Algoritmos y Estructura de Datos I

Transformación de estados y corrección

Idea de la clase

- Aumentaremos nuestros programas con afirmaciones sobre los estados durante la ejecución.
- Para ello, intercalaremos afirmaciones sobre los estados entre las instrucciones.
-
```
x = 0;

x = x + 3;

x = 2 * x;
```

Idea de la clase

- Aumentaremos nuestros programas con afirmaciones sobre los estados durante la ejecución.
- Para ello, intercalaremos afirmaciones sobre los estados entre las instrucciones.
- ```
  x = 0;
  ```
  $\{x = 0\}$
  ```
  x = x + 3;


  x = 2 * x;
  ```
  ```

Idea de la clase

- Aumentaremos nuestros programas con afirmaciones sobre los estados durante la ejecución.
- Para ello, intercalaremos afirmaciones sobre los estados entre las instrucciones.
- ```
  x = 0;
  ```
  $\{x = 0\}$
  ```
  x = x + 3;
  ```
  $\{x = 3\}$
  ```
  x = 2 * x;
  ```
  ```

Idea de la clase

- Aumentaremos nuestros programas con afirmaciones sobre los estados durante la ejecución.
- Para ello, intercalaremos afirmaciones sobre los estados entre las instrucciones.
- ```
  x = 0;
  ```
  $\{x = 0\}$
  ```
  x = x + 3;
  ```
  $\{x = 3\}$
  ```
  x = 2 * x;
  ```
  $\{x = 6\}$
  ```

Ejercicio 1

proc *CuadrupleMasUno*(inout $a : \mathbb{Z}$){
  Pre $\{a = a_0\}$
  Post $\{a = 4 * a_0 + 1\}$
}

```
1  void cuadrupleMasUno(int &a) {
2        a = a + a;
3        a = a + a;
4        a = a + 1;
5  }
```

Ejercicio 1

proc *CuadrupleMasUno*(inout $a : \mathbb{Z}$){
  Pre $\{a = a_0\}$
  Post $\{a = 4 * a_0 + 1\}$
}

```
1  void cuadrupleMasUno(int &a) {
2        {a = a0}
3        a = a + a;
4        {a = a0 + a0}
5        => {a = 2 * a0}
6        a = a + a;
7        {a = 2 * a0 + 2 * a0}
8        => {a = 4 * a0}
9        a = a + 1;
10       {a = 4 * a0 + 1}
11 }
```

Ejercicio 2

proc *suma*(in $a, b : \mathbb{Z}$ outc $: \mathbb{Z}$){
  Pre $\{True\}$
  Post $\{c = a + b\}$
}

```
1  void suma(int a, int b, &c) {
2        c = 0;
3        c = c + a;
4        c = c + b;
5  }
```

Ejercicio 2

proc *suma*(in $a, b : \mathbb{Z}$ out$c : \mathbb{Z}$){
  Pre $\{True\}$
  Post $\{c = a + b\}$
}

```
1  void suma(int a, int b, int &c) {
2        c = 0;
3        {c = 0}
4        c = c + a;
5        {c = 0 + a}
6        ⇒ {c = a}
7        c = c + b;
8        {c = a + b}
9  }
```

Ejercicio 2

```
1  void suma2(int a, int b, int &c) {
2        a = a + b;
3        c = a;
4  }
```

Ejercicio 2

```
1  void suma2(int a, int b, int &c) {
2        {a = a_0}
3        a = a + b;
4        {a = a_0 + b}
5        c = a;
6        {c = a_0 + b}
7  }
```

Ejercicio 3

proc *SumaOResta*(in b : Bool inout n : $\mathbb{Z}$){
  Pre $\{n = n_0\}$
  Post $\{(b \wedge n = n_0 + 1) \vee (\neg b \wedge n = n_0 - 1)\}$
}

```
1  void SumaOResta(bool b, &n) {
2       if(b){
3            n = n + 1;
4       }else {
5            n = n - 1;
6       }
7  }
```

Ejercicio 3

proc *SumaOResta*(in b : Bool inout n : $\mathbb{Z}$){
  Pre $\{n = n_0\}$
  Post $\{(b \wedge n = n_0 + 1) \vee (\neg b \wedge n = n_0 - 1)\}$
}

```
1  void SumaOResta(bool b, &n) {
2       {n = n0}
3       if(b){
4            {b}
5            n = n + 1;
6            {b ∧ n = n0 + 1}
7       }else {
8            {¬b}
9            n = n - 1;
10           {¬b ∧ n = n0 - 1}
11      }
12      {(b ∧ n = n0 + 1) ∨ (¬b ∧ n = n0 - 1)}
13 }
```

Ejercicio 4

proc *SiCeroCambia*(inout $a : \mathbb{Z}$){
  Pre $\{a = a_0\}$
  Post $\{(a_0 = 0 \land a = 1) \lor (a_0 \neq 0 \land a = 0)\}$
}

```
1  void SiCeroCambia(int &a) {
2       if(a == 0){
3            a = 1;
4       }else{
5            a = 0;
6       }
7  }
```

Ejercicio 4

```
1  void SiCeroCambia(int &a) {
2        {a = a_0}
3        if(a == 0){
4              {B : a = 0}
5              ⇒ {a_0 = 0}
6              a = 1;
7              {a_0 = 0 ∧ a = 1}
8        }else{
9              {¬B : a ≠ 0}
10             ⇒ {a_0 ≠ 0}
11             a = 0;
12             {a_0 ≠ 0 ∧ a = 0}
13       }
14       {(a_0 = 0 ∧ a = 1) ∨ (a_0 ≠ 0 ∧ a = 0)}
15 }
```

Ejercicio 5

proc *Maximo*(in $a, b : \mathbb{Z}$, out *result* : $\mathbb{Z}$){
  Pre $\{True\}$
  Post $\{(a \leq b \wedge result = b) \vee (a > b \wedge result = a)\}$
}

```
1  int maximo(int a, int b) {
2      int res;
3      if (a > b) {
4          res = a;
5      } else {
6          res = b;
7      }
8      return res;
9  }
```

Ejercicio 5

```
1  int maximo(int a, int b) {
2      int res;
3      if (a > b) {
4      {B : a > b}
5          res = a;
6          {a > b ∧ res = a}
7      } else {
8      {B : a ≤ b}
9          res = b;
10         {a ≤ b ∧ res = b}
11     }
12     {(a ≤ b ∧ result = b) ∨ (a > b ∧ result = a)}
13     return res;
14 }
```

Ejercicio 6

proc sudoku_esTableroValido (in t: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$, out result: Bool) {

  Pre $\{\text{True}\}$

  Post $\{result = esTableroValido(t)\}$

}

pred esTableroValido (t: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$)
$\{dimensionValida(t) \wedge_L posicionesValidas(t)\}$

pred dimensionValida (t: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$)
$\{length(t) = 9 \wedge_L (\forall i : \mathbb{Z})(0 \leq i < 9 \rightarrow_L length(t_i) = 9)\}$

pred posicionesValidas (t: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$)
$\{(\forall i : \mathbb{Z})(\forall j : \mathbb{Z})(posicionValida(i, j) \rightarrow_L 0 \leq t[i][j] \leq 9)\}$

pred posicionValida (f: $\mathbb{Z}$, c: $\mathbb{Z}$)
$\{indiceValido(f) \wedge indiceValido(c)\}$

pred indiceValido (i: $\mathbb{Z}$) $\{0 \leq i < 9\}$

Ejercicio 6

proc sudoku_esTableroValido (in t: $seq\langle seq\langle \mathbb{Z} \rangle \rangle$, out result: Bool) {
  Pre {True}
  Post {$result = esTableroValido(t)$}
}

```
1  bool sudoku_esTableroValido(Tablero t){
2      bool result;
3      bool dimensionVal = esDimensionValida(&t);
4      bool posicionesVal = posicionesValidas(&t);
5      if (dimensionVal && posicionesVal){
6          result = true;
7      } else {
8          result = false;
9      }
10     return result;
11 }
```

¿Dudas? ¿Preguntas?