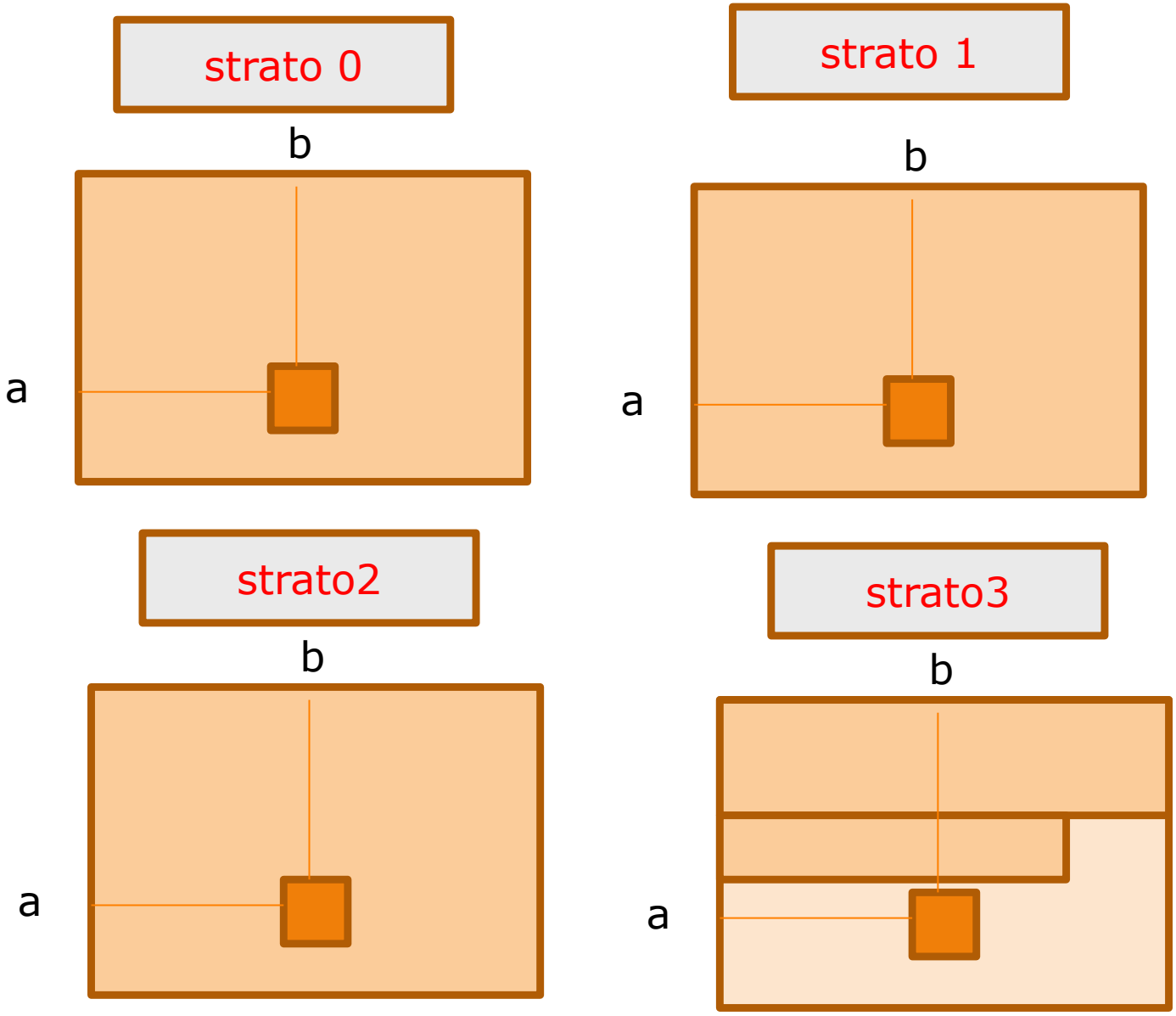
mercoledì 22 febbraio 2012

```
char C[]="buio";
for(int i=0; i<4; i++)
{
 switch(C[i])
  {
    case 'o': cout<<C+i-1;  break;
    case 'a': case 'b': cout<<C+i+1; break;
    case 'i': cout<<C+i;
    default: cout<<"xxx";
  }
}
```

uio xxx ioxxx io

Programmazione) Dato un array int X[lim1][lim2][lim3], il **tassello (a,b),** dove a in [0..lim2-1] e b in [0..lim3-1], di X è costituito dagli elementi X[i][a][b] con i in [0..lim1-1]. Qualora X non sia tutto definito, un tassello potrebbe avere solo alcuni elementi definiti ed altri no.
Si chiede di scrivere una funzione **void F(int*X, int lim1, int lim2, int lim3, int n_ele, tasse T)** che sia corretta rispetto alle seguenti pre e post-condizioni:
PRE=(solo i primi n_ele elementi di X sono definiti, n_ele>0, lim1, lim2 e lim3 sono >0, T è una struttura struct tasse{int a,b;}; tale che T.a in [0..lim2-1]e T.b in [0..lim3-1], X=$\underline{X}$)
POST=(Il tassello T di $\underline{X}$, per quanto riguarda i suoi elementi definiti, è ordinato in X. Tutti gli altri elementi di X restano identici a $\underline{X}$)
NOTA: se riuscite, usate il bubble sort per l'ordinamento, se non riuscite, usate anche un qualsiasi altro algoritmo di ordinamento. E' consigliabile usare almeno una funzione ausiliaria per cui dovrete specificare adeguate pre- e post-condizioni. Inoltre specificate un invariante per il ciclo principale di F.


strato 0
b
a
strato 1
b
a
strato2
b
a
strato3
b
a

Bubble-sort su un array A di k interi, con k>0

```
//A=vA
for(int i=k; i>0; i--) //R
  FAI(A,i);
```

R=(A[0..k-1] è permutazione di vA[0..k-1]) && (A[i+1..k-1] è ordinato)&& (A[i+1..k-1] >=A[0..i])

```
void FAI(int*A, int k) //(A=vA) && (A[0..k-1] def.)
{
 for(int i=0; i<k-1; i++) //R1
   if(A[i]>A[i+1])
   {
      int a=A[i];
      A[i]=A[i+1];
      A[i+1]=a;
    }
}
```

R1=(A è permutazione di vA) &&
(A[i]>=A[0..i-1]) &&(0<=i<=k-1)

POST1=(A è permutazione di vA) &&
(A[k-1]>=A[0..k-2])

in fondo un tassello è una sequenza di elementi, insomma un array come A

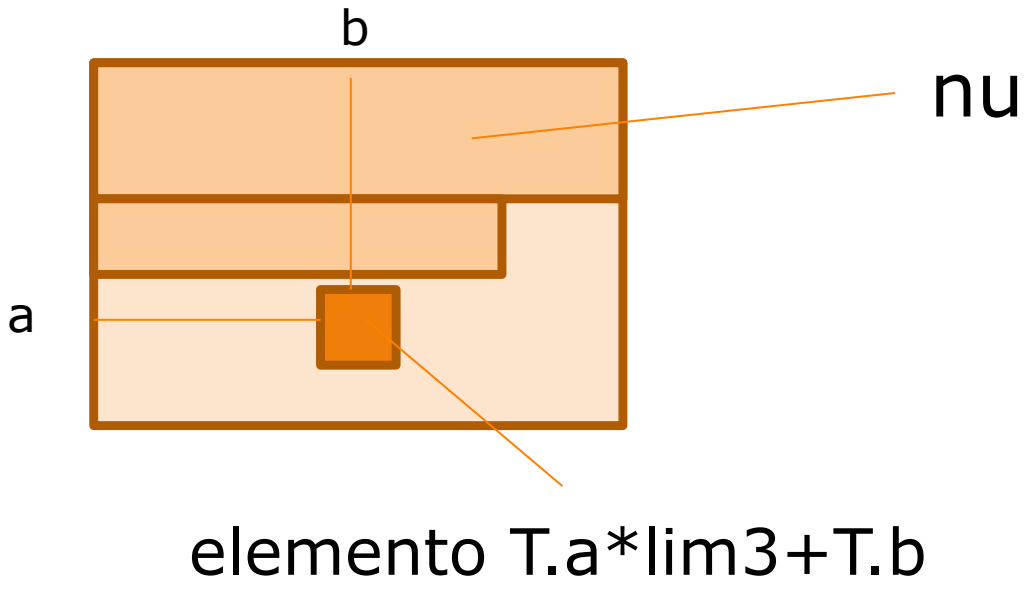
ma quanti elementi ha? Dipende da n_ele

n_ele/(dim_strato) = SP
il tassello avrà CERTAMENTE SP elementi

...ma forse uno in più, dipende da dove si trova l'elemento del tassello nello strato SP che è parzialmente riempito

n_ele%(dim_strato) = nu sono gli elementi definiti dell'ultimo strato

basta chiedersi se T.a*lim3+T.b < nu o no



```
DS=lim2*lim3;
PT= T.a*lim3+T.b
LT= n_ele/DS;
nu=n_ele%(DS);
if(PT < nu)
  LT++;

// LT= lunghezza del tassello T (= n. elementi definiti)

for(int i=LT; i>0; i--) //R
  FAI(A, i, DS, PT);
```

```
void FAI(int*A, int k, int DS, int PT)
{
 for(int i=0; i<k-1; i++)
   if(A[i*DS+PT]>A[(i+1)*DS+PT])
   {
     int a=A[i*DS+PT];
     A[i*DS+PT]=A[(i+1)*DS+PT];
     A[(i+1)*DS+PT]=a;
    }
}
```

```
void F(int*A,int lim1, int lim2, int lim3, int n_ele,
tasse T)
{
int DS=lim2*lim3;
int PT= T.a*lim3+T.b;
int LT= n_ele/DS;
int nu=n_ele%(lim2*lim3);
if(PT < nu)
  LT++;
// LT= lunghezza del tassello T (= n. elementi
definiti)
cout<<"LT="<<LT<<endl;
for(int i=LT; i>0; i--) //R
  FAI(A, i, DS, PT);
}
```