array a più dimensioni

definizione ed esempi

Array a 1 dimensione

char x[20];
int w[50]

x ha tipo char * const
e w int * const

ci dimenticheremo del const

char* p=x ; OK !

char x[50];

x[0] ≡ *(x+0) = *x aritmetica dei puntatori
x[1] ≡ *(x+1)
x[2] ≡ *(x+2)

e così via

e anche x[-5] e *(x-5) sono legali sebbene senza senso in questo caso

possono servire array a 2 dimensioni e anche a 3, 4,… dimensioni

- vogliamo rappresentare in modo semplice la configurazione di una scacchiera
- o abbiamo un elenco di conti correnti con delle informazioni per ciascun numero di conto: saldo, interesse, ecc
- modelli di auto con caratteristiche tecniche
- a 3 dimensioni: marche di auto e per ciascuna marca i modelli con le caratteristiche

sintassi in C

int a[10][20]; ➔ array a 2 dimensioni / matrici

char r[5][10][20]; ➔ a 3 dimensioni / torte

double w[5][6][7][8] ➔ a 4 dimensioni / sequenza di torte

primo elemento= a[0][0], intermedio= a[2][5], ultimo=a[9][19]

r[0][0][0]……..r[4][9][19]

possiamo anche usare l’aritmetica dei puntatori

r[4][9][19] → *(*(*(r+4)+9)+19)

ma dobbiamo prima studiare un po’ per capirla

allocazione in memoria

array a 2 dimensioni = sequenza di array a 1 dimensione

prima la prima riga, poi la seconda, la terza e così via

tutte attaccate senza spazi liberi

array a 3 dimensioni? è una sequenza di array a 2 dimensioni (strati)

prima il primo strato, poi il secondo, il terzo e così via
tutti attaccati e così via per 4, 5, 6,….dimensioni

int x[10]; x ha tipo int * e R-valore = &x[0]

int y[5][10]; y è un array di 5 array di 10 interi

quindi y è un puntatore al primo dei 5 array di 10 int, e il suo tipo è int (*) [10] o int [][10]

quindi y ha tipo int (*) [10]

si osservi che y punta a un oggetto di 10*4 byte

*y è l’oggetto puntato da y e quindi un array di 10 interi
che ha tipo int*, quindi *y ha tipo int *

e y ha tipo int (*)[10], ma hanno lo stesso R-valore

cout << y <<‘ ‘<<*y;

stampa 2 volte &y[0][0] !!

leggiamo valori in un array a 2 dimensioni:

```
int x[5][6];
for(int i=0; i<5;i++)
 for(int j=0; j<6; j++)
   cin >> x[i][j];
```

x viene riempito per righe

prima la riga 0, poi la 1, la 2, fino alla 4

ma visto che le righe sono in memoria proprio nello stesso ordine, prima la 0, poi 1, 2, ecc,
possiamo vedere x come un array ad una dimensione con 30 elementi
per cui

```
int x[5][6];
int* y= &x[0][0]; // IMPORTANTE
for(int i=0; i<30; i++)
   cin >> y[i];
```

stesso effetto del precedente: riempie x

ma a volte mantenere le 2 dimensioni è meglio.
esercizio di trovare l'indice della riga la somma dei cui elementi è massima (in caso di parità, vogliamo l'indice minimo)

```
int x [5][6]=;  //lettura in x
bool prima=true;
int maxv, maxindice;
for(int i=0; i < 5; i++)
{  int somma=0;
    for(int j=0; j<6; j++)
        somma=somma+x[i][j];
   if (prima || somma > maxv)
      {prima=false; maxv=somma; maxindice=i;}
}
```

```
int x [5][6]=;  //lettura in x
bool prima=true; int * y =&x[0][0];
int maxv, maxindice;
for(int i=0; i < 5; i++)
{  int somma=0;
    for(int j=0; j<6; j++)
        somma=somma+ *(y+i*6+j);
   if (prima || somma > maxv)
      {prima=false; maxv=somma; maxindice=i;}
}
```

si calcola la distanza dell'elemento x[i][j] dall'inizio di x

esercizio: calcolare l’indice della colonna a somma massima
Farlo nei 2 modi

esercizio: testare se ci sono colonne identiche e stampare la colonna che ha massimo numero di altre colonne identiche

invariante

se prima = true => non abbiamo fatto niente

se prima= false => allora considerate c>0 colonne e
superc è l'indice della colonna che ripete di più
maxnum il n. di ripetizioni

serve un ciclo interno che calcola il numero di occorrenze di colonne uguali alla colonna corrente c

R1= esaminate le colonne c+1..j-1 e nc è il numero di colonne uguale alla colonna c trovate finora

e un ciclo ancora interno per confrontare la colonna c e quella j

R2= uguale sse finora le 2 colonne sono uguali

```
int x[5][20], superc, maxnum; bool prima=true;
    for(int c=0; c<20;c++)
    { int nc=1;
      for (int j=c+1; j<20; j++)
         {bool uguale=true;
          for(int r=0;r<5&&uguale; r++)
               if(x[r][c]!=x[r][j])
                    uguale=false;
               if(uguale)
                 nc=nc+1;
          }
            if(prima || nc > maxnum)
              {prima=false; superc=c; maxnum=nc;}
      }
 }
  cout<<"supercolonna="<<superc<<" num. rip="<<maxnum<<endl;
}
```