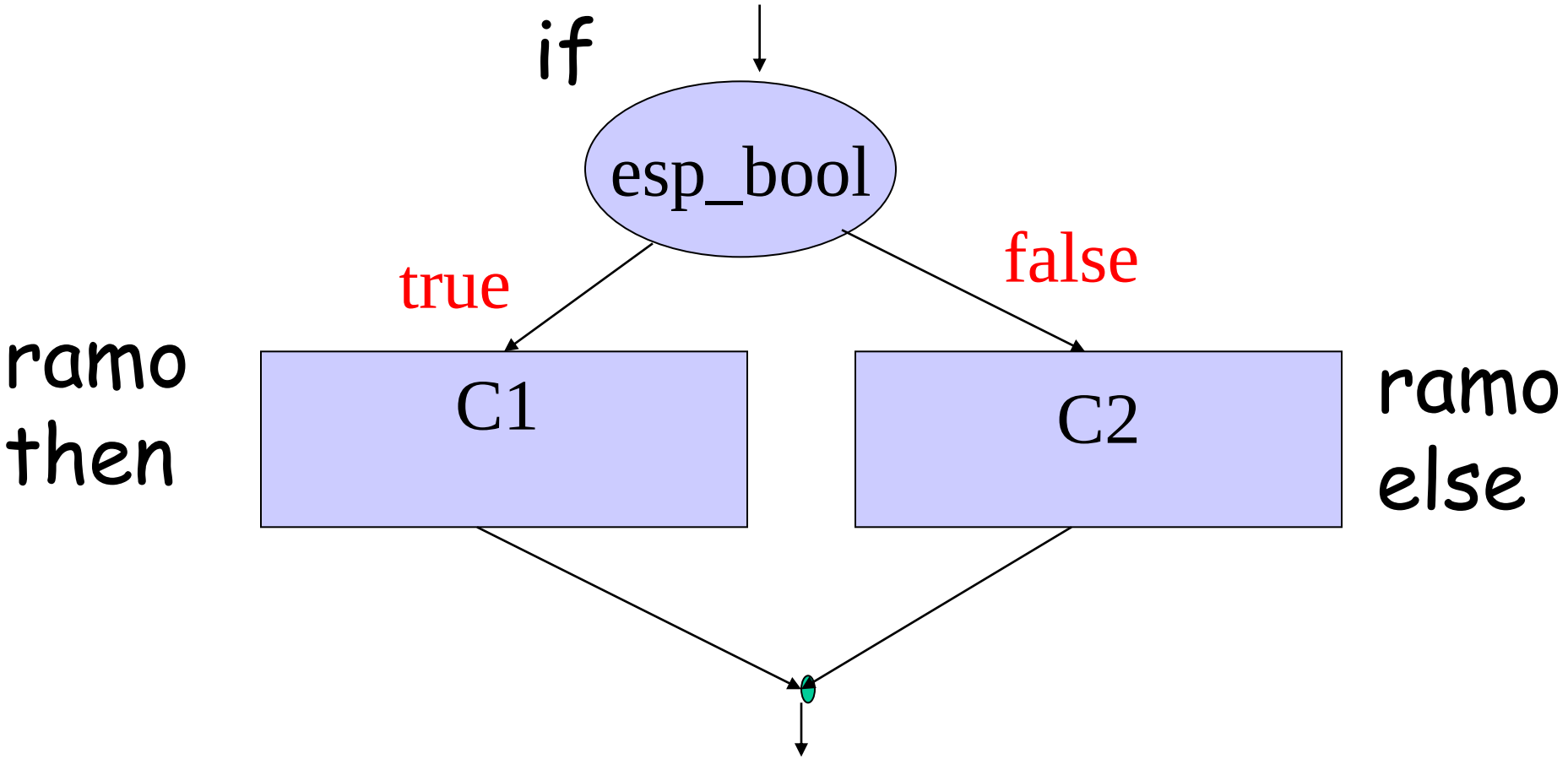
primi esempi di correttezza

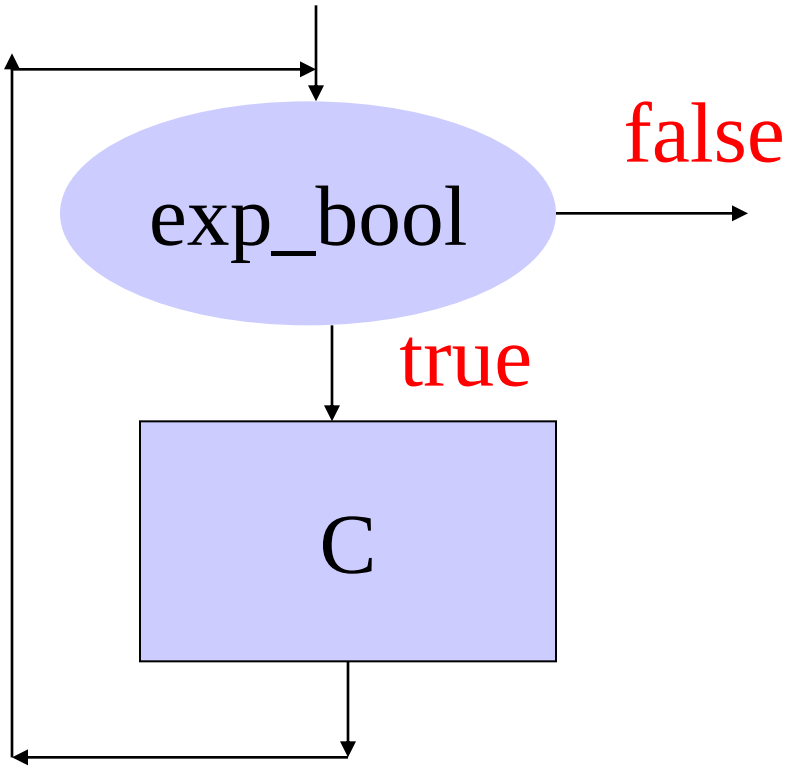
Capitolo 3 del testo

condizionale o if-then-else



1 punto d'entrata ed 1 d'uscita

while



1 punto d’entrata ed 1 d’uscita

esercizio 1

PRE=("input" contiene un intero n>=0, seguito da n interi)

POST=("output" deve contenere n e poi il massimo tra gli n interi che su "input" seguono il valore n)

idea: leggiamo da input un valore alla volta e lo confrontiamo col massimo trovato fino a quel momento
e all'inizio ??? INT_MIN andrà bene !!

PRE=("input" contiene un intero n>=0, seguito da n interi)

POST=("output" deve contenere n e poi se n=0 contiene INT_MIN e altrimenti contiene il massimo tra gli n>0 valori che su "input" seguono il valore n)

comunque se n=0 dovrei fare qualcosa di speciale

```
main()
{
 int n, n_use, x, max;
 ifstream INP("input");
 ofstream OUT("output");
 if(INP && OUT)
  {//PRE
    max=INT_MAX;  INP>>n;
    n_use=n;
    while(n_use>0) // restano da leggere n_use valori
     {                    //se n=n_use allora max = INT_MIN
       INP>>x;            //altrimenti max è massimo dei n-n_use valori letti
       if(x>max)
          max=x;
       n_use=n_use-1;
     }
     OUT<< n<< '  '<<max;   //POST
     INP.close();
     OUT.close();
    }
 else
  cout<<"errore nell'apertura dei file"<<endl;
}
```

INVARIANTE

INVARIANTE

n>=0 e 0<=n_use <=n

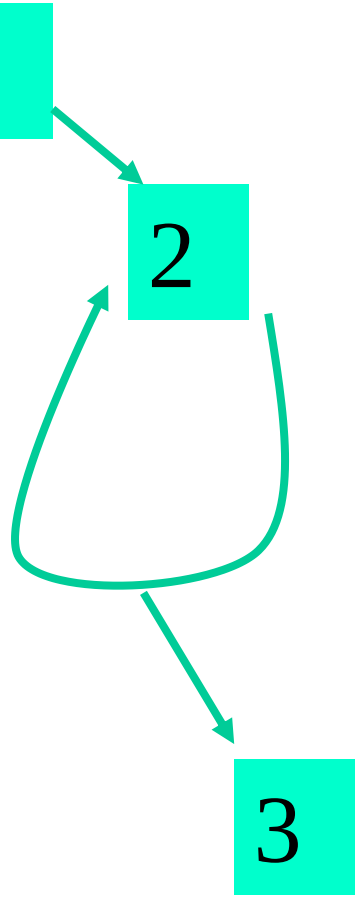
letti n-n_use valori

se n-n_use=0 allora max=INT_MIN

altrimenti max è il massimo tra gli n-n_use valori letti

OSSERVARE che se n>0, quando n_use =0, abbiamo letto n valori e max è il massimo tra questi !

```
max=INT_MAX; INP>>n;
 n_use=n;
 while(n_use>0)  Invariante
  {
    INP>>x;
    if(x>max)
        max=x;
    n_use=n_use-1;
  }
```



1 condizione iniziale

2 invarianza

3 condizione d'uscita

esercizio 2

PRE = come per l'Esercizio 1,

POST=(se n=0, "output" deve contenere la stringa "n=0", se n>0, "output" deve contenere n seguito dal massimo degli n valori che seguono su "input" il valore n)

nel caso n=0 dobbiamo fare qualcosa di speciale che possiamo scegliere

```
INP>>n;
n_use=n;
if(n>0)
 {
  INP>>max;
  n_use = n_use-1;
  while(n_use>0)
   {
     INP>>x;
     if(x>max)
       max=x;
     n_use=n_use-1;
   }
  OUT<< n<< ' '<<max;
 }
else
 OUT<<"n=0";
```

INVARIANTE
n>0
0<=n_use <n
letti n-n_use
 max è il massimo dei valori letti (almeno 1)

Esercizio 3: trovare il massimo e il minimo

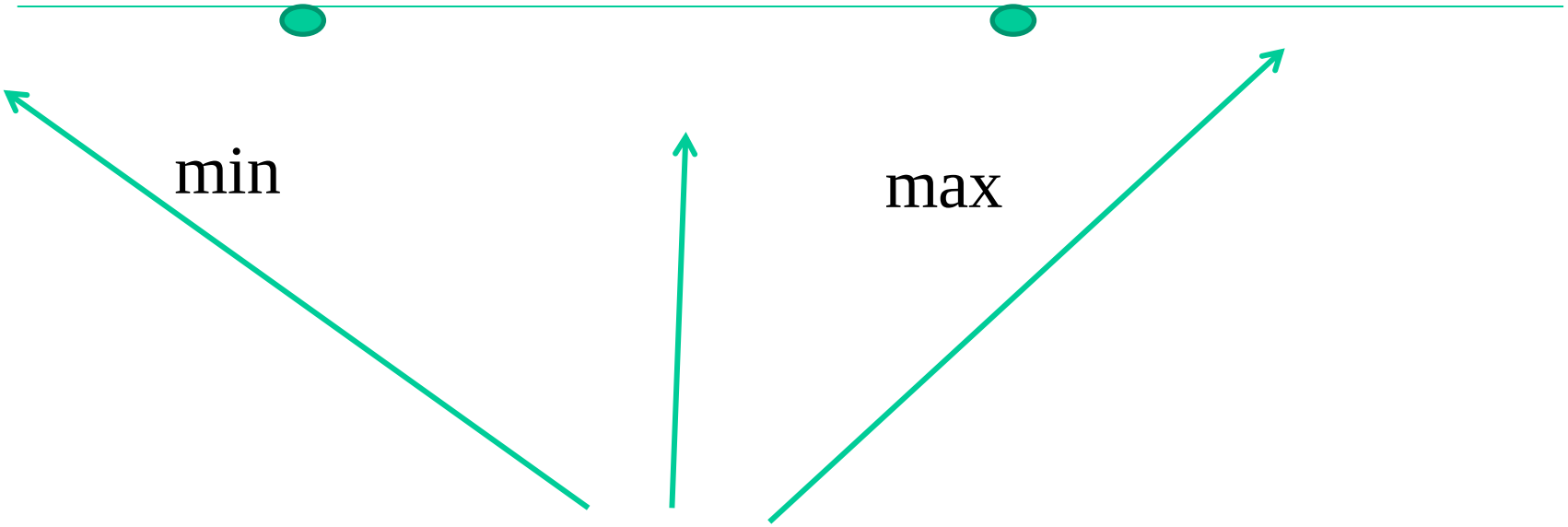
PRE1=("input" contiene n>=0, seguito da (almeno) n interi)

POST1=("output" contiene sempre n, che, a seconda del valore di n, è seguito dai seguenti valori:
i) se n=0, è seguito da INT_MIN e INT_MAX,
ii) se n>=1, è seguito dal massimo e dal minimo dei valori che seguono n su "input")

```
main()
{int n, n_use, max=MIN_INT, min=MAX_INT;
INP>>n; n_use=n;
while(n_use>0)
{
  INP>> x;
  if(max<x)
     max=x;
  if(min>x)
    min=x;
  n_use=n_use-1;
}
OUT<<n<<‘ ‘<<max<<‘ ‘<< min<<endl;
}
```

INVARIANTE
n>=0
0<=n_use<=n
letti n-n_use valori
se letti 0 allora INT_MIN e INT_MAX
altrimenti max e min dei valori letti

ma si può fare meglio? Beh si !!



se leggiamo nuovo x

se cambia max inutile
guardare se cambia min

E' SBAGLIATO !!

```
{int  n, n_use, max=MIN_INT, min=MAX_INT;
  INP>>n; n_use=n;
  while(n_use >0)
  {
    INP>> x;
    if(max<x)
      max=x;
    else
      if(min>x)
        min=x;
     n_use=n_use-1;
}
OUT<<n<<`` `<<max<<`` `<< min<<endl;
}
```

INVARIANTE
n>=0
0<=n_use<=n
letti n-n_use valori
se letti 0 allora INT_MIN e INT_MAX
altrimenti max e min dei valori letti

Ma l’idea di base non è sbagliata, solo non funziona assieme all’inizializzazione

```
max=MIN_INT, min=MAX_INT;
```

```
{int  n, n_use, max,min;
 INP>>n; n_use=n;
 if(n>0)
 { INP>>max; min=max; n_use=n_use-1;
  while(n_use >0)
  {
   INP>> x;
   if(max<x)
     max=x;
   else
     if(min>x)
       min=x;
    n_use=n_use-1;
  }
 OUT<<n<<' '<<max<<' '<< min<<endl;
 }
else OUT<<"problema impossibile"<<endl;
```

cambiare inizializzazione

INVARIANTE
n>0
0<=n_use<n
letti n-n_use valori
max e min sono massimo e minimo dei valori letti

Esercizio 4.2 del testo:

dato X >=1 vogliamo calcolare il minimo esponente tale che $2^{esponente}$ >= X

PRE=(cin contiene intero >0)

```
int X, potenza=1, esponente=0;
cin >> X;
while( X > potenza)
{
      potenza=potenza*2;
      esponente = esponente + 1;
}
cout<< "l'esponente e'=" << esponente<<endl;
```

R = invariante

POST= (esponente è il minimo esponente tale che $2^{esponente} >= X$)

L'invariante R

R=(
  X>=1,
  potenza=$2^{\text{esponente}}$,
  $X > 2^{(\text{esponente}-1)}$
)

test di permanenza nel ciclo: X> potenza

esercizio risolto 4.3: si legge sequenza di caratteri numerici fino a leggere 'a'

esempio: '3' '2' '6' 'a'

il programma deve calcolare l'intero 326

PRE=(cin contiene c1…ck 'a', k>=0, cj è un carattere numerico, per ogni j ∈ [1,k])

programma ?

POST=(calcola NUM(c1…ck))

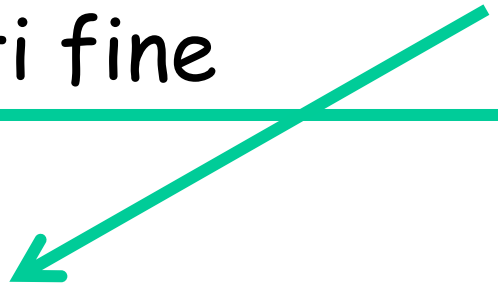
NUM('2' '3' '1')=231 NUM()=0

inizio :

leggi un carattere

se è numerico fai i conti e torna all'inizio

altrimenti fine



fai i conti ?

se abbiamo letto

c1c2c3 abbiamo NUM(c1c2c3) e se c4 !='a'

dobbiamo calcolare NUM(c1c2c3c4)

come ??

NUM(c1c2c3c4)=10*NUM(c1c2c3)+NUM(c4)

```
char q; int num=0, n=1;
IN>>q;
```

R=(n>=1, num=NUM(c1..c(n-1)), q=cn)

```
while(q != 'a')
{
  num=num*10+(q-'0');
  IN >> q;
  n++;
}
```