UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

Programación II – Curso 2018/19

Práctica 1: Enunciado

1. El problema

El problema a resolver en esta Práctica 1 consiste en la implementación de un sistema de recuento para las máquinas de voto electrónico que se utilizarán en los colegios electorales durante las próximas elecciones municipales. El sistema llevará cuenta de los diferentes partidos políticos que han recibido votos en un determinado colegio, el número de votos que tiene cada uno, la cantidad de votos blancos y nulos y la participación; además se permitirá ilegalizar partidos.

Como el objetivo de este trabajo es practicar la independencia de la implementación en los Tipos Abstractos de Datos (TADs), se pide crear dos implementaciones de una LISTA NO ORDENADA, las cuales deberán funcionar de manera intercambiable: una implementación ESTÁTICA y otra DINÁMICA. De este modo el programa principal no deberá realizar ninguna suposición sobre la forma en que está implementado el TAD.

2. TAD Lista

Para mantener la lista de partidos y su información asociada, el sistema utilizará un TAD Lista. Se realizarán dos implementaciones:

1. ESTÁTICA con arrays (*unit* `StaticList.pas`) con tamaño máximo 25.
2. DINÁMICA, simplemente enlazada, con punteros (*unit* `DynamicList.pas`).

2.1. Tipos de datos incluidos en el TAD Lista

`tList`	Representa una lista de partidos
`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'`
`tNumVotes`	Número de votos recibidos (`integer`).
`tItem`	Datos de un elemento de la lista (un partido). Compuesto por: • `partyname`: de tipo `tPartyName` • `numvotes`: de tipo `tNumVotes`
`tPosL`	Posición de un elemento de la lista
`NULL`	Constante usada para indicar posiciones nulas

2.2. Operaciones incluidas en el TAD Lista

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

- `createEmptyList (tList) → tList`

Crea una lista vacía.
PostCD: La lista queda inicializada y no contiene elementos.

- `isEmptyList (tList) → Boolean`

Determina si la lista está vacía.

- `first (tList) → tPosL`

Devuelve la posición del primer elemento de la lista.
PreCD: La lista no está vacía.

- `last (tList) → tPosL`

Devuelve la posición del último elemento de la lista.
PreCD: La lista no está vacía.

- `next (tPosL, tList) → tPosL`

Devuelve la posición en la lista del elemento siguiente al de la posición indicada (o `NULL` si la posición no tiene siguiente).
PreCD: La posición indicada es una posición válida en la lista.

- `previous (tPosL, tList) → tPosL`

Devuelve la posición en la lista del elemento anterior al de la posición indicada (o `NULL` si la posición no tiene anterior).
PreCD: La posición indicada es una posición válida en la lista.

- `insertItem (tItem, tPosL, tList) → tList, Boolean`

Inserta un elemento en la lista antes de la posición indicada. Si la posición es `NULL`, entonces se añade al final. Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario.
PreCD: La posición indicada es una posición válida en la lista o bien nula (`NULL`).
PostCD: Las posiciones de los elementos de la lista posteriores al insertado pueden cambiar de valor.

- `deleteAtPosition (tPosL, tList) → tList`

Elimina de la lista el elemento que ocupa la posición indicada.
PreCD: La posición indicada es una posición válida en la lista.
PostCD: Tanto la posición del elemento eliminado como aquellas de los elementos de la lista a continuación del mismo pueden cambiar de valor.

- `getItem (tPosL, tList) → tItem`

Devuelve el contenido del elemento de la lista que ocupa la posición indicada.
PreCD: La posición indicada es una posición válida en la lista.

- `updateVotes (tNumVotes, tPosL, tList) → tList`

Modifica el número de votos del elemento situado en la posición indicada.
PreCD: La posición indicada es una posición válida en la lista.
PostCD: El orden de los elementos de la lista no se ve modificado.

- `findItem (tPartyName, tList) → tPosL`

Devuelve la posición **del primer elemento de la lista** cuyo nombre de partido se corresponda con el indicado (o `NULL` si no existe tal elemento).

3. Descripción de la tarea

La tarea consiste en implementar un único programa principal (`main.pas`) que procese las operaciones en un colegio electoral con el siguiente formato:

`N partyname`	***[N]ew***: Alta de un partido con su número de votos a 0
`V partyname`	***[V]ote***: Suma un voto al partido indicado. También se utiliza para sumar votos blancos o nulos cuando nos pasan `B (BLANKVOTE)` o `N (NULLVOTE)` como partido, respectivamente
`I partyname`	***[I]llegal:*** Ilegaliza un partido eliminándolo de la lista, y pasando a ser votos nulos todos sus votos
`S totalvoters`	***[S]tats:*** Estadisticas de participación y de votos a cada uno de los partidos. `Totalvoters` representa el número total de votantes censados en el colegio electoral.

En el programa principal se implementará un bucle que procese una a una las operaciones. Para simplificar tanto el desarrollo como las pruebas, el programa no necesitará recibir ningún dato por teclado, sino que leerá y procesará las operaciones contenidas en un fichero (ver documento `EjecucionScript.pdf`). En cada iteración del bucle, el programa leerá del fichero una nueva operación y la procesará. Asimismo, para facilitar la corrección de la práctica todas las operaciones del fichero van numeradas correlativamente.

Para cada línea del fichero de entrada, 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 tal y como se muestra a continuación:

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

donde `CC` es el número de petición, `T` es el tipo de operación (`N`, `V`, `I` o `S`), `XX` son o bien las siglas del partido (`partyname`) o bien el número de votantes del censo (`totalvoters`) (según corresponda) y ␣ indica un espacio en blanco. Nótese que sólo se imprimirán los

parámetros necesarios; por ejemplo, para una petición `[S]tats` se mostraría "`01 S: totalvoters 145`", mientras que para una petición `[N]ew` se mostraría "`02 N: party PBF`".

2. Procesa la petición correspondiente:

- Si la operación es **`[N]ew`**, se incorporará el partido **al final** de la lista de partidos con su número de votos inicializado a 0. Además, se imprimirá el mensaje:

  ```
  *␣New:␣party␣XX
  ```

  Donde `XX` es el `partyname` y `␣` representa un espacio en blanco. El resto de mensajes siguen el mismo formato.

  Si ya existiese un partido con ese *partyname*, se imprimirá el siguiente mensaje:

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

  Se establece que las dos primeras operaciones de los ficheros serán SIEMPRE '`01 N B`' y '`02 N N`' para garantizar que los partidos `BLANKVOTE` y `NULLVOTE` se incluyen en la lista. Además, no es posible incorporar partidos (petición `N`) una vez que la votación ha comenzado (petición `V`). Todos los ficheros de prueba proporcionados, así como los que se usarán en la corrección, cumplen estas dos premisas, por tanto no es necesario comprobarlo.

- Si la operación es **`[V]ote`**, se buscará el partido, se incrementará su contador de votos en 1 y se mostrará el siguiente mensaje:

  ```
  *␣Vote:␣party␣XX␣numvotes␣YY
  ```

  Si no existiese ningún partido con ese *partyname*, se imprimirá el mensaje:

  ```
  +␣Error:␣Vote␣not␣possible.␣Party␣XX␣not␣found.␣NULLVOTE
  ```

  y este voto se contabilizará como voto para el partido `NULLVOTE`.

- Si la operación es **`[I]llegalize`**, se buscará al partido en la lista, se borrará, añadiendo sus votos al partido `NULLVOTE` y se imprimirá el siguiente mensaje:

  ```
  *␣Illegalize:␣party␣XX
  ```

  Si no existiese el partido `XX`, se imprimirá el siguiente mensaje:

  ```
  +␣Error:␣Illegalize␣not␣possible
  ```

  Saldrá ese mismo mensaje al intentar ilegalizar los partidos `BLANKVOTE` y `NULLVOTE`, puesto que no es posible ilegalizarlos.

- Si la operación es **`[S]tats`** se mostrará entonces la lista completa de partidos actuales y el porcentaje de participación de la siguiente forma:

```
Party␣B␣numvotes␣YYB␣(ZZB%)
Party␣N␣numvotes␣YYN
Party␣XX1␣numvotes␣YY1␣(ZZ1%)
Party␣XX2␣numvotes␣YY2␣(ZZ2%)
        …
Party␣XXn␣numvotes␣YYn␣(ZZn%)
Participation:␣KK␣votes␣from␣VV␣voters␣(PP%)
```

Donde `XX` es el `partyName`, `YY` es el número de votos, `ZZ` es el porcentaje de votos recibidos por el partido, `KK` es el número de votantes que participaron, `VV` es el número de total de votantes (`totalVoters`) y `PP` es el porcentaje de participación.

Los porcentajes de votos recibidos por cada partido se calculan sobre el total de voto válido (i.e. votos a partidos + votos blancos, pero sin incluir nulos). El voto en blanco lo recontaremos como un partido más, mientras que para el voto nulo no daremos porcentaje (sólo número) dado que no cuenta como voto válido.

El dato de participación se calcula dando la proporción de personas que han ejercido su derecho a voto (partidos+blanco+nulo) sobre el total de posibles votantes del censo electoral, el cual se adjunta en la propia orden Stats.

3. Liberación de recursos. Una vez que se hayan terminado de procesar todas las peticiones del fichero de entrada, el programa **deberá proceder a vaciar la lista** eliminando todos sus elementos, liberando así los recursos de memoria consumidos.

4. Lectura de ficheros

Para facilitar el desarrollo de la práctica se proporciona un código en Pascal (fichero `read.pas)` que contiene un ejemplo de cómo leer y manejar uno de los ficheros de prueba disponibles en Moodle (`new.txt)`. También se pueden probar de manera conjunta los distintos ficheros de prueba 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 y los criterios de evaluación de la práctica.
Fecha límite de entrega: **Viernes 15 de Marzo de 2019 a las 23:55 horas**
Fechas de los **Controles de Seguimiento**:

- Semana del 25 de Febrero - 1 de Marzo: Implementación y prueba de la `unit StaticList`.
- Días 7, 8, 11 y 12 de Marzo: Implementación y prueba de algunas funcionalidades del `main.pas`: **`New`**, **`Stats`** y **`Vote`**.