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

Algoritmos y Estructura de Datos I

Primer cuatrimestre de 2019

13/05/19

Trabajo Práctico de Implementación (TPI)

Análisis Habitacional Argentino

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.

1.1. Tabla `HOGARES`

- `HOGCODUSU`: Identificador o clave (ℕ) del hogar. Además permite hacer el seguimiento a través de los trimestres.
- `HOGAÑO`: Año de relevamiento.
- `HOGTRIMESTRE`: Trimestre del año de relevamiento.
- `II7`: Régimen de tenencia de los habitantes:
  - 1 - Propietario
  - 2 - Inquilino
  - 3 - Ocupante
- `REGION`: Código de Región:
  - 1 - Gran Buenos Aires
  - 2 - NOA
  - 3 - NEA
  - 4 - Cuyo
  - 5 - Pampeana
  - 6 - 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

1.2. 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:
  - 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

1.3. Funciones C++

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

```
bool esEncuestaValida(eph_h th, eph_i ti);
histogramaHab histHabitacional(eph_h th, eph_i ti);
vector<float> laCasaEstaQuedandoChica(eph_h th, eph_i ti);
bool creceElTeleworkingEnCuidadesGrandes(eph_h t1h, eph_i t1i, eph_h t2h, eph_i t2i);
void ordenarPorRegion(eph_h & th, eph_i & ti);
vector<hogar> muestraHomogenea(eph_h th, eph_i ti);
void corregirRegion(eph_i & ti, eph_h & th);
float indiceGini(eph_h th, eph_i ti);
```

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 vector<int> histogramaHab;
```

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.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 **10 de JUNIO de 2019**.

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. 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. Acceso a las columnas

`aux cantidadItemsIndividuo` : $\mathbb{Z} = 11$;
`aux cantidadItemsHogar` : $\mathbb{Z} = 10$;
`aux @IndCodusu` : $\mathbb{Z} = ord(INDCODUSU)$;
`aux @HogCodusu` : $\mathbb{Z} = ord(HOGCODUSU)$;
`aux @IndAño` : $\mathbb{Z} = ord(INDANO4)$;
`aux @IndTrim` : $\mathbb{Z} = ord(INDTRIMESTRE)$;
`aux @HogAño` : $\mathbb{Z} = ord(HOGANO4)$;
`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` : $\mathbb{Z} = ord(II3)$;

4.2. 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) ∧ 1 ≤ region ≤ 6}
    Post {longitudIgualAMáximaCantidadHabitaciones(th, region, res) ∧
          (∀i : Z) (0 ≤ i < |res| ⟶_L res[i] = cantHogaresCasaConNHabitaciones(th, region, i))}
}
pred longitudIgualAMáximaCantidadHabitaciones (th: eph_h, region: Z, lista: seq⟨Z⟩) {
    (∀h : hogar) ((h ∈ th ∧ h[@Region] = region) ⟶_L |lista| ≥ h[@qHabitaciones])
    ∧ (∃h : hogar) (h ∈ th ∧ h[@Region] = region ∧_L |lista| = h[@qHabitaciones])
}
aux cantHogaresCasaConNHabitaciones ( th: eph_h, region: Z, habitaciones: Z) : Z =
    ∑_{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** $res : seq\langle\mathbb{R}\rangle$) .
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⟨R⟩ ) {
    Pre {esVálida(th, ti)}
    Post {|res| = 6 ∧_L (∀i : Z) (0 ≤ i < 6 ⟶_L res[i] = proporcionDeCasasConHC(th, ti, i + 1))}
}
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 =
    ∑_{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 =
    ∑_{k=0}^{|th|−1} if esHogarValido(th[k], region)] = 0 then 1 else 0 fi ;
```

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) ∧ esVálida(t2h, t2i) ∧_L año(t1i) < año(t2i) ∧ trimestre(t1i) = trimestre(t2i)}
    Post {res = true ↔ proporcionTeleworking(t2h, t2i) > proporcionTeleworking(t1h, t1i)}
}
aux proporcionTeleworking (th: eph_h, ti: eph_i) : ℝ =
    if cantIndividuosQueTrabajan(th, ti) > 0
     then cantIndividuosTrabajandoEnSuVivienda(th, ti)/cantIndividuosQueTrabajan(th, ti) else 0 fi ;
aux cantIndividuosTrabajandoEnSuVivienda (th: eph_h, ti:eph_i) : ℤ =
    ∑_{k=0}^{|ti|-1} if trabaja(ti[k]) ∧ trabajaEnSuVivienda(ti[k], th) ∧
            individuoEnHogarValido(ti[k], th) then 1 else 0 fi ;
aux cantIndividuosQueTrabajan (th: eph_h, ti : eph_i) : ℤ =
    ∑_{k=0}^{|ti|-1} if trabaja(ti[k]) ∧ individuoEnHogarValido(ti[k], th) then 1 else 0 fi ;
pred trabajaEnSuVivienda (i: individuo, th: eph_h) {
    realizaSusTareasEnEsteHogar(i) ∧ suHogarTieneEspaciosReservadosParaElTrabajo(i, th)
}
pred individuoEnHogarValido (i: individuo, th: eph_h) {
    esDeCiudadGrande(i, th) ∧ suHogarEsCasaODepartamento(i, th)
}
```

5. **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) ∧ th0 = th ∧ ti0 = ti}
    Post {|th0| = |th| ∧ |ti0| = |ti|
    ∧ (∀h : hogar) (hogarEnTabla(h, th) ↔ hogarEnTabla(h, th0))
    ∧ (∀i : individuo) (individuoEnTabla(i, ti) ↔ individuoEnTabla(i, ti0))
    ∧_L estanOrdenadosPorRegionYCodusu(th)∧estanOrdenadosPorCodusuDeHogarYComponente(ti, th)}
}
pred estanOrdenadosPorRegionYCodusu (th: eph_h) {
    (∀i : ℤ) (0 ≤ i < |th| ⟶ (∀j : ℤ) (i < j < |th| ⟶_L
        th[i][@Region] < th[j][@Region]
        ∨ (th[i][@Region] = th[j][@Region] ∧ th[i][@HogCodusu] < th[j][@HogCodusu])))
}
pred estanOrdenadosPorCodusuDeHogarYComponente (ti: eph_i, th: eph_h) {
    (∀i : ℤ) (0 ≤ i < |ti| ⟶ (∀j : ℤ) (i < j < |ti| ⟶_L
        suHogarEstaAntes(ti[i], ti[j], th) ∨
        (vivenJuntos(ti[i], ti[j]) ∧ ti[i][@Componente] < ti[j][@Componente])))
}
pred suHogarEstaAntes (i1: individuo, i2: individuo, th: eph_h) {
    (∃h1 : hogar) (hogarEnTabla(h1, th) ∧ ((∃h2 : hogar) (hogarEnTabla(h2, th) ∧_L
        h1[@HogCodusu] = i1[@IndCodusu] ∧ h2[@HogCodusu] = i2[@IndCodusu] ∧
        hogarEstaAntes(h1, h2, th)))
}
pred hogarEstaAntes (h1: hogar, h2: hogar, th: eph_h) {
    (∃i : ℤ) (0 ≤ i < |th| ∧ ((∃j : ℤ) (i < j < |th| ∧_L (th[i] = h1 ∧ th[j] = h2))
```

```
}
```

6. **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.

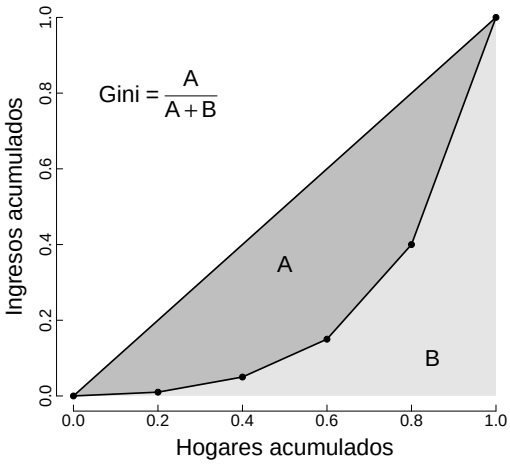
```
proc muestraHomogenea (in th: eph_h, in ti: eph_i, out res: seq⟨hogar⟩) {
    Pre {esVálida(th, ti)}
    Post {(|res| = 0 ∧ ¬existeSolucionMuestraHomogeneaConAlMenos3(th, ti)) ∨
    (|res| ≥ 3∧ esSolucionMuestraHomogenea(res, th, ti)∧¬hayMejorSolucionMuestraHomogenea(res, th, ti))}
}
pred existeSolucionMuestraHomogeneaConAlMenos3 (th: eph_h, ti: eph_i) {
    (∃ muestra : seq⟨hogar⟩) (esSolucionMuestraHomogenea(muestra, th, ti) ∧ |muestra| ≥ 3)
}
pred esSolucionMuestraHomogenea (mh: seq⟨hogar⟩, th: eph_h, ti: eph_i) {
    esMuestraSinRepetidos(mh, th) ∧_L estaOrdenadaPorDiferenciaDeIngresos(res, th, ti)
}
pred esMuestraSinRepetidos (mh: seq⟨hogar⟩, th: eph_h) {
    (∀h : hogar) (hogarEnTabla(h, mh) ⟶ hogarEnTabla(h, th)) ∧ noHayRepetidos(mh)
}
pred estaOrdenadaPorDiferenciaDeIngresos (mh: seq⟨hogar⟩, ti: eph_i) {
    (∃ ingreso : ℤ) (ingreso > 0 ∧ (∀i : ℤ) (0 ≤ i < |mh| − 1 ⟶_L
        (ingresos(mh[i + 1], ti) − ingresos(mh[i], ti) = ingreso)))
}
pred hayMejorSolucionMuestraHomogenea (mh: seq⟨hogar⟩, th: eph_h, ti: eph_i) {
    (∃ mayor : seq⟨hogar⟩) (|mayor| > |mh| ∧ esSolucionMuestraHomogenea(mayor, th, ti))
}
```

7. **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) {
    Pre {esVálida(th, ti) ∧ th0 = th}
    Post {soloModificaRegionGBAaPampeana(th, th0)}
}
pred soloModificaRegionGBAaPampeana (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 ∧ esIgualConRegionPampeana(th0[i], th[i])))
}
pred esIgualConRegionPampeana (original: hogar, nuevo: hogar) {
    original = setAt(nuevo, @Region, 5)
}
```

8. El coeficiente de Gini, G, se utiliza para medir la desigualdad en la distribución de la riqueza. Un valor $G = 0$ representa desigualdad nula. Ocurre solo cuando todas las personas tienen exactamente la misma riqueza. Un valor $G = 1$ representa desigualdad absoluta. Ocurre solo cuando una persona tiene toda la riqueza y el resto nada.

   En este trabajo práctico vamos a implementar una función que permita computar el coeficiente de Gini de ingresos de los hogares. Para eso se requiere contar que la tabla de hogares haya sido previamente ordenada por ingresos de menor a mayor. Calcular el coeficiente de Gini es equivalente a calcular el área A de la siguiente figura




```
proc indiceGini (in th: eph_h, in ti: eph_i, out res: ℝ) {
    Pre {esVálida(th, ti) ∧ tablaHogaresOrdenadaPorIngresosPerCapita(th, ti)}
    Post {res = areaObservada(th,ti) / areaIgualdadTotal()}
}

pred tablaHogaresOrdenadaPorIngresosPerCapita (th: eph_h, ti: eph_i) {
    (∀i : ℤ) ((0 ≤ i < |th| − 1) ⟶_L
        ingresosPerCapitaEnHogar(th[i], ti) ≤ ingresosPerCapitaEnHogar(th[i + 1], ti)
    )
}

aux areaIgualdadTotal : ℝ = 0,5 ;

aux areaObservada (th:eph_h,ti:eph_i) : ℝ = areaIgualdadTotal() − integralDeIngresosObservados(th, ti) ;

aux integralDeIngresosObservados (th:eph_h,ti:eph_i) : ℝ =
    1/|th| Σ_{i=0}^{|th|−1} ingresosHasta(i−1,th,ti) / ingresosHasta(|th|,th,ti) + ingresosPerCapitaEnHogar(th[i],ti) / ingresosHasta(|th|,th,ti) 1/2 ;

aux ingresosHasta (n:ℤ, th:eph_h,ti:eph_i) : ℝ =  Σ_{i=0}^{n} ingresosPerCapitaEnHogar(th[i],ti) ;

aux ingresosPerCapitaEnHogar (h: hogar,ti:eph_i) : ℝ = ingresos(h,ti) / cantHabitantes(h,ti) ;
```

4.3. Predicados y Auxiliares generales

4.4. Usados desde el ejercicio 1

```
pred esVálida (th: eph_h, ti: eph_i) {
    ¬vacia(ti) ∧ ¬vacia(th) ∧ esMatriz(ti) ∧ esMatriz(th) ∧
    cantidadCorrectaDeColumnasI(ti) ∧ cantidadCorrectaDeColumnasH(th) ∧_L
    ¬hayIndividuosSinHogares(ti, th) ∧ ¬hayHogaresSinIndividuos(ti, th) ∧
    ¬hayRepetidosI(ti) ∧ ¬hayRepetidosH(th) ∧
    mismoAñoYTrimestre(ti, th) ∧
    menosDe21MiembrosPorHogar(ti) ∧
    cantidadValidaDormitorios(th) ∧
    valoresEnRangoI(ti) ∧ valoresEnRangoH(th)
}
pred esMatriz (t:seq⟨seq⟨Datos⟩⟩) {
    (∀i : ℤ) ((∀j : ℤ) (0 ≤ i < j < |t| ⟶_L |t[i]| = |t[j]|))
}
pred individuoEnTabla (ind:individuo, ti: eph_i) {
    (∃i : ℤ) (0 ≤ i < |ti| ∧_L ti[i] = ind)
}
pred hogarEnTabla (h:hogar, th:eph_h) {
    (∃i : ℤ) (0 ≤ i < |th| ∧_L th[i] = h)
}
pred vacia (t:seq⟨seq⟨Datos⟩⟩) {
    |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) {
$(i[@IndCodusu] > 0) \wedge$
$(i[@Componente] > 0) \wedge$
$(0 < i[@IndTrimestre] \leq 4) \wedge$
$(0 < i[@Genero] \leq 2) \wedge$
$(i[@Edad] \geq 0) \wedge$
$(i[@Nivel\_Ed] = 0 \vee i[@Nivel\_Ed] = 1) \wedge$

$(-1 \leq i[@Estado] \leq 1) \wedge$
$(0 \leq i[@Cat\_ocup] \leq 4) \wedge$
$(i[@IngresoTot] \geq 0 \vee i[@pIngresoTot] = -1) \wedge$
$(0 < i[@LugarTrabajo] \leq 10)$
}
pred valoresEnRangoH (th:eph$_h$) {
$(\forall h : hogar)\ (hogarEnTabla(h, th) \longrightarrow_L hogarValido(h))$
}
pred hogarValido (h:hogar) {
$(h[@HogCodusu] > 0) \wedge$
$(0 < h[@HogTrimestre] \leq 3) \wedge$
$(0 < h[@Tenencia] \leq 3) \wedge$
$(0 < h[@Region] \leq 6) \wedge$
$(h[@ + 500k] = 0 \vee h[@ + 500k] = 1) \wedge$
$(0 < h[@Tipo] \leq 5) \wedge$
$(h[@qHabitaciones] > 0) \wedge$
$(h[@qDormitorios] \geq 1) \wedge$
$(h[@TrabajaHog] = 1 \vee h[@TrabajaHog] = 2)$
}

4.5. Usados desde el ejercicio 2

pred esCasa (h: hogar) {
$h[@Tipo] = 1$
}

4.6. Usados desde el ejercicio 3

pred hogarConHacinamientoCritico (h: hogar, ti: eph$_i$) {
$cantHabitantes(h, ti) > 3 * h[@qDormitorios]$
}

4.7. Usados desde el ejercicio 4

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

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

pred trabaja (i: individuo) {
$i[@Estado] = 1$
}
pred esDeCiudadGrande (i: individuo, th: eph$_h$) {
$(\exists h : hogar)\ (h \in th \wedge_L esSuHogar(h, i) \wedge h[@ + 500k] = 1)$
}
pred suHogarTieneEspaciosReservadosParaElTrabajo (i: individuo, th: eph$_h$) {
$(\exists h : hogar)\ (h \in th \wedge_L esSuHogar(h, i) \wedge tieneEspaciosReservadosParaElTrabajo(h))$
}
pred suHogarEsCasaODepartamento (i: individuo, th: eph$_h$) {
$(\exists h : hogar)\ (h \in th \wedge_L esSuHogar(h, i) \wedge esCasaODepartamento(h))$
}
pred esCasaODepartamento (h: hogar) {
$h[@Tipo] = 1 \vee h[@Tipo] = 2$
}
pred realizaSusTareasEnEsteHogar (i: individuo) {
$i[@LugarTrabajo] = 6$
}
pred tieneEspaciosReservadosParaElTrabajo (h: hogar) {
$h[@trabajaHogar] = 1$
}

4.8. Usados desde el ejercicio 5

pred vivenJuntos (i1: individuo, i2: individuo) {
$i1[@IndCodusu] = i2[@IndCodusu]$
}

4.9. Usados desde el ejercicio 6

pred noHayRepetidos (th: $seq\langle hogar\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |th| \longrightarrow (\forall j : \mathbb{Z})\ ((0 \leq j < |th| \wedge j \neq i) \longrightarrow_L th[i] \neq th[j]))$
}
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 ;

4.10. Usados desde el ejercicio 7

4.11. Usados desde el ejercicio Yapa

4.12. Acceso a las columnas

aux cantidadItemsIndividuo : $\mathbb{Z} = 11$;
aux cantidadItemsHogar : $\mathbb{Z} = 10$;
aux @IndCodusu : $\mathbb{Z} = ord(INDCODUSU)$;
aux @HogCodusu : $\mathbb{Z} = ord(HOGCODUSU)$;
aux @IndAño : $\mathbb{Z} = ord(INDANO4)$;
aux @IndTrim : $\mathbb{Z} = ord(INDTRIMESTRE)$;
aux @HogAño : $\mathbb{Z} = ord(HOGANO4)$;
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 : $\mathbb{Z} = ord(II3)$;