Práctica 3 Fundamentos da Programación II: Listas Posicionales Ordenadas

O obxectivo desta práctica era manter, de maneira virtual, unha carta de postres, unha despensa e unha lista de pedidos dunha repostería, a través de distintos tipos de datos.

Antes de comezar coa descripción do traballo realizado, cómpre destacar que alteramos o código fonte das listas posicionais ordenadas, tanto das enlazadas como das que non, para engadir a función 'search_element(pl, e)' tal e como está definida nas diapositivas, de forma que nos permita obter a posición (pl) do elemento (e).

Para a implementación do código, utilizamos 3 funcións que se encargan cada unha dunha das partes principais: despensa, carta e pedidos; e unha última para visualizar a Carta de postres unha vez rematada a execución.

A primeira desas funcións, read_pantry(), é a encargada da despensa electrónica. Crea unha lista posicional ordenada chamada Despensa, á que se engade unha lista por ingrediente, que contén o nome do mesmo e a cantidade da que dispoñemos. A Lista Posicional Ordenada ordea os ingredientes por orden alfabética.

A continuación, definimos a función read_desserts(), que se vai a encargar de crear a carta de postres. Recorremos o arquivo liña a liña, obtendo o nome do postre, un dos ingredientes e a cantidade do mesmo. Imos comprobando se o nome do postre xa está na lista nomes, que creamos para poder diferenciar os novos ingredientes dun postre que xa limos ou se trata dun novo postre. Se o postre é novo, reiniciamos a lista posicional Ingredientes, engadimos o nome á lista de comprobación 'nomes', e continuamos como cos outros postres, engadindo á Lista Posicional Ordenada unha lista co nome do ingrediente e a súa cantidade. Finalmente, engádese ao diccionario Carta a lista de ingredientes coas suas cantidades baixo a chave do nome de postre.

Para rematar coas funcións principais, temos a función read_pedidos(), que vai xestionar todo o relativo cos pedidos: aceptalos, rexeitalos, eliminalos da carta, eliminar produtos…
En primeiro lugar, comprobamos que a comanda esté dentro da carta. Se o pedido inicial é correcto, engadímolo á lista posicional ordenada de pedidos, e creamos dúas listas inicialmente baleiras, 'falta' e 'usados', onde engadiremos os produtos que faltan nunha elaboración e os usados, respectivamente.
A continuación, comezamos con dous bucles anidados que recorren a lista de ingredientes e a lista posicional ordenada Despensa, comprobando se existe o ingrediente na despensa. Se efectivamente o ingrediente existe na Despensa, comprobamos se hai a cantidade suficiente de dito produto. En caso positivo, engadimos dito ingrediente á lista de 'usados', en caso contrario engadímolo á lista de 'faltan'.
Despois, comprobamos o tamaño da lista 'faltan', para saber se o postre pudo realizarse ou non. Se a lista ten un elemento ou máis, rexeitamos o pedido e mostramos os produtos que faltan e en que cantidades, que serán os almaceados na lista 'faltan', e eliminamos o postre.
Se a lista está baleira, quere dicir que non falta ningún produto, polo que aceptamos o postre e comprobamos se esta elaboración esgota algún produto, é dicir, tras actualizar a cantidade de produtos dispoñíbeis tras a elaboración, algún produto ten unha nova cantidade dispoñíbe de exactamente 0. Eliminamos os produtos que se esgoten e todos os postres que conteñan ese produto.

Finalmente, a función visualizar_carta() recorre as chaves da Carta, é dicir, os nomes dos postres, e os imprime xunto cos ingredientes necesarios para cociñalo.

En canto ao análise de resultados, o código funciona ben e sen erros con ambas implementacións da lista (array_ordered_positional_list e linked_ordered_positional_list), e ao non ter interacción co usuario e traballar sempre con datos fixos (os arquivos dados) non se producen erros de ningún tipo.