Estructuras de Datos y de la Información
Tipos abstractos de datos

Facultad de Informática
Ingeniería en Informática de Gestión

Enero 2002

Ejercicio 1 Especificar ecuacionalmente los números naturales con las siguientes operaciones:

- generadoras cero y suc,
- suma $+$,
- producto $*$,
- diferencia $-$ de naturales,
- potencia con base y exponente naturales,
- relación de igualdad,
- relación de orden menor o igual $\leq$,
- relación de orden menor estricto $<$,
- cociente div y resto mod de la división entera entre naturales,
- predicados es-par? y es-impar?.

Ejercicio 2 Se desea un tipo abstracto de datos (TAD) que ofrezca el tipo *palabra* junto con algunas operaciones básicas para manipular palabras, cuyas especificaciones informales son como sigue:

pal-vacía : $\longrightarrow$ *palabra*	{ crea la palabra vacía }
pon-letra : *palabra carácter* $\longrightarrow$ *palabra*	{ añade una letra al final de una palabra }
longitud : *palabra* $\longrightarrow$ *nat*	{ devuelve la longitud de una palabra }
letra : *nat palabra* $\longrightarrow$ *carácter*	{ consulta la letra en una posición dada }
es-vacía? : *palabra* $\longrightarrow$ *bool*	{ decide si una palabra es vacía }
es-igual? : *palabra palabra* $\longrightarrow$ *bool*	{ decide si una palabra es igual a otra }
es-inversa? : *palabra palabra* $\longrightarrow$ *bool*	{ decide si una palabra es inversa de otra }
es-capicúa? : *palabra* $\longrightarrow$ *bool*	{ decide si una palabra es capicúa o no }

Especificar ecuacionalmente el TAD distinguiendo las operaciones generadoras y las parciales.

Ejercicio 3 Especificar ecuacionalmente un TAD para describir los *conjuntos* finitos sobre un tipo de elementos dado como parámetro. El TAD debe incluir al menos las siguientes operaciones:

- conjunto vacío,
- añadir un elemento a un conjunto,
- formar un conjunto unitario con un elemento dado,

- relación de pertenencia entre un elemento y un conjunto,
- predicado para saber si un conjunto es vacío o no,
- quitar un elemento a un conjunto,
- unión de conjuntos,
- intersección de conjuntos,
- diferencia de conjuntos,
- cardinal de un conjunto.

Escribir las ecuaciones considerando los siguientes conjuntos de generadoras:

- La constructora del conjunto vacío, la constructora de conjuntos unitarios y la unión de conjuntos.
- La constructora del conjunto vacío y la que añde un elemento a un conjunto.

Implementar el TAD de los conjuntos finitos utilizando

1. vectores de elementos,
2. vectores de elementos sin repeticiones,
3. vectores de elementos sin repeticiones y ordenados, suponiendo en este caso que el tipo de los elementos sobre el que se construyen los conjuntos admite una relación de orden total.

Ejercicio 4 Especificar ecuacionalmente un TAD para describir los *multiconjuntos* finitos sobre un tipo de elementos dado como parámetro. El TAD debe incluir al menos las siguientes operaciones:

- multiconjunto vacío,
- añadir un elemento a un multiconjunto,
- calcular la multiplicidad de un elemento en un multiconjunto,
- predicado para saber si un multiconjunto es vacío o no,
- quitar una aparición de un elemento en un multiconjunto,
- borrar (todas las apariciones de) un elemento en un multiconjunto,
- unión de multiconjuntos,
- intersección de multiconjuntos,
- diferencia de multiconjuntos,
- calcular el cardinal de un multiconjunto, teniendo en cuenta las multiplicidades de los elementos,
- calcular el cardinal de un multiconjunto, contando solamente los elementos distintos.

Suponiendo que el tipo de los elementos sobre el que se construyen los multiconjuntos es el conjunto $\{1, 2, \ldots, N\}$, implementar la especificación de multiconjuntos del ejercicio anterior en términos de vectores de naturales de la forma $M[1..N]$. Calcular el coste en el caso peor de las implementaciones de todas las operaciones sobre multiconjuntos.

Ejercicio 5 Especificar ecuacionalmente un TAD que describe los polinomios con *coeficientes naturales* en una indeterminada, usando un tipo *polinomio* para los polinomios y las operaciones siguientes:

poli-nulo : $\longrightarrow$ *polinomio*	{ representa el polinomio nulo }
suma-mono : *nat nat polinomio* $\longrightarrow$ *polinomio*	{ suma-mono(c, n, p) representa el polinomio $cx^n + p$ }
suma : *polinomio polinomio* $\longrightarrow$ *polinomio*	{ calcula la suma de dos polinomios }
mult : *polinomio polinomio* $\longrightarrow$ *polinomio*	{ calcula el producto de dos polinomios }
coef : *nat polinomio* $\longrightarrow$ *nat*	{ coef(i, p) calcula el coeficiente de x^i en p }
es-nulo? : *polinomio* $\longrightarrow$ *bool*	{ reconoce el polinomio nulo }
grado : *polinomio* $\longrightarrow$ *nat*	{ calcula el grado de un polinomio }
eval : *polinomio nat* $\longrightarrow$ *nat*	{ evalúa un polinomio para un valor dado de la indeterminada }

Escribir una implementación para el TAD de polinomios con coeficientes naturales, representando los polinomios mediante vectores de pares de la forma (coeficiente, exponente), donde el vector está ordenado según el exponente, de menor a mayor. Calcular el coste en el caso peor de las implementaciones de todas las operaciones sobre polinomios.

Ejercicio 6 Modificar adecuadamente la especificación e implementación del TAD de los polinomios para que los coeficientes sean números *enteros* en vez de naturales.

Ejercicio 7 Especificar ecuacionalmente un TAD *consultorio* médico que disponga de las operaciones descritas informalmente a continuación:

- crea: genera un consultorio vacío, sin ninguna información.
- nuevo-médico: modifica un consultorio, dando de alta a un nuevo médico que antes no figuraba en el consultorio.
- pedir-consulta: modifica un consultorio, haciendo que un paciente se ponga a la espera para ser atendido por un médico, el cual debe estar dado de alta en el consultorio.
- siguiente-paciente: consulta el paciente a quien le toca el turno para ser atendido por un médico; éste debe estar dado de alta, y debe tener algún paciente que le haya pedido consulta.
- atender-consulta: modifica un consultorio, eliminando el paciente al que le toque ser atendido por un médico; éste debe estar dado de alta, y debe tener algún paciente que le haya pedido consulta.
- tiene-pacientes?: reconoce si hay o no pacientes a la espera de ser atendidos por un médico, el cual debe estar dado de alta.

En particular, hay que especificar los requerimientos de los géneros que suministran los médicos y los pacientes, teniendo en cuenta las necesidades del resto del programa.

Ejercicio 8 Especificar ecuacionalmente un TAD para describir las *cadenas* finitas sobre un alfabeto dado como parámetro. El TAD debe incluir al menos las siguientes operaciones:

- cadena vacía cad-vacía,
- añ-izq para añadir un carácter por la izquierda a una cadena,
- añ-der para añadir un carácter por la derecha a una cadena,
- unit para generar una cadena unitaria formada por un carácter dado,
- conc para concatenar dos cadenas,
- long para calcular la longitud de una cadena,

- primero para consultar el primer carácter de una cadena,
- elim-prim para eliminar el primer carácter de una cadena,
- último para consultar el último carácter de una cadena,
- elim-ult para eliminar el último carácter de una cadena,
- es-vacía? para decidir si una cadena es vacía o no,
- está? para decidir si un carácter aparece en una cadena o no,

y hay que utilizar las siguientes operaciones como constructoras:

1. cad-vacía y añ-izq,
2. cad-vacía y añ-der,
3. cad-vacía, unit, conc.

Ejercicio 9 Partiendo de la especificación de las cadenas sobre un alfabeto dado, extenderla para especificar las operaciones siguientes:

- *rota-izquierda*$(a_1 \dots a_n) = a_2 \dots a_n a_1$,
- *rota-derecha*$(a_1 \dots a_n) = a_n a_1 \dots a_{n-1}$,
- *i-esimo*$(a_1 \dots a_n) = a_i$,
- *medio*$(a_1 \dots a_n) = a_{(n+1) \operatorname{div} 2}$,
- *ordenada?*$(a_1 \dots a_n) = (\forall i : 1 \leq i < n : a_i \preceq a_{i+1})$, suponiendo que $\preceq$ es una relación de orden sobre los elementos que forman las cadenas.

Ejercicio 10 Usando la especificación de números enteros, especificar un TAD que defina los *números complejos* de componentes enteras, con las siguientes operaciones:

- parte real de un complejo,
- parte imaginaria de un complejo,
- suma de dos complejos,
- resta de dos complejos,
- producto de dos complejos,
- conjugado de un complejo,
- valor absoluto de un complejo.

Ejercicio 11 Especificar el tipo abstracto de datos CIE de los conjuntos de enteros definibles como uniones finitas de intervalos de enteros. Las operaciones a considerar son las siguientes:

vacio : $\rightarrow CIE$
anadir-intervalo : $CIE\ entero\ entero \rightarrow CIE$
elim-intervalo : $CIE\ entero\ entero \rightarrow CIE$
pertenece : $CIE\ entero \rightarrow booleano$
pert-intervalo : $CIE\ entero\ entero \rightarrow booleano$

- vacio genera el conjunto vacío.
- anadir-intervalo construye un CIE a partir del dado como argumento añdiéndole el conjunto de enteros pertenecientes al intervalo cerrado cuyos extremos se dan como argumentos segundo y tercero.
- elim-intervalo construye un CIE a partir del dado como argumento eliminado del mismo el conjunto de enteros perteneciente al intervalo cerrado cuyos extremos se dan como argumento segundo y tercero.

- pertenece nos dice si el entero dado pertenece o no al CIE indicado.
- pert-intervalo nos dice si todos los enteros pertenecientes al intervalo cerrado cuyos extremos se dan como argumento segundo y tercero estn en el CIE indicado como primer argumento.