Introducción a la Programación

Cristian Nahuel Díaz

Primer Semestre de 2022

Clase práctica 7: Más listas, ciclos, invariantes.

Ejercicio 1 - Terminación

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

- i comienza valiendo 0

Ejercicio 1 - Terminación

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

- i comienza valiendo 0
- La lista l no se modifica o sea que $len(l)$ es fijo

Ejercicio 1 - Terminación

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

- i comienza valiendo 0
- La lista *l* no se modifica o sea que *len*(*l*) es fijo
- En cada iteración i se incrementa en 1

Ejercicio 1 - Terminación

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

- i comienza valiendo 0
- La lista l no se modifica o sea que $len(l)$ es fijo
- En cada iteración i se incrementa en 1
- Por lo tanto es inevitable que en algún momento $i = len(l)$, momento en el que la guarda evaluará a False y el ciclo terminará

Ejercicio 1 - Correctitud

$\mathcal{I}$:

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Vale $\mathcal{I}$ en (A)?

- $i = 0$ entonces vale $0 <= i <= len(l)$

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Vale $\mathcal{I}$ en (A)?

- $i = 0$ entonces vale $0 <= i <= len(l)$
- $vr = []$ entonces vale $len(vr) == 0 == i$

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Vale $\mathcal{I}$ en (A)?

- $i = 0$ entonces vale $0 <= i <= len(l)$
- $vr = []$ entonces vale $len(vr) == 0 == i$
- Como i = 0, el rango 0 a i-1 es vacío, entonces la 3er sentencia del invariante vale automáticamente

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Asumamos que $\mathcal{I}$ vale en (B). Sea K el valor de i

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Asumamos que $\mathcal{I}$ vale en (B). Sea K el valor de i

- Tenemos que $0 <= K < len(l)$
- Vale $len(vr) == K$
- Para j tal que $0 <= j <= K - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Asumamos que $\mathcal{I}$ vale en (B). Sea K el valor de i

- Tenemos que $0 <= K < len(l)$
- Vale $len(vr) == K$
- Para j tal que $0 <= j <= K - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

(B):

- Tenemos que $0 <= K < len(l)$
- Vale $len(vr) == K$
- Para j tal que $0 <= j <= K - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Vale $\mathcal{I}$ en (C)?

- Si K es par, se agrega a vr el valor de l[K]. Si es impar, se agrega a vr l[K]+n
- len(vr) se incrementa en 1, entonces vale $len(vr) == K + 1$
- i se incrementa en 1, entonces vale $i == K + 1$ (y que $i - 1 == K$)

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

Vale $\mathcal{I}$ en (C)?

- Si K es par, se agrega a vr el valor de l[K]. Si es impar, se agrega a vr l[K]+n
- len(vr) se incrementa en 1, entonces vale $len(vr) == K + 1$
- i se incrementa en 1, entonces vale $i == K + 1$ (y que $i - 1 == K$)
- Entonces $len(vr) = i$

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

Vale $\mathcal{I}$ en (C)?

- Si K es par, se agrega a vr el valor de l[K]. Si es impar, se agrega a vr l[K]+n
- len(vr) se incrementa en 1, entonces vale $len(vr) == K + 1$
- i se incrementa en 1, entonces vale $i == K + 1$ (y que $i - 1 == K$)
- Entonces $len(vr) = i$
- Para j tal que $0 <= j <= K$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

Vale $\mathcal{I}$ en (C)?

- Si K es par, se agrega a vr el valor de l[K]. Si es impar, se agrega a vr l[K]+n
- len(vr) se incrementa en 1, entonces vale $len(vr) == K + 1$
- i se incrementa en 1, entonces vale $i == K + 1$ (y que $i - 1 == K$)
- Entonces $len(vr) = i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

Vale $\mathcal{I}$ en (C)?

- Si K es par, se agrega a vr el valor de l[K]. Si es impar, se agrega a vr l[K]+n
- len(vr) se incrementa en 1, entonces vale $len(vr) == K + 1$
- i se incrementa en 1, entonces vale $i == K + 1$ (y que $i - 1 == K$)
- Entonces $len(vr) = i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar
- $0 <= K < len(l)$ equivale a $0 <= K + 1 <= len(l)$ y esto equivale a que $0 <= i <= len(l)$

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Vale $\mathcal{I}$ en (C)!

- Entonces $len(vr) = i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar
- $0 <= K < len(l)$ equivale a $0 <= K + 1 <= len(l)$ y esto equivale a que $0 <= i <= len(l)$

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

- Por último, en (D) vale el $\mathcal{I}$ y además que i == len(l)

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

- Por último, en (D) vale el $\mathcal{I}$ y además que i == len(l)
- Entonces vale que para j tal que $0 <= j <= len(l) - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

- Por último, en (D) vale el $\mathcal{I}$ y además que i == len(l)
- Entonces vale que para j tal que $0 <= j <= len(l) - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar
- También que $len(vr) == i == len(l)$

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

- Por último, en (D) vale el $\mathcal{I}$ y además que i == len(l)
- Entonces vale que para j tal que $0 <= j <= len(l) - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar
- También que $len(vr) == i == len(l)$

Ejercicio 1 - Correctitud

```
1  vr: List[int] = []
2  i: int = 0
3  # (A)
4  while i < len(l):
5    # (B)
6    if i % 2 == 0:
7      vr.append(l[i])
8    else:
9      vr.append(l[i]+n)
10   i = i + 1
11   # (C)
12 # (D)
13 return vr
```

$\mathcal{I}$:

- $0 <= i <= len(l)$
- $len(vr) == i$
- Para j tal que $0 <= j <= i - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar

- Por último, en (D) vale el $\mathcal{I}$ y además que i == len(l)
- Entonces vale que para j tal que $0 <= j <= len(l) - 1$, $vr[j] == l[j]$ si j es par y $vr[j] == l[j] + n$ si j es impar
- También que $len(vr) == i == len(l)$

Entonces vale el devuelve: $len(vr) == len(l)$, y en toda posición j entre 0 y len(l)-1: vr[j]==l[j] si j es par, o bien vr[j]==l[j]+n si j es impar