Trabajo Practico de Especificacion

Especificacion y WP(Weakness Precondition)

10 de septiembre de 2024 Algoritmos y Estructuras de Datos 1

pesutipolimardiano

Integrante	LU	Correo electrónico
Nievas, Martin	453/24	`tinnivas@gmail.com`
Bercovich, Maximo	002/01	`email2@dominio.com`
Apellido, Nicolas	003/01	`email3@dominio.com`
Pomsztein, Andy	624/24	`pomszteinandy@gmai.com`

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

1. Enunciados

1.1. Especficiacion

1. grandesCiudades: A partir de una lista de ciudades, devuelve aquellas que tienen más de 50.000 habitantes.
proc grandesCiudades (in ciudades: $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

2. sumaDeHabitantes: Por cuestiones de planificación urbana, las ciudades registran sus habitantes mayores de edad por un lado y menores de edad por el otro. Dadas dos listas de ciudades del mismo largo con los mismos nombres, una con sus habitantes mayores y otra con sus habitantes menores, este procedimiento debe devolver una lista de ciudades con la cantidad total de sus habitantes.
proc sumaDeHabitantes (in menoresDeCiudades: $seq\langle Ciudad\rangle$, in mayoresDeCiudades: $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

3. hayCamino: Un mapa de ciudades está conformado por ciudades y caminos que unen a algunas de ellas. A partir de este mapa, podemos definir las distancias entre ciudades como una matriz donde cada celda i, j representa la distancia entre la ciudad i y la ciudad i (Fig. 2). Una distancia de 0 equivale a no haber camino entre i y i. Notar que la distancia de una ciudad hacia sí misma es cero y la distancia entre A y B es la misma que entre B y A.

proc hayCamino (in distancias: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in desde: $\mathbb{Z}$, in hasta: $\mathbb{Z}$) : Bool

4. cantidadCaminosNSaltos: Dentro del contexto de redes informáticas, nos interesa contar la cantidad de "saltos" que realizan los paquetes de datos, donde un salto se define como pasar por un nodo. Así como definimos la matriz de distancias, podemos definir la matriz de conexión entre nodos, donde cada celda i, j tiene un 1 si hay un único camino a un salto de distancia entre el nodo i y el nodo i, y un 0 en caso contrario. En este caso, se trata de una matriz de conexión de orden 1, ya que indica cuáles pares de nodos poseen 1 camino entre ellos a 1 salto de distancia. Dada la matriz de conexión de orden 1, este procedimiento debe obtener aquella de orden n que indica cuántos caminos de n saltos hay entre los distintos nodos. Notar que la multiplicación de una matriz de conexión de orden 1 consigo misma nos da la matriz de conexión de orden 2, y así sucesivamente.
proc cantidadNSaltos (inout conexion: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in n: $\mathbb{Z}$)

5. caminoMínimo: Dada una matriz de distancias, una ciudad de origen y una ciudad de destino, este procedimiento debe devolver la lista de ciudades que conforman el camino más corto entre ambas. En caso de no existir un camino, se debe devolver una lista vacía.
proc caminoMinimo (in origen: $\mathbb{Z}$, in destino: $\mathbb{Z}$, in distancias: seq⟨seq⟨$\mathbb{Z}$⟩⟩) : seq⟨$\mathbb{Z}$⟩

1.2. WP (Weakest Precondiction)

La funcion poblacionTotal recibe una lista de ciudades donde al menos una de ellas es grande (es decir, supera los 50.000 habitantes) y devuelve la cantidad total de habitantes. Dada la siguiente especificacion:

`proc poblacionTotal` (in ciudades : $seq\langle Ciudad\rangle$) : $\mathbb{Z}$

`requiere` $\{(\exists i : \mathbb{Z})\ (0 \le i < |ciudades| \wedge_L ciudades[i].habitantes > 50,000 \wedge (\forall i : \mathbb{Z})\ (0 \le i < |ciudades| \longrightarrow_L ciudades[i].habitantes \ge 0) \wedge (\forall i, j : \mathbb{Z})\ (0 \le i < j \longrightarrow_L ciudades[i].nombre \neq ciudades[j].nombre))\}$

`asegura` $\{res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes\}$

Con la siguiente implementacion:

```
res := 0;
i := 0;
while (i < ciudades.lenght()) do
res := res + ciudades[i].habitantes;
i := i + 1
endwhile
```

- 1. Demostrar que la implementacion es correcta con respecto a la especificacion.
- 2. Demostrar que el valor devuelto es mayor a 50.000.

2. Predicados Reutilizables

`pred todosPositivos` (s : $seq\langle \mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |s| \longrightarrow_L s[i] \geq 0)$

}

`pred distanciasValidas` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |distancias| \longrightarrow_L todosPositivos(distancias[i]))$

}

`pred diagonalEnCeros` (s : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |s| \wedge 0 \leq j < s[i] \wedge i = j \longrightarrow_L s[i][j] = 0)$

}

`pred esMatrizSimetrica` (s : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |s| \wedge 0 \leq j < |s[i]| \longrightarrow_L s[i][j] = s[j][i])$

}

`pred esMatrizCuadrada` (s : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |s| \longrightarrow_L |s[i]| = |s|)$

}

`pred esCamino` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, c : $seq\langle \mathbb{Z}\rangle$, d : $\mathbb{Z}$, h : $\mathbb{Z}$) {

$(esMatrizCuadrada(distancias) \wedge |c| \geq 2) \wedge_L (\forall e : \mathbb{Z})\ (e \in c \longrightarrow_L 0 \leq e < |distancias| \wedge (c[0] = d \wedge c[|c|-1] = h) \wedge (\forall i : \mathbb{Z})\ (0 \leq i < |c| - 1 \longrightarrow_L distancias[c[i]][c[i+1]] > 0))$

}

3. Resolucion de Ejercicios

3.1. Especificacion

Ejercicio 1:

`proc grandesCiudades` (in ciudades : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

`requiere` {true}

`asegura` $\{|res| = cantidadCiudadesGrandes(ciudades) \wedge sonTodasCiudadesGrandes(res) \wedge (\forall c : Ciudad)\ (c \in res \longrightarrow_L c \in ciudades)\}$

`pred sonTodasCiudadesGrandes` (ciudades : $seq\langle Ciudad\rangle$) {

$((\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes > 50000))$

}

`aux cantidadCiudadesGrandes` (ciudades :$seq\langle Ciudad\rangle$) : $\mathbb{Z}$ $= \sum_{i=0}^{|s|-1}$ (if $s[i].habitantes > 50000$ then 1 else 0 fi) ;

Ejercicio 2:

`proc sumaDeHabitantes` (in menoresDeCiudades : $seq\langle Ciudad\rangle$, in mayoresDeCiudad : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

`requiere` $\{|menoresDeCiudades| = |mayoresDeCiudades| \wedge mismasCiudades(menoresDeCiudades, mayoresDeCiudades \wedge ciudadesDistintas(menoresDeCiudades) \wedge ciudadesDistintas(mayoresDeCiudades))\}$

`asegura` $\{|res| = |menoresDeCiudades| \wedge esLaSuma(res, menoresDeCiudades, mayoresDeCiudades) \wedge mismasCiudades(res, menoresDeCiudades) \wedge mismasCiudades(res, mayoresDeCiudades)\}$

`pred mismasCiudades` (s: $seq\langle Ciudad\rangle$, l: $seq\langle Ciudad\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |s| \longrightarrow_L (\exists j : \mathbb{Z})\ (0 \leq j < |l| \wedge_L s[i].nombre = l[j].nombre))$

}

`pred esLaSuma` (res : $seq\langle Ciudad\rangle$, s: $seq\langle Ciudad\rangle$, l: $seq\langle Ciudad\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |res| \longrightarrow_L (\exists j, k : \mathbb{Z})\ (0 \leq j < |s| \wedge 0 \leq k < |l| \wedge_L s[j].nombre = l[k].nombre \wedge res[i].habitantes = s[j].habitantes + l[k].habitantes))$

}

pred `ciudadesDistintas` (ciudades : $seq\langle Ciudad\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L \neg(\exists j : \mathbb{Z})\ (0 \leq j < |ciudades| \land i \neq j \land_L ciudades[i].nombre = ciudades[j].nombre))$

}

Ejercicio 3:

proc `hayCamino` (in distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in desde : $\mathbb{Z}$, in hasta : $\mathbb{Z}$) : Bool

requiere $\{esMatrizCuadrada(distancias) \land diagonalEnCeros(distancias) \land esMatrizSimetrica(distancias) \land distanciasValidas(distancias) \land (0 \leq desde < |distancias| \land 0 \leq hasta < |distancias|)\}$

asegura $\{res = \text{true} \longleftrightarrow (\exists c : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, c, desde, hasta))\}$

Ejercicio 4:

proc `cantidadDeCaminosNSaltos` (inout conexion : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, n : $\mathbb{Z}$)

requiere $\{conexion = conexion_0 \land esMatrizCuadrada(conexion) \land cerosEnLaDiagonal(conexion) \land esMatrizSimetrica(conexion) \land esMatrizConCerosYUnos(conexion)\}$

asegura $\{|conexion| = |conexion_0| \land_L (\forall i : \mathbb{Z})\ (0 \leq i < |conexion_0| \longrightarrow_L |conexion[i]| = |conexion_0| \land esMatrizDeOrdenN(conexion, conexion_0, n))\}$

pred `esMatrizConCerosYUnos` (conexion : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |conexion| \land 0 \leq j < |conexion[i]| \longrightarrow_L (conexion[i][j] = 0 \lor conexion[i][j] = 1))$

}

pred `esIdentidad` (m : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(esMatrizCuadra) \land_L (\forall i, j : \mathbb{Z})\ (0 \leq i < |m| \land 0 \leq j < |m[i]| \longrightarrow_L ((i = j \land m[i][j] = 1) \lor (i \neq j \land m[i][j] = 0)))$

}

pred `esProducto` (m : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, n : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, o : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, n : $\mathbb{Z}$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |m| \land 0 \leq j < |m[i]| \longrightarrow_L m[i][j] = \sum_{k=0}^{|n|-1} n[i][k] * o[k][j])$

}

pred `esMatrizDeOrdenN` (s : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, l : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\exists lista : seq\langle seq\langle seq\langle \mathbb{Z}\rangle\rangle\rangle)\ ((|lista| = n+1 \land esIdentidad(lista[0]) \land lista[1] = l \land lista[n] = s) \land (\forall i : \mathbb{Z})\ (1 \leq i \leq n \longrightarrow_L (esProducto(lista[i], lista[i-1], lista[1]))))$

}

Ejericio 5:

proc `caminoMinimo` (in origen : $\mathbb{Z}$, in destino : $\mathbb{Z}$, in distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) : $seq\langle \mathbb{Z}\rangle$

requiere $\{(diagonalEnCeros(distancias) \land esMatrizCuadrada(distancias) \land esMatrizSimetrica(distancias) \land distanciasValidas(distancias) \land (0 \leq origen < |distancias| \land 0 \leq destino < |distancias|))\}$

asegura $\{(esCamino(res) \land_L (\forall c : seq\langle \mathbb{Z}\rangle)\ (esCamino(c) \longrightarrow_L distanciaRecorrida(distancias, res) \geq distanciaRecorrida(distancias, c))) \lor (res = [\,]) \longleftrightarrow \neg(\exists c : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, c, origen, destino))\}$

aux `distanciaRecorrida` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, c : $seq\langle \mathbb{Z}\rangle$) : $\mathbb{Z} = \sum_{i=0}^{|c|-2} distancias[c[i]][c[i+1]]$;

4. WP (Weakest Precondition)

Primer item:
Para demostrar la correctitud de la implementación del programa con respecto a su especificación, debemos ver los siguientes puntos:

- 1. **Precondicion, guarda y postcondicion del ciclo**
- 2. **P** $\longrightarrow P_c \Longleftrightarrow P_c \longrightarrow \text{Wp}(\text{res} := 0; \text{Wp}(i := 0, P_c))$(La precondición de la especificación implica la precondición del ciclo.)

- 3. **Correctitud parcial del ciclo**
- 4. **Terminación del ciclo**

4.1. Precondicion, guarda y postcondicion del ciclo

Precondicion del ciclo:
$P_c \equiv \{(P \wedge i = 0 \wedge res = 0)\}$ (Donde P es la precondicion de la especificacion).

Guarda del ciclo:

$\mathbf{B} \equiv \{(i < |ciudades|)\}$

Postcondicion del ciclo:

$Q_c \equiv \{(res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes)\}$

Una vez definido la precondicion, guarda y postcondicion ahora veamos si P (la precondicion de la especificacion) implica la P_c (la precondicion de ciclo) eso lo vamos a hacer calculando la Wp(res := 0 ; Wp(i :=, P_c))

4.2. $P \longrightarrow P_c$.

Cálculo de la Wp(res := 0 ; Wp(i := 0, P_c)):

llamemos 1 a Wp(i := 0, P_c) y 2 Wp(res := 0, 1) (utilizo el axioma 3)

Calculamos 1:

$Wp(i := 0, P_c) \equiv (def(i) \wedge_L P \wedge 0 = 0 \wedge res = 0) \equiv (P \wedge \text{true} \wedge res = 0) \equiv (P \wedge res = 0)$

Una vez calculado 1 lo reemplazamos en 2 y nos queda $Wp(res := 0, P \wedge res = 0)$

Calculamos 2

$Wp(res := 0, P \wedge res = 0) \equiv (def(res) \wedge_L P \wedge 0 = 0) \equiv (P \wedge \text{true}) \equiv P$

Por lo tanto, como Wp(res := 0; Wp(i := 0, P_c)) $\equiv P$, podemos concluir que $P \longrightarrow P_c$. El siguiente paso en la demostración de correctitud consiste en verificar si el ciclo es parcialmente correcto. Para ello, aplicaremos el teorema del invariante, el cual nos permitirá demostrar que una propiedad invariante se mantiene a lo largo de cada iteración del ciclo, garantizando así la correctitud parcial.

4.3. Correctitud parcial del ciclo mediante teorema del invariante.

Para demostrar la correctitud parcial del ciclo nesecitamos declarar el invariante, el cual explicado en palabras va a consistir en mantener a i en un rango adecuado para evitar la indefinicion y que res sea la suma hasta cierto i

Obs: el invariante debe valer antes de cada iteracion y al finalizar cada iteracion.

Sea I el invariante definido de la siguente manera:
$\mathbf{I} \equiv (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes)$

Los siguientes axiomas son los que se deben corroborar para verificar que el ciclo es paracialmente correcto:

- 1. $P_c \longrightarrow I$
- 2. $\{I \wedge B\}S\{I\}$
- 3. $\{I \wedge \neg B\} \longrightarrow Q_c$

1. $P_c \longrightarrow I$ (La precondicion del ciclo implica a el invariante)

Entonces tenemos que $(P \wedge res = 0 \wedge i = 0) \longrightarrow (i \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes)$ pero como $i = 0 \longrightarrow 0 \leq i \leq |ciudades|$. Ademas como consecuencia $res = \sum_{j=0}^{0-1} ciuadades[j].habitantes = \sum_{j=0}^{-1} ciuadades[j].habitantes = 0$ por lo tanto queda desmotrado que $P_c \longrightarrow I$.

2. $\{I \wedge B\}S\{I\} \longleftrightarrow (\{I \wedge B\} \longrightarrow Wp(S_c, I))$

Caculemos el $Wp(S_c, I)$ (Donde S_c es el cuerpo del ciclo) por lo tanto por aximo 3 seria equivalente calcular el $Wp(res := ciudades[i].habitantes, Wp(i := i + 1, I))$.
Llamamos A a $Wp(i := i + 1, I)$ y B $Wp(res := res + ciudades[i.habitantes], A)$

Calculo de A:

$Wp(i := i + 1, I) \equiv (def(i) \wedge_L (0 \leq i + 1 \leq |ciudades| \wedge res = \sum_{j=0}^{(i+1)-1} ciudades[j].habitantes)) \equiv (0 \leq i + 1 \leq$

$|ciudades| \wedge res = \sum_{j=0}^{i} ciuades[j].habitantes)$.

Una vez calculado (A) remplazamos en (B) y queda de la siguiente manera $Wp(res := res{+}ciuadades[j].habitantes, 0 \leq i+1 \leq |ciudades| \wedge res = \sum_{j=0}^{i} ciudades[j].habitantes)$

Calculo de (B):

$Wp(res := res + ciuadades[j].habitantes, 0 \leq i+1 \leq |ciudades| \wedge res = \sum_{j=0}^{i} ciudades[j].habitantes) \equiv ((def(i) \wedge def(ciudades) \wedge 0 \leq i < |ciudades|) \wedge_L (0 \leq i+1 \leq |ciudades| \wedge res + ciudades[i].habitantes = \sum_{j=0}^{i} ciuadades[j].habitantes))$ ahora me fijo que puedo comprimir los rango y restar ciudades[i].habitantes a la sumatoria que seria el ultimo termino. Entonces me queda de la siguiente manera:

$Wp(res := res + ciudades[i].habitantes, (A)) \equiv (0 \leq i < |ciuadades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes)$.

Ahora veamos si vale que $\{I \wedge B\} \longrightarrow Wp(S_c, I)$

Donde $\{I \wedge B\} \equiv (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i < |ciudades|)$ comprimo el rango de i y me queda lo siguente $(0 \leq i < |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes)$. Como $\{I \wedge B\} \equiv Wp(S_c, I)$ entonces vale la implicacion y la tripla de Hoare es valida.

(3). $\{I \wedge \neg B\} \longrightarrow Q_c$

primero que nada definamos $\{I \wedge \neg B\}$

$\{I \wedge \neg B\} \equiv (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i \geq |ciudades|)$ (como i es mayor o igual que la longitud de ciudades y menor o igual a la vez me queda lo siguente)

$\equiv (i = |ciudades| \wedge res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes) \equiv Q_c$

Por lo tanto queda desmotrado que $\{I \wedge \neg B\} \longrightarrow Q_c$ y como consecuencia el ciclo resulta parcialmente correcto. Ahora lo siguiente es verificar que el ciclo termina mediante el teorema de la terminacion.

4.4. Terminacion del ciclo (mediante teroema de la terminacion).

Para verificar que el ciclo concluye debemos verificar los siguientes axiomas del teorema:

- **(4).** $\{I \wedge B \wedge f_v = v_0\} S \{f_v < v_0\}$
- **(5).** $\{I \wedge f_v \leq 0\} \longrightarrow \neg B$

Aclaraciones: f_v es la funcion variante la cual se define como $f_v = (|ciudades| - i)$. ya que tiene que ser un funcion decreciente y como longitud de ciudades es constante e i crece con cada iteracion me da como resultado una funcion decreciente.

Para ejemplificar supongamos ciudades = [(”Buenos Aires”,30000),(”Santa Rosa”,15000),(Rosario,25000),(”Jujuy”,20000)] con la siguiente tabla:

Iteracion	i	res	f_v
0	0	0	4
1	1	30000	3
2	2	45000	2
3	3	70000	1
4	4	90000	0

Tabla 1: visualizacion.

Ya cuando i = 4 la guarda $B \equiv (i < |ciudades|)$ no se cumple, sale del ciclo y se cumple que $f_v \leq 0$.

(4). $\{I \wedge B \wedge f_v = v_0\} S \{f_v < v_0\}$ primero definamos $\{I \wedge B \wedge f_v = v_0\}$

$\{I \wedge B \wedge f_v = v_0\} \equiv (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i < |ciudades| \wedge v_0 = |ciudades| - i)$

$\equiv (0 \leq i < |ciudades| \wedge res = \sum_{j=0}^{i-1}(ciudades[j].habitantes) \wedge v_0 = |ciudades| - i)$ (comprimi los rangos de i).

Ahora lo que debemos hacer para probar que la tripla de Hoare es valida:

$\{I \wedge B \wedge f_v = v_0\} \longrightarrow Wp(res := res + ciudades[i].habitantes, Wp(i := i+1, |ciudades| - i < v_0))$

(utilice axioma 3 de WP).

Llamamo (1) a $Wp(i := i+1, |ciudades| - i < v_0))$ y (2) a $Wp(res := res + ciudades[i].habitantes,$ (1)

Calculamos (1)

$Wp(i := i+1, |ciudades| - i < v_0)) \equiv (def(i) \wedge_L |ciudades| - (i+1) < v_0) \equiv$

$(|ciudades| - i - 1 < v_0)$.

Ahora remplazamos el valor calculado de (1) en (2) y queda $Wp(res := res + ciudades[i].habitantes, |ciudades| -$

$i - 1 < v_0)$

Calculamos ②

$Wp(res := res + ciudades[i].habitantes, |ciudades| - i - 1 < v_0) \equiv$
$(def(i) \wedge def(res) \wedge def(ciudades) \wedge 0 \leq i < |ciudades| \wedge_L |ciudades| - i - 1 < v_0) \equiv$
$(0 \leq i < |ciudades| \wedge |ciudades| - i - 1 < v_0)$

Entonces veamos si $\{I \wedge B \wedge f_v = v_0\} \longrightarrow Wp(S_c, f_v < v_0)$. Y claramente si ya que $v_0 = |ciudades| - i$ y yo se que $v_0 > |ciudades| - i - 1$ por lo tanto $|ciudades| - i > |ciudades| - i - 1$ luego con operaciones aritmeticas trabajo la dsesigualda y llego a $(0 > -1)$ y esos es simpre verdadero.

⑤. $\{I \wedge f_v \leq 0\} \longrightarrow \neg B$

Primero definimos $\{I \wedge f_v \leq 0\}$:
$\{I \wedge f_v \leq 0\} \equiv (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1}(ciudades[j].habitantes) \wedge |ciudades| \leq i) \equiv \ (i = |ciudades| \wedge \sum_{j=0}^{|ciudades|-1}(ciudades[j].habitantes))$
Y desarrollando la $\neg B \equiv (i \geq |ciudades|)$ entonces tenemos que si $(i = |ciudades|) \longrightarrow (i \geq |ciudades|)$.

En conclusión, la correctitud de la implementación del programa ha sido demostrada mediante todo lo detallado anteriormente. Además, se ha comprobado que el ciclo es parcialmente correcto utilizando el teorema del invariante y que termina en una cantidad finita de pasos aplicando el teorema de la terminación