



# TP de Especificación

## Análisis Habitacional Argentino

30 de septiembre de 2021

Algoritmos y Estructuras de Datos I

### Grupo 6

Integrante	LU	Correo electrónico
Romero, Santiago	272/21	santiagooromero1234@gmail.com
Rosselot, Eduardo	924/11	earosselot@gmail.com
Magi, Julian	829/21	magijulian0@gmail.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. Problemas

## 1.1. Encuesta válida

```
proc esEncuestaValida (in th: ephh, in ti: ephi, out result: Bool) {  
  Pre {true}  
  Post {result = true  $\iff$  cumpleRequisitos(th, ti)}  
}  
  
pred cumpleRequisitos (th: ephh, ti: ephi) {  
  encuestaHogaresCorrecta(th)  $\wedge$  encuestaIndividuosCorrecta(ti)  $\wedge$  encuestasSonCompatibles(th, ti)  
}  
  
pred encuestaHogaresCorrecta (th: ephh) {  
  esMatriz(th)  $\wedge$  noEsVacio(th)  $\wedge$  cantidadDeColumnasCorrecta(th, 12)  $\wedge$  noHayHogaresRepetidos(th)  
   $\wedge$  latLongValidas(th)  $\wedge$  hogaresPosibles(th)  $\wedge$  atributosHogaresEnRango(th)  
}  
  
pred encuestaIndividuosCorrecta (ti: ephi) {  
  esMatriz(ti)  $\wedge$  noEsVacio(ti)  $\wedge$  cantidadDeColumnasCorrecta(ti, 11)  $\wedge$  noHayIndividuosRepetidos(ti)  
   $\wedge$  atributosIndividuosEnRango(ti)  
}  
  
pred encuestasSonCompatibles (th: ephh, ti: ephi) {  
  todosHogaresTienenIndividuos(th, ti)  $\wedge$  todosIndividuosTienenHogar(th, ti)  $\wedge$  igualAñoTrimestre(th, ti)  
   $\wedge$  miembrosMenorA20(th, ti)  
}  
  
pred esMatriz (t: seq<seq< $\mathbb{Z}$ >>) {  
  ( $\exists x : \mathbb{Z}$ )( $\forall i : \mathbb{Z}$ )(enRango(i, t)  $\longrightarrow_L$  |t[i]| = x)  
}  
  
pred noEsVacio (t: seq<seq< $\mathbb{Z}$ >>) {  
  |t| > 0  
}  
  
pred cantidadDeColumnasCorrecta (t: seq<seq< $\mathbb{Z}$ >>, n:  $\mathbb{Z}$ ) {  
  |t[0]| = n  
}  
  
pred todosHogaresTienenIndividuos (th: ephh, ti: ephi) {  
  ( $\forall y : hogar$ )( $y \in th \longrightarrow_L (\exists x : individuo)(x \in ti \wedge_L x[@indcodusu] = y[@hogcodusu])$ )  
}  
  
pred todosIndividuosTienenHogar (th: ephh, ti: ephi) {  
  ( $\forall x : individuo$ )( $x \in ti \longrightarrow_L (\exists y : hogar)(y \in th \wedge_L x[@indcodusu] = y[@hogcodusu])$ )  
}  
  
pred noHayIndividuosRepetidos (ti: ephi) {  
  ( $\forall i, j : \mathbb{Z}$ )( $(enRango(i, ti) \wedge enRango(j, ti) \wedge i \neq j$   
   $\longrightarrow_L ((ti[i][@indcodusu] \neq ti[j][@indcodusu]) \vee (ti[i][@componente] \neq ti[j][@componente]))$ )  
}  
  
pred noHayHogaresRepetidos (th: ephh) {  
  ( $\forall i, j : \mathbb{Z}$ )( $(enRango(i, th) \wedge enRango(j, th) \wedge i \neq j) \longrightarrow_L th[i][@hogcodusu] \neq th[j][@hogcodusu]$ )  
}  
  
pred latLongValidas (th: ephh) {  
  ( $\forall x : hogar$ )( $x \in th \longrightarrow_L latitudValida(x[@hoglatitud]) \wedge longitudValida(x[@hoglongitud])$ )  
}
```

Tomamos como válidas las latitudes y longitudes posibles en la Tierra  $[-90, 90]$  y  $[-180, 180]$  respectivamente). Si bien se pueden normalizar números fuera del rango, entendemos que ningún dispositivo de medición lo daría fuera del rango y es mas probable que sea un error en la toma del dato.

```
pred latitudValida (latitud:  $\mathbb{Z}$ ) {  
   $-90 \leq latitud \leq 90$   
}  
  
pred longitudValida (longitud:  $\mathbb{Z}$ ) {  
   $-180 \leq longitud \leq 180$   
}
```

```

pred igualAñoTrimestre (th: ephh, ti: ephi) {
  igualAño(th, ti) ∧ igualTrimestre(th, ti)
}
pred igualAño (th: ephh, ti: ephi) {
  (∃ año : ℤ)((∀ x : hogar)(x ∈ th →L x[@hogaño] = año) ∧ (∀ y : individuo)(y ∈ ti →L y[@indaño] = año))
}
pred igualTrimestre (th: ephh, ti: ephi) {
  (∃ trimestre : ℤ)((∀ x : hogar)(x ∈ th →L x[@hogtrimestre] = trimestre) ∧
  (∀ y : individuo)(y ∈ ti →L y[@indtrimestre] = trimestre))
}
pred miembrosMenorA20 (th: ephh, ti: ephi) {
  (∀ x : hogar)(x ∈ th →L cantidadIndividuosEnHogar(x[@hogcodusu], ti) ≤ 20)
}
aux cantidadIndividuosEnHogar (hogcodusu: ℤ, ti: ephi) : ℤ = ∑i=0|ti|-1 if hogcodusu = ti[i][@indcodusu] then 1 else 0 fi ;
pred hogaresPosibles (th: ephh) {
  (∀ x : hogar)(x ∈ th →L x[@ii2] ≤ x[@iv2])
}
pred atributosIndividuosEnRango (ti: ephi) {
  (∀ x : individuo)(x ∈ ti →L (datoEnRango(x[@ch4], 1, 2) ∧ datoEnRango(x[@nivel_ed], 0, 1) ∧
  datoEnRango(x[@estado], -1, 1) ∧ datoEnRango(x[@cat_ocup], 0, 4) ∧ datoEnRango(x[@pp04g], 1, 10)))
}
pred atributosHogaresEnRango (th: ephh) {
  (∀ x : hogar)(x ∈ th →L (datoEnRango(x[@ii7], 1, 3) ∧ datoEnRango(x[@region], 1, 6) ∧ datoEnRango(x[@mas_500], 0, 1)
  ∧ datoEnRango(x[@iv1], 1, 5) ∧ datoEnRango(x[@ii3], 1, 2)))
}
pred datoEnRango (d: dato, valorMinimo: ℤ, valorMaximo: ℤ) {
  valorMinimo ≤ d ≤ valorMaximo
}

```

## 1.2. Histograma habitacional

La secuencia res, devuelve el histograma, donde cada posición i representa la cantidad de casas con i+1 habitaciones. Ya que consideramos que no tiene sentido que el histograma tenga un valor para casas con 0 habitaciones.

```

proc histHabitacional (in th: ephh, in ti: ephi, in region: ℤ, out res: seq⟨ℤ⟩) {
  Pre {1 ≤ region ≤ 6 ∧ cumpleRequisitos(th, ti)}
  Post {largoCorrecto(th, region, res) ∧ esHistograma(th, region, res)}
}
pred esHistograma (th: ephh, region: ℤ, res: seq⟨ℤ⟩) {
  (∀ i : ℤ)(enRango(i, res) →L res[i] = numCasasConHabEnRegion(th, region, i + 1))
}
aux numCasasConHabEnRegion (th: ephh, region: ℤ, n: ℤ) : ℤ =
  ∑k=0|th|-1 if (th[k][@iv2] = n ∧ th[k][@region] = region ∧ th[k][@iv1] = 1) then 1 else 0 fi ;
pred largoCorrecto (th: ephh, region: ℤ, res: seq⟨ℤ⟩) {
  (∃ i : ℤ)((enRango(i, th) ∧L esRegionyCasa(th, region, i)) ∧L
  (∀ j : ℤ)((enRango(j, th) ∧L esRegionyCasa(th, region, j)) →L (th[j][@iv2] ≤ th[i][@iv2])) ∧ |res| = th[i][@iv2])
}
pred esRegionyCasa (th: ephh, region: ℤ, x: ℤ) {
  th[x][@iv1] = 1 ∧ th[x][@region] = region
}

```

## 1.3. La casa esta quedando chica

```

proc laCasaEstaQuedandoChica (in th: ephh, in ti: ephi, out res: seq⟨ℝ⟩) {
  Pre {cumpleRequisitos(th, ti)}
  Post {|res| = 6 ∧ (∀ i : ℤ)(enRango(i, res) →L res[i] =  $\frac{\text{sumaHogaresHac}(th, ti, i+1)}{\text{sumaHogares}(th, i+1)})$ }
}

```

aux sumaHogares (th:  $eph_h$ , region:  $\mathbb{Z}$ ) :  $\mathbb{Z} = \sum_{i=0}^{|th|} -1$  if  $esRegionyCasa(th, region, i) \wedge \neg masDe500(th, i)$  then 1 else 0 fi ;

esRegionyCasa definido en sección 1.2.

pred masDe500 (th:  $eph_h$ , i:  $\mathbb{Z}$ ) {  
 $th[i][@mas\_500] = 1$   
}

aux sumaHogaresHac (th:  $eph_h$ , ti:  $eph_i$ , region:  $\mathbb{Z}$ ) :  $\mathbb{Z} = \sum_{i=0}^{|th|} -1$  if  $esRegionyCasa(th, region, i) \wedge \neg masDe500(th, i) \wedge hacCritico(th, ti, i)$  then 1 else 0 fi ;

pred hacCritico (th:  $eph_h$ , ti:  $eph_i$ , i:  $\mathbb{Z}$ ) {  
 $\frac{numPersonasCasa(th, ti, i)}{th[i][@iv2]} > 3$   
}

aux numPersonasCasa (th:  $eph_h$ , ti:  $eph_i$ , n:  $\mathbb{Z}$ ) :  $\mathbb{Z} = \sum_{i=0}^{|ti|} -1$  if  $ti[i][@indcodusu] = th[n][@hogcodusu]$  then 1 else 0 fi ;

#### 1.4. Crece el teletrabajo

proc creceElTeleworkingEnCiudadesGrandes ( in t1h:  $eph_h$ , in t1i:  $eph_i$ , in t2h:  $eph_h$ , in t2i:  $eph_i$ , out res: Bool ) {  
Pre {  $(cumpleRequisitos(t1h, t1i) \wedge cumpleRequisitos(t2h, t2i)) \wedge_L$   
 $(añosCorrectos(t1h, t2h) \wedge mismoTrimestre(t1h, t2h))$  }  
Post {  $(res = true) \iff (\frac{proporcionTeleworking(t2h, t2i)}{proporcionTeleworking(t1h, t1i)} > 1)$  }  
}

pred añosCorrectos (t1h:  $eph_h$ , t2h:  $eph_h$ ) {  
 $(\forall i, j : \mathbb{Z})((enRango(i, t1h) \wedge enRango(j, t2h)) \longrightarrow_L t1h[i][@hogaño] < t2h[j][@hogaño])$   
}

pred mismoTrimestre (t1h:  $eph_h$ , t2h:  $eph_h$ ) {  
 $(\forall i, j : \mathbb{Z})((enRango(i, t1h) \wedge enRango(j, t2h)) \longrightarrow_L t1h[i][@hogtrimestre] = t2h[j][@hogtrimestre])$   
}

aux proporcionTeleworking (th:  $eph_h$ , ti:  $eph_i$ ) :  $\mathbb{R} = \frac{cantIndCasaDepTeleworking(th, ti)}{cantIndCasaDep(th, ti)}$  ;

aux cantIndCasaDepTeleworking (th:  $eph_h$ , ti:  $eph_i$ ) :  $\mathbb{Z} = \sum_{k=0}^{|ti|} -1$  if  $viveEnCasaDep500(th, ti, k) \wedge haceTeleworking(th, ti, k)$  then 1 else 0 fi ;

aux cantIndCasaDep (th:  $eph_h$ , ti:  $eph_i$ ) :  $\mathbb{Z} = \sum_{k=0}^{|ti|} -1$  if  $viveEnCasaDep500(th, ti, k)$  then 1 else 0 fi ;

pred viveEnCasaDep500 (th:  $eph_h$ , ti:  $eph_i$ , k:  $\mathbb{Z}$ ) {  
 $(viveEnCasa(th, ti, k) \vee viveEnDep(th, ti, k)) \wedge viveEnMas500(th, ti, k)$   
}

pred haceTeleworking (th:  $eph_h$ , ti:  $eph_i$ , k:  $\mathbb{Z}$ ) {  
 $trabajaEnCasa(ti, k) \wedge tieneHabitacionDeTrabajo(th, ti, k)$   
}

pred viveEnCasa (th:  $eph_h$ , ti:  $eph_i$ , k:  $\mathbb{Z}$ ) {  
 $th[buscarHogar(th, ti[k][@indcodusu])][@iv1] = 1$   
}

pred viveEnDep (th:  $eph_h$ , ti:  $eph_i$ , k:  $\mathbb{Z}$ ) {  
 $th[buscarHogar(th, ti[k][@indcodusu])][@iv1] = 2$   
}

pred viveEnMas500 (th:  $eph_h$ , ti:  $eph_i$ , k:  $\mathbb{Z}$ ) {  
 $th[buscarHogar(th, ti[k][@indcodusu])][@mas500] = 1$   
}

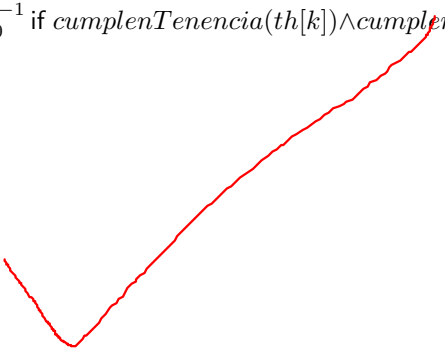
pred trabajaEnCasa (ti:  $eph_i$ , k:  $\mathbb{Z}$ ) {  
 $ti[k][@pp04g] = 6$   
}

pred tieneHabitacionDeTrabajo (th:  $eph_h$ , ti:  $eph_i$ , k:  $\mathbb{Z}$ ) {  
 $th[buscarHogar(th, ti[k][@indcodusu])][@ii3] = 1$   
}

aux buscarHogar (th:  $eph_h$ , hogcodusu:  $\mathbb{Z}$ ) :  $\mathbb{Z} = \sum_{k=0}^{|th|} -1$  if  $th[k][@hogcodusu] = hogcodusu$  then k else 0 fi ;

## 1.5. Costo de Subsidio de Mejora

```
proc costoSubsidioMejora (in th:  $eph_h$ , in ti:  $eph_i$ , in monto:  $\mathbb{Z}$ , out res:  $\mathbb{Z}$ ) {  
  Pre {cumpleRequisitos(th, ti)}  
  Post {cantidadDeSubsidiados(th, ti)  $\times$  monto = res}  
}  
  
aux cantidadDeSubsidiados (th:  $eph_h$ , ti:  $eph_i$ ) :  $\mathbb{Z} = \sum_{k=0}^{|th|-1}$  if cumplenTenencia(th[k])  $\wedge$  cumplenHab(th, ti, k) then 1 else 0 fi;  
  
pred cumplenTenencia (h: hogar) {  
  h[@ii7] = 1  
}  
  
numPersonasCasa definido en 1.3  
  
pred cumplenHab (th:  $eph_h$ , ti:  $eph_i$ , k:  $\mathbb{Z}$ ) {  
  th[k][@ii2] < numPersonasCasa(th, ti, k) - 2  
}
```



## 1.6. Generar Join

```

proc GenerarJoin (in th:  $eph_h$ , in ti:  $eph_i$ , out junta:  $JoinHI$ ) {
  Pre {cumpleRequisitos(th, ti)}
  Post {CantFilasCoincCantInd(ti, junta)  $\wedge$ 
        primerElementoUplaEsHogar(th, junta)  $\wedge$ 
        segundoElementoUplaEsIndividuo(ti, junta)  $\wedge$ 
        CadaUnoConSuHogar(junta)}
}

pred CantFilasCoincCantInd ( ti:  $eph_i$ , junta:  $JoinHI$ ) {
  |junta| = |ti|
}

pred primerElementoUplaEsHogar ( th:  $eph_h$ , junta:  $JoinHI$ ) {
  ( $\forall i : \mathbb{Z}$ )(enRango(i, junta)  $\rightarrow_L$  junta[i]0  $\in$  th)
}

pred segundoElementoUplaEsIndividuo ( ti:  $eph_i$ , junta:  $JoinHI$ ) {
  ( $\forall i : \mathbb{Z}$ )(enRango(i, junta)  $\rightarrow_L$  junta[i]1  $\in$  ti)
}

pred CadaUnoConSuHogar ( junta:  $JoinHI$ ) {
  ( $\forall i : \mathbb{Z}$ )(enRango(i, junta)  $\rightarrow_L$  (junta[i]0[@hogcodusu] = junta[i]1[@indcodusu]))
}

```

## 1.7. Ordenar region y tipo

```

proc ordenarRegionYTipo (inout th:  $eph_h$ , inout ti:  $eph_i$ ) {
  Pre {cumpleRequisitos(th, ti)  $\wedge$  th = TH_0  $\wedge$  ti = TI_0}
  Post {hogaresOrdenadosPorRegion(th)  $\wedge$ 
        hogaresOrdenadosPorCodusuEnRegion(th)  $\wedge$ 
        individuosOrdenadosPorHogcodusu(th, ti)  $\wedge$ 
        individuosOrdenadosPorComponenteEnHogar(th, ti)}
}

pred hogaresOrdenadosPorRegion (th:  $eph_h$ ) {
  ( $\forall i, j : \mathbb{Z}$ )((enRango(i, th)  $\wedge$  enRango(j, th)  $\wedge$  i < j)  $\rightarrow_L$  th[i][@region]  $\leq$  th[j][@region])
}

pred hogaresOrdenadosPorCodusuEnRegion (th:  $eph_h$ ) {
  ( $\forall i, j : \mathbb{Z}$ )((enRango(i, th)  $\wedge$  enRango(j, th)  $\wedge$  i < j)
   $\rightarrow_L$  th[i][@hogcodusu] < th[j][@hogcodusu])
}

pred individuosOrdenadosPorHogcodusu ( th:  $eph_h$ , ti:  $eph_i$ ) {
  ( $\forall i, j, k, h : \mathbb{Z}$ )((enRango(i, ti)  $\wedge$  enRango(j, ti)  $\wedge$  enRango(k, th)  $\wedge$  enRango(h, th)  $\wedge$ 
  viveEnHogar(ti[i], th[k])  $\wedge$  viveEnHogar(ti[j], th[h])  $\wedge$  i < j)  $\rightarrow_L$  k < h)
}

pred individuosOrdenadosPorComponenteEnHogar ( th:  $eph_h$ , ti:  $eph_i$ ) {
  ( $\forall i, j, k : \mathbb{Z}$ )((enRango(i, ti)  $\wedge$  enRango(j, ti)  $\wedge$  enRango(k, th)  $\wedge$ 
  viveEnHogar(ti[i], th[k])  $\wedge$  viveEnHogar(ti[j], th[k])  $\wedge$  i < j)  $\rightarrow_L$  ti[i][@componente] < ti[j][@componente])
}

pred viveEnHogar (x: individuo, h: hogar) {
  x[@indcodusu] = h[@hogcodusu]
}

```

## 1.8. Muestra homogénea

```

proc muestraHomogenea (in th:  $eph_h$ , in ti:  $eph_i$ , out res: seq(hogar)) {
  Pre {cumpleRequisitos(th, ti)}
  Post {(existeSecuenciaDiferencia(th, ti)  $\rightarrow$  esSecuenciaDiferenciaMasLarga(res, th, ti))  $\wedge$ 
        ( $\neg$  existeSecuenciaDiferencia(th, ti)  $\rightarrow$  |res| = 0)}
}

```

```

pred existeSecuenciaDiferencia (th : ephh, ti : ephi) {
  (∃ l : seq⟨hogares⟩)(esSecuenciaDiferenciaIngresos(l, th, ti))
}
pred esSecuenciaDiferenciaMasLarga (l : seq⟨hogar⟩, th : ephh, ti : ephi) {
  esSecuenciaDiferenciaIngresos(l, th, ti) ∧ noHaySeqDifIngMasLarga(l, th, ti)
}
pred esSecuenciaDiferenciaIngresos (l : seq⟨hogar⟩, th : ephh, ti : ephi) {
  esSubsecuencia(l, th) ∧ estaOrdenadoPorIngresos(l, ti) ∧ |l| ≥ 3 ∧ (∀ i : ℤ)(0 ≤ i < |l| - 2 →L
  diferenciaIngresos(l[i], l[i + 1], ti) = diferenciaIngresos(l[i + 1], l[i + 2], ti))
}
pred estaOrdenadoPorIngresos (l : seq⟨hogar⟩, ti : ephi) {
  (∀ i : ℤ)(0 ≤ i < |l| - 1 →L ingresosHogar(l[i], ti) < ingresosHogar(l[i + 1], ti))
}
pred noHaySeqDifIngMasLarga (l : seq⟨hogar⟩, th : ephh, ti : ephi) {
  (∀ sec : seq⟨hogar⟩)(esSecuenciaDiferenciaIngresos(sec, th, ti) →L |l| ≥ |sec|)
}
pred esSubsecuencia (res : seq⟨hogar⟩, th : ephh) {
  (∀ x : hogar)(x ∈ res → x ∈ th)
}
aux ingresosHogar (h : hogar, ti : ephi) : ℤ = ∑i=0|ti|-1 if h[@hogcodusu] = ti[i][@indcodusu] then ti[i][@p47T] else 0 fi ;
aux diferenciaIngresos (x, y : hogar, ti : ephi) : ℤ = ingresosHogar(y, ti) - ingresosHogar(x, ti) ;

```

## 1.9. Corregir región

```

proc corregirRegion (inout th : ephh, in ti : ephi) {
  Pre {cumpleRequisitos(th, ti) ∧ th = TH_0}
  Post {|th| = |TH_0| ∧ mantieneRegion(th, TH_0) ∧ corrigeRegion(th, TH_0)}
}
pred mantieneRegion (th, TH_0 : ephh) {
  (∀ x, y : hogar)(x ∈ TH_0 ∧ y ∈ th ∧ x[@region] ≠ 0 ∧ x[@hogcodusu] = y[@hogcodusu] →L y[@region] = x[@region])
}
pred corrigeRegion (th, TH_0 : ephh) {
  (∀ x, y : hogar)(x ∈ TH_0 ∧ y ∈ th ∧ x[@region] = 0 ∧ x[@hogcodusu] = y[@hogcodusu] →L y[@region] = 5)
}

```

no piden que se mantengan iguales los otros items, es medio raro cómo lo hicieron  
no está bueno permitir que los hogares no estén en el mismo índice que antes

## 1.10. Histograma de anillos concéntricos

```

proc HistogramaDeAnillosCocentricos (in th : ephh, in centro : ℤ × ℤ, in distancias : seq⟨ℤ⟩, out result : seq⟨ℤ⟩) {
  Pre {|distancias| > 0 ∧L distancias[0] ≥ 1 ∧
  encuestaHogaresCorrecta(th) ∧
  secDistanciasCreciente(distancias) ∧
  valoresDeCentroEnRango(centro)}
  Post {|distancia| = |result| ∧L esHistogramaDeAnillos(result, distancia, th)}
}
pred valoresDeCentroEnRango (centro : ent times ℤ) {
  latitudValida((centro)0) ∧ longitudValida((centro)1)
}
pred SecDistanciasCrecientes (distancias : seq⟨ℤ⟩) {
  (∀ i : ℤ)(0 ≤ i < |distancias| - 1 →L distancias[i] < distancias[i + 1])
}
pred esHistogramaDeAnillos (distancias : seq⟨ℤ⟩, result : seq⟨ℤ⟩, th : ephh) {
  (∀ i : ℤ)(1 ≤ i < |result| →L
  result[i] = cantidadHogaresEnAnillo(th, distancia[i], centro) - cantidadHogaresEnAnillo(th, distancia[i - 1], centro) ∧
  (result[0] = cantidadHogaresEnAnillo(th, distancia[0], centro)))
}
aux cantidadHogaresEnAnillo (th : ephh, centro : ℤ × ℤ, distancia : ℤ) : ℤ = ∑k=0|th|-1 if distanciaAlCentro(th[k], centro) <
distancia then 1 else 0 fi ;
aux distanciaAlCentro (h : hogar, centro : ent times ℤ) : ℝ = √((h[@latitud] - (centro)0)2 + (h[@longitud] - (centro)1)2) ;

```

## 1.11. Quitar individuos

```

proc quitarIndividuos (inout th: ephh, inout ti: ephi, in busqueda: seq⟨(ItemIndividuo, dato)⟩, out result: (ephh, ephi))
{
  Pre {CumpleRequisitos(th, ti) ∧ BusquedaValida(busqueda) ∧ th = TH_0 ∧ ti = TI_0}
  Post {|th| + |result0| = |TH_0| ∧
    |ti| + |result1| = |TI_0| ∧
    HogaresSobrantesEnTh(th, TH_0, busqueda, result) ∧
    IndividuosSobrantesEnTi(ti, TI_0, busqueda, result) ∧
    HogaresFiltradosEnResult(th, TH_0, busqueda, result) ∧
    IndividuosFiltradosEnResult(ti, TI_0, busqueda, result)}
}

pred busquedaValida (busqueda: seq⟨(ItemIndividuo, dato)⟩) {
  noHayItemsRepetidos(busqueda) ∧
  (∀ consulta : ℤ × ℤ)(consulta ∈ busqueda → itemIndividuoEsValido(consulta0) ∧ datoValido(consulta))
}

pred noHayItemsRepetidos (busqueda: seq⟨(ItemIndividuo, dato)⟩) {
  (∀ consulta1, consulta2 : ℤ × ℤ)(consulta1 ∈ busqueda ∧ consulta2 ∈ busqueda →L consulta1 ≠ consulta2)
}
  no funciona esto, en el para todo, si consulta1 = consulta2, vale el antecedente pero no el consecuente.
  les conviene usar índices de la secuencia de búsqueda

pred itemIndividuoEsValido (item: ℤ) {
  0 ≤ item ≤ 10
}

pred datoValido (consulta: (ItemIndividuo, dato)) {
  (consulta0 = @indcodusu ∧ datoMayorIgualQue(consulta1, 0)) ∨
  (consulta0 = @componente ∧ datoMayorIgualQue(consulta1, 0)) ∨
  (consulta0 = @indaño ∧ datoMayorIgualQue(consulta1, 0)) ∨
  (consulta0 = @indtrimestre ∧ datoEnRango(consulta1, 1, 4)) ∨
  (consulta0 = @ch4 ∧ datoEnRango(consulta1, 1, 2)) ∨
  (consulta0 = @ch6 ∧ datoMayorIgualQue(consulta1, 0)) ∨
  (consulta0 = @niveled ∧ datoEnRango(consulta1, 0, 1)) ∨
  (consulta0 = @estado ∧ datoEnRango(consulta1, -1, 1)) ∨
  (consulta0 = @catocup ∧ datoEnRango(consulta1, 0, 4)) ∨
  (consulta0 = @p47T ∧ datoMayorIgualQue(consulta1, -1)) ∨
  (consulta0 = @pp04g ∧ datoEnRango(consulta1, 1, 10))
}

pred datoMayorIgualQue (dato: ℤ, valorMinimo: ℤ) {
  valorMinimo ≤ dato
}

// datoEnRango definido en seccion 1.1

pred HogaresSobrantesEnTh (th, TH_0: ephh, busqueda: seq⟨(ItemIndividuo, dato)⟩, result: (ephh, ephi)) {
  (∀ h : hogar)(h ∈ TH_0 ∧ ¬habitantesEnResult(h, result) → h ∉ result0 ∧ h ∈ th)
}

pred HogaresFiltradosEnResult (th, TH_0: ephh, busqueda: seq⟨(ItemIndividuo, dato)⟩, result: (ephh, ephi)) {
  (∀ h : hogar)(h ∈ TH_0 ∧ habitantesEnResult(h, result) → h ∈ result0 ∧ h ∉ th)
}

pred habitantesEnResult (h: hogar, result: (ephh, ephi)) {
  (∃ ind : individuo)(ind ∈ result1 ∧ ind[@indcodusu] = h[@hogcodusu])
}

pred IndividuosFiltradosEnResult (ti, TI_0: ephi, busqueda: seq⟨(ItemIndividuo, dato)⟩, result: (ephh, ephi)) {
  (∀ consulta : ℤ × ℤ)(consulta ∈ busqueda →
  (∀ ind : individuo)(ind ∈ TI_0 ∧ ind[(consulta)0] = (consulta)1 → ind ∈ result1 ∧ ind ∉ ti))
}

pred IndividuosSobrantesEnTi (ti, TI_0: ephi, busqueda: seq⟨(ItemIndividuo, dato)⟩, result: (ephh, ephi)) {
  (∀ consulta : ℤ × ℤ)(consulta ∈ busqueda →
  (∀ ind : individuo)(ind ∈ TI_0 ∧ ind[consulta0] ≠ consulta1 → ind ∉ result1 ∧ ind ∈ ti))
}

```



## 2. Predicados y Auxiliares generales

```
pred enRango (n:  $\mathbb{Z}$ , l: seq( $T$ )) {  
   $0 \leq n < |l|$   
}
```

### 2.1. Auxiliares hogares

```
enum ItemHogar {  
  HOGCODUSU, HOGAÑO, HOGTRIMESTRE, HOGLATITUD, HOGLONGITUD, II7, REGION, MAS_500, IV1, IV2,  
  II2, II3  
}  
aux @hogcodusu :  $\mathbb{Z} = \text{ord}(\text{HOGCODUSU})$ ;  
aux @hogañO :  $\mathbb{Z} = \text{ord}(\text{HOGAÑO})$ ;  
aux @hogtrimestre :  $\mathbb{Z} = \text{ord}(\text{HOGTRIMESTRE})$ ;  
aux @hoglatitud :  $\mathbb{Z} = \text{ord}(\text{HOGLATITUD})$ ;  
aux @hoglongitud :  $\mathbb{Z} = \text{ord}(\text{HOGLONGITUD})$ ;  
aux @ii7 :  $\mathbb{Z} = \text{ord}(\text{II7})$ ;  
aux @region :  $\mathbb{Z} = \text{ord}(\text{REGION})$ ;  
aux @mas_500 :  $\mathbb{Z} = \text{ord}(\text{MAS\_500})$ ;  
aux @iv1 :  $\mathbb{Z} = \text{ord}(\text{IV1})$ ;  
aux @iv2 :  $\mathbb{Z} = \text{ord}(\text{IV2})$ ;  
aux @ii2 :  $\mathbb{Z} = \text{ord}(\text{II2})$ ;  
aux @ii3 :  $\mathbb{Z} = \text{ord}(\text{II3})$ ;
```

### 2.2. Auxiliares individuos

```
enum ItemIndividuo {  
  INDCODUSU, COMPONENTE, INDAÑO, INDTRIMESTRE, CH4, CH6, NIVEL_ED, ESTADO, CAT_OCUP, P47T,  
  PP04G  
}  
aux @indcodusu :  $\mathbb{Z} = \text{ord}(\text{INDCODUSU})$ ;  
aux @componente :  $\mathbb{Z} = \text{ord}(\text{COMPONENTE})$ ;  
aux @indañO :  $\mathbb{Z} = \text{ord}(\text{INDAÑO})$ ;  
aux @indtrimestre :  $\mathbb{Z} = \text{ord}(\text{INDTRIMESTRE})$ ;  
aux @ch4 :  $\mathbb{Z} = \text{ord}(\text{CH4})$ ;  
aux @ch6 :  $\mathbb{Z} = \text{ord}(\text{CH6})$ ;  
aux @nivel_ed :  $\mathbb{Z} = \text{ord}(\text{NIVEL\_ED})$ ;  
aux @estado :  $\mathbb{Z} = \text{ord}(\text{ESTADO})$ ;  
aux @cat_ocup :  $\mathbb{Z} = \text{ord}(\text{CAT\_OCUP})$ ;  
aux @p47T :  $\mathbb{Z} = \text{ord}(\text{P47T})$ ;  
aux @pp04g :  $\mathbb{Z} = \text{ord}(\text{PP04G})$ ;
```

El tp está aprobado  
Les marqué algunas cositas que quedaron pero es un muy buen tp,  
eligieron muy bien cómo especificar cada cosa.  
Felicitaciones!  
Tomás