Curso 2005/2006 Estructura de Datos y de la Información
I. Informática, I. T. Informática de Gestión y de Sistemas
PRÁCTICA 1

1 Objetivo

El objetivo es practicar la independencia de la implementación en los tipos abstractos de datos (TADs). Para ello se realizarán dos implementaciones distintas del TAD tConjunto (una estática con arrays y una dinámica con punteros) y se planteará un problema que deberá ser resuelto con las dos implementaciones (es decir, un mismo programa principal alternará la utilización de una implementación u otra modificando su cláusula `uses`).

2 El TAD tConjunto

Un conjunto es una colección de objetos denominados elementos del conjunto que se caracteriza por:

- Los elementos no están ordenados. Es decir, no podemos hablar del primer elemento del conjunto, del segundo, etc.
- No existen elementos repetidos. Es decir, añadir un elemento a un conjunto en el que ya existe no supone ninguna modificación en dicho conjunto

2.1 Tipos de datos incluidos en el TAD

- tConjunto Representa al conjunto
- tInfo. Dato de un elemento del conjunto (en nuestro caso será char)

2.2 Operaciones incluidas en el TAD

ConjuntoVacio $\rightarrow$ **tConjunto**
{Objetivo: Crear un conjunto vacío
Salida: Un conjunto vacío}

EsConjuntoVacio (tConjunto) $\rightarrow$ **Boolean**
{Objetivo: Determinar si un conjunto está vacío
Entrada: *Conjunto*
Salida: *true* si el conjunto está vacío, *false* en caso contrario.
PreCond: El *Conjunto* debe estar inicializado }

Pertenece (tInfo, tConjunto) $\rightarrow$ **Boolean**
{Objetivo: Determina la pertenencia de un elemento a un conjunto
Entrada: Un *elemento* y un *Conjunto*
Salida: *True* si *elemento* pertenece a *Conjunto*, *False* en caso contrario.
PreCond: El conjunto debe estar inicializado }

Insertar (tInfo, tConjunto) $\rightarrow$ **tConjunto**
{Objetivo: Añade un elemento a un conjunto

Entrada: El *elemento* a insertar y el *Conjunto* donde insertar
Salida: *Conjunto* con el elemento insertado.
PreCond: *Conjunto* está inicializado y se supone memoria suficiente
PosCond: *Conjunto* no se modifica si ya contiene un elemento con *tInfo*}

Cardinal (tConjunto) → Integer
{Objetivo: Cuenta el número de elementos del conjunto
Entrada: *Conjunto*
Salida: Número de elementos}

Intersección ($tConjunto_1$, $tConjunto_2$) → $tConjunto_3$
{Objetivo: Realiza la intersección de dos conjuntos
Entrada: Dos *Conjuntos*
Salida: Un nuevo *conjunto* resultado de la intersección de los conjuntos de entrada
PreCond: Los conjuntos deben estar inicializados y se supone memoria suficiente}

Union ($tConjunto_1$, $tConjunto_2$) → $tConjunto_3$
{Objetivo: Realiza la unión de dos conjuntos
Entrada: Dos *Conjuntos*
Salida: Un nuevo *conjunto* resultado de la unión de los conjuntos de entrada
PreCond: Los conjuntos deben estar inicializados y se supone memoria suficiente}

ConjuntoToStr (tConjunto) → string
{Objetivo: Devuelve el contenido de un conjunto
Entrada: *Conjunto*
Salida: *String* con los elementos del conjunto separados por un espacio}

3 Implementación del TAD tConjunto

Deberán realizarse dos implementaciones del TAD tConjunto, una estática basada en arrays que deberá situarse en la unit ConjuntoEstatico (fichero ConjuntoEstatico.pas) y una dinámica basada en punteros que deberá situarse en la unit ConjuntoDinamico (fichero ConjuntoDinamico.pas). La práctica incluirá un único programa principal (fichero principal.pas) con una cláusula uses que podrá ser:

- uses ConjuntoEstatico; Si se utiliza la implementación estática, o
- uses ConjuntoDinamico; Si se utiliza la implementación dinámica

El programa DEBE FUNCIONAR CORRECTAMENTE con ambas implementaciones.

4 Descripción del problema

4.1 Enunciado

En la facultad de informática se ha decidido crear un juego denominado CuentaLetras. El juego consiste en introducir una palabra de como máximo 10 caracteres. A cada palabra se le asigna una puntuación de la siguiente forma:

- Se cuenta el número *Nc* de consonantes distintas utilizadas.

- Se cuenta el número de vocales distintas utilizadas Nv. Se dará una bonificación *Bv* de 2 puntos si se utilizan todas.
- Se cuenta el número de caracteres raros distintos utilizados *Nr*. Se consideran caracteres raros los siguientes: {h, k, q, w, x, y, z} (Nota: estos caracteres puntuarán, por tanto, doble: como consonantes y raros). Se dará una bonificación *Br* de 4 puntos si se utilizan todos.
- La puntuación resultante será igual a Nc + Nv + Bv + Nr +Br. Es IMPORTANTE destacar que caracteres repetidos en una misma palabra/frase no se tienen en cuenta.
- Gana el jugador que consiga hallar la palabra con mayor puntuación

El programa a realizar sería un calculador de la puntuación para el juego de CuentaLetras. Su funcionamiento deberá ser el siguiente:

- La palabra a calcular se pasa como parámetro (situado entre comillas) al programa principal, por ejemplo tecleando: principal "ayuntamiento"

  Para acceder al parámetro se puede usar la función "paramstr" de la siguiente forma: "palabra:=paramstr(1)". Lee el primer parámetro, lo convierte a string y lo mete en la variable de tipo string palabra.

  También se puede usar la función paramCount para comprobar que el número de parámetros es efectivamente 1: "if (paramCount <> 1) then ..."

  Si usáis el IDE del FreePascal los parámetro al programa principal se le pueden pasar en la opción "Run → Parameters..."
- El resultado de la puntuación sale por pantalla desglosado de la siguiente manera:

  "Total = 10; Nc = 4; Nv = 5; Bv = 1; Nr = 1; Br = 0
- Para simplificar la práctica se consideran los siguientes supuestos

  La palabra introducida es correcta por lo que no será necesario realizar ninguna comprobación en dicho sentido.

  La palabra se introducirá en minúscula y sólo contendrá los caracteres ASCII no extendidos (es decir, no hay letras con acentos ni la "ñ").

4.2 Implementación

La implementación de la práctica debe tener en cuenta los siguientes aspectos:

- El funcionamiento del programa será en modo batch (sin interactividad con el usuario).
- El funcionamiento del programa tiene que hacer uso del TAD tConjunto para el cálculo de la puntuación. No se admitirá realizar el cálculo de ninguna otra forma que no sea usando el TAD tConjunto.
- Se deben crear como mínimo las siguientes funciones en el programa principal:

  CuentaConsonantes(tConjunto)→ Integer: Cuenta el nº de consonantes en el conjunto

  CuentaVocales(tConjunto)→ Integer: Cuenta el nº de vocales en el conjunto

  CuentaRaros(tConjunto)→ Integer: Cuenta el nº de caracteres raros en el conjunto

4.3 Informe

La práctica se acompañará de un pequeño informe, que se incluirá en un fichero en modo texto “informe.txt” en el que, para cada una de las funciones del TAD tConjunto, se razonará que implementación es más eficiente, si la estática o la dinámica (o ambas en caso de ser igual de eficientes). Por ejemplo, un párrafo de informe.txt podría ser:

```
intersección
------------
Esta operación es más eficiente el la implementación estática/dinámica
porque...
```

5 Normas de realización y entrega

- Las prácticas son OBLIGATORIAS y serán realizadas en grupo de DOS PERSONAS.
- La entrega de TODAS las prácticas en las FECHAS indicadas es requisito IMPRESCINDIBLE para aprobar la asignatura en la convocatoria de JUNIO. Para las convocatorias de SEPTIEMBRE y DICIEMBRE las prácticas serán las mismas pero se fijarán otras fechas de entrega.
- Las prácticas tendrán que estar escritas en FreePascal bajo Ubuntu.
- **Fecha límite de entrega: 5 de Mayo de 2006.**
- Forma de entrega: Las prácticas quedarán depositadas en la red según el procedimiento del CECAFI. No se admitirán discos ni papel. El proceso para depositarlas será el siguiente:
  1. Conectarse a las máquinas **xurxo** o **deza**.
  2. Situarse en los directorios:

     /PRACTICAS/ETIX/EDI/P1 (alumnos de **I.T.I. de Gestión**)
     /PRACTICAS/ETIS/EDI/P1 (alumnos de **I.T.I. de Sistemas**)
     /PRACTICAS/EI/EDI/P1 (alumnos de **I. Informática**)
  3. Situarse en el directorio que coincida con el login del usuario y copiar allí la práctica (sólo los ficheros fuentes, no los ejecutables). IMPORTANTE: El nombre del fichero fuente que contenga el programa principal deberá ser principal.pas
  4. Pasada la fecha de entrega no se permitirá el acceso a estos directorios.
- Estructura que deberá de tener cada programa/unit desarrollado

  Encabezamiento del programa/unit. Constará la siguiente información entre comentarios

  ```
  TÍTULO: Prácticas de EDI
  SUBTÍTULO: Practica 1
  AUTOR 1: _____________________________ LOGIN 1: __________
  AUTOR 2: _____________________________ LOGIN 2: ___________
  GRUPO: E.I / E.T.I.X. / E.T.I.S
  FECHA: __/__/____
  ```

  Cuerpo del programa/unit

  - El código irá comentado. Los comentarios han de ser concisos pero explicativos.

- Después de la cabecera de cada procedimiento o función se incluirá lo siguiente: objetivo, entradas, salidas, precondiciones (condiciones que han de cumplir las entradas para el correcto funcionamiento de la subrutina) y poscondiciones (otras consecuencias de la ejecución de la subrutina que no quedan reflejadas en la descripción del objetivo o de las salidas)

- Criterios de valoración de la práctica:

  Eficacia: que se cumplan las especificaciones

  Control de Errores: Control intensivo de todos los errores de ejecución que sea posible detectar

  Claridad: que el programa se pueda entender con facilidad, que contenga comentarios oportunos, indentación adecuada, nombres de variables significativos, etc.

  Modular: que esté construido con módulos intercambiables y reutilizables.