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

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2021

TPI: Análisis Habitacional Argentino

Entrega: 12 de Noviembre

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.

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.cpp`: Aquí es donde van a volcar sus implementaciones.
- `ejercicios.h`: *headers* de las funciones que tienen que implementar.
- `auxiliares.cpp` y `auxiliares.h`: Donde es posible volcar funciones auxiliares.
- `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. Para ello recomendamos:
  1. Lanzar el CLION.
  2. Cerrar el proyecto si hubiese uno abierto por *default*: `File->Close Project`
  3. En la ventana de Bienvenida de CLION, seleccionar `Open Project`
  4. Seleccionar la carpeta del proyecto `src`.
  5. Si es necesario, cargar el CMakeList.txt nuevamente mediante `Tools->CMake->Reload CMake Project`
  6. No olvidarse descomprimir el GTEST.

Es importante recalcar que la especificación de los ejercicios elaborada por la materia es la guía sobre la que debe basarse el equipo a la hora de implementar los problemas.

3. Entregable

La fecha de entrega del TPI es el **12 de NOVIEMBRE de 2021**.

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 subirse en un archivo comprimido en la solapa Trabajos Prácticos en la tarea SUBIR TPI.
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. Funciones C++

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

```
bool esEncuestaValida (eph_h th, eph_i ti);
vector <int> histHabitacional (eph_h th, eph_i ti, int region);
vector<pair<int,float>> laCasaEstaQuedandoChica (eph_h th, eph_i ti);
bool creceElTeleworkingEnCiudadesGrandes (eph_h t1h, eph_i t1i, eph_h t2h, eph_i t2i);
int costoSubsidioMejora(eph_h th, eph_i ti, int monto);
join_hi generarJoin(eph_h th, eph_i ti);
void ordenarRegionYCodusu (eph_h & th, eph_i & ti);
vector<hogar> muestraHomogenea(eph_h& th, eph_i& ti);
void corregirRegion(eph_h & th, eph_i ti);
vector<int> histogramaDeAnillosConcentricos(eph_h th, eph_i ti,pair<int,int> centro,
                                            vector<int> distancias);
pair<eph_h,eph_i> quitarIndividuos(eph_i & ti,eph_h & th,vector<pair<int,dato>> busqueda);
```

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 pair < hogar, individuo > par_hi;
typedef vector < par_hi > join_hi;
```

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);
```

Uso de pair

Vamos a utilizar el container pair que pertenece a la librería estándar del C++ y está definido en el header **utility**. Este es un container que puede poner juntos dos elementos de cualquier tipo. En nuestro caso, vamos a utilizarlo para dos valores enteros.

Para asignar u acceder al primero se utiliza la propiedad *first*, y para el otro... *second*. En el siguiente ejemplo[1], veremos varias maneras de declarar, asignar e imprimir los valores de containers pair.

```
#include <iostream>
#include <utility>
using namespace std;

int main()
{
    pair <int, char> PAIR1 ;
    pair <string, double> PAIR2 ("GeeksForGeeks", 1.23) ;
    pair <string, double> PAIR3 ;
```

[1] https://www.geeksforgeeks.org/pair-in-cpp-stl/

```
    PAIR1.first = 100;
    PAIR1.second = 'G' ;

    PAIR3 = make_pair ("GeeksForGeeks␣is␣Best",4.56);

    cout << PAIR1.first << "␣" ; // espacio en blanco para separar los elementos
    cout << PAIR1.second << endl ;

    cout << PAIR2.first << "␣" ;
    cout << PAIR2.second << endl ;

    cout << PAIR3.first << "␣" ;
    cout << PAIR3.second << endl ;

    return 0;
}
```

5. Especificación

En esta sección se encuentra la Especificiación de los ejercicios a resolver, a partir del enunciado del TPE. A continuación, repetimos el detalle del contenido de las tablas de Individuos y Hogares.

5.1. Tabla HOGARES

- HOGCODUSU: Identificador o clave ($\mathbb{N}$) del hogar. Además permite hacer el seguimiento a través de los trimestres.
- HOGAÑO: Año de relevamiento.
- HOGTRIMESTRE: Trimestre del año de relevamiento.
- HOGLATITUD: Latitud (geolocalización del hogar).
- HOGLONGITUD: Longitud (geolocalización del hogar).
- II7: Régimen de tenencia de los habitantes:
  - 1 - Propietario
  - 2 - Inquilino
  - 3 - Ocupante
- REGION: Código de Región:
  - 1 - Gran Buenos Aires
  - 40 - NOA
  - 41 - NEA
  - 42 - Cuyo
  - 43 - Pampeana
  - 44 - Patagonia
- MAS_500: El hogar se ubica dentro de un aglomerados de más de 500.000 habitantes:
  - 0 - NO
  - 1 - SI
- IV1: Tipo de hogar (por observación):
  - 1 - Casa
  - 2 - Departamento
  - 3 - Pieza de inquilinato
  - 4 - Pieza en hotel/pensión
  - 5 - Local no construido para habitación
- IV2: Cantidad total de ambientes o habitaciones (sin contar baño/s, cocina, pasillo/s, lavadero, garage).

- II2: De esos, ¿cuántos usan habitualmente para dormir?
- II3: ¿Utiliza alguno exclusivamente como lugar de trabajo (para consultorio, estudio, taller, negocio, etc.)?
  - 1 - Si
  - 2 - No

5.2. Tabla PERSONAS

- INDCODUSU: Identificador único o clave ($\mathbb{N}$) del hogar. Se corresponde a un HOGCODUSU de la tabla hogares. Además permite hacer el seguimiento a traés de los trimestres.
- COMPONENTE: Número de orden que identifica a una persona dentro de un hogar.
- INDAÑO: Año de relevamiento.
- INDTRIMESTRE: Trimestre del año de relevamiento.
- CH4: Género:
  - 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
  - -1 - No informado
- 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
- p47T: Monto de ingreso total individual. Puede ser -1 si no fue informado.
- PP04G: Dónde realiza principalmente sus tareas:
  - 0 - Sin Dato
  - 1 - En un local / oficina / establecimiento negocio / taller / chacra / finca
  - 2 - En puesto o kiosco fijo callejero
  - 3 - En vehículos: bicicleta / moto / autos / barcos / botes (no incluye servicio de transporte)
  - 4 - En vehículo para transporte de personas y mercaderías-aéreos, marítimo, terrestre (incluye taxis, colectivos, camiones, furgones, transporte de combustible, mudanzas, etc.)
  - 5 - En obras en construcción, de infraestructura, minería o similares
  - 6 - En este hogar
  - 7 - En el hogar del socio o del patrón
  - 8 - En el domicilio / local de los clientes
  - 9 - En la calle / espacios públicos / ambulante / de casa en casa / puesto móvil callejero
  - 10 - En otro lugar

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.

5.3. Tipos:

type $dato = \mathbb{Z}$

type $individuo = seq\langle dato \rangle$

type $hogar = seq\langle dato \rangle$

type $eph_i = seq\langle individuo \rangle$

type $eph_h = seq\langle hogar \rangle$

type $joinHI = seq\langle hogar \times individuo \rangle$

Tanto *individuo* como *hogar* se pueden pensar como una línea en las matrices eph_i y eph_h.

5.4. Acceso a las columnas

aux `cantidadItemsIndividuo` : $\mathbb{Z} = 11$;
aux `cantidadItemsHogar` : $\mathbb{Z} = 12$;
aux `@IndCodusu` : $\mathbb{Z} = ord(INDCODUSU)$;
aux `@HogCodusu` : $\mathbb{Z} = ord(HOGCODUSU)$;
aux `@IndAño` : $\mathbb{Z} = ord(INDAÑO)$;
aux `@IndTrim` : $\mathbb{Z} = ord(INDTRIMESTRE)$;
aux `@HogAño` : $\mathbb{Z} = ord(HOGANO)$;
aux `@HogTrim` : $\mathbb{Z} = ord(HOGTRIMESTRE)$;
aux `@Componente` : $\mathbb{Z} = ord(COMPONENTE)$;
aux `@Nivel_Ed` : $\mathbb{Z} = ord(NIVEL\_ED)$;
aux `@Estado` : $\mathbb{Z} = ord(ESTADO)$;
aux `@Cat_Ocup` : $\mathbb{Z} = ord(CAT\_OCUP)$;
aux `@Edad` : $\mathbb{Z} = ord(CH6)$;
aux `@Genero` : $\mathbb{Z} = ord(CH4)$;
aux `@IngresoTot` : $\mathbb{Z} = ord(p47T)$;
aux `@LugarTrabajo` : $\mathbb{Z} = ord(PP04G)$;
aux `@Tenencia` : $\mathbb{Z} = ord(II7)$;
aux `@Region` : $\mathbb{Z} = ord(REGION)$;
aux `@+500k` : $\mathbb{Z} = ord(MAS\_500)$;
aux `@Tipo` : $\mathbb{Z} = ord(IV1)$;
aux `@qHabitaciones` : $\mathbb{Z} = ord(IV2)$;
aux `@qDormitorios` : $\mathbb{Z} = ord(II2)$;
aux `@trabajaHogar` : $\mathbb{Z} = ord(II3)$;
aux `@Latitud` : $\mathbb{Z} = ord(HOGLATITUD)$;
aux `@Longitud` : $\mathbb{Z} = ord(HOGLONGITUD)$;

5.5. Problemas

1. **proc encuestaVálida**(**in** $th : eph_h$, **in** $ti : eph_i$, **out** res : Bool).
   El procedimiento devuelve verdadero si se verifica:
   - Que th y ti son matrices, es decir, que en el interior de cada una, todos sus vectores tienen la misma longitud
   - Que existe al menos un hogar en th y un individuo en ti
   - Que la cantidad de columnas (tamaño de los vectores) es igual a la cantidad variables de la tabla (o de enumerados de **Item**)
   - Que los hogares tienen individuos asociados y viceversa, es decir, que no hay individuos sin hogares ni hogares sin individuos
   - Que no hay individuos ni hogares repetidos
   - Que el año y trimestre de relevamiento es el mismo para todos los registros
   - Que la cantidad de miembros del hogar es menor o igual a 20
   - Que el atributo `IV2` es mayor o igual al atributo `II2`
   - Que todos los atributos categóricos tienen valores en el rango esperado. Por ejemplo, el atributo `REGION` sólo debería tener valores entre 1 y 6 inclusive

```
proc encuestaVálida (in th: eph_h, in ti: eph_i, out res: Bool) {
    Pre {True}
    Post {res = true ↔ esVálida(th, ti)}
}
```

2. **proc histHabitacional**(**in** $th : eph_h$, **in** $ti : eph_i$, **in** $region : \mathbb{Z}$, **out** $res : seq\langle\mathbb{Z}\rangle$) .
Dada una encuesta válida, se desea construir el histograma habitacional correspondiente a una región dada como parámetro de entrada. El histograma se representa con una secuencia de enteros **res** donde la i-ésima posición contiene la cantidad de hogares de tipo casa con i habitaciones en la `región` recibida por parámetro. El largo de la secuencia de salida depende de la máxima cantidad de habitaciones en la región.

```
proc histHabitacional (in th: eph_h, in ti: eph_i, in region: Z, out res: seq⟨Z⟩) {
    Pre {esVálida(th, ti) ∧ valorRegionValido(region)}
    Post {longitudIgualAMáximaCantidadHabitaciones(th, region, res) ∧
          (∀i : Z) (0 ≤ i < |res| ⟶_L res[i] = cantHogaresCasaConNHabitaciones(th, region, i + 1))}
}
pred longitudIgualAMáximaCantidadHabitaciones (th: eph_h, region: Z, lista: seq⟨Z⟩) {
    (∀h : hogar) ((h ∈ th ∧_L h[@Region] = region) ⟶_L |lista| ≥ h[@qHabitaciones])
    ∧ (∃h : hogar) (h ∈ th ∧_L h[@Region] = region ∧ |lista| = h[@qHabitaciones])
}
aux cantHogaresCasaConNHabitaciones ( th: eph_h, region: Z, habitaciones: Z) : Z =
    Σ_{i=0}^{|th|-1} (if esCasa(th[i]) ∧ th[i][@qHabitaciones] = habitaciones ∧ th[i][@Region] = region then 1 else 0 fi) ;
```

3. **proc laCasaEstaQuedandoChica**(**in** $th : eph_h$, **in** $ti : eph_i$, **out** $res : seq\langle\mathbb{Z} \times \mathbb{R}\rangle$) .
Dada una encuesta válida, se pide calcular por cada una de las 6 regiones argentinas, la proporción de hogares tipo casa con hacinamiento crítico (HC). Los hogares con hacinamiento crítico son aquellos en los cuales hay en promedio más de tres personas por cuarto. Las casas deben estar ubicadas en aglomeraciones de menos de 500.000 habitantes. Agrupar los cálculos por región en una secuencia ordenada de acuerdo al código de la columna `REGION`.

```
proc laCasaEstaQuedandoChica (in th: eph_h, in ti: eph_i, out res: seq⟨Z × R⟩) {
    Pre {esValida(th, ti)}
    Post {|res| = 6 ∧ sonLasProporcionesDeCasasConHCPorRegion(res, th, ti) ∧
    estanOrdenadasPorRegion(res)}
}
pred sonLasProporcionesDeCasasConHCPorRegion (res: seq⟨Z × R⟩, th: eph_h, ti: eph_i) {
    (∀i : Z) (0 ≤ i < |res| ⟶_L
    valorRegionValido(res[i]_0) ∧ res[i]_1 = proporcionDeCasasConHC(th, ti, res[i]_0))
}
aux proporcionDeCasasConHC (th: eph_h, ti: eph_i, region: Z) : R =
    if cantHogaresValidos(th, region) > 0
     then cantHogaresValidosConHC(th, ti, region)/cantHogaresValidos(th, region) else 0 fi ;
pred esHogarValido (h: hogar, region: Z) {
    esCasa(h) ∧ h[@Region] = region ∧ h[@ + 500k] = 0
}
aux cantHogaresValidosConHC (th: eph_h, ti: eph_i, region: Z) : Z =
    Σ_{k=0}^{|th|-1} if esHogarValido(th[k], region) ∧ hogarConHacinamientoCritico(th[k], ti) then 1 else 0 fi ;
aux cantHogaresValidos (th: eph_h, region: Z) : Z =
    Σ_{k=0}^{|th|-1} if esHogarValido(th[k], region)] = 0 then 1 else 0 fi ;
pred hogarConHacinamientoCritico (h: hogar, ti: eph_i) {
    cantHabitantes(h, ti) > 3 * h[@qDormitorios]
}
pred ordenadasPorRegion (res: seq⟨Z × R⟩) {
```

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

}

4. **proc creceElTeleworkingEnCiudadesGrandes**(**in** $t1h : eph_h$, **in** $t1i : eph_i$, **in** $t2h : eph_h$, **in** $t2i : eph_i$, **out** res : Bool)
Dadas dos encuestas válidas, detectar si hay un incremento (en proporción) interanual del Teleworking en ciudades de más de 500.000 habitantes. Para ello calcular la proporción de individuos que realizan sus tareas laborales en su hogar (`PP04G`), entre dos encuestas de años diferentes, pero del mismo trimestre. Específicamente, t1h y t1i son anteriores a t2h y t2i. Verificar solo para hogares tipo casas o departamentos. Para simplificar el análisis, se considerarán como haciendo Teleworking, los hogares que tengan ambientes reservados para el trabajo.

`proc creceElTeleworkingEnCiudadesGrandes` (in t1h: eph_h, in t1i: eph_i, in t2h: eph_h, in t2i: eph_i, out res: Bool) {

`Pre` $\{esVálida(t1h, t1i) \land esVálida(t2h, t2i) \land_L año(t1i) < año(t2i) \land trimestre(t1i) = trimestre(t2i)\}$

`Post` $\{res = true \leftrightarrow proporcionTeleworking(t2h, t2i) > proporcionTeleworking(t1h, t1i)\}$

}

`aux proporcionTeleworking` (th: eph_h, ti: eph_i) : $\mathbb{R}$ =
if $cantIndividuosQueTrabajan(th, ti) > 0$
then $cantIndividuosTrabajandoEnSuVivienda(th, ti)/cantIndividuosQueTrabajan(th, ti)$ else 0 fi ;

`aux cantIndividuosTrabajandoEnSuVivienda` (th: eph_h, ti:eph_i) : $\mathbb{Z}$ =
$\sum_{k=0}^{|ti|-1}$ if $trabaja(ti[k]) \land\ trabajaEnSuVivienda(ti[k], th) \land$
$individuoEnHogarValido(ti[k], th)$ then 1 else 0 fi ;

`aux cantIndividuosQueTrabajan` (th: eph_h, ti : eph_i) : $\mathbb{Z}$ =
$\sum_{k=0}^{|ti|-1}$ if $trabaja(ti[k]) \land individuoEnHogarValido(ti[k], th)$ then 1 else 0 fi ;

`pred trabajaEnSuVivienda` (i: individuo, th: eph_h) {

$realizaSusTareasEnEsteHogar(i) \land suHogarTieneEspaciosReservadosParaElTrabajo(i, th)$

}

`pred individuoEnHogarValido` (i: individuo, th: eph_h) {

$esDeCiudadGrande(i, th) \land suHogarEsCasaODepartamento(i, th)$

}

`pred trabaja` (i: individuo) {

$i[@ESTADO] = 1$

}

`aux año` (ti: eph_i) : $\mathbb{Z} = ti[0][@IndAño]$;

`aux trimestre` (ti: eph_i) : $\mathbb{Z} = ti[0][@IndTrimestre]$;

`pred esDeCiudadGrande` (i: individuo, th: eph_h) {

$(\exists h : hogar)\ (h \in th \land_L esSuHogar(h, i) \land h[@ + 500k] = 1)$

}

`pred suHogarTieneEspaciosReservadosParaElTrabajo` (i: individuo, th: eph_h) {

$(\exists h : hogar)\ (h \in th \land_L esSuHogar(h, i) \land tieneEspaciosReservadosParaElTrabajo(h))$

}

`pred suHogarEsCasaODepartamento` (i: individuo, th: eph_h) {

$(\exists h : hogar)\ (h \in th \land_L esSuHogar(h, i) \land esCasaODepartamento(h))$

}

`pred esCasaODepartamento` (h: hogar) {

$h[@Tipo] = 1 \lor h[@Tipo] = 2$

}

`pred realizaSusTareasEnEsteHogar` (i: individuo) {

$i[@LugarTrabajo] = 6$

}

pred tieneEspaciosReservadosParaElTrabajo (h: hogar) {

$h[@trabajaHogar] = 1$

}

5. **proc costoSubsidioMejora**(**inout** $th : eph_h$, **in** $ti : eph_i$, **in** $monto : \mathbb{Z}$, **out** $res : \mathbb{Z}$) .
Dada una encuesta válida, y un monto de subsidio se desea calcular el costo total de implementar un subsidio de mejora habitacional para aquellos hogares que sean casas de tenencia propia y cuya cantidad de habitaciones que utilizan para dormir (atributo II2) sea estrictamente inferior a la cantidad de habitantes menos dos.

proc costoSubsidioMejora (in th: eph_h, in ti: eph_i, in monto: $\mathbb{Z}$, out res: $\mathbb{Z}$) {

Pre $\{esVálida(th, ti) \land monto > 0\}$

Post $\{res = \sum_{k=0}^{|th|-1}$ if $tieneCasaPropia(th[k]) \land tieneCasaChica(th[k], ti)$ then $monto$ else 0 fi$\}$

}

pred tieneCasaPropia (h: hogar) {

$h[@II7] = 1$

}

pred tieneCasaChica (h: hogar, ti: eph_i) {

$cantHabitantes(h, ti) - 2 > h[@II2]$

}

6. **proc generarJoin**(**inout** $th : eph_h$, **in** $ti : eph_i$, **out** $junta : joinHI$) .
Dada una encuesta válida, devuelve la combinación de las tables de hogares e individuos generando la tabla *junta* tal que cada fila es una 2-upla de la cual el primer elemento es de tipo hogar y el segundo de tipo individuo que satisfacen la condición que el valor en HOGCODUSU del primero es igual al valor de INDCODUSU del segundo.

proc generarJoin (in th: eph_h, in ti: eph_i, out junta: joinHI) {

Pre $\{esVálida(th, ti)\}$

Post $\{todosEstanEnElJoin(th, ti, junta) \land noSobraNinguno(th, ti, junta) \land sinRepetidos(junta)\}$

}

pred todosEstanEnElJoin (th: eph_h, ti: eph_i, junta: joinHI) {

$(\forall h : hogar)\ ((\forall i : individuo)\ (h \in th \land i \in ti \land h[@HOGCODUSU] = i[@INDCODUSU] \longrightarrow (h, i) \in junta)))$

}

pred noSobraNinguno (th: eph_h, ti: eph_i, junta: joinHI) {

$(\forall t : hogar \times individuo)\ (t \in junta \longrightarrow (t_0[@HOGCODUSU] = t_1[@INDCODUSU] \land t_0 \in th \land t_1 \in ti)))$

}

pred sinRepetidos (junta: joinHI) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |junta| \longrightarrow_L \#apariciones(junta, junta[i]) = 1)$

}

7. **proc ordenarRegionYCODUSU**(**inout** $th : eph_h$, **inout** $ti : eph_i$) .
Dada una encuesta válida, ordenar la encuesta de hogares `th` de acuerdo a: 1) El código de región: todos los del gran buenos aires primero, luego los de NOA y así sucesivamente (siguiendo el orden dado por el número de categoría). 2) Dentro de cada región, ordenar de forma creciente por `CODUSU`.

Además ordenar la encuesta individuos `ti`, según: 1) El `CODUSU` de th luego de realizar el ordenamiento, 2) Dentro del mismo hogar, ordenar por `COMPONENTE` de menor a mayor.

proc ordenarRegionYCODUSU (inout th: eph_h, inout ti: eph_i) {

Pre $\{esVálida(th, ti) \land th0 = th \land ti0 = ti\}$

Post $\{|th0| = |th| \land |ti0| = |ti|$
$\land\ (\forall h : hogar)\ (hogarEnTabla(h, th) \leftrightarrow hogarEnTabla(h, th0))$
$\land\ (\forall i : individuo)\ (individuoEnTabla(i, ti) \leftrightarrow individuoEnTabla(i, ti0))$

$\wedge_L estanOrdenadosPorRegionYCodusu(th) \wedge estanOrdenadosPorCodusuDeHogarYComponente(ti, th)\}$

}

`pred estanOrdenadosPorRegionYCodusu` (th: eph_h) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |th| \longrightarrow_L (\forall j : \mathbb{Z})\ (i < j < |th| \longrightarrow_L$
$(th[i][@Region] < th[j][@Region]$
$\vee\ (th[i][@Region] = th[j][@Region] \wedge th[i][@HogCodusu] < th[j][@HogCodusu]))))$

}

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

$(\forall i : \mathbb{Z})\ (0 \leq i < |ti| \longrightarrow_L (\forall j : \mathbb{Z})\ (i < j < |ti| \longrightarrow_L$
$(suHogarEstaAntes(ti[i], ti[j], th) \vee$
$(vivenJuntos(ti[i], ti[j]) \wedge ti[i][@Componente] < ti[j][@Componente]))))$

}

`pred suHogarEstaAntes` (i1: individuo, i2: individuo, th: eph_h) {

$(\exists h1 : hogar)\ ((\exists h2 : hogar)\ (hogarEnTabla(h1, th) \wedge hogarEnTabla(h2, th) \wedge$
$h1[@HogCodusu] = i1[@IndCodusu] \wedge h2[@HogCodusu] = i2[@IndCodusu] \wedge$
$hogarEstaAntes(h1, h2, th)))$

}

`pred hogarEstaAntes` (h1: hogar, h2: hogar, th: eph_h) {

$(\exists i : \mathbb{Z})\ (0 \leq i < |th| \wedge_L ((\exists j : \mathbb{Z})\ (i < j < |th| \wedge_L (th[i] = h1 \wedge th[j] = h2))$

}

`pred vivenJuntos` (i1: individuo, i2: individuo) {

$i1[@IndCodusu] = i2[@IndCodusu]$

}

8. **proc muestraHomogenea**(**in** $th : eph_h$, **in** $ti : eph_i$, **out** $res : seq\langle hogar\rangle$) .
Dada una encuesta válida, encontrar una secuencia de hogares lo más larga posible tal que la diferencia de ingresos totales sea la misma para cada par de hogares consecutivos. Debe estar ordenada de menor a mayor por cantidad de ingresos. De no encontrarse una secuencia de al menos 3 elementos, devolver una secuencia vacía.

`proc muestraHomogenea` (in th: eph_h, in ti: eph_i, out res: $seq\langle hogar\rangle$) {

`Pre` $\{esVálida(th, ti)\}$

`Post` $\{(|res| = 0 \wedge \neg existeSolucionMuestraHomogeneaConAlMenos3(th, ti)) \vee$
$(|res| \geq 3 \wedge\ esSolucionMuestraHomogenea(res, th, ti) \wedge \neg hayMejorSolucionMuestraHomogenea(res, th, ti))\}$

}

`pred existeSolucionMuestraHomogeneaConAlMenos3` (th: eph_h, ti: eph_i) {

$(\exists\ muestra : seq\langle hogar\rangle)\ (esSolucionMuestraHomogenea(muestra, th, ti) \wedge |muestra| \geq 3)$

}

`pred esSolucionMuestraHomogenea` (mh: $seq\langle hogar\rangle$, th: eph_h, ti: eph_i) {

$esMuestraDeHogares(mh, th) \wedge_L ordenadaPorIngresosConMismaDiferencia(mh, th, ti)$

}

`pred esMuestraDeHogares` (mh: $seq\langle hogar\rangle$, th: eph_h) {

$(\forall h : hogar)\ (hogarEnTabla(h, mh) \longrightarrow hogarEnTabla(h, th))$

}

`pred ordenadaPorIngresosConMismaDiferencia` (mh: $seq\langle hogar\rangle$, ti: eph_i) {

$(\exists\ dif : \mathbb{Z})\ (dif > 0 \wedge (\forall i : \mathbb{Z})\ (0 \leq i < |mh| - 1 \longrightarrow_L$
$(\text{ingresos}(mh[i+1], ti) - \text{ingresos}(mh[i], ti) = dif)))$

}

`pred hayMejorSolucionMuestraHomogenea` (mh: $seq\langle hogar\rangle$, th: eph_h, ti: eph_i) {

$(\exists\ mayor : seq\langle hogar\rangle)\ (|mayor| > |mh| \wedge esSolucionMuestraHomogenea(mayor, th, ti))$

}

9. **proc corregirRegion**(**inout** $th : eph_h$, **in** $ti : eph_i$) .

Dada una encuesta válida, se desea agrupar las regiones de Gran Buenos Aires y Pampeana. Para cada hogar de Gran Buenos Aires, cambiar la región del hogar a Pampeana. Todo lo demás deberá permanecer igual.

```
proc corregirRegion (inout th: eph_h, in ti: eph_i) {
    Pre {esVálida(th, ti) ∧ th0 = th}
    Post {cambiaRegionesGBAaPampeana(th, th0)}
}
pred cambiaRegionesGBAaPampeana (th: eph_h, th0: eph_h) {
    |th| = |th0| ∧_L (∀i : ℤ) (0 ≤ i < |th| ⟶_L
        ((th0[i][@Region] ≠ 1 ∧ th0[i] = th[i]) ∨
        (th0[i][@Region] = 1 ∧ cambiaRegion(th0[i], th[i]))))
}
pred cambiaRegionGBAaPampeana (original: hogar, nuevo: hogar) {
    original[@HogCodusu] = nuevo[@HogCodusu] ∧ original[@HogAño] = nuevo[@HogAño] ∧
    original[@HogTrim] = nuevo[@HogTrim] ∧ original[@Tenencia] = nuevo[@Tenencia] ∧
    nuevo[@Region] = 43 ∧ original[@ + 500k] = nuevo[@ + 500k] ∧
    original[@Tipo] = nuevo[@Tipo] ∧ original[@qHabitaciones] = nuevo[@qHabitaciones] ∧
    original[@qDormitorios] = nuevo[@qDormitorios] ∧ original[@trabajaHogar] = nuevo[@trabajaHogar]
}
```

10. **proc quitarIndividuos**(**inout** $th : eph_h$, **inout** $ti : eph_i$, **in** $busqueda : seq\langle(ItemIndividuo, dato)\rangle$, **out** $result : (eph_h, eph_i)$)
Dada una encuesta válida y una búsqueda de individuos válida, se desea quitar los individuos que coinciden con todos los términos de búsqueda, y en base a este resultado se desea también quitar los hogares correspondientes (comparando el CODUSU). Una búsqueda se define como una lista de pares ordenados $(item : ItemIndividuo, valor : dato)$, y para que sea válida debe ocurrir que el $item$ sea un valor válido de $ItemIndividuo$, sin repetirse en la búsqueda; y $valor$ es el valor que se desea que el individuo tenga para su $item$ asociado. Los individuos y hogares quitados deben ser devueltos en $result$.

    Ejemplo:

    quitarIndividuos$(TH, TI, \langle(INDAÑO, 2020), (NIVEL_ED, 1)\rangle, result)$ debería devolver en $result$ tablas de hogares e individuos que contenga solamente registros de 2020 de personas que tienen estudios universitarios completos; mientras que TH y TI deberán contener el resto de los registros originales.

```
proc quitarIndividuos (inout th: eph_h, inout ti: eph_i, in busqueda: seq⟨(ItemIndividuo, dato)⟩, out result: (eph_h, eph_i)) {
    Pre {esVálida(th, ti) ∧ esBusquedaValida(busqueda) ∧ th = th0 ∧ ti = ti0}
    Post {((quedanIndividuosEnEncuesta(busqueda, ti0) ∧ esVálida(th, ti)) ∨
        (¬quedanIndividuosEnEncuesta(busqueda, ti0) ∧ encuestaVacia(th, ti))) ∧
        ((hayIndividuosEnResultado(busqueda, ti0) ∧ esVálida(result_0, result_1)) ∨
        (¬hayIndividuosEnResultado(busqueda, ti0) ∧ encuestaVacia(result_0, result_1))) ∧
        individuosCorrectos(busqueda, ti, ti0, result) ∧
        hogaresCorrectos(busqueda, th, th0, ti0, result)}
}
pred esBusquedaValida (busqueda: seq⟨(ItemIndividuo, dato)⟩) {
    |busqueda| > 0 ∧
    (∀p : (ItemIndividuo, dato)) (
    ((p_0 = IndCodusu ∧ p_1 > 0) ∨
    (p_0 = Componente ∧ p_1 > 0) ∨
    (p_0 = IndTrimestre ∧ p_1 <= 4) ∨
    (p_0 = Genero ∧ 0 < p_1 <= 2) ∨
    (p_0 = Edad ∧ p_1 >= 0) ∨
    (p_0 = Nivel_Ed ∧ 0 < p_1 < 2) ∨
    (p_0 = Estado ∧ −1 <= p_1 <= 1) ∨
    (p_0 = Cat_ocup ∧ 0 <= p_1 <= 4) ∨
    (p_0 = IngresoTot ∧ p_1 >= 0) ∨
    (p_0 = pIngresoTot ∧ p_1 = −1) ∨
    (p_0 = LugarTrabajo ∧ 0 < p_1 <= 10)) ∧
    (∀i : ℤ) ((∀j : ℤ) ((0 ≤ i, j < |busqueda| ∧ i ≠ j) ⟶_L busqueda[i]_0 ≠ busqueda[j]_0))))
```

}

pred `quedanIndividuosEnEncuesta` (busqueda: $seq\langle(ItemIndividuo, dato)\rangle$, ti0: eph_i) {

$(\exists i : individuo)\ (i \in ti0 \wedge_L \neg cumpleCondicion(busqueda, i))$

}

pred `hayIndividuosEnResultado` (busqueda: $seq\langle(ItemIndividuo, dato)\rangle$, ti0: eph_i) {

$(\exists i : individuo)\ (i \in ti0 \wedge_L cumpleCondicion(busqueda, i))$

}

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

$|th| = 0 \wedge |ti| = 0$

}

pred `individuosCorrectos` (busqueda: $seq\langle(ItemIndividuo, dato)\rangle$, ti: eph_i, ti0: eph_i, result: (eph_h, eph_i)) {

$|ti0| = |result_1| + |ti| \wedge$
$(\forall i : individuo)\ (i \in result_1 \longrightarrow (i \notin ti \wedge i \in ti0 \wedge_L cumpleCondicion(busqueda, i))) \wedge$
$(\forall i : individuo)\ (i \in ti \longrightarrow (i \notin result_1 \wedge i \in ti0 \wedge_L \neg cumpleCondicion(busqueda, i)))$

}

pred `cumpleCondicion` (busqueda: $seq\langle(ItemIndividuo, dato), i : individuo\rangle$) {

$(\forall b : (ItemIndividuo, dato))\ (b \in busqueda \longrightarrow_L i[ord(b_0)] = b_1)$

}

pred `hogaresCorrectos` (busqueda: $seq\langle(ItemIndividuo, dato)\rangle$, th: eph_h, th0: eph_h, ti0: eph_i, result: (eph_h, eph_i)) {

$(\forall h : hogar)\ (h \in result_0 \leftrightarrow (h \in th0 \wedge hayIndividuoQueCumpleCondicionEnHogar(h, busqueda, ti0))) \wedge$
$(\forall h : hogar)\ (h \in th \leftrightarrow (h \in th0 \wedge hayIndividuoQueNoCumpleCondicionEnHogar(h, busqueda, ti0)))$

}

pred `hayIndividuoQueCumpleCondicionEnHogar` (h:hogar, busqueda: $seq\langle(ItemIndividuo, dato)\rangle$, ti0: eph_i) {

$(\exists i : individuo)\ (i \in ti0 \wedge cumpleCondicion(busqueda, i) \wedge_L h[@HOGCODUSU] = i[@INDCODUSU])$

}

pred `hayIndividuoQueNoCumpleCondicionEnHogar` (h:hogar, busqueda: $seq\langle(ItemIndividuo, dato)\rangle$, ti0: eph_i) {

$(\exists i : individuo)\ (i \in ti0 \wedge \neg cumpleCondicion(busqueda, i) \wedge_L i[@INDCODUSU] = h[@HOGCODUSU])$

}

11. **proc histogramaDeAnillosConcentricos**(**in** $th : eph_h$, **in** $ti : eph_i$, **in** $centro : \mathbb{Z} \times \mathbb{Z}$, **in** $distancias : seq\langle\mathbb{Z}\rangle$, **out** $result : seq\langle\mathbb{Z}\rangle$) .
Dada una tabla de hogares, un punto central ("centro") y una lista no vacía y estrictamente creciente de distancias, devuelve otra lista que contiene la cantidad de hogares que se encuentran en los anillos concéntricos determinados por las distancias con respecto al punto central.

proc `histogramaDeAnillosConcentricos` (in th: eph_h, in ti: eph_i, in centro: $\mathbb{Z}\times\ \mathbb{Z}$, in distancias:$seq\langle\mathbb{Z}\rangle$, out result:$seq\langle\mathbb{Z}\rangle$) {

`Pre` $\{esValida(th, ti) \wedge |distancias| > 0 \wedge estrictamenteCrecientes(distancias)\}$

`Post` $\{hogaresEnAnillosConcentricos(distancias, centro, result, th)\}$

}

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

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

}

pred `hogaresEnAnillosConcentricos` (distancias: $seq\langle\mathbb{Z}\rangle$, centro: $\mathbb{Z}\times\ \mathbb{Z}$, result: $seq\langle\mathbb{Z}\rangle$, th: eph_h) {

$result[0] = cantHogaresEnAnillo(0, distancias[0], centro) \wedge$
$(\forall i : \mathbb{Z})\ (1 \leq i < |distancias| - 1 \longrightarrow_L result[i] = cantHogaresEnAnillo(distancias[i], distancias[i + 1], centro))$

}

`aux cantHogaresEnAnillo` (distDesde: $\mathbb{Z}$, distHasta: $\mathbb{Z}$, centro: $\mathbb{Z}\times\mathbb{Z}$) : $\mathbb{Z}$ =
$\sum_{k=0}^{|th|}$ if $hogarEnAnillo(distDesde, distHasta, centro, th[k])$ then 1 else 0 fi ;

`pred hogarEnAnillo` (distDesde: $\mathbb{Z}$, distHasta: $\mathbb{Z}$, centro: $\mathbb{Z}\times\mathbb{Z}$, h: hogar) {

$distDesde < distanciaEuclideana(centro, h[@HOGLATITUD], h[@HOGLONGITUD]) <= distHasta$

}

`aux distanciaEuclideana` (centro: $\mathbb{Z}\times\mathbb{Z}$, latitud: $\mathbb{Z}$, longitud: $\mathbb{Z}$) : $\mathbb{R}$ =
$\sqrt[2]{(centro_0 - latitud)2 + (centro_1 - longitud)2}$;

5.6. Predicados y Auxiliares generales

5.7. Usados desde el ejercicio 1

`pred esVálida` (th: eph_h, ti: eph_i) {
$\neg vacia(ti) \wedge \neg vacia(th) \wedge esMatriz(ti) \wedge esMatriz(th) \wedge$
$cantidadCorrectaDeColumnasI(ti) \wedge cantidadCorrectaDeColumnasH(th) \wedge_L$
$\neg hayIndividuosSinHogares(ti, th) \wedge \neg hayHogaresSinIndividuos(ti, th) \wedge$
$\neg hayRepetidosI(ti) \wedge \neg hayRepetidosH(th) \wedge$
$mismoAñoYTrimestre(ti, th) \wedge$
$menosDe21MiembrosPorHogar(th, ti) \wedge$
$cantidadValidaDormitorios(th) \wedge$
$valoresEnRangoI(ti) \wedge valoresEnRangoH(th)$
}
`pred esMatriz` (t:$seq\langle seq\langle Datos\rangle\rangle$) {
$(\forall i : \mathbb{Z})\ ((\forall j : \mathbb{Z})\ (0 \leq i < j < |t| \longrightarrow_L |t[i]| = |t[j]|))$
}
`pred individuoEnTabla` (ind:individuo, ti: eph_i) {
$(\exists i : \mathbb{Z})\ (0 \leq i < |ti| \wedge_L ti[i] = ind)$
}
`pred hogarEnTabla` (h:hogar, th:eph_h) {
$(\exists i : \mathbb{Z})\ (0 \leq i < |th| \wedge_L th[i] = h)$
}
`pred vacia` (t:$seq\langle seq\langle Datos\rangle\rangle$) {
$|t| = 0$
}
`pred cantidadCorrectaDeColumnasI` (ti:eph_i) {
$(\forall i : individuo)\ (individuoEnTabla(i, ti) \rightarrow |i| = cantidadItemsIndividuo)$
}
`pred cantidadCorrectaDeColumnasH` (th: eph_h) {
$(\forall h : hogar)\ (hogarEnTabla(h, th) \rightarrow |h| = cantidadItemsHogar)$
}
`pred hayIndividuosSinHogares` (ti:eph_i, th:eph_h) {
$(\exists i : individuo)\ (individuoEnTabla(i, ti) \wedge_L \neg hayHogarConCodigo(th, i[@IndCodusu])$
}
`pred hayHogarConCodigo` (th:eph_h, c:$\mathbb{Z}$) {
$(\exists h : hogar)\ (hogarEnTabla(h, th) \wedge_L h[@HogCodusu] = c)$
}
`pred hayHogaresSinIndividuos` (ti:eph_i, th:eph_h) {
$(\exists h : hogar)\ (hogarEnTabla(h, th) \wedge_L \neg hayIndividuoConCodigo(ti, h[@HogCodusu])$
}
`pred hayIndividuoConCodigo` (ti:eph_i, c:Z) {
$(\exists i : individuo)\ (individuoEnTabla(i, ti) \wedge_L i[@IndCodusu] = c)$
}
`pred hayRepetidosI` (ti:eph_i) {
$(\exists n1 : \mathbb{Z})\ (0 \leq n1 < |ti| \wedge (\exists n2 : \mathbb{Z})\ ($
$0 \leq n2 < |ti| \wedge n1 \neq n2 \wedge_L mismoCodusuYComponente(ti[n1], ti[n2])$
$))$
}
`pred mismoCodusuYComponente` (i1: individuo, i2: individuo) {
$i1[@IndCodusu] = i2[@IndCodusu] \wedge i1[@Componente] = i2[@Componente]$
}
`pred hayRepetidosH` (th: eph_h) {
$(\exists n1 : \mathbb{Z})\ (0 \leq n1 < |th| \wedge (\exists n2 : \mathbb{Z})\ ($

$0 \leq n1 < |th| \wedge n1 \neq n2 \wedge_L th[n1][@HogCodusu] = th[n2][@HogCodusu]$
))
}
pred mismoAñoYTrimestre (ti: eph$_i$, th: eph$_h$) {
$(\exists año : \mathbb{Z})$ $((\exists trimestre : \mathbb{Z})$ (
$(\forall i : individuo)$ (
$individuoEnTabla(i, ti) \longrightarrow_L$
$i[@IndAño] = año \wedge i[@IndTrimestre] = trimestre$
) $\wedge$ $(\forall h : hogar)$ (
$hogarEnTabla(h, th) \longrightarrow_L$
$h1[@HogAño] = año \wedge h1[@HogTrimestre] = trimestre$
)
))
}
pred menosDe21MiembrosPorHogar (th:eph$_h$, ti:eph$_i$) {
$(\forall i : \mathbb{Z})$ $(0 \leq i < th \longrightarrow_L cantHabitantes(th[i], ti) < 21)$
}
aux cantHabitantes (h: hogar, ti: eph$_i$) : $\mathbb{Z}$ =
$\sum_{k=0}^{|ti|-1}$(if $esSuHogar(h, ti[k])$ then 1 else 0 fi) ;
pred esSuHogar (h: hogar, i: individuo) {
$h[@HogCodusu] = i[@IndCodusu]$
}
pred cantidadValidaDormitorios (th:eph$_h$) {
$(\forall h : hogar)$ $(hogarEnTabla(h, th) \longrightarrow_L h[@qHabitaciones] \geq h[@qDormitorios])$
}
pred valoresEnRangoI (ti:eph$_i$) {
$(\forall i : individuo)$ $(individuoEnTabla(i, ti) \longrightarrow_L individuoValido(i))$
}
pred individuoValido (i:individuo) {
$(i[@IndCodusu] > 0) \wedge$
$(i[@Componente] > 0) \wedge$
$(0 < i[@IndTrimestre] \leq 4) \wedge$
$(0 < i[@Genero] \leq 2) \wedge$
$(i[@Edad] \geq 0) \wedge$
$(i[@Nivel\_Ed] = 0 \vee i[@Nivel\_Ed] = 1) \wedge$
$(-1 \leq i[@Estado] \leq 1) \wedge$
$(0 \leq i[@Cat\_ocup] \leq 4) \wedge$
$(i[@IngresoTot] \geq 0 \vee i[@pIngresoTot] = -1) \wedge$
$(0 \leq i[@LugarTrabajo] \leq 10)$
}
pred valoresEnRangoH (th:eph$_h$) {
$(\forall h : hogar)$ $(hogarEnTabla(h, th) \longrightarrow_L hogarValido(h))$
}
pred hogarValido (h:hogar) {
$(h[@HogCodusu] > 0) \wedge$
$(0 < h[@HogTrimestre] \leq 4) \wedge$
$(0 < h[@Tenencia] \leq 3) \wedge$
$valorRegionValido(h[@Region]) \wedge$
$(h[@ + 500k] = 0 \vee h[@ + 500k] = 1) \wedge$
$(0 < h[@Tipo] \leq 5) \wedge$
$(h[@qHabitaciones] > 0) \wedge$
$(h[@qDormitorios] \geq 1) \wedge$
$(h[@TrabajaHog] = 1 \vee h[@TrabajaHog] = 2)$
}
pred valorRegionValido (r: $\mathbb{Z}$) {
$((r = 1) \vee (40 \leq r \leq 44))$
}
pred esCasa (h: hogar) {
$h[@Tipo] = 1$
}
aux ingresos (h: hogar, ti: eph$_i$) : $\mathbb{Z}$ =
$\sum_{i=0}^{|ti|-1}$ if $ti[i][@IndCodusu] = h[@HogCodusu] \wedge ti[i][@IngresoTot] > -1$ then $ti[i][@IngresoTot]$ else 0 fi ;