UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2018

05/11/18

Trabajo Práctico de Implementación (TPI)

Encuesta Permanente de Hogares del INDEC (EPH)

1. Introducción

El Trabajo Práctico de Implementación consiste en que cada grupo debe implementar en C++ todas las funciones propuestas en el TP de Especificación. Para ello deben seguir la especificación de este enunciado, y no la propia que habían realizado en el transcurso del Trabajo Práctico de Especificación.

A continuación, repetimos el detalle del contenido de las tablas de Individuos y Hogares.

DISEÑO DE REGISTROS DE LA BASE PERSONAS

- ***CODUSU***: Código único (mayor a cero) para distinguir VIVIENDAS u HOGARES. Todos los individuos que vivan en un mismo HOGAR tendran a su vez el mismo CODUSU.
- ***COMPONENTE***: Código único (mayor a cero) que se asigna a los individuos que conforman cada hogar de la vivienda. Todos los integrantes de una VIVIENDA son encuestados (generan un registro).
- ***AÑO***: Año de relevamiento.
- ***TRIMESTRE***: Trimestre del año de relevamiento.
- ***CH4***: Género: Trimestre del año de relevamiento.
  - 1 - Varón
  - 2 - Mujer
- ***CH6***: Cuantos años cumplidos tiene.
- ***NIVEL_ED***: Estudios universitarios completos
  - 0 - NO
  - 1 - SI
- ***ESTADO***: Condición de actividad
  - 0 - Desocupado, Inactivo
  - 1 - Ocupado
- ***CAT_OCUP***: Categoría Ocupacional (Para ocupados y desocupados con ocupación anterior)
  - 0 - Ns./Nr.
  - 1 - Patrón
  - 2 - Cuenta propia
  - 3 - Obrero o empleado,
  - 4 - Trabajador familiar sin remuneración
- ***PP3E_TOT***: Total de horas que trabajó en la semana en la ocupación principal.
- ***PP04D_COD***: Código de ocupación principal del Clasificador Nacional de Ocupaciones.
- ***P21***: Monto de ingresos de la ocupación principal.

DISEÑO DE REGISTROS DE LA BASE HOGARES

- ***CODUSU***: Código único (mayor a cero) para distinguir VIVIENDAS u HOGARES. Todos los individuos que vivan en un mismo HOGAR tendran a su vez el mismo CODUSU.
- ***AÑO***: Año de relevamiento.
- ***TRIMESTRE***: Trimestre del año de relevamiento.
- ***ITF***: Monto de Ingreso Total Familiar.
- ***II7***: Régimen de tenencia de los habitantes
  - 1 - Propietario
  - 2 - Inquilino
  - 3 - Ocupante

En el transcurso de la encuesta, puede ocurrir que alguno de los ítems mencionados no haya sido completado, ya sea porque no fue informado o por un error del encuestador. Esa situación la representaremos con el entero -1.

1.1. Funciones C++

La declaración de cada una de las funciones a implementar es la siguiente:

```
bool esEncuestaValida(eph_i ti, eph_h th);
hogar mayorIngresoPorPersonaActiva(eph_i ti, eph_h th);
float porcHogaresNoPropMiemMay3(eph_i ti, eph_h th);
bool generoGPosiblesClientes(eph_i ti, eph_h th, int g);
void ordenarPorITF(eph_i & ti, eph_h th);
lista_nivel_ed nivelEducativoXHogar(eph_i ti, eph_h th);
lista_ev_hogares crecimientoHogarenoVsIngresos(eph_i ti1, eph_h th1,eph_i ti2,
                                               eph_h th2);
```

Donde declaramos las siguientes estructuras de datos

```
typedef vector<int> individuo;
typedef vector<int> hogar;
typedef vector<individuo> eph_i;
typedef vector<hogar> eph_h;
typedef tuple<int,float> nivel_ed_hogar;
typedef vector<nivel_ed_hogar> lista_nivel_ed;
typedef tuple<int,float,float,int> evolucion_hogar;
typedef vector<evolucion_hogar> lista_ev_hogares;
```

Funciones adicionales para leer y grabar encuestas:

```
void leerEncuesta(string filename, eth_i & ti, eth_h & th);
void grabarEncuesta(eth_i ti, eth_h th, string filename);
```

2. Consignas

- Implementar todas las funciones que se encuentran en el archivo ejercicios.h. Para ello, deberán usar la especificación que se encuentra en la última sección del presente enunciado.
- Extender el conjunto de casos de tests de manera tal de lograr una cobertura de líneas del 100 %. En caso de no poder alcanzarla, explicar el motivo. La cobertura debe estar chequeada con herramientas que se verán en laboratorio de la materia.
- No está permitido el uso de librerías de C++ fuera de las clásicas: **math**, **vector**, **tuple**, las de input-output, etc. Consultar con la cátegra por cualquier librería adicional.

Dentro del archivo que se que se descarguen desde la página de la materia van a encontrar los siguientes archivos y carpetas:

- `definiciones.h`: Aquí están los renombres mencionados arriba junto con la declaración del **enum** Item.
- `ejercicios.h`: *headers* de las funciones que tienen que implementar.
- `ejercicios.cpp`: Aquí es donde van a volcar sus implementaciones.

- `main.cpp`: Punto de entrada del programa.
- `tests`: Estos son algunos Tests Suites provistos por la materia. Aquí deben completar con sus propios Tests para lograr la cobertura pedida.
- `lib`: Todo lo necesario para correr Google Tests. Aquí no deben tocar nada.
- `datos`: Aquí están los datos que se usan en los tests y datos reales correspondientes a los años 2016 y 2017 de la EPH en CABA.
- `CMakeLists.txt`: Archivo que necesita CLion para la compilación y ejecución del proyecto. **NO** deben sobreescribirlo al importar los fuentes desde CLion.

Es importante recalcar que la especificación de los ejercicios elaborada por la materia puede diferir del enunciado propuesto para el TPE. Por ello, recomendamos fuertemente utilizar el enunciado anterior solo como guía y basarse en la especificación a la hora de implementarlas.

3. Entregable

La fecha de entrega del TPI es el **16 de NOVIEMBRE de 2018**.

1. Entregar una implementación de las funciones anteriormente descritas que cumplan el comportamiento detallado en la Especificación. El entregable debe estar compuesto por todos los archivos necesarios para leer y ejecutar el proyecto y los casos de test adicionales propuestos por el grupo.
2. El proyecto debe enviarse en un archivo comprimido, sin el directorio de compilación de CLion (binarios) y el directorio de GTEST. Enviarlo en formato **.zip** por correo a **algo1-tm-doc@dc.uba.ar**, poniendo como título: "Entrega TPI - Grupo XX", donde XX es el número de grupo que será asignado por la cátedra.
3. **Importante: Utilizar la especificación diseñada para este TP, no la solución del TPE!**.
4. **Importante: Es condición necesaria que la implementación pase todos los casos de tests provistos en el directorio tests. Estos casos sirven de guía para la implementación, existiendo otros TESTS SUITES *secretos* en posesión de la materia que serán usados para la corrección.**

4. Especificación

En esta sección se encuentra la Especificiación de los ejercicios a resolver, a partir del enunciado del TPE. La implementación de cada ejercicio DEBE SEGUIR OBLIGATORIAMENTE ESTA ESPECIFICACIÓN.

Todos aquellos auxiliares que no se encuentren definidos inmediatamente después del *proc*, se encuentran en la sección de Predicados y Auxiliares comunes.

4.1. Definición de columnas

Dado el tipo

enum ItemIndividuo { CODUSU, ANO4, TRIMESTRE, COMPONENTE, ... }

y

enum ItemHogar { CODUSU, ANO4, TRIMESTRE, ITF, ... }

Definimos los siguientes auxiliares:

```
aux cantidadItemsIndividuo : ℤ = 12;
aux cantidadItemsHogar : ℤ = 5;
aux @Codusu : ℤ = ord(CODUSU);
aux @Ano4 : ℤ = ord(ANO4);
aux @Trim : ℤ = ord(TRIMESTRE);
aux @Componente : ℤ = ord(COMPONENTE);
aux @Nivel_Ed : ℤ = ord(NIVEL_ED);
aux @Estado : ℤ = ord(ESTADO);
aux @Cat_Ocup : ℤ = ord(CAT_OCUP);
aux @Edad : ℤ = ord(CH6);
aux @Genero : ℤ = ord(CH4);
aux @PP3E_Tot : ℤ = ord(PP3E_TOT);
aux @PP04D_Cod : ℤ = ord(PP04D_COD);
aux @P21 : ℤ = ord(P21);
aux @Itf : ℤ = ord(ITF);
aux @Prop : ℤ = ord(II7);
```

4.2. Problemas

1. **proc esEncuestaVálida**(**in** $ti : eph_i$, **in** $th : eph_h$, **out** $result$: Bool)

   proc esEncuestaVálida (in ti: eph_i, in th: eph_h, out result: Bool) {
   Pre $\{True\}$
   Post $\{result \leftrightarrow encuestaValida(ti, th)\}$
   }

2. **proc mayorIngresoPorPersonaActiva**(**in** $th : eph_h$, **in** $ti : eph_i$, **out** $res : hogar$)

   Dada la encuesta th y ti (hogares e individuos, respectivamente), devuelve el hogar que posee la persona activa con mayor ingreso mensual.

   proc mayorIngresoPorPersonaActiva (in ti: eph_i, in th: eph_h, out res: hogar) {
   Pre $\{encuestaValida(ti, th) \wedge_L hayUnActivoDeMayorIngreso(ti)\}$
   Post $\{res \in th \ \wedge_L \ (\exists\, p : individuo)\ p \in ti \ \wedge_L \ individuoEnHogar(p, res) \wedge esElActivoDeMayorIngreso(p, ti))\}$
   }

   pred hayUnActivoDeMayorIngreso (ti: eph_i) {
   $(\exists\, p : individuo)\ p \in ti \ \wedge_L \ trabaja(p) \wedge p[@p21] > 0 \ \wedge \ (\forall\, q : individuo)\ (q \in ti \ \wedge \ q \neq p) \ \longrightarrow_L (trabaja(q) \ \wedge \ p[@p21] > q[@p21])$
   }
   pred esElActivoDeMayorIngreso (p: individuo, ti : eph_i) {
   $(\forall\, q : individuo)\ (q \in ti \ \wedge \ q \neq p) \longrightarrow_L trabaja(q) \ \wedge \ p[@p21] > q[@p21])$
   }

3. **proc porcHogaresNoPropMiemMay3**(**in** $ti : eph_i$, **in** $th : eph_h$, **out** $result : \mathbb{R}$).

   Dada la encuesta th y ti (hogares e individuos, respectivamente), se devuelve el porcentaje de hogares donde no hay miembros propietarios del hogar y donde la cantidad de miembros en el hogar es mayor e igual a tres.

   proc porcHogaresNoPropMiemMay3 (in ti: eph_i, in th: eph_h,out result:$\mathbb{R}$) {
   Pre $\{encuestaValida(ti, th)\}$
   Post $\{result = \frac{100}{|th|} \ \sum_{i=0}^{|th|-1} hogaresNoPropMay3(th[i], ti))\}$
   }

   aux hogaresNoPropMay3 (h: hogar, ti: eph_i) : $\mathbb{Z}$ =
   if $\neg esPropietario(h) \wedge (\exists\, si : seq\langle individuo\rangle)\ losIndividuosDelHogar(ti, si, h) \wedge_L |si| \geq 3$ then 1 else 0 fi ;
   aux esPropietario (h: hogar) : Bool $= h[@Prop] = 1$;

4. **proc generoGPosiblesClientes**(**in** $ti : eph_i$, **in** $th : eph_h$, **in** $g : \mathbb{Z}$, **out** $result$: Bool).
   Para un análisis de mercado y con el objetivo de lanzar paquetes turísticos para solteros, se quiere establecer si hay mas personas universitarias activas menores a 45 años de un género g que se recibe como parámetro, que otro género. Se va a suponer, como una aproximación, que las personas solteras viven solas. El ejercicio debe devolver verdadero en caso afirmativo.

   proc generoGPosiblesClientes (in ti: eph_i,in th: eph_h, in g: $\mathbb{Z}$, out result:Bool) {
   Pre $\{encuestaValida(ti, th)\}$
   Post $\{result = true \leftrightarrow \#hogaresXGenero(ti, th, g) > \#hogaresXNoGenero(ti, th, g)\}$
   }

   aux #hogaresXGenero (ti: eph_i,th: eph_h, g: $\mathbb{Z}$) : $\mathbb{Z}$ =
   $\sum_{i=0}^{|th|-1}$ if $esPerfilBuscado(th[i], ti) \wedge hogarSoloGeneroG(th[i], ti, g)$ then 1 else 0 fi ;
   aux #hogaresXNoGenero (ti: eph_i,th: eph_h, g: $\mathbb{Z}$) : $\mathbb{Z}$ =
   $\sum_{i=0}^{|th|-1}$ if $esPerfilBuscado(th[i], ti) \wedge \neg(hogarSoloGeneroG(th[i], ti, g))$ then 1 else 0 fi ;
   pred esPerfilBuscado (h: hogar, ti: eph_i) {
   $(\exists\, si : seq\langle individuo\rangle)\ losIndividuosDelHogar(ti, si, h) \wedge_L |si| = 1 \wedge (\exists\, i : individuo)\ i \in ti \wedge_L individuoEnHogar(i, h) \wedge trabaja(i) \wedge i[@Edad] < 45 \wedge i[@NivelEd] = 1$
   }
   pred hogarSoloGeneroG (h: hogar, ti: eph_i, g: $\mathbb{Z}$) {
   $(\forall\, i : individuo)\ i \in ti \longrightarrow_L individuoEnHogar(i, h) \wedge i[@Genero] = g$
   }

5. **proc ordenarPorITF**(**inout** $ti : eph_i$, **in** $th : eph_h$)

   Dada la encuesta th y ti (hogares e individuos, respectivamente), se pide ordenar la tabla de individuos en forma descendente segun el ingreso total familiar. En caso de empate ordenar ascendentemente por CODUSU. Dentro de cada hogar, los individuos se ordenan por el código de componente, de menor a mayor.

`proc ordenarPorITF` (inout ti: eph_i, in th: eph_h) {

`Pre` $\{encuestaValida(ti, th) \land ti = ti_0\}$

`Post` $\{mismosIndividuos(ti, ti_0) \land_L estaOrdenado(ti, th)\}$

}

`pred mismosIndividuos` (t1: eph_i, t2 : eph_i) {

$estaContenida(t1, t2) \land estaContenida(t2, t1)$

}

`pred estaOrdenado` (ti: eph_i, th : eph_h) {

$(\forall i : \mathbb{Z})\ 0 \leq i < |ti|-1 \longrightarrow_L (mayorITF(ti[i], ti[i+1], th) \lor menorComponente(ti[i], ti[i+1], th) \lor menorCODUSU(ti[i], ti[i+1], th))$

}

`pred mayorITF` (ind1: individuo, ind2 : individuo, th: eph_h) {

$(\exists i, j : \mathbb{Z})\ 0 \leq i, j < |th| \land_L individuoEnHogar(ind1, th[i]) \land individuoEnHogar(ind2, th[j]) \land th[i][@Itf] > th[j][@Itf]$

}

`pred menorCODUSU` (ind1: individuo, ind2 : individuo, th: eph_h) {

$igualITF(ind1, ind2, th)\ \land\ ind1[@Codusu] < ind2[@Codusu]$

}

`pred menorComponente` (ind1: individuo, ind2 : individuo, th: eph_h) {

$igualITF(ind1, ind2, th)\ \land\ mismoCodusu(ind1, ind2)\ \land\ ind1[@Componente] < ind_2[@Componente]$

}

`pred estaContenida` (t1: eph_i, t2 : eph_i) {

$(\forall i : individuo)\ i \in t1 \longrightarrow_L i \in t2$

}

`pred mismoCodusu` (ind1: individuo, ind2 : individuo) {

$ind1[@Codusu] = ind2[@Codusu]$

}

`pred igualITF` (ind1: individuo, ind2 : individuo, th: eph_h) {

$(\exists i, j : \mathbb{Z})\ 0 \leq i, j < |th| \land_L individuoEnHogar(ind1, th[i]) \land individuoEnHogar(ind2, th[j]) \land th[i][@Itf] = th[j][@Itf]$

}

6. **proc nivelEducativoXHogar**(**in** $th : eph_h$, **in** $ti : eph_i$, **out** $result : seq\langle \mathbb{Z}, \mathbb{R} \rangle$).

Dada la encuesta th y ti (hogares e individuos, respectivamente), se busca establecer los máximos niveles educativos por hogar en la distribución habitacional. El ejercicio debe devolver una secuencia de tuplas $result$ de dos elementos. El primero es el código de nivel educativo, y el segundo el porcentaje de hogares (del total), que poseen ese máximo nivel educativo entre sus componentes.

`proc nivelEducativoXHogar` (in th: eph_h,in ti: eph_i,out result:$seq\langle \mathbb{Z}, \mathbb{R} \rangle$) {

`Pre` $\{encuestaValida(ti, th)\}$

`Post` $\{|result| = 2 \land (\forall i : \mathbb{Z})\ esNivelEducativo(i) \longrightarrow_L \left(result[i]_0 = i \land result[i]_1 = 100 * \frac{\#hogaresNivelEducativo(i,ti,th)}{|th|}\right)\}$

}

`pred esNivelEducativo` (i: $\mathbb{Z}$) $\{0 \leq i \leq 1\}$

`aux #hogaresNivelEducativo` (ne: $\mathbb{Z}$, th: eph_h, ti: eph_i) : $\mathbb{Z}\ = \sum_{j=0}^{|th|-1}$ if $esMaxNivelEducativo(ne, th[i], ti)$ then 1 else 0 fi ;

`pred esMaxNivelEducativo` (ne : $\mathbb{Z}$, h: hogar, ti:eph_i) {

$(\exists i : individuo)\ i \in ti\ \land_L\ (individuoEnHogar(i, h)\ \land\ i[@NivelEd] = ne\ \land$
$(\forall j : individuo)\ j \in ti\ \longrightarrow_L\ (individuoEnHogar(j, h) \land j[@NivelEd] \leq ne))$

}

7. **proc crecimientoHogareñoVsIngresos**(**in** $t1i : eph_i$, **in** $t1h : eph_h$, **in** $t2i : eph_i$, **in** $t2h : eph_h$, **out** $result : seq\langle < \mathbb{Z}, \mathbb{R}, \mathbb{R}, \mathbb{Z} > \rangle$).

Dadas dos encuestas $t1$ y $t2$ correspondientes a dos trimestres distintos, devolver una secuencia $result$ de tuplas. Cada tupla contiene un análisis de la evolución de los hogares que pertenecen al cuartil correspondiente a sus ingresos (ITF). La tupla contiene la siguiente información: número de cuartil (el primer cuartil corresponde al 25 % de hogares con mayores ingresos), incremento porcentual promedio de componentes de hogares en ese cuartil, incremento porcentual promedio de ingresos de hogares del cuartil, cantidad de hogares promedio de donde surgieron los datos.

`proc crecimientoHogareñoVsIngresos` (in t1i: eph_i,in t1h: eph_h, in t2i: eph_i,in t2h: eph_h, out result: $seq\langle < \mathbb{Z}, \mathbb{R}, \mathbb{R}, \mathbb{Z} > \rangle$) {

`Pre` $\{encuestaValida(t1i, t1h)\ \land\ encuestaValida(t2i, t2h)\ \land_L\ sonDistintosTrimestres(t1i, t2i) \land |t1h| \geq 4 \land |t2h| \geq 4\}$

`Post` $\{|result| = 4\ \land_L\ (\forall k :< \mathbb{Z}, \mathbb{R}, \mathbb{R}, \mathbb{Z} >)\ k \in result \leftrightarrow$
$(1 \leq k_0 \leq 4\ \land_L\ esIncCompHogarCuartil(k_0, k_1, t1i, t1h, t2i, t2h)\ \land\ esIncIngresoHogarCuartil(k_0, k_2, t1h, t2h)\ \land$
$esPromCasosEnCuartil(k_0, k_3, t1h, t2h))\ \land ordenadaAscendentePorCuartil(result))\}$

}

`pred ordenadaAscendentePorCuartil` (s: $seq\langle < \mathbb{Z}, \mathbb{R}, \mathbb{R}, \mathbb{Z} > \rangle$) {

$(\forall i : \mathbb{Z})\ 0 \leq i < |s| - 1 \longrightarrow_L s[i]_0 < s[i+1]_0$

}

pred sonDistintosTrimestres (t1i: eph_i, t2i: eph_i) {
$(\forall\, i : individuo)\ i \in t1i\ \wedge\ (\forall\, j : individuo)\ j \in t2i\ \longrightarrow_L\ (i[@Ano4] \neq j[@Ano4]\ \vee\ i[@Trim] \neq j[@Trim]$
}
pred esIncCompHogarCuartil (Z: c, rComp: $\mathbb{R}$, t1i: eph_i, t1h: eph_h, t2i: eph_i, t2h: eph_h) {
$rComp =$ if $promedCmopHogCuart(t1i, t1h, c) > 0$ then $100 * \frac{promedCompHogCuart(t2i,t2h,c) - promedCompHogCuart(t1i,t1h,c)}{promedCompHogCuart(t1i,t1h,c)}$ else 0 fi
}
pred esIncIngresoHogarCuartil (Z: c, rIng: $\mathbb{R}$, t1h: eph_h, t2h: eph_h) {
$rIng =$ if $promedIngHogarCuart(t1h, c) > 0$ then $100 * \frac{promedIngHogarCuart(t2h,c) - promedIngHogarCuart(t1h,c)}{promedIngHogarCuart(t1h,c)}$ else 0 fi
}
pred esPromCasosEnCuartil (c: $\mathbb{Z}$, q: $\mathbb{Z}$, t1h: eph_h, t2h: eph_h) $\{q = promCasosCuartil(t1h, t2h, c)\}$
pred estaEnCuartil (t: eph_h, h: hogar, c: $\mathbb{Z}$) {
$(\exists\, tord : eph_h)\ esVersionOrdenadaPorITFyCODUSU(t, tord)\ \wedge_L$
$h \in subseq(tord, limIzquierdoCuartil(tord, c), limDerechoCuartil(tord, c))$
}
pred esVersionOrdenadaPorITFyCODUSU (th: eph_h, tord: eph_h) {
$hogaresUnicos(tord) \wedge mismosHogares(th, tord)\ \wedge\ (\forall\, i : \mathbb{Z})\ 0 \leq i < |tord| - 1 \longrightarrow_L tord[i][@Itf] > tord[i+1][@Itf] \vee$
$(tord[i][@Itf] = tord[i+1][@Itf] \wedge tord[i][@CODUSU] < tord[i+1][@CODUSU])$
}
pred mismosHogares (ta: eph_h, tb : eph_h) {
$estaContenida(ta, tb) \wedge estaContenida(tb, ta)$
}
pred estaContenida (ta: eph_h, tb : eph_h) {
$(\forall\, h : hogar)\ h \in ta \longrightarrow_L h \in tb$
}
aux promedCompHogarCuart (ti: eph_h, th : eph_h, q: $\mathbb{Z}$) : $\mathbb{R}$ =
$\sum_{i=0}^{|th|-1}$ if $estaEnCuartil(th, th[i], q)$ then $\frac{\#individuosEnHogar(th[i],ti)}{\#hogaresEnCuartil(th,q)}$ else 0 fi ;
aux promedIngHogarCuart (th : eph_h, c: $\mathbb{Z}$) : $\mathbb{R}$ = $\sum_{j=0}^{|th|-1}$ if $estaEnCuartil(th, th[i], c)$ then $\frac{th[i][@Itf]}{\#hogaresEnCuartil(th,c)}$ else 0 fi ;
aux #hogaresEnCuartil (th: eph_h, c: $\mathbb{Z}$) : $\mathbb{Z}$ = $\sum_{i=0}^{|th|-1}$ if $estaEnCuartil(th, th[i], c)$ then 1 else 0 fi ;
aux limIzquierdoCuartil (th: eph_h, c: $\mathbb{Z}$) : $\mathbb{Z}$ = $\lfloor 0{,}25 * |th| * (c-1) \rfloor$;
aux limDerechoCuartil (th: eph_h, c: $\mathbb{Z}$) : $\mathbb{Z}$ = $\lfloor 0{,}25 * |th| * c \rfloor$;
aux promCasosCuartil (t1h: eph_h, t2h: eph_h, c: $\mathbb{Z}$) : $\mathbb{R}$ = $\frac{1}{2}(\#hogaresEnCuartil(t1h, c) + \#hogaresEnCuartil(t2h, c))$;

4.3. Predicados y Auxiliares comunes

pred encuestaValida (ti: eph_i, th: eph_h) {
$|ti| > 0 \wedge |th| > 0\ \wedge_L$
$esMatrizIndividuo(ti)\ \wedge esMatrizHogar(th)\ \wedge_L$
$individuosValidos(ti)\ \wedge\ hogaresValidos(th)\ \wedge_L$
$individuosDistintos(ti)\ \wedge\ hogaresUnicos(th)\ \wedge_L$
$tablasCompatibles(ti, th)$
}
pred esMatrizIndividuo (ti: eph_i) {
$(\forall\, i, j : \mathbb{Z})(0 \leq i < |ti|\ \wedge\ 0 \leq j < |ti|) \longrightarrow_L (|ti[i]| = |ti[j]|\ \wedge\ |ti[i]| = cantidadItemsIndividuo)$
}
pred esMatrizHogar (t: eph_h) {
$(\forall\, i, j : \mathbb{Z})(0 \leq i < |t|\ \wedge\ 0 \leq j < |t|) \longrightarrow_L (|t[i]| = |t[j]|\ \wedge\ |t[i]| = cantidadItemsHogar)$
}
pred individuosValidos (ti: eph_i) {
$(\forall\, p : individuo)\ p \in ti \longrightarrow_L individuoValido(p)$
}
pred individuoValido (p: individuo) {
$p[@Codusu] > 0 \wedge p[@Componente] > 0 \wedge p[@Ano4] > 0 \wedge trimestreEnRango(p[@Trim]) \wedge nivelEdEnRango(p[@Nivel\_Ed]) \wedge$
$estadoEnRango(p[@Estado]) \wedge catOcupEnRango(p[@Cat\_Ocup]) \wedge edadEnRango(p[@Edad]) \wedge horasEnRango(p[@PP3E\_Tot]) \wedge$
$p21EnRango(p[@P21])\ \wedge\ p[@PP04D\_Cod] \geq -1\ \wedge\ \neg(trabaja(p) \longrightarrow p[@P21] = 0)\ \wedge\ (\neg trabaja(p) \longrightarrow p[@PP3E\_Tot] <=$
$0)\ \wedge\ (trabaja[p] \longrightarrow p[@Edad] > 10)$
}
pred hogaresValidos (th: eph_h) $\{(\forall\, h : hogar)\ h \in th \longrightarrow_L hogarValido(h)\}$
pred hogarValido (h: hogar) {
$h[@Codusu] > 0\ \wedge\ h[@Ano4] > 0\ \wedge\ trimestreEnRango(h[@Trim])\ \wedge\ h[@Itf] \geq 0\ \wedge\ pII7EnRango(h[@Prop])$
}
pred trimestreEnRango (i: $\mathbb{Z}$) $\{1 \leq i \leq 4\}$
pred nivelEdEnRango (i: $\mathbb{Z}$) $\{0 \leq i \leq 1\ \vee\ noSabeNoContesta(i)\}$
pred estadoEnRango (i: $\mathbb{Z}$) $\{0 \leq i \leq 1\ \vee\ noSabeNoContesta(i)\}$
pred catOcupEnRango (i: $\mathbb{Z}$) $\{0 \leq i \leq 4\ \vee\ noSabeNoContesta(i)\}$
pred edadEnRango (i: $\mathbb{Z}$) $\{0 \leq i \leq 110\ \vee\ noSabeNoContesta(i)\}$
pred p21EnRango (i: $\mathbb{Z}$) $\{\ i \geq 0\ \vee\ noSabeNoContesta(i)\}$

pred pII7EnRango (i: $\mathbb{Z}$) $\{1 \leq i \leq 3 \ \vee \ noSabeNoContesta(i)\}$

pred horasEnRango (i: $\mathbb{Z}$) $\{0 \leq i < 56 \ \vee \ noSabeNoContesta(i)\}$

pred noSabeNoContesta (i: $\mathbb{Z}$) $\{i = -1\}$

pred trabaja (p: individuo) $\{p[@Estado] = 1\}$

pred individuosDistintos (ti: eph_i) {

$(\forall p1, p2 : individuo)\ (p1 \in ti \ \wedge \ p2 \in ti \ \wedge \ p1 \neq p2) \longrightarrow_L (p1[@Codusu] \neq p2[@Codusu] \ \vee \ p1[@Componente] \neq p2[@Componente])$

}

pred hogaresUnicos (th: eph_h) {

$(\forall p1, p2 : hogar)\ (p1 \in th \ \wedge \ p2 \in th \ \wedge \ p1 \neq p2) \longrightarrow_L p1[@Codusu] \neq p2[@Codusu]$

}

pred tablasCompatibles (ti: eph_i, th: eph_h) {

$mismoPeriodo(ti, th) \ \wedge \ individuosConHogar(ti, th) \ \wedge \ hogaresConIndividuos(ti, th) \ \wedge \ ingresoHogarEnRango(ti, th)$

}

pred mismoPeriodo (ti: eph_i, th: eph_h) {

$(\forall i : individuo) \ \ i \in ti \ \wedge \ (\forall h : hogar)\ h \in th \longrightarrow_L (i[@Ano4] = h[@Ano4] \ \wedge \ i[@Trim] = h[@Trim])$

}

pred individuosConHogar (ti: eph_i, th: eph_h) {

$(\forall i : individuo)\ i \in ti \longrightarrow_L ((\exists h : eph_h) \ \ h \in th \ \wedge_L \ individuoEnHogar(i, h))$

}

pred hogaresConIndividuos (ti: eph_i, th: eph_h) {

$(\forall h : hogar)\ h \in th \longrightarrow_L ((\exists i : individuo) \ \ i \in ti \ \wedge_L \ individuoEnHogar(i, h))$

}

pred ingresoHogarEnRango (ti: eph_i, th: eph_h) {

$(\forall h : hogar)\ h \in th \ \longrightarrow_L h[@Itf] \leq ingresoTotalHogar(ti, h)$

}

pred individuoEnHogar (i: individuo, h : hogar) {

$i[@Codusu] = h[@Codusu]$

}

pred losIndividuosDelHogar (ti: eph_i, si: $seq\langle individuo\rangle$, h: hogar) {

$\#individuosEnHogar(h, ti) = |si| \wedge individuosDistintos(si) \wedge (\forall i : individuo)\ i \in si \longrightarrow_L individuoEnHogar(i, h)$

}

aux ingresoTotalHogar (ti: eph_i, h: hogar) : $\mathbb{Z} = \sum_{i=0}^{|ti|-1}$ if $individuoEnHogar(ti[i], h)$ then $ti[i][@p21]$ else 0 fi ;

aux #individuosEnHogar (h: hogar,ti: eph_i) : $\mathbb{Z} = \sum_{i=0}^{|ti|-1}$ if $individuoEnHogar(ti[i], h)$ then 1 else 0 fi ;