Titolo nota 29/10/2015

Scrivere una procedura in C con intestazione

void once (int a[], int b[], int dima, int dimb, int *num)

che presi due array a e b e le loro dimensioni, restituisce nella variabile puntata da num il numero dei valori di a che compaiono esattamente una volta in b

```
main()
{ int v1[5], int v2[4];
  int x;
  ⋮
  once(v1, v2, 5, 4, &x);
}
```

Titolo nota 29/10/2015

a | 10 | 20 | 10 | -3 |

b | 8 | 7 | -3 | 10 | 20 | 20 |

che valore devo lasciare in *num ??

1 + 1 + 1 = 3

```
void once (int a[], int b[], int dima, int dimb, int *num)
{ int i, int j, int ~~conta~~ = 0; *num = 0;
  for (i = 0; i < dima; i++)
  { int occ = 0;
    for (j = 0; j < dimb; j++)
      if (a[i] == b[j]) occ = occ + 1;
    if (occ == 1) ~~conta = conta~~ + 1; }
      *num = *num + 1;
  ~~*num = conta~~;
}
```

in questo ciclo a[i] è un valore fissato

Titolo nota 29/10/2015

Possiamo rimpiazzare il secondo for con una iterazione indeterminata

```
{ int occ = 0; J = 0;
  while (J < dimb && occ < 2)
  { if (a[i] == b[J]) occ = occ + 1;
    J = J + 1;
  }
  if (occ == 1) *num = *num + 1;
}
```

rimpiazza il corpo del `for(i=0, ....)` di prima

Titolo nota 29/10/2015

Specifichiamo formalmente il problema, ovvero:

"Il numero dei valori di a che occorrono esattamente una volta in b"

$$\#\{ i \mid i \in [\phi, \text{dim}b) \wedge b[i] = n \}$$

fissato un valore n indica quanti sono (gli indici di) i valori in b uguali a n

$$\#\left\{ J \mid J \in [\phi, \text{dim}a) \wedge \#\{ i \mid i \in [\phi, \text{dim}b) \wedge b[i] = a[J] \} = 1 \right\}$$

Titolo nota 29/10/2015

Scrivere una funzione in C

int check (int a[], int dim)

che controlla che NON ci siano in a tre elementi contigui con lo stesso valore.

es. | 2 | 3 | 5 | -1 | 7 | 7 | 8 | 7 | -1 | true (cioè 1)

| 2 | 3 | 5 | -1 | 7 | 7 | 7 | 8 | -1 | false (cioè ϕ)

"Non esiste un indice i tale che

$a[i] = a[i+1] \wedge a[i] = a[i+2]$" (quali possono essere i valori leciti per i?)

$$\neg \left(\exists i \,.\, i \in [\phi, dim-2) \wedge a[i] = a[i+1] \wedge a[i] = a[i+2] \right)$$

$i \in [\phi, dim-3]$

Titolo nota 29/10/2015

Come si risolve un problema del tipo

(*) $\neg(\exists i.\ i \in [x,y) \wedge P(i))$

la proprietà P deve essere esprimibile in sintassi C

```
i = x; trovato = ~~0~~; 1
while (i<y && !trovato)
    if P(i) trovato = ~~1~~; 0
    else i = i+1;
```

Alla fine di questo ciclo, il valore di verità della formula (*) è il valore di !trovato

Titolo nota 29/10/2015

Per risolvere il problema di prima istanziamo lo schema con

$x = 0$, $y = dim - 2$ $\quad P(i) \equiv a[i] == a[i+1]$ && $a[i] == a[i+2]$

```
int check (int a[], int dima)
{ int i = 0; int trovato = 0;
  while (i < dim-2 && !trovato)
    if (a[i] == a[i+1] && a[i] == a[i+2])
        trovato = 1;
    else i = i+1;
  return (!trovato);
}
```

Titolo nota 29/10/2015

Domande: possiamo scriverlo così?

```
while ( i < dima - 2 )        && ! trovato
{  salva = a[i];
   if ( salva == a[i+1] && salva == a[i+2] )
        i = dima - 2;        <-   trovato = 1
   else  i = i+1;            <-
}
return
```

NO

Titolo nota 29/10/2015

while (A && B) { - - - - }

dopo il ciclo sono sicuro che vale

$\neg (A \wedge B)$ $\neg A \vee \neg B$

```
int true = 0;
while (i < dim && !true)
 { if (i+1 < dim && a[i]==a[i+1])
        if (i+2 < dim && a[i+1]==a[i+2])
              true = 1;
   i = i+1; }
return (!true);
```

Titolo nota 29/10/2015

Il C valuta un && in questo modo A && B (CAND conditional and)

- valuta A. Se A è <u>false</u> calcola come risultato <u>false</u>
  Se A è <u>true</u> calcola come risultato B

```
i+1 < dim && a[i] < a[i+1]
```

Il CAND NON è commutativo

in C `x>y && 3/0 == 1`

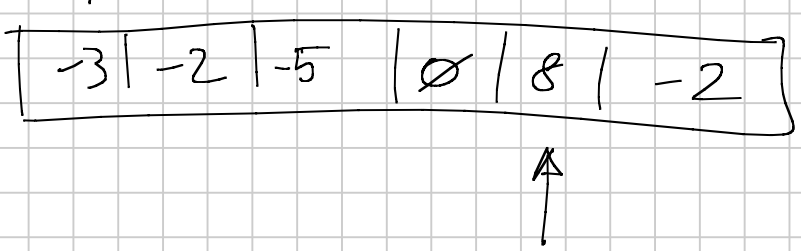
non genera errore se x non è maggiore di y

Titolo nota 29/10/2015

Scrivere in C una funzione

int SUM (int a[], int dim)

che calcola la somma di tutti i valori di a che precedono il primo valore positivo in a.



-10

Se non ci sono valori positivi nell'array restituisce la somma di tutti i valori dell'array.

```
int sum (int a[], int dim)
{ int i; int ris = 0;
  i = 0; int trovatopos = 0;
  while (i < dim && !trovatopos)
    if (a[i] > 0) trovatopos = 1;
      else { ris = ris + a[i]; i = i+1; }

  return (ris);
}
```

```
while ( i < dim && a[i] < 0)
  ris = ris + a[i];
```