Ficha 1

1 Especificações

1. a)

```
int fa (int x, int y) {
    // pre: True
    ...
    // pos: (m == x || m == y) && (m >= x && m >= y)
    return m;
}
```

A função fa recebe 2 inteiros e retorna o maior entre eles.

b)

```
int fb (int x, int y) {
    // pre: x >= 0 && y >= 0
    ...
    // pos: x % r == 0 && y % r == 0
    return r;
}
```

A função fb recebe 2 inteiros não negativos e retorna o divisor comum entre x e y.

c)

```
int fc (int x, int y) {
    // pre: x > 0 && y > 0
    ...
    // pos: r % x == 0 && r % y == 0
    return r;
}
```

A função fb recebe 2 inteiros positivos e retorna o divisor comum entre x e y.

(...) resto é idêntico

2. a)

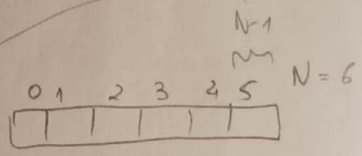
```
int prod (int x, int y) {
    // pre: True

    ...

    // pos: r == x * y
    return r;
}
```

b)

```
int mdc (int x, int y) {
    // pre: x > 0 && y > 0
    ...
    // pos: (x % r==0 && y % r==0) && (forall_{r'} x % r'==0 && y % r'==0
                                                  ⇓
                                               r >= r')
    return r;
}
```

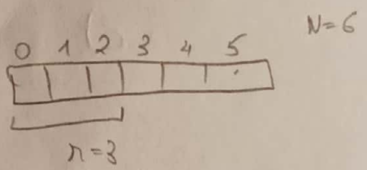
c)

```
int sum (int v[], int N) {
    // pre: N>0
    ...
    // pos: r == Sum_{0<= K<N} v[K]
    return r;
}
```



d)

```
int maxOrd ( int v[], int N) {
    // pre: N>0
    ...
    // pos: 1<=r<=N && (forall_{0<= k<r-1} v[k] <= v[k+1])
    return r;              && (v[r] < v[r-1] || r == N)
}
```



ou o de seguida é menor

ou termina o array

e) `int isSorted ( int v[], int N){`

0 1 2 3 4   N=5

```
   // pre: N>=0
   ...
   // pos: (r==-1 && exists_{0<=i<N} v[i] > v[i+1]) || (r==0 &&
                                  forall_{0<=i<N} v[i] <= v[i+1])
   return r;
}
```

② Correção

1\. a) $\{True\}$

$r = x + y;$

$\{r \geq x\}$

① Se $x = 0$
 $y = -1$

$\cancel{-1 \geq 0}$
 $r \geq x$ (F)

② $\{y \geq 0\}$
 $r = x + y;$
 $\{r \geq x\}$

③

$$\dfrac{\begin{array}{l} y \geq 0 \Rightarrow y \geq 0 \\ y \geq 0 \Rightarrow x + y \geq x \\ y \geq 0 \Rightarrow (r \geq x)[r \backslash x+y] \end{array}}{\{y \geq 0\}\ r = x + y;\ \{r \geq x\}}\ \text{Atr}$$

b) $\{True\}$

$x = x + y;\ y = x - y;\ x = x - y;$

$\{x == y\}$

① Se $x = 0$
 $y = -1$

$x = 0 - 1 = -1$
 $y = -1 - (-1) = 0$
 $x = -1 - 0 = -1$

$x = -1$
 $y = 0$
 $x = -1$

② $\{y == x\}$ $x = x + y;$ $y = x - y;$ $x = x - y;$ $\{x == y\}$

Fiz 1º isto para alterar isto

③

$$\dfrac{\dfrac{\dfrac{\begin{array}{l} y == x \Rightarrow y == x \\ y == x \Rightarrow 2y == x + y \end{array}}{\{y == x\}\ x = x + y;\ \{R'\}}\ \text{Atr} \quad \dfrac{\begin{array}{l} R' \Rightarrow -x + 2y == 0 \quad R' \Rightarrow 2y == x \\ R' \Rightarrow x - 2x + 2y == 0 \end{array}}{\{R'\}\ y = x - y;\ \{R\}}\ \text{Atr}}{\{y == x\}\ x = x + y;\ y = x - y\ \{R\}}\ \text{Seq} \quad \dfrac{\begin{array}{l} R \Rightarrow x - 2y == 0 \\ R \Rightarrow x - y == y \end{array}}{\{R\}\ x = x - y;\ \{x == y\}}\ \text{Atr}}{\{y == x\}\ x = x + y;\ y = x - y;\ x = x - y;\ \{x == y\}}\ \text{Seq}$$

c) $\{True\}$

$x = x+y;\ y = x-y;\ x = x-y;$

$\{x \neq y\}$

é inverso ao b)

nesse caso

① se $x = 1$
$y = 1$
⋮

② $\{x \neq y\}$ ....

③

d) $\{True\}$

if $(x>y)$ $r = x-y;$ else $r = y-x;$

$\{r>0\}$

① se $x = 0$
$y = 0$

$r = 0-0 = 0$
e $r>0$ é falso

② $\{x \mathrel{!=} y\}$
if $(x>y)$ $r = x-y;$ else $r = y-x;$
$\{r>0\}$

③

$$\cfrac{\cfrac{x>y \Rightarrow x>y \ \checkmark \qquad x>y \Rightarrow x-y>0}{\{x \mathrel{!=} y \wedge x>y\}\ r = x-y;\ \{r>0\}}\ \text{Atr} \qquad \cfrac{x<y \Rightarrow x<y \ \checkmark \qquad x<y \Rightarrow y-x>0}{\{x \mathrel{!=} y \wedge x<y\}\ r = y-x;\ \{r>0\}}\ \text{Atr}}{\{x \mathrel{!=} y\}\ \text{if}\ (x>y)\ r = x-y;\ \text{else}\ r = y-x;\ \{r>0\}}\ \text{if}$$

3 Invariantes

2. a)

```
int minInd (int v[], int N){
    // pre: N>0
    int i = 1, r=0;
    // inv: 0<=r<N && forall_{0<=k<i} v[r]<=v[k] && i<=N
    while (i<N) {
        if (v[i] < v[r]) r = i;
        i = i+1;
    }
    // pos: 0<=r<N && forall_{0<=k<N} v[r] <= v[k]
    return r;
}
```

V: N-i

(olhando para a pos e pre)

N=6

0	1	2	3	4	5
5	10	8	0	1	7

r = 3

Consideremos I inicialmente como:

I: 0<=r<N && forall_{0<=k<i} v[r] <= v[k]

→ Provar $\boxed{\text{Init}}$

(I)[i\1] (I)[r\0]

{N>0} i=1; r=0; {I}

N>0 ⇒ 0<=0<N && forall_{0<=k<1} v[0] <= v[k]

0<N ✓

único k possível é 0, logo v[0]<=v[0] ✓

Provou-se assim Init.

→ Provar $\boxed{\text{Util}}$

I ∧ ¬c ⇒ Q

0<=r<N && forall_{0<=k<i} v[r]<=v[k] ∧ i>=N ⇒ 0<=r<N && forall_{0<=k<N} v[r]<=v[k]

No fim temos i==N

logo com i>=N e faltava i<=N falha por em $\boxed{\text{I: } i<=N}$

→ Provar $\boxed{\text{Pos}}$

$\{I \wedge C\}\ S\ \{I\}$

$0<=\lambda<N$ && forall_{0<=k<i} $v[\lambda] < v[k]$ && $i<=N$ $\wedge$ $i<N$ if $(v[i] < v[\lambda])\ \lambda = i;\ i = i+1;$ $\{I\}$

Sai que $i<N$

Podemos ver isto de 2 formas:

→ Prática → Se $i<N$ então caso condição if seja True

$\lambda = i$
e como $i<N$
e $0<=\lambda<N$
então está visto para variável λ

e como $i = i+1$
no fim o máximo que pode ficar é $i==N$
e em I temos $i<=N$
logo está provado

Caso cond. if seja False ajeita-se

→ Prova

O → provado ou igual

$0<=\lambda<N$ && forall_{0<=k<i} $v[\lambda]<=v[k]$ && $i<N$ $\wedge$ $v[i] < v[\lambda]$ $\Rightarrow$ $0<=i<N$ && forall_{0<=k<i} $v[i] <= v[k]$ && $i+1<=N$

$v[\lambda] <= v[k]$ e $v[i] < v[\lambda]$ logo $v[i] <= v[k]$

$0<=\lambda<N$ && forall_{0<=k<i} $v[\lambda]<=v[k]$ $\wedge$ $i<N$ $\wedge$ $v[i] >= v[\lambda]$ $\Rightarrow$ $0<=\lambda<N$ && forall_{0<=k<=i} $v[\lambda] <= v[k]$ && $i+1<=N$

$k = \lambda$ a i

logo $v[\lambda] <= v[i]$

$\{P \wedge v[i]<v[\lambda]\}\ \lambda = i;\ R$ Atr $\quad$ $\{P \wedge v[i] >= v[\lambda]\} \Rightarrow R$

if

$\{P\}$ if $(v[i] < v[\lambda])\ \lambda = i;\ \{R\}$

$0<=\lambda<N$ && forall_{0<=k<=i} $v[\lambda] <= v[k]$ && $i+1<=N$ $(\Leftrightarrow i<=N)$

$R \Rightarrow 0<=\lambda<N$ && forall_{0<=k<i+1} $v[\lambda] <= v[k]$ && $i+1<=N$

$R \Rightarrow 0<=\lambda<N$ && forall - ...

Atr

$\{R\}\ i = i+1;\ \{I\}$

Seq

$\underbrace{\{0<=\lambda<N \text{ \&\& forall}_{0<=k<i}\ v[\lambda] <= v[k] \wedge i<N\}}_{P}$ if $(v[i] < v[\lambda])\ \lambda = i;$ $\quad i = i+1;$ $\quad \{I\}$

b)
```
int minimo (int v[], int N) {
    // pre: N>0
    int i=1, r= v[0];
    // inv: (forall_{0<=k<i} r<= v[k]) && (exists_{0<=p<N} r== v[p])
    while (i!=N) {
        if (v[i]<r) r = v[i];
        i= i+1;
    }
    // pos: (forall_{0<=k<N} r<= v[k]) && (exists_{0<=p<N} r== v[p])
    return r;
}
```

V: N-i

0	1	2	3
20	10	30	50
r=20	r=10	r=10	r=10
i=0	i=1	i=2	i=3

N=4

	r	i	N
v[0]	20	1	4
v[1]	10	2	4
	10	3	4
	10	4	4

I poderia ser: (forall_{0<=k<i} r<= v[k]) && (exists_{0<=p<N} r== v[p])

→ Prova $\boxed{\text{Init}}$

{N>0} i=1; r= v[0]; {I}

N>0 ⇒ forall_{0<=k<1} v[0]<= v[k] (k=0)
&& exists_{0<=p<N} v[0] == v[p] (p=0)

→ Prova $\boxed{\text{Util}}$

$I \wedge \underbrace{i==N}_{\neg C}$ ⇒ (forall_{0<=k<N} r <= v[k]) && (exists_{0<=p<N} r == v[p])

→ Prova $\boxed{\text{Pres}}$

{I ∧ C} if(v[i]< r) r = v[i]; i=i+1 {I}

(i!=N)

forall_{0<=k<N} r<= v[k] && (exists_{0<=p<N} r== v[p]) ∧ v[i]<r

⇓

forall_{0<=k<=i} v[i] <= v[k] && (exists_{0<=p<N} v[i] == v[p])

forall_{0<=k<N} r<= v[k] && exists_{0<=p<N} r== v[p] ∧ v[i]>=r ⇒ forall_{0<=k<i+1} r<= v[k] && exists_{0<=p<N} r== v[p]

p=i

{P ∧ (v[i]<r)} r= v[i]; {R}   Atr

{P ∧ v[i]>=r} ⇒ {R}

R ⇒ (forall_{0<=k<i+1} r <= v[k]) && exists_{0<=p<N} r== v[p]   Atr

{P} if (v[i]<r) r= v[i] {R}   if

{R} i= i+1; {I}

{(forall_{0<=k<N} r<= v[k]) && (exists_{0<=p<N} r== v[p])} if (v[i]<r) r= v[i]; i=i+1;   {I}   Seq

c) int Soma (int v[], int N) {

```
    // pre: N>0
    int i=0, r=0;
    // inv: r == Sum_{0<=k<i} v[k]
    while (i!=N) {
        r = r + v[i];
        i = i+1;
    }
    // pos: r == Sum_{0<=k<N} v[k]
    return r;
}
```

$\boxed{V: \ N-i}$

0	1	2	3	4	N = 5
10	5	7	2	8	

r	i	N
0	0	5
10	1	5
15	2	5
22	3	5
24	4	5
32	5	5

Olhando para a pos e pre, um possível I seria:

$r == Sum\_\{0 <= k < i\} \ v[k]$

→ Provar $\boxed{Init}$ $\{P\} \ldots \{I\}$

$\{N>0\}$ i=0; r=0; $\{I\}$

$N>0 \Rightarrow 0 == Sum\_\{0<=k<0\} \ v[k]$ ✓

$\sum_{k=0}^{-1} v[k] = 0$

$$\sum_{k=0}^{i-1} v[k] = r$$

$$\sum_{k=0}^{i} v[k] = \sum_{k=0}^{i-1} + v[i] = r + v[i]$$

→ Provar $\boxed{Util}$ $I \wedge \neg c \Rightarrow Q$

$\underbrace{r == Sum\_\{0<=k<i\} \ v[k] \wedge i==N}_{r == Sum\_\{0<=k<N\} \ v[k]} \Rightarrow r == Sum\_\{0<=k<N\} \ v[k]$ ✓

→ Provar $\boxed{Pres}$ $\{I \wedge c\}$ S $\{I\}$

$\{r==Sum\_\{0<=k<i\} \ v[k] \wedge i!=N\}$ r=r+v[i]; i=i+1; $\{r == Sum\_\{0<=k<i\} \ v[k]\}$

$r==Sum\_\{0<=k<i\} v[k] \wedge i!=N$

$\Rightarrow r+v[i] == Sum\_\{0<=k<i+1\} \ v[k]$ ✓

$Inc \Rightarrow r+v[i] == Sum\_\{0<=k<i+1\} \ v[k]$ Ah

$\{Inc\}$ r = r+v[i] $\{R\}$

$R \Rightarrow r == Sum\_\{0<=k<i+1\} \ v[k]$ Ah

$\{R\}$ i=i+1; $\{I\}$

Seq

$\{Inc\}$ r=r+v[i]; i=i+1; $\{I\}$

d)
```
int quadrado1 (int x) {
    // pre: x >= 0
    int a = x, b = x, r = 0;
    // inv: a * b + r == x^2
    while (a != 0) {
        if (a % 2 != 0) r = r + b;
        a = a/2; b = b*2;
    }
    // pos: r == x^2
    return r;
}
```

V: a

$a >= 0$ pq $a = x$ e $x >= 0$ e $a = a/2$

$x = 5$

a	b	r
5	5	0
2	10	5
1	20	5
0	40	25

Possível inv seria: $a*b + r == x^2$

→ Provar Init {P} ... {I}

$\{x >= 0\}\ a = x; b = x; r = 0;\ \{a*b + r == x^2\}$

$x >= 0 \Rightarrow x * x + 0 == x^2$

$x >= 0 \Rightarrow x^2 == x^2$ ✓

→ Prova Util $I \wedge c \Rightarrow Q$

$a*b + r == x^2 \wedge a == 0 \Rightarrow r == x^2$ ✓

→ Provar Pres {I ∧ c} S {I}

$\{a*b + r == x^2 \wedge a != 0\}$ if $(a\%2 != 0)$ $r = r + b;$ $a = a/2;$ $b = b*2;$ $\{a*b + r == x^2\}$

$a*b + r == x^2 \wedge a\%2 != 0 \wedge a != 0$ — a é do tipo $a = 2k+1$

$\Downarrow$

$\frac{a}{2} * b * 2 + r + b == x^2$ ✓

$(a+1)*b + r == x^2 \Leftrightarrow a*b + r + b == x^2$

$a*b + r == x^2 \wedge a\%2 == 0 \wedge a != 0 \Rightarrow \frac{a}{2}*b*2 + r == x^2$ ✓ pois é par

$$\frac{a*b + r == x^2 \wedge a \% 2 != 0 \wedge a != 0 \Rightarrow \{I \wedge c \wedge a\%2 != 0\}\ r = r + b;\ \{R'\}}{\{I \wedge c \wedge a\%2 != 0\}\ r = r+b;\ \{R'\} \quad \{I \wedge c \wedge a\%2 == 0\} \Rightarrow \{R'\}}\ \text{Ah}$$

$$\frac{\{I \wedge c\}\ \text{if}(a\%2 != 0)\ r = r+b;\ \{R'\} \quad \frac{R' \Rightarrow \frac{a}{2}*b*2 + r == x^2}{\{R'\}\ a = a/2\ \{R\}}\ \text{Ah} \quad \frac{R \Rightarrow a*b*2 + r == x^2}{\{R\}\ b = b*2;\ \{I\}}\ \text{Ah}}{\{I \wedge c\}\ S\ \{I\}}\ \text{Seq}$$

e) int quadrado2 (int x) {

```
    // pre : x >= 0
    int r = 0, i = 0, p = 1;
    // inv: r == i^2  ∧  i <= x  ∧  p == 2*i+1
    while ( i < x ) {
        i = i + 1;
        r = r + p;
        p = p + 2;
    }
    // pos : r == x^2
    return r;
}
```

V: x - i

x = 5

r	i	p
0	0	1
1	1	3
4	2	5
9	3	7
16	4	9
25	5	11

Possível I olhando para a pos e para a nossa tabela

Seria I : r == i^2

→ Testar $\boxed{\text{Init}}$ {P} ... {I}

{x >= 0} r = 0; i = 0; p = 1; { r == i^2 }

x >= 0 ⟹ 0 == 0^2 ✓

→ Testar $\boxed{\text{Util}}$ I ∧ ¬c ⟹ Q

r == i^2 ∧ i >= x ⟹ r == x^2

Sse i <= x

logo falta acrescentar ao I : r == i^2 ∧ i <= x

→ Testar $\boxed{\text{Pres}}$ {I ∧ c} S {I}

r == i^2 ⟹ r + p == i^2 + 2i + 1 Se e Só Se p == 2*i + 1 (que é o que falta acrescentar ao nosso invariante)

✓

então i < x

r == i^2 ∧ i < x ⟹ r + p == (i+1)^2 ∧ i + 1 <= x

$$\frac{\dfrac{\{I \wedge c\}\ i = i + 1;\ \{R'\} \quad \dfrac{R' \Rightarrow r + p == i^2 \wedge i <= x}{\{R'\}\ r = r + p\ \{R\}}\ \text{Atr} \quad \dfrac{R \Rightarrow I}{\{R\}\ p = p + 2;\ \{I\}}\ \text{Atr}}{\{r == i^2 \wedge i <= x \wedge i < x\}\ i = i + 1;\ r = r + p;\ p = p + 2;\ \{r == i^2 \wedge i <= x\}}\ \text{Seq}$$

0 1 2 3 4 | $N=5$

10	20	30	5	10

$r=3$

f) int maxPord (int v[], int N) {

// pre: $\boxed{N>0}$ se $r=0$ $N>=0$

$\boxed{V: N-r}$

int r = 1;

// inv: $\boxed{1<=r<=N \ \&\& \ forall\_\{0<=k<r-1\}\ v[k] <= v[k+1]}$

while (r<N && v[r-1] <= v[r])

r = r+1;

// pos: 1<=r<=N && (forall_{0<=k<r-1} v[k] <= v[k+1])

return r;   && ($\underbrace{v[r] < v[r-1]}_{\text{prox elem é menor}}$ || r==N) — em termina o array

}

$N=3$

0 1 2

10	20	30

$r=3$

I poderia ser: 1<=r<=N && forall_{0<=k<r-1} v[k] <= v[k+1]

→ Prova $\boxed{Init}$

{N>0} r = 1; { 1<=r<=N && (forall_{0<=k<r-1} v[k] <= v[k+1]) && (v[1] < v[0] || 1==N)

N>0 => 1<=1<=N ✓ && (forall_{0<=k<r-1} v[k] <= v[k+1]) && (v[r] < v[0] || 1==N)

→ Prova $\boxed{Util}$ Inv ∧ ¬c → Q

1<=r<=N && forall_{0<=k<r-1} v[k] <= v[k+1]

∧ => 1<=r<=N && (forall_{0<=k<r-1} v[k] <= v[k+1])

(r>=N ① || v[r-1] > v[r] ②)   && (v[r] < v[r-1] || r==N)

Se ① então 1<=r<=N && forall_{0<=k<r-1} v[k] <= v[k+1]) ∧ r>=N => ✓ // ✓

Se ② então 1<=r<=N && forall_{0<=k<r-1} v[k] <= v[k+1]) ∧ v[r-1] > v[r] => // ✓

✓

→ Prova $\boxed{Pres}$ {Inv ∧ c} S {I}

verdade pois ✓   r<N   r<N   k pode ser r-1 logo v[r-1] <= v[r]

1<=r<=N && forall_{0<=k<r-1} v[k] <= v[k+1] ∧ (r<N && v[r-1] <= v[r]) => 1<= r+1 <=N && forall_{0<=k<r} v[k] <= v[k+1]

{ 1<=r<=N && forall_{0<=k<r-1} v[k] <= v[k+1]) ∧ r<N && v[r-1] <= v[r]} r = r+1; { 1<=r<=N && forall_{0<=k<r-1} v[k] <= v[k+1] }

g) int procura (int x, int a[], int N){

//pre: N>0 $\boxed{V: N-i}$

int p = -1, i = 0; //inv: $\boxed{(p==-1 \;\&\&\; \text{forall\_}\{0<=k<i\}\; a[k]!=x \;\&\&\; i<=N) \,||\, (0<=p<i \;\&\&\; x==a[p])}$

while (p==-1 && i<N) {

if (a[i]==x) p=i;

i = i+1;

}

// pos: (p==-1 && forall_{0<=k<N} a[k]!=x) ||
((0<=p<N) && x==a[p])

return p;

}

x = 10

0	1	2	3	4
5	7	1	10	2
i=0	i=1	i=2	i=3	i=4
p=-1	p=-1	p=-1	p=3	

e pela pos:

Invariante I seria: $(p==-1 \;\&\&\; \text{forall\_}\{0<=k<i\}\; a[k]!=x) \,||\, (0<=p<i \;\&\&\; x==a[p])$

→ Provar $\boxed{\text{Init}}$ {P}... {I}

{N>0} p=-1; i=0; {I}

é sempre verdade logo ✓

$N>0 \Rightarrow (-1==-1 \;\&\&\; \text{forall\_}\{0<=k<0\}\; a[k]!=x) \,||\, (0<=-1<0 \;\&\&\; x==a[-1])$

→ Provar $\boxed{\text{Util}}$ I ∧ nc ⇒ Q

temos i>=N logo falta provar I: i<=N para isto estar sempre em N

$(p==-1 \;\&\&\; \text{forall\_}\{0<=k<i\}\; a[k]!=x) \,||\, (0<=p<i \;\&\&\; x==a[p]) \Rightarrow (p==-1 \;\&\&\; \text{forall\_}\{0<=k<N\}\; a[k]!=x) \,||\, (0<=p<N \;\&\&\; x==a[p])$

∧ (p!=-1 || i>=N)

✓

Logo I seria agora $(p==-1 \;\&\&\; \text{forall\_}\{0<=k<i\}\; a[k]!=x) \,||\, (0<=p<i \;\&\&\; x==a[p]) \;\&\&\; i<=N$

✓

→ Provar $\boxed{\text{Pres}}$ {I ∧ c} S {I}

a[i]!=x

p==-1 && x

$(p==-1 \;\&\&\; \text{forall\_}\{0<=k<i\}\; a[k]!=x) \,||\, (0<=p<i \;\&\&\; x==a[p]) \;\&\&\; i<=N \wedge p==-1 \wedge i<N \wedge a[i]!=x \Rightarrow (p==-1 \;\&\&\; \text{forall\_}\{0<=k<i+1\}\; a[k]!=x) \,||\, (0<=p<i+1 \;\&\&\; x==a[p]) \;\&\&\; i+1<=N$

por i<N logo i+1<=N

⊕

Atr

{I ∧ p==-1 && i<N ∧ a[i]==x} p=i; {R}

I ∧ p==-1 && i<N ∧ a[i]!=x ⇒ R

R ⇒ $(p==-1 \;\&\&\; \text{forall\_}\{0<=k<i+1\}\; a[k]!=x) \,||\, (0<=p<i+1 \;\&\&\; x==a[p]) \wedge i+1<=N$

Atr

{I ∧ c} if(a[i]==x) p=i; {R} if {R} i = i+1 {I}

Seq

{I ∧ c} if (a[i]==x) p=i; i=i+1; {I}

$(p==-1$ && forall_{0<=k<i} a[k] != x) ||
(0<=p<i && x==a[p]) && i<=N

$\wedge i<N$ $\wedge p==-1$ $\wedge a[i]==x$

$\Longrightarrow$

(i==-1 && forall_{0<=k<i+1} a[k] != x) ||
(0<=i<i+1 && x==a[i]) $\wedge$ i+1<=N $\rightarrow$ i<=N logo i<N

Ah

{I $\wedge$ p==-1 && i<N $\wedge$ a[i]==x} p=i; {R}

b)

```
int procura2(int x, int a[], int N){
    //pre: N>0 && forall_{0<=k<N-1} a[k-1] <= a[k]
    int p=-1, i=0;
    //inv: ??
    while (p==-1 && i<N && x >= a[i]){
        if (a[i]==x) p=i;
        i = i+1;
    }
    //pos: (p==-1 && forall_{0<=k<N} a[k] != x) || ((0<=p<N) && x == a[p])
    return p;
}
```

todos elem [0 ... N-1] estão ord. crescente

V: N-i

igual ao anterior? Falta ver = Pos

x=20 N=4

0	1	2	3
10	20	30	5

i=0 p=-1; i=1 p=1; i=2 p=1

Possível I: (p==-1 && forall_{0<k<i} a[k] != x) ||
(0<=p<i && x==a[p])

→ Provar Init {P} ... {I}

N>0 && forall_{0<=k<N-1} a[k-1] <= a[k] $\Longrightarrow$ (-1==-1 && forall_{0<k<0} a[k] != x) ||
(0<=p<0 && x==a[0])

→ Provar Util I $\wedge$ ¬c $\Rightarrow$ Q

(p==-1 && forall_{0<k<i} a[k] != x) ||
(0<=p<i && x==a[p])
$\wedge$ p!=-1 $\wedge$ i>=N $\wedge$ x<a[i]

$\Longrightarrow$ (p==-1 && forall_{0<=k<N} a[k] != x) || ((0<=p<N) &&
x == a[p])

falta acrescentar ao I i<=N pois assim i>=N, 0<i<=N > i==N

Soma dos 1^{os} n^{os} inteiros

$0 - 1 + 1 = 0$ ✓

j) int triangulo (int n){

// pre: (n>=0)

int r=0, (i=1);

$r == \sum_{k=1}^{i-1} k = \frac{i(i-1)}{2}$ ou

V: $n - i + 1$

// inv: $\boxed{r == Sum\_\{1<=k<i\} k}$

while (i != n+1) {

r = r+i;

i = i+1;

}

$r == \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

// pos $r == \frac{n * (n+1)}{2}$;

return r;

}

n = 4

r	i	n
0	1	4
1	2	4
3	3	4
6	4	4
10	5	4

A cada i, os i-1 1os nos somados dão o r

Possível I: $r == Sum\_\{1<=k<i\} k$

→ Provar $\boxed{Init}$ {P} ... {I}

{n>=0} r=0; i=1; {r == Sum_{1<=k<i-1} k}

n>=0 ⟹ 0 == $\underbrace{Sum\_\{1<=k<1\} k}$ ✓

$r = \sum_{k=1}^{0} k = 0$

→ Provar $\boxed{Util}$ I ∧ ¬c ⟹ Q

{r == Sum_{1<=k<i} k ∧ i == n+1} ⟹ r == Sum_{1<=k<n+1} k ✓

n+1

→ Provar $\boxed{Pres}$ {I ∧ C} S {I}

$r == \sum_{k=1}^{i-1} k$

r == Sum_{1<=k<i} k ∧ i != n+1 ⟹ r+i == Sum_{1<=k<i+1} k

✓ ⟹ $r + i == \sum_{k=1}^{i} k$

$\sum_{k=1}^{i-1} k + i$

✓ R ⟹ r == Sum_{1<=k<i+1} k

$$\frac{\{I \wedge C\} \quad r = r+i; \quad \{R\}}{} \text{Ah} \qquad \frac{\{R\} \quad i = i+1; \quad \{I\}}{} \text{Ah}$$

Seq

{r == Sum_{1<=k<i} k ∧ i != n+1} r=r+i; i=i+1; {r == Sum_{1<=k<i} k}

I   C   I

k) int triangle2 (int n) {

// pre : n >= 0

int r = 0, i = n;

// inv: $\boxed{(n+i+1)(n-i)/2 \wedge i >= 0}$

while (i > 0) {

r = r + i;

i = i - 1;

}

// pos: $r == \frac{n*(n+1)}{2}$

return r;

}

$\boxed{V: i}$

$\sum_{k=a}^{b} k = \frac{(a+b)(b-a+1)}{2}$

n = 3

r	i
0	3
3	2
5	1
6	0

$i = n \quad + r = 0$

$i = (n-1 \quad + r = n$

$i = n-2 \quad r = (n-1)+n$

$i = n-3 \quad r = (n-2) + (n-1) + n$

0 (n-i)
1
2
3

desta forma como no ex. anterior

$r == \frac{(n+i+1)(n-(i+1)+1)}{2}$

Logo I possível, I : $\frac{(n+i+1)(n-i)}{2} == r$

→ Prova $\boxed{Init}$ {Q}...{I}

$\{n>=0\}\ r = 0;\ i = n;\ \{r == \frac{(n+i+1)(n-i)}{2}\}$

$n >= 0 \Rightarrow 0 == \frac{(2n+1)(n-n)}{2}$ ✓

0 == 0.

→ Prova $\boxed{Util}$ $I \wedge \neg c \Rightarrow Q$

$r == \frac{(n+i+1)(n-i)}{2} \wedge i <= 0 \Rightarrow r == \frac{n*(n+1)}{2}$

i tem de ser 0 e com i <= 0 falta por no I i >= 0

→ Prova $\boxed{Pres}$ {I ∧ c} S {I}

✓

$r = \frac{n^2 + \cancel{in} + n - (\cancel{n i} + i^2 + i)}{2} = \frac{n^2 + n - i^2 - i}{2}$

⇓

$r+i == \frac{n^2 - \cancel{in} + n + \cancel{in} - i^2 + i}{2} = \frac{n^2+n-i^2+i}{2}$

$r == \frac{n^2+n-i^2+i}{2} - i \frac{(2)}{} = \frac{n^2+n-i^2+i-2i}{2}$

$= \frac{n^2+n-i^2-i}{2}$

$r == \frac{(n+i+1)(n-i)}{2} \wedge i >= 0 \wedge i > 0 \Rightarrow r+i == \frac{(n+i)(n-i+1)}{2} \wedge i >= 1$

i >= 1

i - 1 >= 0

$R \Rightarrow r == \frac{(n+i-\cancel{1}+\cancel{1})(n-i+1)}{2} \wedge$

$\{I \wedge c\}\quad r = r + i;\quad \{R\}$

$\{R\}\quad i = i - 1;$

Ah

Seg

$\{I \wedge c\} \quad r = r + i; \quad i = i - 1; \quad \{I\}$

b) int mod (int x, int y) {

V: r - (x % y) → copiei

// pre: x >= 0 && y > 0

int r = x; // inv: $\boxed{r \geq 0 \; \&\& \text{ exists\_\{q\} } x == q*y + r}$

while (y <= r) {

r = r - y;

}

// pos: 0 <= r < y && exists_{q} x == q*y + r.

return r;

}

x = 6

y = 4

daqui onde se retira

r	y	x
~~6~~	4	6
2	4	6

Loop possível I: $r \geq 0 \wedge$ exists_{q} x == q*y + r

→ Provar $\boxed{\text{Init}}$ {P} ... {I}

{x >= 0 && y > 0} r = x; {r ≥ 0 ∧ exists_{q} x == q*y + r}

x >= 0 && y > 0 ⟹ x >= 0 && exists_{q} x == q*y + r ∧ r ≥ 0

→ Provar $\boxed{\text{Util}}$ I ∧ ¬c ⟹ Q

r ≥ 0 ∧ exists_{q} x == q*y + r ∧ y > r

0 <= r < y

⟹ 0 <= r < y && exists_{q} x == q*y + r

→ Provar $\boxed{\text{Pres}}$ {I ∧ c} S {I}

x == (q-1)*y + r

r ≥ y

r >= 0 && exists_{q} x == q*y + r ∧ y <= r ⟹ r - y >= 0 && exists_{q} x == q*y + r - y

{I ∧ c} r = r - y; {I}

m) float valor1 (float x, float coef[], int N){

// pre: N>=0

float r=0; int i=0;

$\boxed{V: N-i}$

// inv: $\boxed{r == Sum\_\{0<=k<i\}\ x\hat{}k * coef[k] \wedge i<=N}$

while (i<N){

r = r + pow(x,i) * coef[i];

i = i+1;

}

// pos: r = Sum_{0<=k<N} x^k * coef[k]

return r;

}

x=2, N=4

	0	1	2	3
Coef	2	0	3	5

$2^0 * 2 + 2^1 * 0 + 2^2 * 3 + 2^3 * 5$

i=0, i=1, i=2, i=3

i=0 r = $2^0 * 2$

e só depois se incrementa

logo possível I:

r == Sum_{0<=k<i} x^k * coef[k]

→ Provar $\boxed{Init}$ {P} .. {I}

{N>=0} r=0, i=0; {r == Sum_{0<=k<i} x^k * coef[k]}

N>=0 ⟹ 0 == Sum_{0<=k<0} x^k * coef[k]

0==0 ✓ → $\sum_{k=0}^{-1} \cdots = 0$

→ Provar $\boxed{Util}$ $I \wedge \neg c \Rightarrow Q$

r == Sum_{0<=k<i} x^k * coef[k] ∧ i>=N ⟹ r == Sum_{0<=k<N} x^k * coef[k]

falta acrescentar ao I $\boxed{i<=N}$ (i<=N ∧ i>=N > i==N)

→ Provar $\boxed{Pres}$ {I ∧ c} S {I}

r == Sum_{0<=k<i} x^k * coef[k] ∧ i<N ⟹ r + x^i * coef[i] == Sum_{0<=k<i+1} x^k * coef[k] ∧ i+1≤N (Atr)

R ⟹ r == Sum_{0<=k<i+1} x^k * coef[k] ∧ i+1≤N (Atr)

{I∧C} r = r+pow(x,i) * coef[i] {R} . {R} i = i+1; {I} (Seq)

{I∧C} r = r+pow(x,i) * coef[i]; i = i+1; {I}

```
n) float valor2 (float x, float coef[], int N) {
      // pre: N >= 0
      float r=0; int i=N;
      // inv: [            ]
      while (i>0) {
          i = i-1;
          r = (r*x) + coef[i];
      }
      // pos: r = Sum_{0<=k<N} x^k * coef[k]
      return r;
}
```

V: i

$x = 2$ $N = 4$

	0	1	2	3
coef	2	0	3	5

$i = 4$

$i=3$ $r = (0*2) + 5 = 5$	C_3
$i=2$ $r = 5*2 + 3$	$x*C_3 + C_4$
$i=1$ $r = 5*2*2 + 3*2 + 0$	$x^2 * C_3 + C_4 * x + C_5$
$i=0$ $r = 5*2*2*2 + 3*2*2 + 2$	$x^3 * C_3 + C_4 * x^2 + C_5 * x + C_6$

$$r == \sum_{k=i}^{N} x^k * coef[K]$$

$\Downarrow$

N=4

0	1	2	3
2	0	3	5

C_3

$C_4 + C_3 * x$

C_6 $C_5 + C_4 * x + C_3 * x^2$

$+ C_5 * x + C_4 * x^2 + C_3 * x^3$