Capitolo 8

Esercizi su funzioni

Lavoriamo su un array a 3 dimensioni . Si vuole realizzare:

int F(int A[][5][10], int limite1, int n_ele)

- A ha limite1 strati e solo i suoi primi n_ele elementi sono definiti (0<n_ele<=limite1*5*10)

- F deve restituire l'indice dello strato a somma massima

- Se ci sono più strati a somma massima, basta l'indice di uno qualsiasi di questi strati

idea: ci saranno un certo numero di strati pieni (di elementi definiti) ed un ultimo strato (possibile) non completamente pieno.
E' facile calcolare la somma degli strati pieni

Quanti strati pieni ci sono?
n_ele/5*10 e
n_ele%(5*10) sono gli elementi definiti nell'ultimo strato

lo strato incompleto è fatto da alcune righe piene a da un'ultima riga incompleta:
(n_ele%(5*10))/10 e (n_ele%(5*10))%10

PRE=(0< n_ele<=limite1*5*10, i primi n_ele elem. di A sono definiti)

```
int F(int A[][5][10], int limite1, int n_ele)
{int n_sp=n_ele/(5*10), n_eu=n_ele%(5*10);
int sommaM=INT_MIN, indice=0;
for(int i=0; i<n_sp;i++) //R1
{
  int S=somma(*(A+i));
  if(S>sommaM)
      {sommaM=S; indice=i;}
}
//POST1
ultimo strato e poi return indice
```

} POST=(A[indice] ha somma massima)

POST1=(n_sp=0 => indice=0 && sommaM=INT_MIN)
&& (n_sp>0 => A[indice] strato max tra A[0..n_sp-1])

R1=(i=0 => indice=0 && sommaM=INT_MIN) &&
(i>0 => A[indice] strato max tra A[0..i-1])
&&(0<=i<=n_sp)

Fare prova di correttezza del for 1

```
//POST1
int S=somma_inc(*(A+n_sp), n_eu);
if(S>sommaM) indice=n_sp;
//POST=(A[indice] strato a somma massima di A)
```

attenzione ai casi limite: n_ele>0 in PRE

-n_sp=0

-n_ue=0

-non possono essere entrambi 0

```
if(n_eu >0)
{
   int S=somma_inc(*(A+n_sp), n_eu);
    if(S>sommaM) indice=n_sp;
}
return indice;
```

funzione somma

```
//PRE=(X è int[5][10], piena di valori)
int somma(int X[][10])
{int somma=0;
for(int i=0; i<5;i++)// somma = somma X[0..i-1]
  for(int j=0; j<10; j++)// somma X[0..i-1]+X[i][0..j-1]
    somma=somma+X[i][j];
return somma;
}
//POST=(somma=somma degli elementi di X)
```

PRE=(X è int[5][10] e solo i suoi primi k elementi sono definiti)

```
int somma_inc(int X[][10], int k)
{
  int n_r=k/10, n_eur=k%10, int somma=0;
for(int i=0; i<n_r; i++)
  for(int j=0; j<10;j++)
    somma=somma+X[i][j];

for(int i=0;i<n_eur;i++)
    somma=somma+X[n_r][i];
 } //POST=(somma = somma primi k elementi di X)
```

ma ogni strato è contiguo.
Possiamo trattarlo come un array ad una dimensione e così scompare la differenza tra somma e somma_inc

```
//PRE=(X ha i primi lim elem definiti)
int T(int* X, int lim)
{
int somma=0;
for(int i=0; i<lim; i++)
   somma=somma+X[i];
}//POST=(somma= somma dei primi lim elementi di X)
```

PRE=(0<n_ele<=limite1*5*10, i primi n_ele elem. di A sono definiti)

```
int F(int A[][5][10], int limite1, int n_ele)
{int n_sp=n_ele/(5*10), n_eu=n_ele%(5*10);
int sommaM=INT_MIN, indice=0;
for(int i=0; i<n_sp;i++) //R1
{
  int S=T(**(A+i), 50);
  if(S>sommaM)
      {sommaM=S; indice=i;}
}
if(n_eu>0)
 {int S=T(**(A+n_sp), n_eu); if(S>sommaM) indice=n_sp;}
return indice;
```

} POST=(A[indice] ha somma massima)

Problema (2): calcolare indice della **fetta** a somma massima di int A[limite1][limite2][limite3]

fetta 0 è composta da A[0][0], A[1][0], A[2][0],.,A[limite1-1][0]

fetta i è composta da A[0][i], A[1][i], A[2][i],...A[limite1-1][i]

ma ancora c'è il problema che solo i primi n_ele elementi di A sono definiti con n_ele>0

PRE=(A punta al primo di limite1*limite2*limite3 interi di cui solo i primi n_ele sono definiti, con n_ele>0)

int G(int*A, int limite1, int limite2, int limite3, int n_ele)

POST=(G restituisce l'indice di una fetta di A che ha somma degli elementi definiti massima rispetto alle altre fette di A)

```
for(int f=0; f<limite2; f++)
{ int sommaf=0;
   for(int s=0; s<limite1; s++)
      for(int c=0; c<limite3; c++)
         sommaf=sommaf+*(A+s*limite2*limite3+f*limite3+c);
  if(sommaf>sommaM)
     {sommaM=sommaf; indice=f;}
}
```

come evitare gli elementi indefiniti delle fette?

come evitare le fette vuote?