programa deberá rellenar completamente la tabla con datos leídos desde teclado. Además el programa solicitará un DNI a buscar. En caso de que se encuentre el DNI en la tabla, el programa mostrará por pantalla los datos de la persona (DNI y nombre) y en caso contrario mostrará un mensaje de error indicando que no se ha encontrado a la persona con el DNI introducido.

6. Diseñe un programa que solicite una frase por teclado, sustituya los espacios en blanco por asteriscos '*' y a continuación la muestre por pantalla. Para leer una cadena de caracteres por teclado incluido los espacios y tabulaciones se puede utilizar:
   - La función **gets** definida en la librería **cstdio** de la siguiente forma:

     ```
     char cadena[150];
     gets(cadena); //No tiene en cuenta el límite de la cadena
     ```

   - El método **getline** de **cin** definido en la librería **iostream** de la siguiente forma:

     ```
     char cadena[150];
     cin.getline(cadena,150); //150 es el tamaño máximo de la cadena
     ```

7. Diseñe un programa que solicite una frase por teclado, elimine los espacios en blanco (compactar la frase) y la muestre por pantalla.

8. Diseñe un programa que solicite una frase por teclado, la invierta y la muestre por pantalla. Por ejemplo, si se introduce la frase "Hola caracola" debe mostrar por pantalla "alocarac aloH".

9. Diseñe un programa que solicite una frase por teclado sin signos de puntuación ni acentos e indique si es un palíndromo o no. Un palíndromo es una frase que, atendiendo sólo a sus letras se lee igual tanto de izquierda a derecha como de derecha a izquierda, por ejemplo "**dábale arroz a la zorra el abad**".
   Nota: Se debe reutilizar el código de los ejercicios 6 al 8 y hacer uso de la función **`strcmp()`** de la clase `cstring`.

10. Diseñe un programa que solicite el nombre y los dos apellidos de una persona en tres cadenas diferentes, las una en una única cadena y la muestre por pantalla.

11. Diseñar un programa que declare dos vectores (VNombres y VApellidos) de 10 elementos de tipo cadena e implemente el siguiente menú:

```
                MENU

   1. Pedir Vector VNombres
   2. Pedir Vector VApellidos
   3. Mostrar Vectores
   4. Generar Nombres de Personas
   5. Salir

Elija opción:
```

Donde cada opción tiene la siguiente funcionalidad:

1) Rellena el vector VNombres con nombres propios de personas introducidos desde teclado.
2) Rellena el vector VApellidos con 2 apellidos de personas introducidos desde teclado.
3) Muestra por pantalla cada vector en una línea y sus valores separados por comas (,).
4) Solicitará por pantalla cuántos nombres de personas se tienen que generar de manera aleatoria. Para generar un nombre de persona, el programa deberá escoger un nombre del vector VNombres y dos apellidos del vector VApellidos de manera aleatoria.
5) Salir de programa.

12. Diseñe un programa que solicite por teclado el nombre y apellidos de una persona en dos cadenas distintas, una para el nombre y otra para los apellidos. A continuación deberá crear otra cadena con el nombre completo de la persona pero con el formato "`Apellidos, Iniciales del Nombre`" y se mostrará por pantalla.

    Ejemplo: si introducimos "María José" y "Pérez Díaz" se mostrará por pantalla "Pérez Díaz, M.J."

13. Diseñe un programa que declare dos vectores de 10 elementos de tipo entero e implemente el siguiente menú:

```
MENU

   1. Pedir Vector 1
   2. Pedir Vector 2
   3. Mostrar Vectores
   4. Comprobar Vectores
   5. Salir

Elija opción:
```

    Donde cada opción tiene la siguiente funcionalidad:
    1) Rellena el vector 1 con valores introducidos desde teclado.
    2) Rellena el vector 2 con valores introducidos desde teclado.
    3) Muestra por pantalla cada vector en una línea y sus valores separados por comas (,).
    4) Comprueba si ambos vectores contienen los mismo valores o no mostrando un texto por pantalla indicando el resultado de la comprobación.
    5) Salir de programa.

14. Diseñe un programa que declare un vector de 15 elementos reales y la rellene completamente con valores leídos desde teclado. A continuación pida una posición **P** del vector a partir del cual muestre el elemento más pequeño y su posición. Ejemplo:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2.1	**0.2**	**5.3**	**2.4**	**14.5**	**-10.5**	**3.7**	**0.8**	**4.9**	**81**	**92.3**	**2.6**	**0.15**	**5.1**	**0.4**

    Si P es 2 el valor más pequeño desde la posición 2 a la 14 es -10.5 y su posición es la 5
    Si P es 6 el valor más pequeño desde la posición 6 a la 14 es 0.15 y su posición es la 12

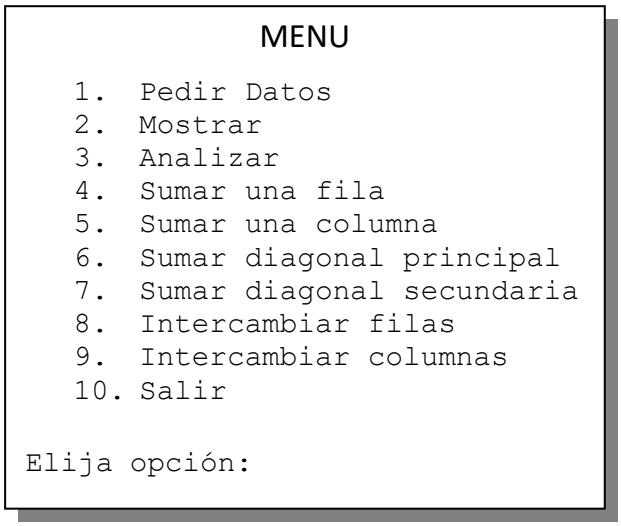
15. Modifique el programa del ejercicio anterior, para que dado un vector de 15 elementos lo ordene de menor a mayor. **Nota**: repita el bucle anterior dando a P valores de 0 a 13 y cada vez que encuentre el valor más pequeño, intercambie la posición de dicho valor con la posición que indique P.

16. Diseñe un programa que declare un vector de 1000 caracteres (tipo char) y a continuación solicite el número de elementos del vector (**N**) a ser rellenados, y rellenará las primeras **N** posiciones del vector con vocales generadas aleatoriamente del siguiente modo:

    1) Se genera un número aleatorio **X** en el intervalo [0, 20].
    2) Si X está comprendido en el intervalo **[0,4]** se pone la vocal '**a**' en el vector.
    3) Si X está comprendido en el intervalo **[5,7]** se pone la vocal '**e**' en el vector.
    4) Si X está comprendido en el intervalo **[8,12]** se pone la vocal '**i**' en el vector.
    5) Si X está comprendido en el intervalo **[13,16]** se pone la vocal '**o**' en el vector.
    6) Si X está comprendido en el intervalo **[17,20]** se pone la vocal '**u**' en el vector.

    Finalmente el programa mostrará por pantalla la frecuencia y el porcentaje de aparición de cada vocal. Ejemplo de mensaje: La vocal 'a' ha aparecido 234 veces (34.4%)

17. Diseñe un programa que declare una tabla de 10 filas por 10 columnas de números enteros e implemente el siguiente menú:



    Donde cada opción tiene la siguiente funcionalidad:

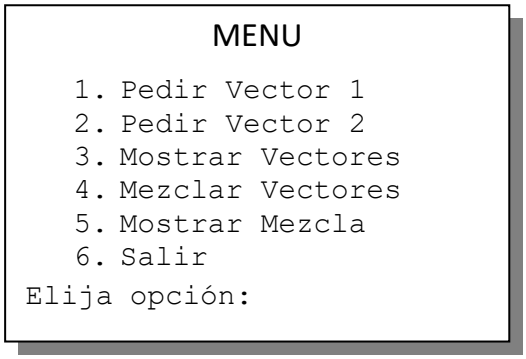
    1) Rellena la tabla con valores positivos introducidos desde teclado. Si el valor introducido es negativo se deberá solicitar nuevamente hasta que sea positivo.
    2) Muestra el contenido de la tabla de modo tabular, es decir, en 10 filas y 10 columnas.
    3) Muestra cuántos valores son pares y cuantos son impares.
    4) Muestra la suma de los valores de la fila cuyo índice ha sido solicitado previamente y que debe estar entre el rango [0,9].
    5) Muestra la suma de los valores de la columna cuyo índice ha sido solicitado previamente y que debe estar entre el rango [0,9].
    6) Muestra la suma de los valores de la diagonal principal de la tabla.
    7) Muestra la suma de los valores de la diagonal secundaria de la tabla.

8) Intercambia los valores de dos filas de la tabla cuyos índices han sido solicitados previamente. Estos índices deben estar en el rango [0,9] y no deben coincidir.
9) Intercambia los valores de dos columnas de la tabla cuyos índices han sido solicitados previamente. Estos índices deben estar en el rango [0,9] y no deben coincidir.
10) Salir de programa.

18. Diseñe un programa para mezclar vectores de valores enteros. El programa definirá las siguientes variables locales:

```
int Vuno[15], Vdos[15], Vfus[30];
int numuno, numdos; /* Nº de elementos almacenados en los vectores
                       Vuno y Vdos respectivamente */
```

Además se implementará el siguiente menú:



Donde cada opción tiene la siguiente funcionalidad:

1) Pedirá el número de elementos a rellenar del vector **`Vuno`** y lo almacena en la variable **`numuno`**. Posteriormente solicitará por teclado tantos elementos del vector como indique **`numuno`**. Los valores del vector deberán ser introducidos de manera ascendente.
2) Pedirá el número de elementos a rellenar del vector **`Vdos`** y lo almacena en la variable **`numdos`**. Posteriormente solicitará por teclado tantos elementos del vector como indique **`numdos`**. Los valores del vector deberán ser introducidos de manera ascendente.
3) Mostrará por pantalla el contenido de cada vector en una línea y sus valores separados por comas (,).
4) Copiará en el vector **`Vfus`**, el contenido de los vectores **`Vuno`** y **`Vdos`** de manera ordenada y creciente.
5) Mostrará por pantalla el contenido del vector **`Vfus`** en una línea y sus valores separados por comas (,).
6) Salir de programa.

19. Diseñe un programa que permita jugar al juego de **adivinar una palabra**. Para ello:

- Se definirán las siguientes variables locales:

```
char palabraSecreta[50]; //Almacena la palabra a adivinar.
int puntos;   //Almacena los puntos conseguidos por el jugador.
```

- Se comenzará estableciendo los parámetros iniciales del Juego, del siguiente modo:
  a) Se le pedirá al usuario que introduzca la palabra a adivinar, teniendo en cuenta que dicha palabra no se podrá ver por pantalla y que en su lugar se mostrará una cadena de asteriscos. Además la cadena leída deberá almacenarse siempre en mayúsculas. Por todo anterior, al alumno le serán de utilidad las siguientes funciones:

- **`getch()`** (librería **`conio.h`**), que lee un carácter desde teclado sin producir eco en pantalla.
- **`strupr()`** (librería **`cstring`**), que convierte una cadena de texto a mayúsculas.

b) Se establecerá a 9 la cantidad de puntos de partida del usuario.

- Posteriormente comenzará el juego, mostrándole al usuario una palabra con tantos guiones '-' como letras tenga la palabra oculta y le pedirá una letra. Si la letra se encuentra en la palabra oculta se mostrará en su/s posición/es correspondiente/s junto con el resto de guiones y letras ya descubiertas. Si por el contrario la letra no está, se le quitará un punto.

  El juego finalizará cuando el jugador adivine la palabra, en cuyo caso se le dará la enhorabuena y se le mostrarán los puntos conseguidos, o cuando se quede con 0 puntos, momento en el cual se le informará de tal situación y se le mostrará cuál era la palabra oculta.

  Nota: La letra solicitada al usuario deberá ser transformada a mayúsculas. Para ello el alumno podrá utilizar la función **`toupper()`** que se encuentra en la librería **`cctype.`**

20. Diseñe un programa para analizar la frecuencia de entrada de números enteros comprendidos entre 0 y 24 inclusive. Para ello el programa definirá las siguientes variables locales:

```
#define TAMA 25
int Datos[1000]; //Vector donde se almacenan los valores a analizar
int NDatos;      //Nº de valores almacenados en Datos
int Valores[TAMA];  //Vector que contiene la frecuencia de aparición
```

El vector **`Valores`** contabiliza el número de veces que aparece cada valor del intervalo [0,24] en el vector **`Datos`**. Por ejemplo, si el contenido del vector **`Valores`** fuese el que se muestra a continuación:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	…	24
2	0	5	2	14	2	0	0	4	8	9	2	1	5	0	0	2	1	12

significaría que en el vector **Datos**, el **0** aparece **2** veces, el **1** no ha parecido (**0** veces), el **2** aparece **5** veces, el **3** aparece **2** veces, el **4** aparece **14** veces, etc.

Se deberá implementar el siguiente menú:

```
MENU

   1. Pedir datos
   2. Analizar datos
   3. ¿Están todos?
   4. Valor repetido
   5. Valor más repetido
   6. Mostrar datos
   7. Mostrar Análisis
   8. Salir

Elija opción:
```

Donde cada opción tiene la siguiente funcionalidad:

1) Solicitará al usuario si desea introducirlos manualmente (**m**) o generarlos de forma aleatoria (**a**).
   - Si el usuario inserta una **m**, se le pedirán los valores y se guardarán dentro del vector **`Datos`** hasta que se introduzca el nº **-1**. El valor de **`NDatos`** deberá ser actualizado consecuente con el número de datos introducidos.
     Nota: Si el usuario introduce un valor fuera del rango [**0,24**] que no sea -1 (indica que la entrada de datos manual ha terminado), será rechazado y se volverá a solicitar otro.
   - Si el usuario introduce la **a**, se generará un valor aleatorio para **`NDatos`** en el intervalo [**1,1000**] y posteriormente se generarán y añadirán al vector **`Datos`** tantos valores aleatorios en el intervalo **[0,24]** como indique **`NDatos`**.
2) Actualiza el vector **`Valores`** mediante el análisis del vector **`Datos`**. El análisis consiste en contabilizar las veces que se repite cada número en **`Datos`** y guardar dicha cantidad en **`Valores`**. Por ejemplo, si el nº **5** se repite **20** veces en **`Datos`**, en la posición con índice **5** de **`Valores`** se guardará el nº **20**.
   En resumen, cada **posición** del vector **`Valores`** indicará cuantas veces se repite el valor **posición** en el vector **`Datos`**.
3) Mostrará un mensaje afirmativo por pantalla si todos los valores del intervalo [0,24] aparecen al menos una vez en el vector **`Datos`**, en caso contrario mostrará un mensaje indicando que hay valores del intervalo **[0,24]** que no están.
4) Solicita por teclado un valor del intervalo **[0,24]** y mostrará por pantalla un mensaje indicando cuántas veces está repetido. Si el usuario introduce un valor fuera del rango deberá volver a preguntar.
5) Muestra por pantalla qué valor del intervalo **[0,24]** es el más repetido.
6) Muestra por pantalla todos los valores del vector **`Datos`**.
7) Muestra por pantalla todos los valores del análisis (vector **`Valores`)**.