

PROBLEM SOLVING SU ARRAY

Lo GUARDATE ANCHE GLI APPUNTI DI BARBONI
(CORSO A) DEL 2/10

```

int member (int a[], int dim, int x) {
  int i=0;
  int Trovato = 0;
  while (i < dim && !Trovato) {
    if (a[i] == x) Trovato = 1;
    i++;
  }
  return Trovato;
}

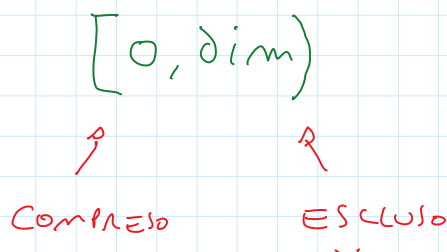
```

UNA
VARIABILE
USATA IN
QUESTO
MODO
VIENE DETTA
"FLAG"

RISPONDE A QUESTA DOMANDA:

$$\exists i \in [0, dim) . a[i] == x \quad ?$$

ESISTE UN INDICE i COMPRESO TRA 0 E
 dim (QUEST'ULTIMO ESCLUSO) TALE CHE
 $a[i]$ È UGUALE A x ?



RAPPRESENTA L'INTERVALLO
CHE COMPRENDE I VALORI
DA 0 A dim

COMPRESO
0

ESCLUSO
dim

DA 0 A dim

CORRISPONDE A $[0, \text{dim}-1]$

Alcuni esempi:

- $\exists i \in [0, \text{dim}) . a[i] < 0$

ESISTE UN ELEMENTO NEGATIVO?

- $\exists i \in [0, \text{dim}) . a[i] \% 2 == 0$

ESISTE UN NUMERO PARI?

- $\exists i \in [0, \text{dim}-1) . a[i] == a[i+1]$

ESISTONO DUE ELEMENTI CONSECUTIVI UGUALI?

↓
L'ULTIMO ELEMENTO
NON HA UN SUCCESSORE!!

IN GENERALE

$\exists i \in [0, \text{dim}) . P$

↑
PÙÒ VARIARE

← PROPRIETÀ DELL'ELEMENTO CHE STO CERCANDO

Ricerca di un numero negativo

$$\exists i \in [0, \text{dim}) . a[i] < 0$$

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

ESTREMI DELL'INTERVALLO

CONSIDERIAMO OMI QUESTI :

- $\forall i \in [0, \text{dim})$. $a[i] < 0$

SONO TUTTI ?
NEGATIVI .

- $\forall i \in [0, \text{dim})$. $a[i] \% 2 == 0$

SONO TUTTI
PARI ?

- $\forall i \in [0, \text{dim}-1)$. $a[i] == a[i+1]$

SONO TUTTI
UGUALI ?

|||

$\forall i \in [1, \text{dim})$. $a[i] == a[0]$

TIPICA SOLUZIONE SBAGLIATA

```

int TuttiPari (int a[], int dim) {
    int pari = 0;
    for (int i = 0; i < dim; i++) {
        if (a[i] % 2 == 0) pari = 1;
        else pari = 0;
    }
    return pari;
}
    
```

a	2	4	6	5
	0	1	2	3
	P	P	P	P

pari 0 // // //
OK

FATE

||

FATE

LE

SIMULAZIONI

SU CARTA

DEI VOSTRI

PROGRAMMI !!

a

2	5	7	4
0	1	2	3
P	P	P	P

pa:

ϕ

 $\neq \phi$
 ϕ
1
No

OK

IDEA :

SONO TUTTI PARI

SE E SOLO SE

NESSUNO È DISPARI

IN TERMINI LOGICI :

$$\forall i \in [0, \text{dim}) . a[i] \% 2 == 0 \quad \text{VERA}$$

SE E SOLO SE

$$\exists i \in [0, \text{dim}) . a[i] \% 2 == 1 \quad \text{FALSA}$$

⇒ QUESTO LO SAPPIAMO FARE, SOLO CHE DEVO
DARE IL RISULTATO CONTRARIO (FALSO) ↑

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

ALTRO MODO EQUIVALENTE :

```
int TuttiPari (int a[], int dim) {  
    int i = 0;  
    int OK = 1;  
    while (i < dim && OK) {  
        if (a[i] % 2 == 1) OK = 0;  
        i++;  
    }  
    return OK;  
}
```

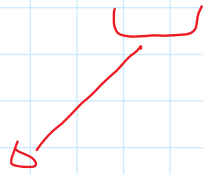
NON C'È IL !

PROBLEMI DI CONTESIO

→ CONTARE QUANTI NUMERI PARI CI SONO IN UN ARRAY

$$\# \{ i \mid i \in [0, \text{dim}) \wedge a[i] \% 2 == 0 \}$$

AND LOGICO



CARDINALITA' DELL'INSIEME

(OSSIA IL NUMERO DI ELEMENTI CHE CONTIENE)

INSIEME DEGLI INDICI
I CUI ELEMENTI CORRISPONDENTI
SONO PARI

8	7	3	2
0	1	2	3

$$\{0, 3\}$$

↑
INSIEME DEGLI INDICI DI ELEMENTI PARI

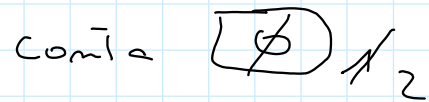
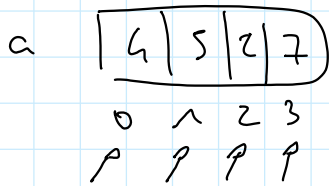
$$\# \{0, 3\} = 2$$

$$\# \{5, 7, 9\} = 3$$

⋮

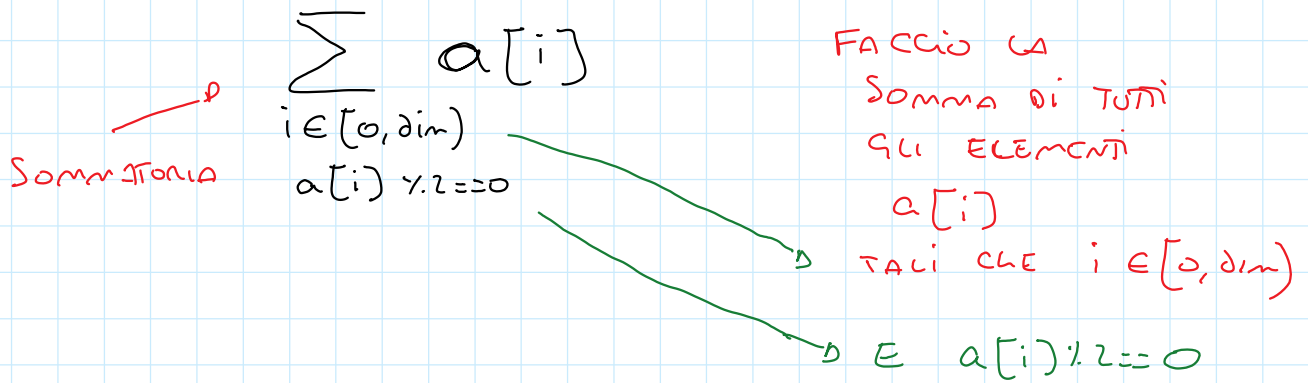
```
int contaPari (int a[], int dim) {  
    int conta = 0;   
    for (int i=0; i < dim; i++) {  
        if (a[i] % 2 == 0) conta++;  
    }  
    return conta;  
}
```

UNA
VARIABLE
USATA
IN QUESTO
MODO È
DETTA
"CONTATORE"



→ PROBLEMI di ACCREAZIONE

→ CALCOLARE LA SOMMA di TUTTI i PARI



```
int sommaPositivi (int a[], int dim) {
```

```
    int somma = 0
```

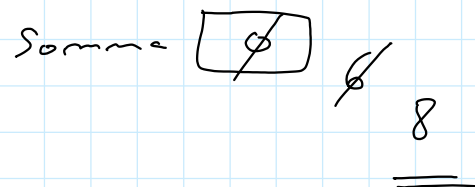
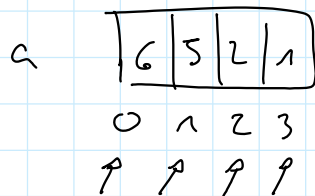
← UNA VARIABILE USATA IN QUESTO MODO È DETTA "ACCUMULATORE"

```
    for (int i = 0; i < dim; i++) {
```

```
        if (a[i] % 2 == 0) somma += a[i];
    }
```

$x += y$;
 " "
 $x = x + y$;

```
    return somma;
}
```



→ RICERCA DEL MINIMO ELEMENTO

→ SCRIVERE UNA FUNZIONE CHE RESTITUISCE

UN VALORE X

TALE CHE

$$\exists i \in [0, \text{dim}). (a[i] = x \wedge \forall j \in [0, \text{dim}). x \leq a[j])$$

↑ ↑ ↑
X È PRESENTE E X È MINORE
O UGUALE DI
TUTTI GLI ELEMENTI

PARTIAMO SCRIVENDO UNA FUNZIONE CHE

CALCOLA

$$\forall j \in [0, \text{dim}). x \leq a[j]$$

CORRISPONDE A VERIFICARE

$$\exists j \in [0, \text{dim}). x > a[j] \text{ È FALSO}$$

```

int minoreDiTutti (int a[], int dim, int x) {
    int i=0;
    int ok=1;
    while (i < dim && ok) {
        if (x > a[i]) ok=0;
        i++;
    }
    return ok;
}
    
```

ORA CHE HO LA FUNZIONE `minoreDiTutti`

VERIFICARE

$$\exists i \in [0, \text{dim}). (a[i] = x \wedge \forall j \in [0, \text{dim}). x \leq a[j])$$

CORRISPONDE A FARE

$$\exists i \in [0, \text{dim}). (a[i] = x \wedge \text{minoreDiTutti}(a, \text{dim}, a[i]))$$

DEVE
RESTITUIRE
X

```
int minore (int a[], int dim) {
    int i = 0;
    int trovato = 0;
    int min;
    while (i < dim && !trovato) {
        if (minoreDiTutti(a, dim, a[i])) {
            min = a[i];
            trovato = 1;
        }
    }
    return min;
}
```

POSSIAMO RISOLVERE PROBLEMI
COMPLESSI COMPONENTE
(MODULARMENTE) SOLUZIONI DI

PROBLEMI PIU' SEMPLICI

RAGIONANDO SUL PROBLEMA MI RENDO
CONTO CHE POSSO TROVARE UNA SOLUZIONE
PIU' FUNDA CHE SI "PORTA DIETRO" NELLO
SCANDIRE L'ARRAY IL VALORE MINIMO TROVATO
FINO A QUEL MOMENTO

```
int minimo (int a[], int dim)  
int min = a[0];  
for (int i = 1; i < dim; i++) {  
    if (a[i] < min)  
        min = a[i];  
}  
return min;  
}
```

min
RAPPRESENTA
IL VALORE
MINIMO CHE
HO INCONTRO
FINO A QUEL
MOMENTO

← STO
ASSUMENDO
CHE ESISTA
ALMENO
UN
ELEMENTO
(dim > 0)

DOMANDE PRATICHE:

→ COME FACCIO A FARE ESERCIZI?

- ↳ VEDREMO ALTM ESERCIZI DA
SOLGERE SU CARTA
- ↳ IN LABORATORIO

→ COME SI SCRIVONO LE FUNZIONI?

↳ LE FUNZIONI DEVONO ESSERE DEFINITE (SCRITTE)
PRIMA DELLA LORO CHIAMATA

```

int somma (int x, int y) {
    :
}

main() {
    :
    somma(3, 4);
    :
}

```

SE
INVERTO
IL COMPILATORE
C MI
DA
UN ERRORE

OK



POTREI SCRIVERE
SOMMA DOPO IL MAIN
A PATTO DI SCRIVERE
IL PROTOTIPO DI SOMMA
somma (int x, int y);
PRIMA DEL main