mercoledì 4 ottobre 2017 11:16

Controllare che in un array ogni elemento compaia (occorre) esattamente 2 volte.

cardinalità di un insieme.

Se l'insieme è finito, la sua cardinalità è uguale al numero degli elementi.
La cardinalità di un insieme I, si indica con $\#I$

$$\#\{1,2,3\} = 3$$

a di dimensione dim

$$\left(\forall i \in [0, dim) . \#\{j \mid j \in [0, dim) \wedge a[j] = a[i]\} = 2\right)$$

insieme degli elementi j per i quali vale questa proprietà

a | 3 | -2 | -2 | 5 | 3 | 5 | vale la proprietà

a | 4 | -2 | -2 | 5 | 3 | 5 | non vale la proprietà

mercoledì 4 ottobre 2017 11:16

Funzione che conta quante volte compare in a un elemento el.

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

int duevolte (int a[], int dim)
{ int i = 0;
  int ok = 1;
  while (i < dim && ok)
    if (conta (a[i], a, dim) != 2) ok = 0;
      else i++;
  return ok;
}
```

mercoledì 4 ottobre 2017 11:16

Nell'array a di dimensione dim esiste un elemento che compare esattamente due volte.

$$\left(\exists\, i \in [0, dim) \,.\, \#\left\{ j \mid j \in [0, dim) \wedge a[i] = a[j] \right\} = 2\right)$$

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

$$\left(\forall i \in [0, dim) \cdot \#\{j \mid j \in [0, dim) \wedge a[i] = a[j]\} = 2\right)$$

mercoledì 4 ottobre 2017 11:16

```
int duevolte (int a[], int dim)
{ int ok = 1;
  i = 0;
  while (i < dim && ok)
    { int conta = 0;
      for (int j=0; j < dim; j++)
          if (a[j] == a[i]) conta ++;
      if (conta != 2) ok = 0;
            else i++;
    }
  return ok;
}
```

mercoledì 4 ottobre 2017 11:16

Dati due array a e b, di dimensione dima e dimb, rispettivamente, controllare che tutti gli elementi di a che compaiono in b, compaiano in a esattamente due volte.

$A \Rightarrow B \equiv \neg A \vee B$

se A è falso l'implicazione è vera

$$(\forall i \in [0, dima).$$
$$(\exists j \in [0, dimb).\ a[i] = b[j]) \Rightarrow$$
$$\#\{z \mid z \in [0, dima) \wedge a[i] = a[z]\} = 2)$$

mercoledì 4 ottobre 2017 11:16

```
int conta (int el, int a[], int dim)
{ ..... }

int check (int a[], int dima, int b[], int dimb)
{ int ok=1;
  int i=0;
  while (i<dim && ok)
    if (member(a[i], b, dimb))
        if(conta(a[i], a, dima)!=2) ok=0;
        else i++;
    else i++;
  return ok;
}
```

```
int check (int a[], int dima, int b[], int dimb)
{ int ok = 1;
  int i = 0;
  while (i < dima && ok)
    { int j = 0;
      int trovatoinb = 0
      while (j < dimb && !trovatoinb)
      { if (a[i] == b[j]) trovatoinb = 1;
          else  j++;
      if (trovatoinb)
            {
              int conta = 0;
              for(int z = 0; z < dima ; z++)
                 if (a[z] == a[i]) conta ++;

              if (conta != 2) ok = 0;
            } else i++;
      else i++;
    }

  return ok;
}
```

mercoledì 4 ottobre 2017 11:16

Cercare il valore massimo in un array

a | -2 | 3 | -5 | 7 | → 7

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

Se volessimo, oltre al valore massimo, anche l'indice, utilizzeremmo un parametro puntatore.

Se ci sono più elementi uguali al valore massimo, vogliamo l'indice del primo.

a | 5 | 7 | 2 | 7 |
0 1 2 3

risultato 7

modifica del parametro l'indice 1

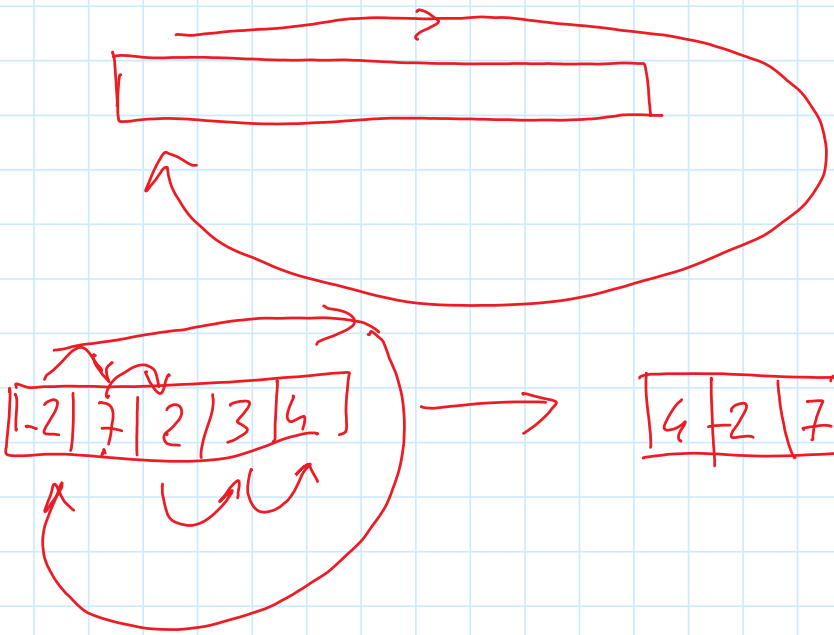
```
int max(int a[], int dim, int *ind)
{ int m = a[0];
  *ind = 0;
  for(int i=0; i<dim; i++)
    if(a[i] >= m) { m = a[i];
                    *ind = i;
                  }
  return m;
}
```

>= indica dell'ultima occorrenza del valore massimo.

```
main()
{ int a[10]
  int ind;
  int m;
  m = max(a, 10, &ind);
```

SHIFT CIRCOLARE

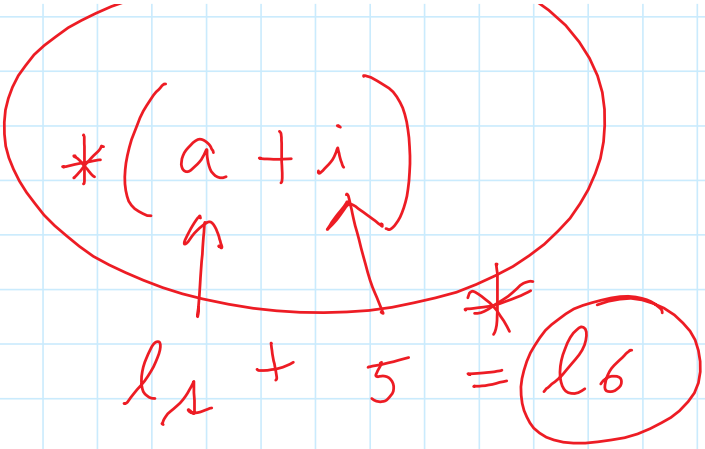
mercoledì 4 ottobre 2017 11:16




```
void shiftr (int a[], int dim)
{ int temp = a[dim-1];
  for(int i = dim-1; i > 0; i--)
      a[i] = a[i-1];
  a[0] = temp;
}
```

~~for (int i = 1; i < dim; i++)~~
~~a[i] = a[i-1];~~

a[i]



Controlliamo se gli elementi >0 sono in numero maggiori di quelli $<=0$ in un array a di dimensione dim

mercoledì 4 ottobre 2017 11:16

$$\#\{j \mid j \in [0, dim) \wedge a[j] > 0\}$$

$$>$$

$$\#\{j \mid j \in [0, dim) \wedge a[j] <= 0\}$$

```
int check (int a[], int dim)
{ int nmag = 0;
  int nmin = 0;
  for (int i = 0; i < dim; i++)
      if (a[i] > 0) nmag++;
      else          nmin++;
  return nmag > nmin;
}
```

```
if (nmag > nmin) return 1;
else             return 0;
```

mercoledì 4 ottobre 2017 11:16

Cercare il massimo e il minimo

```
void maxmin (int a[], int dim, int *max,
             int *min)
{
  *max = a[0];
  *min = a[0];
  for (int i=1; i<dim; i++)
    if (a[i] > *max) *max = a[i];
    else if (a[i] < *min) *min = a[i];
}

main()
{
  int a[10];
  int max;
  int min;
  :
  maxmin(a, 10, &max, &min);
```

mercoledì 4 ottobre 2017 11:16

```
int maxmin (int a[], int dim, int *min)
{ int max = a[0];
  *min = a[0];
  for (i = 1; i < dim; i++)
     if (a[i] > max) max = a[i];
     else if (a[i] < *min) *min = a[i];
  return max;
}
main()
{ int a[10];
  int max;
  int min;
  :
max = maxmin (a, 10, &min);
```