

## Práctico 2 – Parte 1

Docentes: Silvia Pelozo, Felipe Espósito, Emmanuel Gunther

1. El tipo de los valores de verdad es muy elemental pero no siempre es incluido como tipo concreto en los lenguajes de programación. Especificá el TAD *booleanos* incluyendo constructores (*verdadero* y *falso*), las operaciones usuales ( $\wedge, \vee, \Rightarrow, \equiv, \neg$ ), y las ecuaciones que las describen.
2. Algunos lenguajes de programación no incluyen el tipo *lista* como tipo concreto. Especificá el TAD *lista* de valores de tipo  $A$  a partir de los constructores usuales:

**TAD  $lista_A$**   
**constructores**  
 $[\ ] : lista_A$   
 $\triangleright : A \times lista_A \rightarrow lista_A$

Las operaciones a incluir son:  $\triangleleft$  para insertar un elemento por detrás, *tamaño* para observar la longitud de la lista, *cabeza* para observar el primer elemento de la lista, *cola* para eliminar el primer elemento de la lista, *es\_vacia* para observar si la lista es vacía, *concat* para concatenar dos listas, y  $\cdot$  para observar el elemento que se encuentra en cierta posición dada.

Notar que algunas operaciones tienen restricciones.

3. El TAD *cola* es ampliamente usado para resolver una gran variedad de problemas. Este uso se puede extender aún más si se incluyen operaciones para agregar, observar y eliminar elementos por ambos extremos. Una variante con estas características se denomina *cola doble* o *bicola*. Especificá este TAD que además de las 5 operaciones habituales del TAD *cola*, incluye operaciones para: devolver el último elemento, eliminar el último elemento, y agregar un elemento al comienzo de la bicola. Detallá los constructores, operaciones y ecuaciones necesarias.
4. Implementá el TAD *booleanos* del ejercicio 1 utilizando un número natural.
5. Un programa de computación simbólica permite a sus usuarios manipular *polinomios con coeficientes enteros*. Para ello se utilizará un TAD, cuyos constructores son los siguientes:

**TAD *polinomio***  
**constructores**  
 $0 : polinomio$   
 $-x^+ + : \mathbf{nat} \times \mathbf{nat} \times polinomio \rightarrow polinomio$

El constructor  $0$  crea el polinomio cuyos coeficientes son todos ceros. El constructor  $nx^k +$  agrega el factor con coeficiente  $n$  y grado  $k$ . Notá que no hay restricciones en la forma de construir un polinomio, por lo que puede aparecer más de un factor con mismo grado.

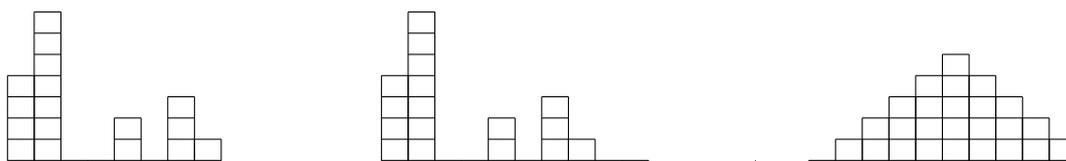
- a) Especificá las operaciones: evaluar en  $X$ , devolver la suma de todos los coeficientes de los factores de grado  $k$ , eliminar todos los factores con grado  $k$ .
  - b) Luego escribí un programa que *normalice* un polinomio, es decir, devuelva un polinomio equivalente pero con, a lo sumo, un factor de cada grado.  
**Ayuda:** Puede ser necesario introducir operaciones auxiliares para programar este procedimiento.
  - c) Implementá el TAD *polinomio* utilizando el TAD *lista* especificado en el ejercicio 2.
6. Se quiere diseñar un sistema para montar en los colectivos del sistema público de transporte de alguna ciudad, que lleve la cuenta de la cantidad de pasajes emitidos y la cantidad de monedas utilizadas para abonarlos. Se cuenta con dos dispositivos: una *tiqueteadora* que emite pasajes y un *monedero* que almacena monedas de 10, 25 y 50 centavos. El chofer opera ambos dispositivos de manera manual: introduciendo cada moneda de manera clasificada según su valor, y pulsando un botón para emitir el ticket. Se quiere resolver dos problemas: (1) verificar al terminar cada ronda que el monto de dinero acumulado en el monedero se corresponde con la cantidad de tickets emitidos; y (2) emitir un ticket cuando el monto introducido es suficiente (habiendo introducido las monedas utilizadas para pagar y posiblemente habiendo sacado monedas para dar vuelto).

- a) Especificá el monedero de la siguiente manera. Tendrá como constructores uno para crear el monedero vacío, y tres que introducen una moneda de cada tipo. Además dispondrá de operaciones para: devolver el número de monedas dentro del monedero para cada tipo, y sacar una moneda para cada tipo. Describí el comportamiento incluyendo las ecuaciones que sean necesarias, asumiendo que el monedero tiene capacidad infinita y que las operaciones para sacar monedas sólo pueden llevarse a cabo cuando existen monedas del tipo adecuado.
- b) Escribí un programa que resuelva cada uno de los problemas (1) y (2), asumiendo que el valor de un pasaje es  $N$ .

**Ayuda:** Para llevar la cuenta de los tiques emitidos puedes utilizar el TAD contador visto en clase.

7. Implementá el TAD *bicola* del ejercicio 3 utilizando la idea de *arreglo circular*.
8. Considerá un TAD muy parecido al tipo *pila*, que además de las 5 operaciones características, dispone una operación que invierte el orden de los elementos en la pila.
  - a) Especificá el TAD *pila reversible* que tienen los mismos constructores y operaciones que el TAD *pila*, y además la operación *invertir*.
 

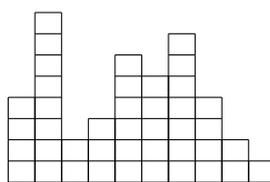
**Ayuda:** Puede ser conveniente introducir operaciones auxiliares para caracterizar la operación *invertir*.
  - b) Implementá este TAD utilizando arreglos. ¿Se te ocurre una implementación de manera que todas las operaciones sean de orden constante?
9. Los histogramas son gráficos en forma de barras verticales como, por ejemplo, los siguientes tres:



Para crear tales histogramas, se pueden utilizar tres constructores: uno para construir el histograma inicial (consistente de 1 sola columna vacía), otro para agregar una nueva columna vacía a (la derecha de) un histograma existente, y una tercera para agregar un cuadrado a la última columna (la que está más a la derecha).

El primer histograma de ejemplo se construye a partir del histograma inicial, utilizando 4 veces el constructor que agrega un cuadrado, una vez el que agrega una columna, 7 veces el que agrega un cuadrado, 3 veces el que agrega una columna, 2 veces el que agrega un cuadrado, 2 veces el que agrega una columna, 3 veces el que agrega un cuadrado, una vez el que agrega una columna y finalmente una vez el que agrega un cuadrado.

- a) Especificá el TAD *hist*, con sus 3 constructores, las ecuaciones entre ellos que sean necesarias y al menos tres operaciones: una para observar el número de columnas, otra para observar el número de cuadrados en una columna particular, y finalmente otra que aplicada a histogramas con igual número de columnas devuelva un nuevo histograma con la suma de cuadrados columna a columna. Por ejemplo, si se suman el segundo y tercer histogramas mostrados más arriba, se obtiene:



**Ayuda:** Es posible que necesites definir operaciones auxiliares para enunciar las ecuaciones que describen las operaciones mencionadas.

- b) Implementá el TAD *hist* de la manera que creas adecuada. Justificá tu elección mencionando sus ventajas frente a otras posibles implementaciones.

10. Implementá el TAD  $listas_A$  del ejercicio 2 con punteros, utilizando la estructura de datos de *lista ligada*. Considerá diferentes variaciones de esta estructura que permitan implementar las operaciones de manera mas o menos eficiente.
11. La siguiente es una especificación del TAD *conjunto* de elementos del tipo  $A$ :

**TAD  $conjunto_A$**   
**constructores**  
 $\emptyset : conjunto_A$   
 $\{-\} : A \rightarrow conjunto_A$   
 $\cup : conjunto_A \times conjunto_A \rightarrow conjunto_A$   
**operaciones**  
 $\in : A \times conjunto_A \rightarrow booleano$   
 $\cap : conjunto_A \times conjunto_A \rightarrow conjunto_A$   
**ecuaciones**  
 $e \in \emptyset = falso$   
 $e \in \{e'\} = (e = e')$   
 $e \in C \cup C' = (e \in C) \vee (e \in C')$   
 $\emptyset \cap C = C$   
 $\{e\} \cap C = (e \in C \rightarrow \{e\})$   
 $\neg e \in C \rightarrow \emptyset$   
 $(C'' \cup C') \cap C = (C'' \cap C) \cup (C' \cap C)$

Implementá el TAD *conjunto* de enteros utilizando como estructura de datos listas (abstractas), manteniendo sus elementos ordenados y sin repetición.

12. El TAD *rect* de los rectángulos cuyos lados tienen longitud entera, tiene tres constructores: uno para crear un rectángulo de  $1 \times 1$ ; otro para agregar una fila (de alto 1) a un rectángulo ya creado; y el último para agregar una columna (de ancho 1) a un rectángulo ya creado. Por ejemplo, la siguiente secuencia de rectángulos



se crea a partir del rectángulo inicial (de  $1 \times 1$ ) uno de  $4 \times 5$  agregando sucesivamente una fila, una columna, una columna, una columna, una fila, una fila y una columna. Las líneas internas fueron dibujadas sólo para ilustrar lo que fue ocurriendo, en realidad lo que importa son los rectángulos:



Observá que hay diferentes formas de construir un mismo rectángulo. Por ejemplo, la secuencia que comienza con el rectángulo inicial y agrega sucesivamente una columna, una columna, una fila, una fila, una columna, una fila y una columna construye el mismo rectángulo que la secuencia de arriba.

- Especificá el TAD *rect*, con sus 3 constructores, las ecuaciones entre ellos que sean necesarias y al menos cuatro operaciones que se aplican sobre un rectángulo y devuelven: la altura, el ancho, su perímetro, y su superficie respectivamente.
- Implementar el TAD *rect* de la manera que creas conveniente.