FACULTADE DE INFORMÁTICA
Programación II – Curso 2018/19

Práctica 2: Enunciado

1. El problema

En esta segunda práctica se amplían las funcionalidades de la primera, pasando ahora a gestionarse simultáneamente los resultados de las votaciones de varios centros (o colegios) electorales. El sistema se encargará de llevar un recuento de votos actualizado de todos los centros, pudiendo mostrar además estadísticas electorales de los mismos. Para conseguirlo, se usarán distintas estructuras de datos.

El objetivo de esta práctica es comprender el funcionamiento e implementar varios tipos abstractos de datos (TADs) así como manejar interdependencias entre ellos.

2. Estructuras de datos

Para resolver este problema se utilizarán 4 tipos abstractos de datos (TADs):

- Una Cola (TAD RequestQueue) para gestionar las peticiones recibidas del fichero.
- Una Lista Ordenada (TAD PartyList) para almacenar las votaciones a los partidos correspondientes a un único centro electoral.
- Una Lista Ordenada (TAD CenterList) para almacenar los distintos centros electorales.
- Un TAD Manager que se encarga de gestionar conjuntamente las dos estructuras anteriores.

2.1. Unit SharedTypes

Algunos tipos de datos comunes se definirán en esta unit (SharedTypes.pas), ya que son utilizados simultáneamente en varios TADs.

`tPartyName`	Nombre (siglas) del partido político (`string`). Hay dos constantes de este tipo utilizadas para representar dos "partidos" que siempre se incluirán en la lista para contabilizar: • Votos en blanco. Constante `BLANKVOTE` con valor `'B'` • Votos nulos. Constante `NULLVOTE` con valor `'N'`
`tCenterName`	Nombre de un centro electoral (`string`).
`tNumVotes`	Número de votos (`integer`).

2.2. TAD RequestQueue

Emplearemos un TAD Cola o **RequestQueue** (unit `RequestQueue.pas`) para ir almacenando, por orden de llegada, las operaciones a procesar. La implementación de dicho TAD deberá ser **estática**.

2.2.1. Tipos de datos incluidos en el TAD RequestQueue

`tQueue`	Representa la cola de operaciones.
`tRequest`	Tipo de petición recibida (`char`).
`tItemQ`	Datos de un elemento de la cola (una operación): • `request` de tipo `tRequest`. • `code` de tipo `string`. • `param1` de tipo `string`. • `param2` de tipo `string`.
`tPosQ`	Posición de un elemento en la cola de peticiones
`NULLQ`	Constante utilizada para representar posiciones nulas en la cola

2.2.2. Operaciones incluidas en el TAD RequestQueue

A mayores de las especificaciones abajo indicadas, una precondición común para todas las operaciones (salvo `createEmptyQueue`) es que la cola debe estar previamente inicializada:

- `createEmptyQueue (tQueue) → tQueue`
  Crea una cola vacía.
  PostCD: La cola queda inicializada y vacía.

- `isEmptyQueue (tQueue) → Boolean`
  Determina si la cola está vacía.

- `enqueue (tQueue, tItemQ) → tQueue, Boolean`
  Inserta un nuevo elemento (`tItemQ`) en la cola. Devuelve `verdadero` si se ha podido insertar, `false` en caso contrario.

- `front (tQueue) → tItemQ`
  Devuelve el contenido (`tItemQ`) del frente de la cola (i.e. el elemento más antiguo).
  PreCD: la cola no está vacía.

- `dequeue (tQueue) → tQueue`
  Elimina el elemento que está al frente de la cola.
  PreCD: la cola no está vacía.

2.3. TAD PartyList

Se realizará una implementación **dinámica y simplemente enlazada** del TAD Lista **Ordenada** (unit `PartyList.pas`) para almacenar una lista de partidos.

2.3.1. Operaciones incluidas en el TAD PartyList

Las especificaciones del TAD PartyList son idénticas a las del TAD Lista de la Práctica 1, únicamente cambia la de la siguiente operación:

- `insertItem (tItem, tList) → tList, Boolean`
  Inserta un elemento de forma **ordenada** en función del campo `partyname`. Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario. **PostCD: Las posiciones de los elementos de la lista posteriores al insertado pueden cambiar de valor.**

Asimismo, la operación `findItem` no cambia su especificación pero su implementación debe tener en cuenta que ahora la lista está ordenada.

2.4. TAD CenterList

Se realizará una implementación **estática** del TAD Lista **Ordenada** (unit `CenterList.pas`) para mantener la lista de centros electorales.

2.4.1. Tipos de datos incluidos en el TAD CenterList

`tListC`	Representa una lista de centros electorales
`tItemC`	Datos de un elemento de la lista (un centro): • `centername:` de tipo `tCenterName` • `totalvoters:` de tipo `tNumVotes` • `validvotes:` de tipo `tNumVotes` • `partylist:` de tipo `tList`
`tPosC`	Posición de un elemento en la lista de centros
`NULLC`	Constante utilizada para representar posiciones nulas en la lista de centros

2.4.2. Operaciones incluidas en el TAD CenterList

Las operaciones son las mismas que las del TAD Lista de la Práctica 1 renombrando los tipos de datos y las operaciones para evitar conflictos con el TAD PartyList. Todos los identificadores del TAD CenterList pasarán a terminar en C. Cambia la especificación de las siguientes operaciones:

- `insertItemC (tItemC, tListC) → tListC, Boolean`
  Inserta un elemento de forma **ordenada** en función del campo `centername`. Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario. **PostCD: Las posiciones de los elementos de la lista posteriores al insertado pueden cambiar de valor.**

- `updateListC (tList, tPosC, tListC) → tListC`
  Modifica la lista de partidos del elemento situado en la posición indicada.
  PreCD: La posición indicada es una posición válida en la lista.

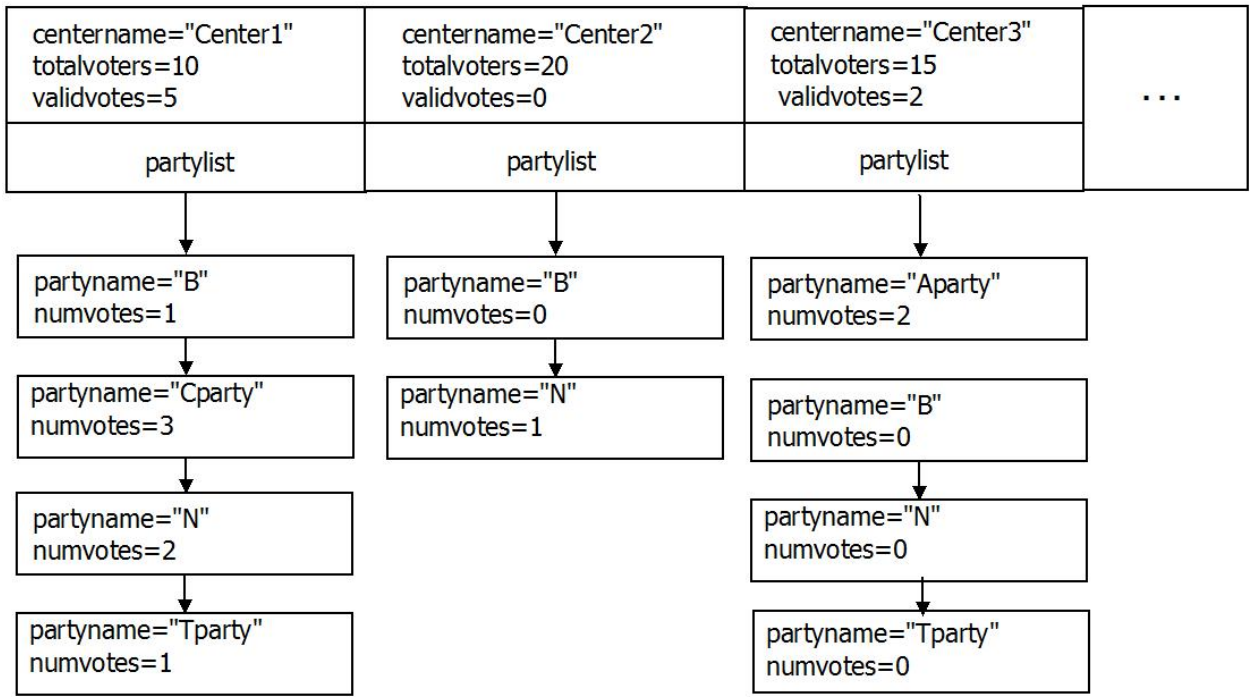
- `updateValidVotesC (tNumVotes, tPosC, tListC) → tListC`
  Modifica el número de votos válidos del elemento situado en la posición indicada.
  PreCD: La posición indicada es una posición válida en la lista.

- `findItemC (tCenterName, tListC) → tPosC`
  Devuelve la posición **del primer elemento de la lista** cuyo nombre de centro se corresponda con el indicado (o `NULLC` si no existe tal elemento).

2.5. TAD Manager

Este TAD (unit `Manager.pas`) implementará una multilista basándose en el TAD CenterList y donde cada centro tiene una lista de partidos (TAD PartyList).

tManager



2.5.1. Tipos de datos incluidos en el TAD Manager

`tManager`	Representa la multilista

2.5.2. Operaciones incluidas en el TAD Manager

Una precondición común para todas estas operaciones (salvo para `createEmptyManager`) es que la multilista debe estar previamente inicializada:

- `createEmptyManager (tManager) → tManager`
  Crea una multilista vacía.
  PostCD: La multilista queda inicializada y no contiene elementos.

- `insertCenter (tCenterName, tNumVotes, tManager) → tManager, Boolean`
  Inserta un elemento **por orden de centername** inicializando `totalvoters` con el valor indicado y `validvotes` a 0. Se inicializa su lista de partidos (`partylist`) con los partidos NULLVOTE y BLANKVOTE. Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario.
  PostCD: Las posiciones de los elementos de la lista posteriores al insertado pueden cambiar de valor.

- `insertPartyInCenter(tCenterName,tPartyName,tManager)→ tManager, Boolean`
  Incluye un nuevo partido en la lista de partidos (`partylist`) en el centro indicado (centername), inicializando su número de votos a 0. Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario.

- `deleteCenters (tManager) → tManager, integer`
  Elimina los centros electorales cuyo número de votos válidos es 0, así como su lista de partidos. Devuelve el número de centros que han sido eliminados.

- `deleteManager(tManager) → tManager`
  Elimina todos los elementos de la multilista.

- `showStats(tManager)`
  Muestra las estadísticas de votación y participación de todos los centros.

- `voteInCenter(tCenterName, tPartyName, tManager)→ tManager, boolean`
  Incrementa en uno el número de votos del partido `partyname` en el centro `centername`. Si el partido no existe en ese centro, se actualiza el número de votos nulos. Devuelve true si el voto es válido y false si es nulo.
  PreCD: el centro electoral es válido.

3. Descripción de la tarea

Implementar un programa principal (`main.pas`) que **use los TAD SharedTypes, RequestQueue y Manager** y que procese las peticiones recibidas con el siguiente formato:

`C centername totalvoters`	***[C]reate***: Se incorpora el centro electoral `centername` y cuyo número total de votantes es `totalvoters`.
`N centername partyname`	***[N]ew:*** Alta de un partido en el centro con 0 votos***.***
`V centername partyname`	***[V]ote:*** Se emite un voto para un partido en un centro electoral.
`R`	***[R]emove:*** Elimina los centros con 0 votos.
`S`	***[S]tats:*** Muestra estadísticas de participación y voto.

El programa leerá y procesará las operaciones contenidas en un fichero (ver documento `EjecucionScript.pdf`). Se insertarán primero todas las peticiones en la cola de peticiones. Después, un bucle ejecutará las operaciones. Para cada petición en cola, el programa:

1. **Muestra una cabecera con la operación a realizar.** Esta cabecera está formada por una primera línea con 20 asteriscos y una segunda línea que indica la operación

```
********************
CC␣T:␣center/party␣XX␣totalvoters/party␣YY
```

donde `CC` es el número de petición; `T` es el tipo de operación (`C`, `N`, `V`, `R` o `S`); `XX` es, o bien el nombre del centro (`centername`) o bien las siglas del partido (`partyname`). De forma similar, `YY` es o bien las siglas del partido (`partyname`) o bien el número de votantes (`totalvoters`).

2. **Procesa la petición** correspondiente:

- Si la operación es **`[C]reateCenter`**, se añade el centro a la multilista con el número de votantes igualado al parámetro indicado. Además, se mostrará:

  ```
  *␣Create:␣center␣XX␣totalvoters␣YY
  ```

  donde `XX` es el nombre del centro electoral (`centername`), `YY` es el número total de votantes censados en ese centro. Si ya existiese un centro electoral con el nombre indicado (`centername`) o hubiese algún problema al crear el centro, se mostrará:

  ```
  +␣Error:␣Create␣not␣possible
  ```

- Si la operación es **`[N]ew`**, se añadirá el partido al centro indicado y se mostrará:

  ```
  *␣New:␣center␣XX␣party␣YY
  ```

donde XX es el nombre del centro e YY es el del partido. Si ya existiese en el centro ese partido o hubiese algún problema con la inserción, se mostrará:

```
+ Error: New not possible
```

- Si la operación es **[V]ote**, se incrementará el número de votos de ese partido en el centro y se mostrará:

  ```
  * Vote: center XX party YY
  ```

  donde XX es el nombre del centro, YY el del partido. Si el partido en cuestión no existiese para el centro indicado, se mostrará:

  ```
  + Error: Vote not possible. Party YY not fount in center XX. NULLVOTE
  ```

- Si la operación es **[R]emove**, se eliminarán todos los centros con 0 votos válidos, mostrando el siguiente mensaje para cada uno de ellos:

  ```
  * Remove: center XX
  ```

  Si no existiese ningún centro a eliminar se mostrará:

  ```
  * Remove: no centers removed
  ```

- Si la operación es **[S]tats,** se mostrarán las estadísticas de voto de la siguiente forma:

  ```
  Center CC1
   Party XX1 numvotes YY1 (ZZ1%)
   …
   Party N numvotes YY
   …
   Party XXn numvotes YYn (ZZn%)
   Participation: KK votes from VV voters (PP%)

   Center CC2
          …
  ```

  donde CC es el nombre del centro. El resto de estadísticas se muestran de la misma forma que en la práctica 1.

3. Liberación de recursos. Al finalizar, el programa **deberá proceder a vaciar la lista de centros** (CenterList), liberando así los recursos de memoria reservados.

4. Lectura de ficheros

Se proporciona un código en Pascal (fichero `read.pas`) con un ejemplo de cómo leer los ficheros de prueba disponibles.

Se dejará a disposición de los estudiantes un conjunto de ficheros de prueba con el conjunto de operaciones mínimas exigidas para poder aprobar la práctica (ver documento `EjecucionScript.pdf`). Se podrán probar de manera conjunta con el script proporcionado en la carpeta `script` (`script.sh`). Finalmente, para que el `script` no dé problemas se recomienda **NO copiar directamente el texto de este documento**, ya que el formato PDF puede incluir caracteres invisibles que darían por incorrectas salidas válidas.

5. Información importante

El documento `NormasEntrega.pdf`, disponible en la página web de la asignatura, detalla claramente las normas de entrega.
Fecha límite de entrega: **Viernes 3 de Mayo de 2018 a las 23:55 horas**
Fechas de los **Controles de Seguimiento:**

- Semana del 8-12 de Abril: Implementación parcial, pero funcional, del programa principal `main.pas`, incluyendo las operaciones: Create y Stats.