

Caso #1, 5%

Topic: #gettingintoC

Due date: 08-24-2021, 9:00pm

Send to: epinedatec@gmail.com

Subject: Estructuras - Caso #1

Groups of: 1

Description

Proceda a programar en C++ los siguientes ejercicios:

1. Dado un arreglo de strings retornar subgrupos de strings que coinciden entre ellas en dos palabras cualesquiera sin repetición y sacando todas las combinaciones posibles, indicando las dos palabras de match

ejemplo

Input: ['estoy escribiendo la tarea', 'mañana voy en carro', 'estoy en el carro escribiendo la poesía', 'voy mañana a ver la tarea']

Output:

estoy escribiendo la tarea, voy mañana a ver la tarea -> tarea la
mañana voy en carro, estoy en el carro escribiendo la poseía -> carro en
estoy escribiendo la tarea, estoy en el carro escribiendo la poseía -> estoy la
estoy escribiendo la tarea, estoy en el carro escribiendo la poseía -> escribiendo estoy
estoy escribiendo la tarea, estoy en el carro escribiendo la poseía -> escribiendo la

2. Diseñe por medio de un struct la definición de un spreadsheet (hoja de calculo), debe poder definir valores enteros ubicados en diferentes celdas, también celdas que contienen operaciones de SUM y dichas operaciones indican cuales son las celdas que suma. Cree un programa que evalúe el struct y retorne otro struct que contenga las sumas resueltas.

	A	B	C	D	E	F	G
1							
2							
3						10	
4			23			4	
5			21			1	
6			66			2	
7			2			3	
8							
9		112					
10							
11							
12					45	200	
13					67		
14					88		
15							
16							

Caso #2, 5%**Topic:** #sorting**Due date:** 09-16-2021, 9:00pm**Send to:** epinedatec@gmail.com**Subject:** Estructuras - Caso #2**Groups of:** 1**Description**

Proceda a codear en C++ las siguientes funciones:

1. `void sortBySelectionSort(float[] *pValoresAOrdenar);` el cual debe recibir un puntero a un arreglo de números enteros y proceder a ordenar los números usando el algoritmo de selectionsort.
2. Implemente una lista enlazada simple que permita almacenar en memoria jugadores. Los jugadores nodo de la lista deben implementarse con un struct y deben incluir un número único y un nombre. La lista simple debe implementar las siguientes funciones:
 - `isEmpty()`, retorna true en caso de estar vacía
 - `getQuantity()`, retorna la cantidad de elementos existentes en la lista
 - `addPlayer(int pNumber, string pNombre)`, agrega un nuevo jugador al final de la lista, retorna true si logró agregarlo sin errores
 - `removePlayer(int pNumber)`, busca un jugador con ese número y lo remueve de la lista, retorna true si efectivamente lo encontró y lo removió
 - `insertPlayer(int pNumber, string pNombre, int pPosition)`, inserta un nuevo jugador en una posición específica de la lista, siendo el primer elemento la posición 0. Si la posición no se existe se agrega al final.
 - `listPlayers()`, procede a imprimir la lista de jugadores
3. `void sortByInsertionSort(ListaDinamica *pListaJugadores);` la función debe recibir un puntero a una lista de jugadores creada en el ejercicio 2. El algoritmo procede a ordenar la lista con el algoritmo de insertionsort usando el nombre del jugador como llave del ordenamiento. [string::compare - C++ Reference \(cplusplus.com\)](#)

El caso debe cumplir:

- respetar las prácticas estudiadas en clean code
- la lista, nodos y los dos algoritmos de ordenamiento deben quedar en archivos .h separados
- el idioma dentro del código fuente debe ser inglés
- cree un programa que tengan un main que permita hacer las pruebas de los 3 ejercicios: crear un arreglo de varios elementos y que se demuestre que efectivamente lo ordena, que se pruebe que funcionan bien las funciones de la lista y lo mismo que dicha lista quede efectivamente bien ordenada
- cualquier sospecha de copia anulará el trabajo, tenga en cuenta que ninguno de los ejercicios amerita hacer uso de código o librerías copiadas de internet
- la tarea deberá entregarla en un correo como un link de github, cualquier explicación adicional la agregan al readme.md y el último commit no debe superar la fecha y hora de entrega