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

Algoritmos y Estructura de Datos I

Primer cuatrimestre de 2018

21/5/18

Trabajo Práctico de Implementación (TPI)

Encuesta Permanente de Hogares del INDEC (EPH)

1. Introducción

Para este TP van a tener que implementar en C++ todas las funciones propuestas en el TP de Especificación.

La representación de la encuesta sufrirá algunas modificaciones con respecto a la presentado en el TPE para semejarlo a los datos reales. Se agregarán dos columnas nuevas y se cambiarán algunos valores.

Cabe agregar, que las encuestas provistas para el TPI son adaptaciones de los datos reales provistos desde la página del INDEC. Se han integrado en un solo archivo la información proveniente de dos tablas: Una tabla específica correspondiente a los HOGARES y otra tabla de detalle de los INDIVIDUOS dentro de cada HOGAR. Los datos NO han sido modificados, excepto el CODUSU, ya que era una variable alfanumérica, ni completados faltantes. Debido a esto último, aparecen valores en elementos de la tabla como **-9** o **-1**, que significan ausencia de respuesta o ausencia de datos respectivamente.

1.1. Especificación de los datos de las encuestas

A continuación enumeramos cada columna junto a su descripción:

- ***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. Además los hogares conservan el mismo CODUSU entre encuestas de diferentes años.
- ***COMPONENTE***: Código único (mayor a cero) que se asigna a los individuos que conforman cada hogar de la vivienda. Este código es el mismo entre encuestas del mismo año para la misma persona.
- ***AÑO***: Año de relevamiento.
- ***NIVEL_ED***: Estudios universitarios completos
  - 0 - NO
  - 1 - SI
- ***ESTADO***: Condición de actividad
  - 1 - Ocupado
  - 2 - Desocupado
  - 3 - Inactivo
  - 4 - Menor de 10 años
- ***EDAD***: Edad del individuo
- ***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 definida en ***PP04D_COD***.
- ***ITF***: Monto de Ingreso Total Familiar.

- ***IX_Tot***: Cantidad de miembros del hogar.
- ***IX_Mayeq10***: Cantidad de miembros del hogar de 10 y más años.
- **P47T**: Suma de los ingresos laborales y no laborales del individuo
- **CAT_INAC:** Categoría de inactividad.
  - 1: Jubilado
  - 2: Rentista
  - 3: Estudiante
  - 4: Ama de casa
  - 5: Menor de 6 años
  - 6: Discapacitado
  - 7: Otros

1.2. Funciones C++

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

```
bool esEncuestaValida(eph t);
int laMejorEdad(eph t);
float promedioIngresoProfesional(eph t);
int hogarDeMayorIngreso(eph t);
individuo mejorNoProfesional(eph t);
bool sigoEstudiando(eph t);
individuo empleadoDelAnio(eph t);
bool noTieneAQuienCuidar(eph t);
bool pareto(eph t);
int elDeMayorIncrementoInterAnual(eph t1, eph t2);
vector<tuple<int,float>> mejorQueLaInflacion(eph t1, eph t2, float infl);
void ordenar(eph &t);
void agregarOrdenado(eph &t, individuo ind);
void quitar(eph &t, individuo ind);
```

Donde declaramos las siguientes estructuras de datos os renombres

```
typedef vector<int> individuo;
typedef vector<individuo> eph;
typedef tuple<int,float> paritaria_exitosa;
typedef vector<paritaria_exitosa> lista_exitosos;
```

Funciones adicionales para leer y grabar encuestas:

```
eph leerEncuesta(string filename);
void grabarEncuesta(ept h, 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 cátedra. 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 cátedra 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 **04 de Junio 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. El proyecto debe entregarse en un archivo comprimido, sin el directorio de compilación de CLion (binarios), con el número de grupo en el nombre.
2. **Importante: Utilizar la especificación diseñada para este TP, no la solución del TPE!**.
3. **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 cátedra 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 Item { CODUSU, COMPONENTE, ANO4, NIVEL_ED, ESTADO, ... }

Definimos los siguientes auxiliares:

`fun cantidadItems` : $\mathbb{Z} = 15$;
`fun @Codusu` : $\mathbb{Z} = ord(CODUSU)$;
`fun @Ano4` : $\mathbb{Z} = ord(ANO4)$;
`fun @Componente` : $\mathbb{Z} = ord(COMPONENTE)$;
`fun @Nivel_Ed` : $\mathbb{Z} = ord(NIVEL\_ED)$;
`fun @Estado` : $\mathbb{Z} = ord(ESTADO)$;
`fun @Cat_Ocup` : $\mathbb{Z} = ord(CAT\_OCUP)$;
`fun @Edad` : $\mathbb{Z} = ord(EDAD)$;
`fun @PP3E_Tot` : $\mathbb{Z} = ord(PP3E\_TOT)$;
`fun @PP04D_Cod` : $\mathbb{Z} = ord(PP04D\_COD)$;
`fun @P21` : $\mathbb{Z} = ord(P21)$;
`fun @P47T` : $\mathbb{Z} = ord(P47T)$;
`fun @Itf` : $\mathbb{Z} = ord(ITF)$;
`fun @IX_Tot` : $\mathbb{Z} = ord(IX\_TOT)$;
`fun @IX_MayeEq10` : $\mathbb{Z} = ord(IX\_MAYEEQ10)$;
`fun @Cat_Inac` : $\mathbb{Z} = ord(CAT\_INAC)$;

4.2. Problemas

1. **proc esEncuestaVálida**(**in** $t : eph$, **out** $result$: Bool).

   proc esEncuestaVálida (in t: eph, out result: Bool) {
   Pre $\{True\}$
   Post $\{result \leftrightarrow encuestaValida(t)\}$
   }

2. **proc laMejorEdad**(**in** $t : eph$, **out** $result : \mathbb{Z}$).

   proc laMejorEdad (in t: eph, out result: $\mathbb{Z}$) {
   Pre $\{encuestaValida(t)\}$
   Post $\{hayEdad(t, result) \wedge_L (\forall\, e : \mathbb{Z})\ hayEdad(t, e) \longrightarrow_L promedioIngresoXEdad(t, result) \geq promedioIngresoXEdad(t, e)\}$
   }

   pred hayEdad (t: eph, edad: $\mathbb{Z}$) $\{(\exists\, p : individuo)\ p \in t\ \wedge_L\ p[@Edad] = edad\}$
   fun promedioIngresoXEdad (t: eph, edad: $\mathbb{Z}$) : $\mathbb{Z}$ = $\frac{sumaIngresoXEdad(t,edad)}{cantidadEdad(t,edad)}$;
   fun sumaIngresoXEdad (t: eph, edad: $\mathbb{Z}$) : $\mathbb{Z}$ =
   $\sum_{i=0}^{|t|-1}$ if $t[i][@Edad] = edad\ \wedge\ t[i][@P47T] \geq 0$ then $t[i][@P47T]$ else 0 fi ;
   fun cantidadEdad (t: eph, edad: $\mathbb{Z}$) : $\mathbb{Z}$ = $\sum_{i=0}^{|t|-1}$ if $t[i][@Edad] = edad\ \wedge\ t[i][@P47T] \geq 0$ then 1 else 0 fi ;

3. **proc promedioIngresoProfesional**(**in** $t : eph$, **out** $result : \mathbb{Z}$).

   proc promedioIngresoProfesional (in t: eph, out result: $\mathbb{Z}$) {
   Pre $\{encuestaValida(t)\ \wedge_L\ cantProfesionales(t) > 0\}$
   Post $\{result = ingresoPromedioProfXHora(t)\}$
   }

4. **proc hogarDeMayorIngreso**(**in** $t : eph$, **out** $result : \mathbb{Z}$).

   proc hogarDeMayorIngreso (in t: eph, out result: $\mathbb{Z}$) {
   Pre $\{encuestaValida(t)\ \wedge\ |t| > 1\}$
   Post $\{esHogar(t, result) \wedge_L (\forall\, cod : \mathbb{Z})\ (esHogar(t, cod) \longrightarrow_L$
   $esMayorIngreso(t, cod, result)\ \vee\ igualIngresoMenorIndice(t, cod, result)\}$
   }

   pred esHogar (t: eph, c: $\mathbb{Z}$) $\{(\exists\, p : individuo)\ p \in t\ \wedge_L\ p[@Codusu] = c\}$
   pred esMayorIngreso (t: eph, cod: $\mathbb{Z}$, result: $\mathbb{Z}$) $\{ingresoHogar(t, result) > ingresoHogar(t, cod)\}$
   fun ingresoHogar (t: eph, cod: $\mathbb{Z}$) : $\mathbb{Z}$ = $\sum_{i=0}^{|t|-1}$ if $t[i][@Codusu] = cod\ \wedge\ noHuboAntes(t, i, cod)$ then $t[i][@ITF]$ else 0 fi ;
   pred noHuboAntes (t: eph, i: $\mathbb{Z}$, cod: $\mathbb{Z}$) $\{(\forall\, j : \mathbb{Z})\ 0 \leq j < i \longrightarrow_L t[j][@Codusu] \neq cod\}$
   pred igualIngresoMenorIndice (t: eph, cod: $\mathbb{Z}$, result: $\mathbb{Z}$) $\{ingresoHogar(t, result) = ingresoHogar(t, cod)\ \wedge$
   $indiceCodusu(t, result) \leq indiceCodusu(t, cod)\}$
   fun indiceCodusu (t: eph, cod: $\mathbb{Z}$) : $\mathbb{Z}$ = $\sum_{i=0}^{|t|-1}$ if $noHuboAntes(t, i, cod)$ then i else 0 fi ;

5. **proc mejorNoProfesional**(**in** $t : eph$, **out** $result : individuo$).
   //CIMAPP = con ingreso mayor al promedio profesional
   proc mejorNoProfesional (in t: eph, out result: $\mathbb{Z}$) {
   Pre $\{EncuestaValida(t)\ \wedge_L\ noProfesionalesCIMAPP(t) > 0\ \wedge_L\ cantProfesionales(t) > 0\}$
   Post $\{result \in t\ \wedge_L\ esNoProfesionalCIMAPP(t, result)\}$
   }

   fun noProfesionalesCIMAPP (t: eph) : $\mathbb{Z}$ = $\sum_{i=0}^{|t|-1}$ if $esNoProfesionalCIMAPP(t, i)$ then 1 else 0 fi ;
   pred esNoProfesionalCIMAPP (t: eph, p: individuo) $\{p[@Nivel\_Ed] = 0\ \wedge\ p[@Estado] = 1\ \wedge\ p[@PP3E\_Tot] > 0\ \wedge_L$
   $\frac{p[@P21]}{4 \times p[@PP3E\_Tot]} > ingresoPromedioProfXHora(t)\}$

6. **proc sigoEstudiando**(**in** $t : eph$, **out** $result$: Bool).

   proc sigoEstudiando (in t: eph, out result: Bool) {
   Pre $\{encuestaValida(t)\}$
   Post $\{res \leftrightarrow desocupadosNoUniversitarios(t) > desocupadosUniversitarios(t)\}$
   }

   fun desocupadosNoUniversitarios (t:eph) : $\mathbb{Z}$ = $\sum_{i=0}^{|t|-1}$ if $t[i][@Nivel\_Ed] = 0\ \wedge\ t[i][@Estado] = 2$ then 1 else fi ;
   fun desocupadosUniversitarios (t:eph) : $\mathbb{Z}$ = $\sum_{i=0}^{|t|-1}$ if $t[i][@Nivel\_Ed] = 1\ \wedge\ t[i][@Estado] = 2$ then 1 else fi ;

7. **proc empleadoDelAño**(**in** $t : eph$, **out** $result : individuo$).

   proc empleadoDelAño (in t: eph, out result: individuo) {
   Pre {$encuestaValida(t)\ \wedge_L\ hayEmpleadoDelAnio(t)$}
   Post {$esEmpleadoDelAnio(t, result)$}
   }

   pred hayEmpleadoDelAño (t: eph) {$(\exists p : individuo)\ esEmpleadoDelAnio(t, p)$}
   pred esEmpleadoDelAño (t: eph, p: individuo) {$p \in t\ \wedge_L\ p[@Cat\_Ocup] \neq 1\ \wedge\ (\exists pat : individuo)\ pat \in t\ \wedge_L\ p[@Cat\_Ocup] = 1\ \wedge\ pat[@P21] < p[@P21]$]}

8. **proc noTieneAQuienCuidar**(**in** $t : eph$, **out** $result$: Bool).

   proc noTieneAQuienCuidar (in t: eph, out result: Bool) {
   Pre {$encuestaValida(t)$}
   Post {$(\exists p : individuo)\ p \in t\ \wedge_L\ p[@Cat\_Ocup] = 4\ \wedge\ noTieneMenores(t, p)$}
   }

   pred noTieneMenores (t: eph, p: individuo) {$p[@IX\_TOT] - p[IX\_Mayeq10] = 0$}

9. **proc pareto**(**in** $t : eph$, **out** : Bool).

   proc pareto (in t: eph, out result: Bool) {
   Pre {$encuestaValida(t)$}
   Post {$(\exists s : seq\langle < individuo >\rangle)\ contenida(s, t)\ \wedge_L\ |s| = 0{,}2 * |t|\ \wedge_L\ esGrupoSelecto(t, s)$}
   }

   pred esGrupoSelecto (s: $seq\langle individuo\rangle$) {$sumaIngresos(s) > 0{,}8 * sumaIngresos(t)$}
   fun sumaIngresos (s: $seq\langle individuo\rangle$) : $\mathbb{Z}$ = $\sum_{i=0}^{|s|-1}$ if $s[i][@P47T] \geq 0$ then $s[i][@P47T]$ else 0 fi ;

10. **proc elDeMayorIncrementoInterAnual**(**in** $t1 : eph$, **in** $t2 : eph$, **out** $result : \mathbb{Z}$).
    Nota: Dado que estamos utilizando datos reales, se debieron relajar varias precondiciones que teníamos en el TPE, tal como que ambas encuestas tienen el mismo número de datos y los individuos estaban exactamente ordenados, etc.

    proc elDeMayorIncrementoInterAnual (in t1: eph, in t2: eph, out result: $\mathbb{Z}$) {
    Pre {$encuestaValida(t1)\ \wedge\ encuestaValida(t2)\ \wedge_L\ hayIndividuo(t1, t2)$}
    Post {$0 \leq result < |t2|\ \wedge_L\ (\exists p : individuo)\ p \in t1\ \wedge_L\ esParecido(p1, t2[result])$
    $\wedge_L\ esElDeMayorIncremento(t1, t2, p1, t2[result])$}
    }

    pred hayIndividuo (t1: eph, t2: eph) {$(\exists p1, p2 : individuo)\ seMantiene(t1, t2, p1, p2)$}
    pred seMantiene (t1: eph, t2: eph, p1: individuo, p2:individuo) {$p1 \in t1\ \wedge\ p2 \in t2\ \wedge_L$
    $esParecido(p1, p2) \wedge_L\ mismaActividad(p1, p2)\ \wedge\ p1[@P21] > 0\ \wedge p2[@P21] > 0$}
    pred esParecido (p1: individuo, p2:individuo) {$columnasIgual(p1, p2)\ \wedge\ columnasDistintas(p1, p2)$}
    pred columnasIgual (p1: individuo, p2:individuo) {$p1[@Componente] = p2[@Componente]\ \wedge\ p1[@Codusu] = p2[@Codusu]$}
    pred columnasDistintas (p1: individuo, p2: individuo) {$p1[@Ano4] < p2[@Ano4]\ \wedge\ (p1[@Nivel\_Ed] = 1 \longrightarrow p2[@Nivel\_Ed] \neq 0)\ \wedge\ pasanLosAnios(p1, p2)$}
    pred esElDeMayorIncremento (t1: eph, t2: eph, p1: individuo, p2: individuo) {$(\forall q1 : individuo, q2 : individuo)$
    $seMantiene(t1, t2, p1, p2) \longrightarrow_L q2[@P21] - q1[@P21] \leq p2[@P21] - p1[@P21]$}
    pred mismaActividad (p1: individuo, p2: individuo) {$p1[@PP04D\_Cod] = p2[@PP04D\_Cod]$}
    pred pasanLosAnios (p1: individuo, p2: individuo) {$p1[@Edad] \leq p2[@Edad] \leq p1[@Edad] + p2[@Anio] - p1[@Ano4]$}

11. **proc mejorQueLaInflacion**(**in** $t1 : eph$, **in** $t2 : eph$, **in** $infl : \mathbb{R}$, **out** $result : seq\langle < \mathbb{Z}, \mathbb{R} >\rangle$).

    proc mejorQueLaInflacion (in t1: eph, in t2: eph, in infl: $\mathbb{R}$, out result: $seq\langle < \mathbb{Z}, \mathbb{R} >\rangle$) {
    Pre {$encuestaValida(t1)\ \wedge\ encuestaValida(t2)$}
    Post {$ordenadaDescTupla(result)\ \wedge\ (\forall k :< \mathbb{Z}, \mathbb{R} >)\ k \in result \leftrightarrow$
    $(esCodOcupacion(t1, t2, k_0)\ \wedge_L\ inflacion(t1, t2, k_0) > infl\ \wedge\ inflacion(t1, t2, k_0) = k_1$}
) }

    pred ordenadaDescTupla (s: $seq\langle < \mathbb{Z}, \mathbb{R} >\rangle$) {$(\forall i : \mathbb{Z})\ 0 \leq i < |s| - 1 \longrightarrow_L s[i]_0 \geq s[i+1]_0$}
    pred esCodOcupacion (t1: eph, t2: eph, c: $\mathbb{Z}$) {$c \geq 0\ \wedge\ (\exists p1 : individuo)\ p1 \in t1\ \wedge_L\ p1[@PP04D\_Cod] = c\ \wedge\ t[i][@P21] \geq 0\ \wedge\ (\exists p2 : individuo)\ p2 \in t2\ \wedge_L\ p2[@PP04D\_Cod] = c\ \wedge\ t[i][@P21] > 0$}
    fun inflacion (t1: eph, t2: eph, c: $\mathbb{Z}$) : $\mathbb{Z}$ = if $promedioPaga(t1, c) > 0$ then $\frac{promedioPaga(t2,c)}{promedioPaga(t1,c)} - 1$ else 0 fi ;
    fun promedioPaga (t: eph, c: $\mathbb{Z}$) : $\mathbb{Z}$ = $\sum_{i=0}^{|t|-1}$ if $t[i][@PP04D\_Cod] = c\ \wedge\ t[i][@P21] > 0$ then $\frac{paga(t,i)}{cantCodOcup(t,c)}$ else 0 fi ;
    fun paga (t: eph, i: $\mathbb{Z}$) : $\mathbb{Z}$ = $t[i][@P21]$;
    fun cantCodOcup (t: eph, c: $\mathbb{Z}$) : $\mathbb{Z}$ = $\sum_{i=0}^{|t|-1}$ if $t[i][@PP04D\_Cod] = c\ \wedge\ t[i][@P21] > 0$ then 1 else 0 fi ;

12. **proc ordenar**(**inout** $t : eph$).

    `proc ordenar` (inout t: eph) {
        `Pre` $\{t = t_0 \ \wedge \ encuestaValida(t)\}$
        `Post` $\{mismosIndividuos(t_0, t) \ \wedge_L \ estaOrdenada(t)\}$
    }

    `pred mismosIndividuos` (t1: eph, t2: eph) $\{|t1| = |t2| \wedge_L contenida(t1, t2) \wedge contenida(t2, t1)\}$

13. **proc agregarOrdenado**(**inout** $t : eph$, **in** $p : individuo$).

    `proc agregarOrdenado` (inout t: eph, in p: individuo) {
        `Pre` $\{t = t_0 \ \wedge \ encuestaValida(t) \ \wedge \ individuoValido(p)\}$
        `Post` $\{p \in t_0 \longrightarrow t = t_0 \ \wedge$
            $p \notin t_0 \longrightarrow estaAgregado(t, t_0, p) \ \wedge \ estaOrdenada(t)\}$
    }

    `pred estaAgregado` (t: eph, t0: eph, p: individuo) $\{|t| = |t0| + 1 \ \wedge_L \ (\exists j : \mathbb{Z}) \ 0 \leq j < |t| \ \wedge_L \ subseq(t, 0, j) = subseq(t0, 0, j) \ \wedge$ $subseq(t, j + 1, |t|) = subseq(t0, j, |t0|) \ \wedge \ t[j] = p\}$

14. **proc quitar**(**inout** $t : eph$, **in** $r : individuo$).

    `proc quitar` (inout t: eph, in p: individuo) {
        `Pre` $\{t = t_0 \ \wedge \ encuestaValida(t) \ \wedge \ individuoValido(p)\}$
        `Post` $\{p \notin t_0 \longrightarrow t = t_0$
            $p \in t_0 \longrightarrow noEsta(t, t_0, p) \ \wedge \ estaOrdenada(t)\}$
    }

    `pred noEsta` (t: eph, t0: eph, p: individuo) $\{estaAgregado(t0, t, p)\}$

4.3. Predicados y Auxiliares comunes

`pred encuestaValida` (t: eph) $\{|t| > 0 \ \wedge_L \ esMatriz(t) \ \wedge_L individuosValidos(t) \wedge_L$ $individuosDistintos(t) \ \wedge \ mismoAnio(t) \ \wedge \ hogaresCoherentes(t)\}$

`pred esMatriz` (t: eph) $\{(\forall i, j : \mathbb{Z})(0 \leq i < |t| \ \wedge \ 0 \leq j < |t| \wedge) \longrightarrow_L |t[i]| = |t[j]|\}$

`pred individuosValidos` (t: eph) $\{(\forall p : individuo) \ p \in t \longrightarrow_L individuoValido(p)\}$

`pred individuoValido` (p: individuo) $\{|p| = cantidadItems() \ \wedge_L \ p[@Codusu] > 0 \ \wedge \ p[@Componente] > 0 \ \wedge \ p[@Ano4] > 0 \ \wedge$ $nivelEdEnRango(p[@Nivel\_Ed]) \wedge estadoEnRango(p[@Estado]) \wedge catOcupEnRango(p[@Cat\_Ocup]) \wedge edadEnRango(p[@Edad]) \wedge$ $p[@PP3E\_Tot] \geq -1 \ \wedge \ p[@Itf] \geq 0 \ \wedge \ p21EnRango(p[@P21]) \ \wedge \ p47EnRango(p[@P47T]) \ \wedge \ p[@PP04D\_Cod] \geq -1 \ \wedge$ $catInacEnRango(p[@Cat\_Inac]) \ \wedge \ p[@IX\_Mayeq10] > 0 \ \wedge \ p[IX\_Tot] > 0 \ \wedge \ adultosMenoresATotal(p) \ \wedge \ \neg trabaja(p) \longrightarrow$ $p[@P21] = 0 \ \wedge \ \neg trabaja(p) \Leftrightarrow p[@PP3E\_Tot] = -1 \ \wedge \ p[@PP3E\_Tot] = -1 \Leftrightarrow p[@PP04D\_Cod] = -1\}$

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

`pred estadoEnRango` (i: $\mathbb{Z}$) $\{1 \leq i \leq 4 \ \vee \ noSabeNoContesta(i)\}$

`pred catOcupEnRango` (i: $\mathbb{Z}$) $\{0 \leq i \leq 4 \ \vee \ noSabeNoContesta(i)\}$

`pred edadEnRango` (i: $\mathbb{Z}$) $\{0 \leq i \leq 110\}$

`pred catInacEnRango` (i: $\mathbb{Z}$) $\{0 \leq i \leq 7\}$

`pred p21EnRango` (i: $\mathbb{Z}$) $\{i = -9 \ \vee \ i \geq 0\}$

`pred p47EnRango` (i: $\mathbb{Z}$) $\{i = -9 \ \vee \ i = -1 \ \vee \ i \geq 0\}$

`pred noSabeNoContesta` (i: $\mathbb{Z}$) $\{i = 9 \ \vee \ i = 99 \ \vee \ i = 999 \ \vee \ i = 9999\}$

`pred adultosMenoresATotal` (p: individuo) $\{p[@IX\_Mayeq10] \leq p[IX\_Tot]\}$

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

`pred individuosDistintos` (t: eph) $\{(\forall p1, p2 : individuo) \ p1 \in t \wedge p2 \in t \wedge p1 \neq p2 \longrightarrow_L p1[@Codusu] \neq p2[@Codusu] \ \vee$ $p1[@Componente] \neq p2[@Componente]\}$

`pred mismoAnio` (t: eph) $\{(\forall p1, p2 : individuo) \ p1 \in t \ \wedge \ p2 \in t \longrightarrow_L p1[@Ano4] = p2[@Ano4]\}$

`pred hogaresCoherentes` (t: eph) $\{(\forall p1, p2 : individuo) \ p1 \in t \ \wedge \ p2 \in t \ \wedge \ mismoHogar(p1, p2) \longrightarrow_L mismoITF(p1, p2) \ \wedge$ $mismaCantidadMiembros(p1, p2) \ \wedge \ mismaCantidadAdultos(p1, p2)\}$

`pred mismoHogar` (p1: individuo, p2: individuo) $\{p1[@Codusu] = p2[@Codusu]\}$

`pred mismoITF` (p1: individuo, p2: individuo) $\{p1[@Itf] = p2[@Itf]\}$

`pred mismaCantidadMiembros` (p1: individuo, p2: individuo) $\{p1[@IX\_Tot] = p2[@IX\_Tot]\}$

`pred mismaCantidadAdultos` (p1: individuo, p2: individuo) $\{p1[@IX\_Mayeq10] = p2[@IX\_Mayeq10]\}$

`pred esProfesional` (p: individuo) $\{t[i][@Nivel\_Ed] = 1 \ \wedge \ p[@Estado] = 1\}$

`fun cantProfesionales` (t: eph) : $\mathbb{Z} = \sum_{i=0}^{|t|-1}$ if $esProfesionalConIngresos(t[i])$ then 1 else 0 fi ;

`pred esProfesionalConIngresos` (p: individuo) $\{esProfesional(p) \ \wedge \ declaroIngresosYHoras(p)\}$

`pred declaroIngresosYHoras` (p: individuo) $\{p[P21] > 0 \ \wedge \ p[PP3E\_Tot] > 0\}$

`fun ingresoPromedioProfXHora` (t: eph) : $\mathbb{Z} = \frac{\sum_{i=0}^{|t|-1} \text{ if } esProfesionalConIngresos(t[i]) \text{ then } \frac{t[i][@P21]}{4 \times t[i][@PP3E\_Tot]} \text{ else } 0 \text{ fi}}{cantProfesionales(t)}$;

`pred contenida` (s: $seq\langle < individuo > \rangle$, t: eph) $\{(\forall p : individuo) \ p \in s \longrightarrow_L p \in t\}$

`pred mayorCodusu` (p1: individuo, p2: individuo) $\{p1[@Codusu] > p2[@Codusu]\}$

`pred igualCodusuMayorEdad` (p1: individuo, p2: individuo) $\{p[@Codusu] = p[@Codusu] \ \wedge \ p[@Edad] \geq p[@Edad]\}$

`pred estaOrdenada` (t: eph) $\{(\forall i : \mathbb{Z}) \ 0 \leq i < |t| - 1 \longrightarrow_L mayorCodusu(t[i], t[i + 1]) \ \vee \ igualCodusuMayorEdad(t[i], t[i + 1])\}$