Informática Industrial Curso 2020-2021

Practica 1

`ejercicio01`

Realizar una función que reciba dos parámetros:

1) por referencia constante un vector de enteros.
2) un entero (`pivote`).

y devuelva un entero indicando cuantos de los elementos del vector son mayores que el `pivote`.

En el programa principal se leerán los datos de un fichero de texto que contendrá un entero por línea. El nombre de este fichero se pasará como primer argumento al invocar el programa. El segundo argumento será el valor del pivote, si no está disponible, el pivote valdrá 11.

Tras invocar a la función, se mostrará, por salida estándar en dos líneas:

1. '`Pivote: `' seguido del valor considerado para el pivote.
2. '`Resultado: `' seguido del valor devuelto por la función.

En caso de producirse algún error, el programa deberá devolver un código de error distinto de 0.

Ficheros de datos: `datosEnteros1.dat datosInt2.dat datosInteg3.dat`

`ejercicio02`

Realizar una función que reciba dos parámetros:

1) por referencia constante un vector de enteros.
2) un entero (`pivote`).

y devuelva un vector de enteros con los elementos son mayores que el `pivote`.

En el programa principal se leerán los datos de un fichero de texto que contendrá un entero por línea. El nombre de este fichero se pasará como primer argumento al invocar el programa. El segundo argumento será el valor del pivote, si no está disponible, el pivote valdrá 2.

Tras invocar a la función, se mostrará, por salida estándar:

1. '`Pivote: `' seguido del valor considerado para el pivote.
2. '`Resultado: `' seguido de los elementos devueltos por la función (en una sola línea separados por una coma y un espacio. Puede haber una coma al final de la línea).

En caso de producirse algún error, el programa deberá devolver un código de error distinto de 0.

Ficheros de datos: `datosEnteros1.dat datosInt2.dat datosInteg3.dat`

`ejercicio03`

Realizar una función que reciba dos parámetros:

1) por referencia constante un vector de enteros.
2) un entero (`pivote`).

y devuelva un vector de enteros con los elementos que, estando en posiciones pares del vector, sean mayores que el `pivote`.

En el programa principal se leerán los datos de un fichero de texto que contendrá un entero por línea. El nombre de este fichero se pasará como primer argumento al invocar el programa. El segundo argumento será el valor del pivote, si no está disponible, el pivote valdrá 5.

Tras invocar a la función, se mostrará, por salida estándar:

1. '`Pivote: `' seguido del valor considerado para el pivote.
2. '`Resultado: `' seguido de los elementos devueltos por la función (en una sola línea separados por una coma y un espacio. Puede haber una coma al final de la línea).

En caso de producirse algún error, el programa deberá devolver un código de error distinto de 0.

Ficheros de datos: `datosEnteros1.dat datosInt2.dat datosInteg3.dat`

ejercicio04

Realizar una función que reciba:

1) por referencia constante un vector de enteros.
2) por referencia un vector de enteros (`mayores`).
3) por referencia un vector de enteros (`menores`).
4) un entero (`pivote`).

y devuelva en `mayores` los elementos que son mayores que `pivote`, en `menores` los que son menores; y como parámetro de salida el número de los que son iguales.

En el programa principal se leerán los datos de un fichero de texto que contendrá un entero por línea. El nombre de este fichero se pasará como primer argumento al invocar el programa. El segundo argumento será el valor del pivote, si no está disponible, el pivote valdrá 10.

Tras invocar a la función, se mostrará, por salida estándar:

1. '`Pivote: `' seguido del valor considerado para el pivote.
2. '`Mayores: `' seguido (en la siguiente línea) de los elementos mayores (uno por línea)
3. '`Menores: `' seguido (en la siguiente línea) de los elementos menores (uno por línea)
4. '`Iguales: `' seguido del número de iguales.

En caso de producirse algún error, el programa deberá devolver un código de error distinto de 0.

Ficheros de datos: `datosEnteros1.dat datosInt2.dat datosInteg3.dat`

ejercicio05

Ídem que el anterior, pero si el nombre del fichero que pasa el usuario no termina con '`.dat`' se la añadirá antes de acceder al fichero.

ejercicio06

Ídem que el anterior, pero los vectores de elementos mayores y menores no se volcaran por salida estándar sino que se escribirán en sendos ficheros que tengan el mismo nombre que el entrada (sin el `.dat`) pero terminado en '`_mayores.dat`' y '`_menores.dat`' respectivamente.

ejercicio07

Ídem que el anterior, pero los nombres de los ficheros creados serán '_mayores_<*valor pivote*>.dat' y '_menores_<*valor pivote*>.dat' respectivamente, donde '<*valor pivote*>' será la representación decimal del valor del pivote.

ejercicio08

Realizar una función que, dado un parámetro entero, devuelva el número de bits a 1 que tiene su representación binaria.

En el programa principal leerá enteros desde entrada estándar, mostrando el resultado de la función seguida de la representación hexadecimal del número (precedida por `' 0x'`). El programa terminará cuando el usuario introduzca un 0.

En caso de producirse algún error, el programa deberá devolver un código de error distinto de 0.

ejercicio09

Realizar una función a la que se le pasa un entero sin signo y devuelve un `string` con la representación binaria del número (sin ceros a las izquierda).

En el programa principal leerá enteros sin signo desde entrada estándar, mostrando el resultado de la función seguida de la representación hexadecimal del número (precedida por `' 0x'`). El programa terminará cuando el usuario introduzca un 0.

En caso de producirse algún error, el programa deberá devolver un código de error distinto de 0.

ejercicio10

Realizar una función que reciba:

1) por referencia constante un vector de `bool`.
2) un entero sin signo como índice `inicial`.
3) un entero sin signo como índice `final`.

y devuelva un entero sin signo que tenga sus bits a 1 ó 0 dependiendo de si el valor booleano correspondiente es verdad o falso. El valor del bit 0 corresponderá al elemento `inicial`, el bit 1 al `inicial+1` y así sucesivamente hasta el final.

Al programa principal se le pasarán como argumentos en línea de comandos: el nombre del fichero de datos (añadir `.dat` si no tiene esa extensión), el índice `inicial` y el índice `final`. Si no se indican los índices se utilizarán los límites del vector.

El fichero de datos tendrá en cada línea alguna de las siguientes palabras “verdad”, “si”, “true”, “1” (que deberán considerarse como `true`), “falso”, “no”, “false”, “0” (que deberán considerarse como `false`).

Tras invocar a la función, se mostrará, por salida estándar: el entero devuelto en decimal, el entero devuelto en hexadecimal (precedido por `' 0x'`) y la representación binaria del entero generado usando la función del `ejercicio09` (precedida por `' 0b'`).

En caso de producirse algún error, el programa deberá devolver un código de error distinto de 0.

Ficheros de datos: `datosBooleanos2.dat datosBool1.dat`

`ejercicio11`

Función que dado un entero rellene un vector de booleanos a verdad o falso dependiendo del valor de los bits contenidos en el mismo. El bit más significativo, distinto de 0, deberá quedar en la posición 0 del vector.

En el programa principal leerá enteros desde entrada estándar, mostrando el vector devuelto por la función (separando los elementos por coma y espacio), utilizando “V” para los que son verdad y “F” para los que son falsos.

El programa terminará cuando se introduzca un 0.

En caso de producirse algún error, el programa deberá devolver un código de error distinto de 0.

`ejercicio12 (opcional)`

Realizar una función que reciba por referencia un vector de `double` y lo ordene por el método de la burbuja.

En el programa principal se leerán los datos de un fichero cuyo nombre se pasará como primer parámetro al programa. Si el nombre del fichero no termina con '`.dat`' se la añadirá antes de acceder al mismo.

Se mostrará por salida estándar y por líneas los datos ordenados en punto fijo con cuatro decimales y alineados al punto decimal.

En caso de producirse algún error, el programa deberá devolver un código de error distinto de 0.

Ficheros de datos: `datosDouble1.dat datosDoblePrec2.dat`

`ejercicio13 (opcional)`

Ídem que el anterior pero utilizando el algoritmo de selección directa para la ordenación.

Ficheros de datos: `datosDouble1.dat datosDoblePrec2.dat`

Guía de estilo

Al escribir el código fuente se debe de respetar las siguientes reglas:

- Indicar autor y fecha en las primeras líneas del fichero fuente.
- Usar dos espacios para cada nivel de sangrado, utilizando caracteres espacio.
- En cada línea no deberá haber más de una instrucción; aunque una instrucción puede ocupar varias líneas, en ese caso las líneas de continuación deberán tener dos niveles de sagrado más que la inicial (4 espacios más).
- No utilizar '`using namespace std`' sino especificar el espacio de nombre cuando sea necesario (`std::cout`, `std::endl`, `std::string`, `std::vector`, etc.)
- Dejar un espacio antes y después de cada operador, en especial de los de asignación (=, +=, -=, …), comparación (>, <, ==, …) y los de envío y recepción de `stream` (<< y >>).
- Las líneas deberán tener, preferiblemente, menos de 72 caracteres y nunca superar los 80.
- Utilizar mensajes de depuración allí donde sea conveniente. Se utilizará la salida de error (`std::cerr`) para mostrar dichos mensajes.
- Declarar las variables justo antes de ser utilizadas y, si es necesario, hacer la declaración y la inicialización en la misma instrucción.