



# Trabajo práctico 1: Especificación y WP

## Elecciones Nacionales

15 de septiembre de 2023

Algoritmos y Estructuras de Datos

`sudo_rm-rf_/*`

Integrante	LU	Correo electrónico
Rocca, Santiago	152/23	santiagrocca17@gmail.com
Fisz, Maximiliano	586/19	maximilianofisz@gmail.com
Gomez, Abril	574/20	goskema@gmail.com
López, Gonzalo	1017/22	gonzalo.esloga.uba@gmail.com



**Facultad de Ciencias Exactas y Naturales**  
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (+54 +11) 4576-3300

<http://www.exactas.uba.ar>

# 1. Especificación

## 1.1. General

### 1.1.1. Predicados Universales

```
pred noHayRepetidos (in escrutinio : seq⟨ℤ⟩) {  
    (∀x : ℤ)(0 ≤ x < |escrutinio| →L ((∀y : ℤ)(0 ≤ y < |escrutinio| ∧ ¬(x = y) →L ¬(escrutinio[x] = escrutinio[y])))  
}  
pred cantVotosValidos (in escrutinio : seq⟨ℤ⟩) {  
    ((∀x : ℤ)(0 ≤ x < |escrutinio| →L (escrutinio[x] ≥ 0))  
}  
pred escrutinioValdio (in escrutinio: seq⟨ℤ⟩) {  
    |escrutinio| ≥ 2  
}  
pred EleccionValida (in escrutinio: seq⟨ℤ⟩) {  
    nohayRepetidos(escrutinio) ∧ cantVotosValidos(escrutinio) ∧ escrutinioValido(escrutinio)  
}  
pred umbralElectoral (in escrutinioSen : seq⟨ℤ⟩) {  
    ((∀x : ℤ)(0 ≤ x < |escrutinio| →L (escrutinioSen[x] > 3))  
}  
pred minimoDePartidos (in escrutinio: seq⟨ℤ⟩) {  
    |escrutinio| ≥ 3  
}
```

### 1.1.2. Auxiliares

```
aux sumaDeVotos (in escrutinio : seq⟨ℤ⟩) : ℤ =  $\sum_{i=0}^{|escrutinio|-1} escrutinio[i]$ ;  
aux porcentajeDeVotos (in escrutinio: seq⟨ℤ⟩, in votosPartido: ℤ) : ℝ = sumaDeVotos(escrutinio)-1 * votosPartido * 102;  
aux bancasDe (in indicePartido: ℤ, in bancas, in dHont seq⟨seq⟨ℤ⟩⟩) : ℤ =  
 $\sum_{p=0}^{bancas-1} if\ cocienteGanador(indicePartido, p, dHont)\ then\ 1\ else\ 0$ ;
```

## 1.2. hayBallotage

### 1.2.1. Main

```
proc hayBallotage (in escrutinio : seq⟨ℤ⟩) : Bool  
    requiere {eleccionValida(escrutinio)}  
    asegura {res = ¬((partidoMayorA45%(escrutinio) ∨ (partidoMayorA40%ConDiferencia(escrutinio)))}
```

### 1.2.2. Predicados Especificos

```
pred partidoMayorA45% (in escrutinio : seq⟨ℤ⟩) {  
    (∃n : ℤ)(0 ≤ n < |escrutinio| - 1 ∧L (porcentajeDeVotos(escrutinio, escrutinio[n]) > 45)  
}  
pred partidoMayorA40%ConDiferencia (in escrutinio : seq⟨ℤ⟩) {  
    (∃n : ℤ)(0 ≤ n < |escrutinio| - 1 ∧L (porcentajeDeVotos(escrutinio, escrutinio[n]) > 40) ∧L  
    ¬(∀x : ℤ)(0 ≤ x < |escrutinio| - 1 ∧ (¬(n = x) →L ((escrutinio[n] - escrutinio[x]) > 10))  
}
```

## 1.3. hayFraude

### 1.3.1. Main

```
proc hayFraude ((in escrutinio_Presidencial: seq⟨ℤ⟩, in escrutinio_Senadores: seq⟨ℤ⟩, in escrutinio_Diputados: seq⟨ℤ⟩) :  
Bool  
    requiere {umbralElectoral(escrutinio_Diputados) ∧ eleccionValida(escrutinio_Presidencial) ∧  
    eleccionValida(escrutinio_senadores) ∧ eleccionValida(escrutinio_diputados) ∧  
    minimoDePartidos(escrutinio_Senadores) ∧  
    (|escrutinio_Presidencial| = |escrutinio_Senadores| = |escrutinio_Diputados|)}  
    asegura {res = ¬(((sumaDeVotos(escrutinio_Presidencial) = sumaDeVotos(escrutinio_Senadores)) ∧  
    (sumaDeVotos(escrutinio_Presidencial) = sumaDeVotos(escrutinio_Diputados))))}
```

## 1.4. obtenerSenadoresEnProvincia

### 1.4.1. Main

```
proc obtenerSenadoresEnProvincia (in escrutinio : seq⟨ℤ⟩) : ℤ × ℤ
  requiere {eleccionValida(escrutinio) ∧ minimoDePartidos(escrutinio)}
  asegura {(∃!x : ℤ)(0 ≤ x < |escrutinio| - 1 ∧L ((∃!y : ℤ)(0 ≤ y < |escrutinio| - 1 ∧L ((∀i : ℤ)(0 ≤ i < |escrutinio| - 1 ∧ ¬(i = x) ∧ ¬(i = y) →L escrutinio[i] < escrutinio[resy] < escrutinio[resx]))))))}
```

### 1.4.2. Predicados Especificos

## 1.5. calcularDHondtEnProvincia

### 1.5.1. Main

```
proc calcularDHondtEnProvincia (in cant_bancas: ℤ, in escrutinio: seq⟨ℤ⟩) : seq⟨seq⟨ℤ⟩⟩
  requiere {eleccionValida(escrutinio) ∧ umbralElectoral(escrutinio) ∧ cant_bancas > 0}
  asegura {(∀r : ℤ)(0 ≤ n < cant_bancas) ∧L (∀x : ℤ)(0 ≤ n < |escrutinio|) →L (res[x][n] =  $\frac{\text{escrutinio}[x]}{n+1}$ )}
```

## 1.6. obtenerDiputadosEnProvincia

### 1.6.1. Main

```
proc obtenerDiputadosEnProvincia (in cant_bancas: ℤ, in escrutinio: seq⟨ℤ⟩, in dHondt: seq⟨seq⟨ℤ⟩⟩) : seq⟨ℤ⟩
  requiere {eleccionValida(escrutinio) ∧ umbralElectoral(escrutinio)}
  asegura {(∀r : ℤ)(0 ≤ r < |escrutinio| - 1 →L res[r] = bancasDe(r, cant_bancas, dHondt))}
pred cocienteGanador (in indicePartido: ℤ, in bancaEnDisputa: ℤ, in dHont: seq⟨seq⟨ℤ⟩⟩) {
  res = True ↔ (∀i : ℤ)(0 ≤ i < |dHont| - 1 ∧ ¬(i = indicePartido) →L dHont[bancaEnDisputa][indicePartido] > dHont[bancaEnDisputa][i])
}
```

## 1.7. validarListasDiputadosEnProvincia

### 1.7.1. Main

```
proc (in cant_bancas: ℤ, in listas: seq⟨seq⟨dni : ℤ × genero : ℤ⟩⟩ (Bool) :
  requiere {(cant_bancas > 0) ∧ (dni > 0) ∧ (1 ≤ genero ≤ 2)}
  asegura {(∀ partido : ℤ)(0 ≤ partido < |listas|) →L (cantCandidatosCorrecta(cant_bancas, listas[partido]) ∧ altGenero(listas[partido]))}
```

### 1.7.2. Predicados Especificos

```
pred cantCandidatosCorrecta (cant_bancas: ℤ, partido: seq⟨dni : ℤ × genero : ℤ⟩) {
  cant_bancas = |partido|
}
pred altGenero (partido: seq⟨dni : ℤ × genero : ℤ⟩) {
  ((∀n : ℤ)(n > 0)) →L (((n mód 2 = 0) →L (partido[n, 1] = 1)) ∧L ((n mód 2 = 1) →L (partido[n, 1] = 2))) ∨L ((n mód 2 = 0) →L (partido[n, 1] = 2 ∧L (n mód 2 = 1) →L partido[n, 1] = 1))
}
```

## 2. Implementaciones y demostraciones de correctitud

### 2.1. Implementaciones

#### 2.1.1. hayBallotage

```
1      res := true
2      tans := 0
3      primero := 0
4      segundo := 0
5      i := 0
6      suma := 0
7      while (escrutinio.size() > i) do
8          suma:= suma + escrutinio[i]
9          i := i + 1
10     endwhile
11     i := 0
12     while (escrutinio.size() > i) do
13         escrutinio[i] := (escrutinio[i] * 100)/suma
14         i := i + 1
15     endwhile
16     i := 0
17     while (escrutinio.size() > i) do
18         if (segundo < escrutinio[i])
19             segundo := escrutinio[i]
20         else:
21             skip
22         endif
23         if (primero < segundo)
24             trans := primero
25             primero := segundo
26             segundo := trans
27         else:
28             skip
29         endif
30         i := i + 1
31     endwhile
32     if (primero > 45)
33         res := false
34     else
35         if ((primero > 40) && (primero - segundo >= 10))
36             res := false
37         else
38             skip
39         endif
40     endif
41     endif
```

Código 1: ()

### 2.1.2. hayFraude

```
1      sumaPres := 0
2      sumaDip := 0
3      sumaSen := 0
4      i := 0
5      while (escrutinio_Presidencial.size() > i) do
6          sumaPres:= sumaPres + escrutinio_Presidencial[i]
7          sumaDip:= sumaDip + escrutinio_Diputados[i]
8          sumaSen := sumaSen + escrutinio_Senadores[i]
9          i := i + 1
10     endwhile
11     res := true
12     if (sumaPres = sumaDip && sumaPres = sumaSen)
13         res := false
14     else:
15         skip
16     endif
```

Código 2: ()

### 2.1.3. obtenerSenadoresEnProvincia

```
1      trans := 0
2      primero := 0
3      segundo := 0
4      i := 0
5      while (escrutinio.size() > i) do
6          if (escrutinio[segundo] < escrutinio[i])
7              segundo := i
8          else:
9              skip
10         endif
11         if (escrutinio[primero] < escrutinio[segundo])
12             trans := primero
13             primero := segundo
14             segundo := trans
15         else:
16             skip
17         endif
18         i := i + 1
19     endwhile
20     res_{0} := primero
21     res_{1} := segundo
```

Código 3: ()

#### 2.1.4. validarListasDiputadosEnProvincia

```
1      res := true
2      i := 0
3      while (listas.size() > i) do
4          if (listas[i].size() != cant_bancas)
5              res:= false
6          else:
7              skip
8          endif
9          i := i + 1
10     endwhile
11     i := 0
12     j := 1
13     while (listas.size() > i) do
14         genero := listas[i][0][1]
15         while (listas[i].size() > j) do
16             if (listas[i][j][1] == genero)
17                 res:=false
18             else:
19                 genero := listas[i][j][1]
20                 j := j + 1
21             endif
22         endwhile
23         i := i + 1
24     endwhile
```

Código 4: ()