

# Laboratorio de Programación - Labo05

## Programación de Ciclos con Invariantes

Algoritmos y Estructuras de Datos I

Departamento de Computación, FCEyN, Universidad de  
Buenos Aires.

# Repaso - Ciclos

Sintaxis de un ciclo:

---

```
1 while(B) {  
2     // cuerpo del ciclo  
3 }
```

---

- ▶ El ciclo se repite continuamente mientras la guarda B se cumpla. Cada repetición es una iteración.
- ▶ El ciclo termina cuando no se cumpla la guarda.
- ▶ Al salir, en caso de que el ciclo terminara, el estado resultante es el mismo que el del final de la última iteración.

## Repaso - Teorema del Invariante

Sea  $P_C$  la precondition del ciclo,  $Q_C$  la postcondición,  $B$  la guarda e  $I$  un invariante del ciclo. Si se cumple:

1.  $P_C \Rightarrow I$ ,
2.  $\{I \wedge B\}$  cuerpo del ciclo  $\{I\}$ ,
3.  $I \wedge \neg B \Rightarrow Q_C$ ,

entonces el ciclo es **parcialmente** correcto (si termina, termina en  $Q_C$ ).

---

```
1 // Vale I
2 while(B){
3     // Vale I && B
4     Cuerpo del ciclo
5     // Vale I
6 }
7 // Vale Qc
```

---

## Repaso: Teorema de corrección de un ciclo

- **Teorema.** Sean un predicado  $I$  y una función  $fv : \mathbb{V} \rightarrow \mathbb{Z}$  (donde  $\mathbb{V}$  es el producto cartesiano de los dominios de las variables del programa), y supongamos que  $I \Rightarrow \text{def}(B)$ . Si

1.  $P_C \Rightarrow I$ ,
2.  $\{I \wedge B\} S \{I\}$ ,
3.  $I \wedge \neg B \Rightarrow Q_C$ ,
4.  $\{I \wedge B \wedge v_0 = fv\} S \{fv < v_0\}$ ,
5.  $I \wedge fv \leq 0 \Rightarrow \neg B$ ,

... entonces la siguiente tripla de Hoare es válida:

$$\{P_C\} \text{ while } B \text{ do } S \text{ endwhile } \{Q_C\}$$

# Ejemplo

---

```
1  bool hayMayorACero(vector<int> v) {
2      bool encuentre = false;
3      int i = 0;
4      int n = v.size();
5      while(i<n) {
6          encuentre = encuentre || v[i] > 0;
7          i = i+1;
8      }
9      return encuentre;
10 }
```

---

► Sea  $v = \{-1, -2, 3, -3, 4\}$

## Principio de iteración

Iteración	i	v[i]	encontre
-----------	---	------	----------

---

## Final de iteración

Iteración	i	v[i]	encontre
-----------	---	------	----------

---

# Ejemplo

---

```
1  bool hayMayorACero(vector<int> v) {
2      bool encuentre = false;
3      int i = 0;
4      int n = v.size();
5      while(i<n) {
6          encuentre = encuentre || v[i] > 0;
7          i = i+1;
8      }
9      return encuentre;
10 }
```

---

► Sea  $v = \{-1, -2, 3, -3, 4\}$

## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>

## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>

# Ejemplo

---

```
1  bool hayMayorACero(vector<int> v) {
2      bool encuentre = false;
3      int i = 0;
4      int n = v.size();
5      while(i<n) {
6          encuentre = encuentre || v[i] > 0;
7          i = i+1;
8      }
9      return encuentre;
10 }
```

---

► Sea  $v = \{-1, -2, 3, -3, 4\}$

## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>

## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>

# Ejemplo

---

```
1  bool hayMayorACero(vector<int> v) {
2      bool encuentre = false;
3      int i = 0;
4      int n = v.size();
5      while(i<n) {
6          encuentre = encuentre || v[i] > 0;
7          i = i+1;
8      }
9      return encuentre;
10 }
```

---

► Sea  $v = \{-1, -2, 3, -3, 4\}$

## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>
3	2	3	<b>false</b>

## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>
3	3	-3	<b>true</b>



# Ejemplo

---

```
1  bool hayMayorACero(vector<int> v) {
2      bool encuentre = false;
3      int i = 0;
4      int n = v.size();
5      while(i<n) {
6          encuentre = encuentre || v[i] > 0;
7          i = i+1;
8      }
9      return encuentre;
10 }
```

---

► Sea  $v = \{-1, -2, 3, -3, 4\}$

## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>
3	2	3	<b>false</b>
4	3	-3	<b>true</b>

## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>
3	3	-3	<b>true</b>
4	4	4	<b>true</b>

# Ejemplo

```
1  bool hayMayorACero(vector<int> v) {
2      bool encuentre = false;
3      int i = 0;
4      int n = v.size();
5      while(i<n) {
6          encuentre = encuentre || v[i] > 0;
7          i = i+1;
8      }
9      return encuentre;
10 }
```

► Sea  $v = \{-1, -2, 3, -3, 4\}$

## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>
3	2	3	<b>false</b>
4	3	-3	<b>true</b>
5	4	4	<b>true</b>

## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>
3	3	-3	<b>true</b>
4	4	4	<b>true</b>
5	5	⊥	<b>true</b>

# Ejemplo

## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>
3	2	3	<b>false</b>
4	3	-3	<b>true</b>
5	4	4	<b>true</b>

## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>
3	3	-3	<b>true</b>
4	4	4	<b>true</b>
5	5	$\perp$	<b>true</b>

Es un invariante?

►  $I \equiv i \leq n$  ?

# Ejemplo

## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>
3	2	3	<b>false</b>
4	3	-3	<b>true</b>
5	4	4	<b>true</b>

## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>
3	3	-3	<b>true</b>
4	4	4	<b>true</b>
5	5	$\perp$	<b>true</b>

## Es un invariante?

►  $I \equiv i \leq n$  ? 👍

►  $I \equiv i \leq n \wedge (\text{encontre} = \text{true} \vee \text{encontre} = \text{false})$  ?

# Ejemplo

## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>
3	2	3	<b>false</b>
4	3	-3	<b>true</b>
5	4	4	<b>true</b>

## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>
3	3	-3	<b>true</b>
4	4	4	<b>true</b>
5	5	$\perp$	<b>true</b>

## Es un invariante?

►  $I \equiv i \leq n$  ? 👍

►  $I \equiv i \leq n \wedge (\text{encontre} = \mathbf{true} \vee \text{encontre} = \mathbf{false})$  ? 👍

►  $I \equiv 0 \leq i \leq n \wedge \text{encontre} = \mathbf{true} \leftrightarrow (\exists k : \mathbb{Z}) 0 \leq k \leq i \wedge v[k] > 0$  ?

# Ejemplo


## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>
3	2	3	<b>false</b>
4	3	-3	<b>true</b>
5	4	4	<b>true</b>



## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>
3	3	-3	<b>true</b>
4	4	4	<b>true</b>
5	5	$\perp$	<b>true</b>

## Es un invariante?

►  $I \equiv i \leq n$  ? 

►  $I \equiv i \leq n \wedge (\text{encontre} = \text{true} \vee \text{encontre} = \text{false})$  ? 

►  $I \equiv 0 \leq i \leq n \wedge \text{encontre} = \text{true} \leftrightarrow (\exists k : \mathbb{Z}) 0 \leq k \leq i \wedge v[k] > 0$  ?  

# Ejemplo


## Principio de iteración


Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>
3	2	3	<b>false</b>
4	3	-3	<b>true</b>
5	4	4	<b>true</b>



## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>
3	3	-3	<b>true</b>
4	4	4	<b>true</b>
5	5	$\perp$	<b>true</b>

## Es un invariante?

►  $I \equiv i \leq n$  ? 

►  $I \equiv i \leq n \wedge (\text{encontre} = \text{true} \vee \text{encontre} = \text{false})$  ? 

►  $I \equiv 0 \leq i \leq n \wedge_L \text{encontre} = \text{true} \leftrightarrow (\exists k : \mathbb{Z}) 0 \leq k \leq i \wedge_L v[k] > 0$  ?   


►  $I \equiv 0 \leq i \leq n \wedge_L \text{encontre} = \text{true} \leftrightarrow (\exists k : \mathbb{Z}) 0 \leq k < i \wedge_L v[k] > 0$  ?

# Ejemplo


## Principio de iteración


Iteración	i	v[i]	encontre
1	0	-1	<b>false</b>
2	1	-2	<b>false</b>
3	2	3	<b>false</b>
4	3	-3	<b>true</b>
5	4	4	<b>true</b>



## Final de iteración



Iteración	i	v[i]	encontre
1	1	-2	<b>false</b>
2	2	3	<b>false</b>
3	3	-3	<b>true</b>
4	4	4	<b>true</b>
5	5	$\perp$	<b>true</b>

## Es un invariante?

►  $I \equiv i \leq n$  ? 

►  $I \equiv i \leq n \wedge (\text{encontre} = \text{true} \vee \text{encontre} = \text{false})$  ? 

►  $I \equiv 0 \leq i \leq n \wedge_L \text{encontre} = \text{true} \leftrightarrow (\exists k : \mathbb{Z}) 0 \leq k \leq i \wedge_L v[k] > 0$  ?   


►  $I \equiv 0 \leq i \leq n \wedge_L \text{encontre} = \text{true} \leftrightarrow (\exists k : \mathbb{Z}) 0 \leq k < i \wedge_L v[k] > 0$  ?   




# Ejemplo

## Principio de iteración

Iteración	i	v[i]	encontre
1	0	-1	false
2	1	-2	false
3	2	3	false
4	3	-3	true
5	4	4	true

## Final de iteración

Iteración	i	v[i]	encontre
1	1	-2	false
2	2	3	false
3	3	-3	true
4	4	4	true
5	5	⊥	true

## Es un invariante?

►  $I \equiv i \leq n$  ? 👍

►  $I \equiv i \leq n \wedge (\text{encontre} = \text{true} \vee \text{encontre} = \text{false})$  ? 👍

►  $I \equiv 0 \leq i \leq n \wedge_L \text{encontre} = \text{true} \leftrightarrow (\exists k : \mathbb{Z}) 0 \leq k \leq i \wedge_L v[k] > 0$  ? 🤔  
👎

►  $I \equiv 0 \leq i \leq n \wedge_L \text{encontre} = \text{true} \leftrightarrow (\exists k : \mathbb{Z}) 0 \leq k < i \wedge_L v[k] > 0$  ? 🦵  
💪



No todos nos van a servir para poder demostrar la correctitud parcial del ciclo! (en particular  $I \wedge \neg B \Rightarrow Q_c$ )

## Ejercicio

Calcular la suma de todos los números primos positivos mayores a 2 hasta  $n$  (inclusive) respetando el siguiente invariante:

$$I \equiv 3 \leq i \leq n + 1 \wedge suma = \sum_{k=3}^{i-1} \text{if } esPrimo(k) \text{ then } k \text{ else } 0 \text{ fi}$$

## Ejercicio

Calcular la suma de todos los números primos positivos mayores a 2 hasta  $n$  (inclusive) respetando el siguiente invariante:

$$I \equiv 3 \leq i \leq n + 1 \wedge suma = \sum_{k=3}^{i-1} \text{if } esPrimo(k) \text{ then } k \text{ else } 0 \text{ fi}$$

---

```
1  int suma = 0;
2  int i = 3;
3  while(i <= n) {
4      if (esPrimo(i)){
5          suma = suma + i;
6      }
7      i++;
8  }
9  return suma;
```

---

## Variante 1

Calcular la suma de todos los números primos positivos mayores a 2 hasta  $n$  (inclusive) respetando el siguiente invariante:

$$I \equiv 3 \leq i \leq n + 2 \wedge i \bmod 2 = 1 \wedge$$

$$suma = \sum_{k=3}^{i-2} \text{if } esPrimo(k) \text{ then } k \text{ else } 0 \text{ fi}$$

## Variante 1

Calcular la suma de todos los números primos positivos mayores a 2 hasta  $n$  (inclusive) respetando el siguiente invariante:

$$I \equiv 3 \leq i \leq n + 2 \wedge i \bmod 2 = 1 \wedge$$

$$suma = \sum_{k=3}^{i-2} \text{if } esPrimo(k) \text{ then } k \text{ else } 0 \text{ fi}$$

---

```
1  int suma = 0;
2  int i = 3;
3  while(i <= n) {
4      if (esPrimo(i)){
5          suma = suma + i;
6      }
7      i = i + 2;
8  }
9  return suma;
```

---

## Variante 2

Calcular la suma de todos los números primos positivos mayores a 2 hasta  $n$  (inclusive) respetando el siguiente invariante:

$$I \equiv 1 \leq i \leq n \wedge i \bmod 2 = 1 \wedge$$
$$suma = \sum_{k=i+2}^n \text{ if } esPrimo(k) \text{ then } k \text{ else } 0 \text{ fi}$$

## Variante 2

Calcular la suma de todos los números primos positivos mayores a 2 hasta  $n$  (inclusive) respetando el siguiente invariante:

$$I \equiv 1 \leq i \leq n \wedge i \bmod 2 = 1 \wedge$$

$$suma = \sum_{k=i+2}^n \text{if } esPrimo(k) \text{ then } k \text{ else } 0 \text{ fi}$$

---

```
1  int suma = 0;
2  int i = n;
3  if(i % 2 == 0 )
4      i--;
5  while(i > 2) {
6      if (esPrimo(i)){
7          suma = suma + i;
8      }
9      i -= 2;
10 }
11 return suma;
```

---