Metodología de la Programación

Grado en Ingeniería Informática de Gestión y Sistemas de Información
Escuela de Ingeniería de Bilbao (UPV/EHU)
Departamento de Lenguajes y Sistemas Informáticos
Curso: 1^o
Curso académico: 2020-2021
Grupo 01
Tema 2: Especificación y documentación formal de programas
2 puntos

25-02-2021

Enunciado

Índice

Lista de figuras

****

1 Primer programa: División entre los elementos de $A(1..n)$ y el producto de $B(1..n)$ y $C(1..n)$ y división entre los elementos de $A(1..n)$ y $C(1..n)$ (*1 punto*)

Documentar el programa que se muestra en la figura 1. Dicho programa realiza lo siguiente:

- Recibe como datos de entrada tres vectores de enteros $A(1..n)$, $B(1..n)$ y $C(1..n)$ no vacíos que contienen solo elementos positivos (mayores o iguales que 1). Además, cada elemento de $A(1..n)$ es múltiplo del producto de los elementos correspondientes de $B(1..n)$ y $C(1..n)$. Dicho de otra forma, el resto de la división entera (mod) entre cada elemento de $A(1..n)$ y el producto de los elementos correspondientes de $B(1..n)$ y $C(1..n)$ es cero.
- Devuelve como resultado, los vectores $A(1..n)$, $B(1..n)$ y $C(1..n)$ modificados de la siguiente forma: para cada posición k comprendida entre 1 y n, se tendrá en $A(k)$ el cociente de la división entera entre el valor inicial de $A(k)$ y el producto de los valores iniciales de $B(k)$ y $C(k)$; se tendrá en $B(k)$ el cociente de la división entera entre el valor inicial de $A(k)$ y el valor inicial de $C(k)$; y se tendrá en $C(k)$ el valor inicial de $A(k)$.

En el programa de la figura 1, div representa la división entera. Ejemplos: $19\ div\ 2 = 9$; $19\ div\ 3 = 6$; $19\ div\ 4 = 4$; $17\ div\ 3 = 5$; $8\ div\ 12 = 0$.

Nótese que el hecho de que cada elemento de $A(1..n)$ sea múltiplo del producto de los correspondientes elementos de $B(1..n)$ y $C(1..n)$, hace posible recuperar los valores iniciales de $A(1..n)$ después de haber sido divididos por el producto de los correspondientes elementos de $B(1..n)$ y $C(1..n)$. Por ejemplo, si cogemos por un lado 30 y por otro 5 y 3, tenemos que $30\ div\ (5*3) = 2$ y a partir de 2 podemos volver al punto de partida: $2*(5*3) = 30$. Ello es debido a que 30 es múltiplo de $5*3$ (es decir, 15). En cambio, si cogemos por un lado 30 y por otro 4 y 3, tenemos que $30\ div\ (4*3) = 2$ y a partir de 2 no podemos volver al punto de partida: $2*(4*3) = 24$. Ello es debido a que 30 no es múltiplo de $4*3$ (es decir, 12).

(1) {Precondición} i := 1;	(1) (0,080 puntos)
(2) {Aserción intermedia}	(2) (0,005 puntos)
while (3) {Invariante} (4) {Expresión cota E} i $\leq$ n **loop**	(3) (0,200 puntos); (4) (0,015 puntos)
(5) {Aserción intermedia} aux := A(i) div (B(i) $*$ C(i));	(5) (0,010 puntos)
(6) {Aserción intermedia} A(i) := aux;	(6) (0,010 puntos)
(7) {Aserción intermedia} B(i) := aux $*$ B(i);	(7) (0,150 puntos)
(8) {Aserción intermedia} i := i + 1;	(8) (0,150 puntos)
(9) {Aserción intermedia} C(i $-$ 1) := B(i $-$ 1) $*$ C(i $-$ 1);	(9) (0,150 puntos)
(10) {Aserción intermedia}	(10) (0,150 puntos)
end loop;	
(11) {Postcondición}	(11) (0,080 puntos)

Figura 1: Primer programa: División entre los elementos de $A(1..n)$ y el producto de $B(1..n)$ y $C(1..n)$ y división entre los elementos de $A(1..n)$ y $C(1..n)$.

2 Segundo programa: Cantidad de números $abundantes$ pertenecientes al intervalo $[r..w]$ (*1 punto*)

Se dice que un número entero positivo (≥ 1) x es **abundante** si la suma de los divisores de x que pertenecen al intervalo $[1..x-1]$ es mayor que x. Por ejemplo, 12 es abundante porque $1+2+3+4+6 > 12$. También 18 es abundante puesto que $1+2+3+6+9 > 18$. En cambio, 8 no es abundante porque $1+2+4 \leq 8$. Tampoco 15 es abundante porque $1+3+5 \leq 15$. El 1 no es abundante y los números primos tampoco son abundantes.

Se pide realizar lo siguiente:

(a) (*0,010 puntos*) Definir el predicado $divisor(x, y)$ que exprese que el número positivo x es un divisor del número no negativo y. Por tanto, el predicado ha de expresar que x es mayor o igual que 1, que y es mayor o igual que 0 y que el resto de dividir y por x es 0.

(b) (*0,100 puntos*) Definir el predicado $abundante(x)$ que exprese que x es un número abundante. Por tanto, el predicado ha de expresar que x es mayor o igual que 1 y que la suma de los divisores de x que pertenecen al intervalo $[1..x-1]$ es mayor que x. Al definir este predicado, hay que utilizar el predicado del apartado anterior.

(c) (*0,890 puntos*) Documentar el programa que se muestra en la figura 2. Para ello, hay que utilizar el predicado definido en el apartado anterior siempre que sea posible. El programa de la figura 2 realiza lo siguiente:

- Recibe como datos de entrada dos números enteros r y w, donde r es mayor o igual que 1 y w es mayor o igual que r.
- Devuelve, en la variable $cantidad$, el número de elementos del intervalo $[r..w]$ que son abundantes.

La función $es\_abundante$ que aparece en el programa de la figura 2, es una implementación del predicado $abundante$ del apartado *(b)*. La función $es\_abundante$ estará escrita en el lenguaje de programación que se está utilizando, mientras que el predicado $abundante$ estará escrito en el lenguaje de la lógica de primer orden (o lógica de predicados).

(1) {Precondición}	(1) (0,020 puntos)
i := r − 1;	
(2) {Aserción intermedia}	(2) (0,005 puntos)
cantidad := 0;	
(3) {Aserción intermedia}	(3) (0,005 puntos)
while (4) {Invariante} (5) {Expresión cota E} i ≤ w − 1 **loop**	(4) (0,250 puntos); (5) (0,020 puntos)
(6) {Aserción intermedia}	(6) (0,010 puntos)
i := i + 1;	
(7) {Aserción intermedia}	(7) (0,210 puntos)
if es_abundante(i) **then**	
(8) {Aserción intermedia}	(8) (0,010 puntos)
cantidad := cantidad + 1;	
(9) {Aserción intermedia}	(9) (0,210 puntos)
end if;	
(10) {Aserción intermedia}	(10) (0,050 puntos)
end loop;	
(11) {Postcondición}	(11) (0,100 puntos)

Figura 2: Segundo programa: cantidad de números abundantes pertenecientes al intervalo $[r..w]$.