array e for

5.2 del testo

l'array è una struttura dati che permette di mettere assieme molti oggetti dello stesso tipo facilitandone l'accesso

char X[100]; dichiara 100 variabili carattere che si chiamano:

X[0], X[1], ……, X[99]

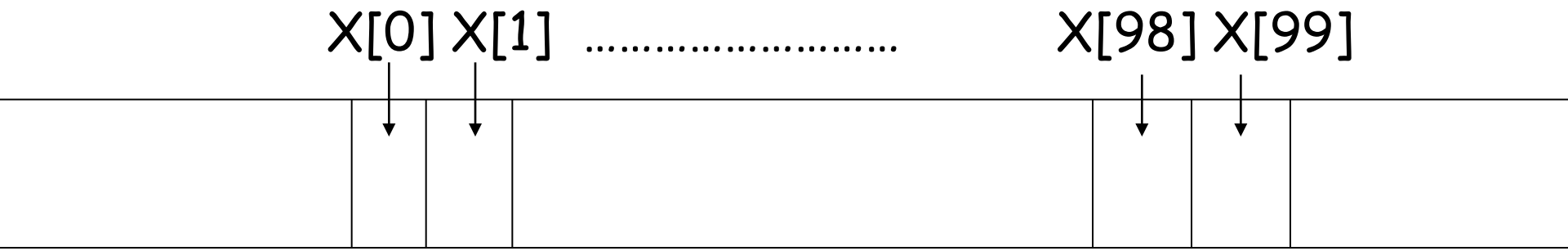
attenzione : si parte sempre dall'elemento 0

il C++ PRESCRIVE che questi elementi abbiano SEMPRE L-valori contigui

X[11] e X[12] allora

&X[12] = &X[11] +1 ricordarsi che ogni char occupa 1 byte

e quindi &X[99] = &X[0]+99



RAM

lo stesso per array di qualsiasi tipo usando il n. di byte del TIPO

array a 2, 3, 4, 5, …. dimensioni:

double X[5][10];

float Y[3][4][10];

int Z[10][10][20][30];

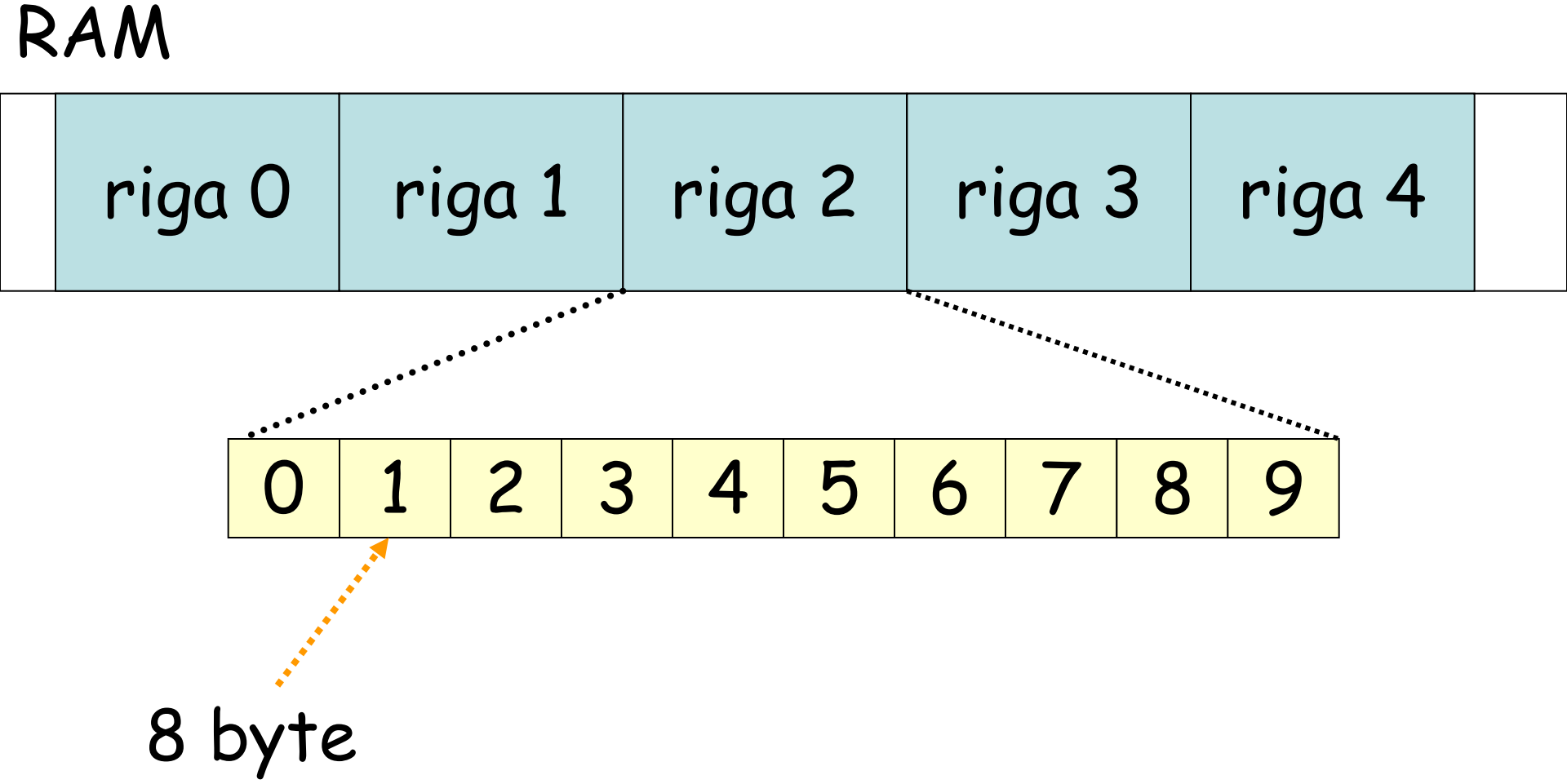
e così via

limite della prima dimensione è 5, della seconda è 10

elementi: X[0][0] X[4][9]
Z[0][0][0][1]

Y[3][0][1] non esiste

di nuovo gli elementi sono accostati nella RAM per righe: double X[5][10]



e float Y[3][4][10]; ?

ogni strato è un array [4][10] visto prima

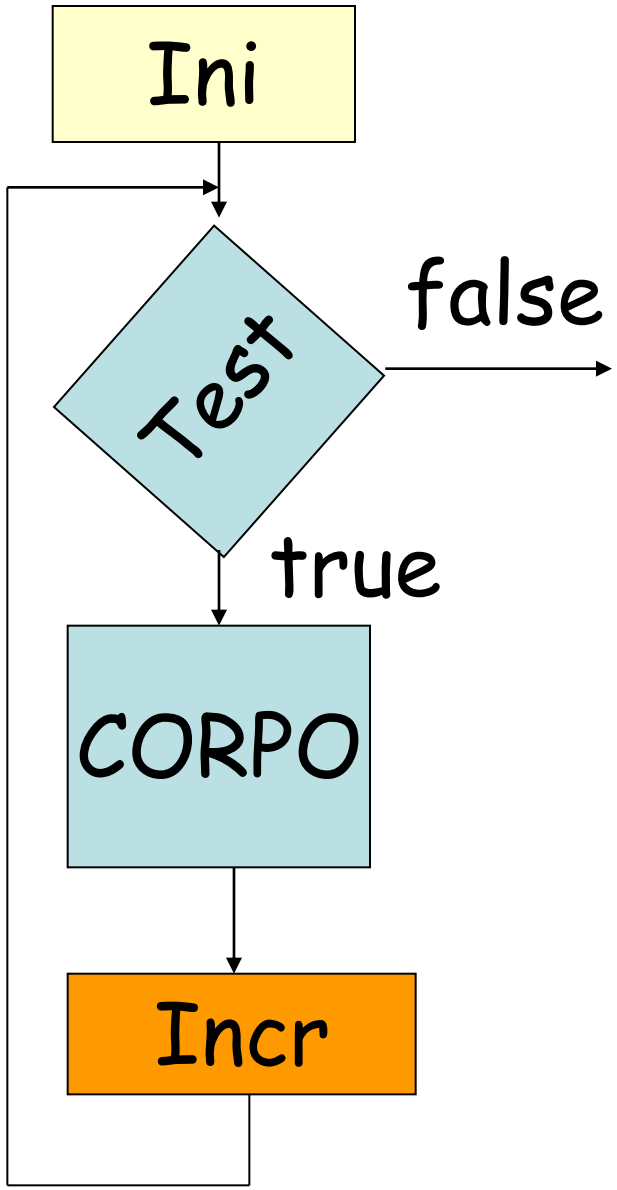
inizializzare un array da cin:

```
int A[3][10][10], i=0,j,z;
while(i<3)
{ j=0;
 while(j<10)
 {z=0;
   while(z<10)
      {cin >> A[i][j][z]; z++}
  j++; }
i++;}
```

anche:

```
int A[3][10][10]={10, 0, 3, ……};
int A[3][10][10]={}; // tutto a 0
```

```
for ( Ini ; Test ; Incr)
{ CORPO}
```



inizializzare un array da cin col for:

```
int A[3][10][10];
for(int i=0; i<3; i++)
 for(int j=0; j<10; j++)
   for(int z=0; z<10; z++)
      cin >> A[i][j][z];
```

visibilità di i, j e z?

pag. 59 del testo:

```
int i=4; cout<<i<<endl;
for(int i=5; i<10; i++)
{
  cout<<i<<endl;
  int i=0;
  cout<<i<<endl;
  i++;
}
cout<<i<<endl;
```

for e while

```
for ( Ini ; Test ; Incr)
{ CORPO}
```

è equivalente a

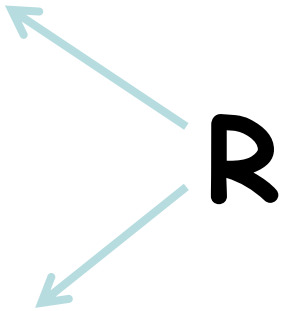
```
{Ini;
while (Test)
{CORPO; Incr;}
}
```

in certi casi è più comodo il for in altri il while

Regola di prova del for è la stessa del while

PRE <for(ini; test; incr) C > POST

è come



PRE <ini; while(test) C;incr> POST

somma degli elementi di un array a 2 dimensioni

```
int Z[10][20], somma=0;

for(int i=0; i<10; i++) //somma=righe 0..i-1
{

   int sommarig=0;

   for(int j=0; j<20; j++)  //sommarig= elem 0..j-1

     sommarig=sommarig+Z[i][j] ;

  somma=somma+sommarig;

}
```

esercizio: trovare le posizioni del minimo e del massimo valore in un array ad una dimensione di 100 elementi

PRE =(A[100] definito, sia $\underline{A}$ il valore)

POST=(posmin e posmax $\in$ [0,99],
A[posmin] <=A[0..99] e
A[posmax]>=A[0..99]), A=$\underline{A}$

```
int posmin=0, posmax=0;
for(int i=1; i<100; i++)  //R
  {
    if(A[posmin]>A[i])
       posmin=i;
     else
    if(A[posmax]<A[i])
       posmax=i;
  }
```

R=(0<=i<=100, 0<=posmin <i, 0<=posmax< i,) &&
(A[posmin]<=A[0..i-1]) &&
(A[posmax]>=A[0..i-1]) &&(A[0..i-1]=A[0..i-1])

esercizio:
in un array a 2 dimensioni float B[10][20].
calcolare l'indice della riga a somma minima

PRE=(B[10][20] è definito, e $\underline{B}$ il suo valore iniziale)

POST=(index è l'indice di una riga a somma minima, B=$\underline{B}$)

che tipo ha un array ?

```
char A[100];
cout<< A<< &A[0];   // stampa 2 indirizzi uguali
```

Quindi A ha tipo char* ?

Il tipo esatto di A è char[100],

ma in C++ char[100] è equivalente a char[] e a char *

A è una costante di tipo puntatore a carattere:

char * const

provate a compilare A=A+1; // da errore

PERCHE'?

se cambiassi A perderei l'accesso all'array e questo non può essere

GIUSTO ??

OSSERVARE

```
char A[100], *p =A; // OK char[100] è
                    // char*

char B[100];

B=p;  // ERRORE B è costante
```

```
int A[100], *p=A;
```

A[20] e p[20] sono esattamente la stessa cosa

la cosa pericolosa è che :

```
int *q; q[30]=1;   // è considerato OK
                   // dal compilatore C++
```

eventualmente ci sarà errore RUN TIME

ARITMETICA dei puntatori in C++:

- dato double * p;
- p+1 == p + 8 ha ancora tipo double*
- *p = oggetto puntato da p
-*(p+1) = oggetto puntato da p+1
-(p+k)= p+ k*8

anche se p non è un array !!!

char * p; p+5 ?

double *q, q-10 ?

double P[100]; P+4?

insomma sapere il tipo puntato e quindi la sua dimensione è essenziale

sembra facile, ma si complica con gli array a più dimensioni

tipo e taglia degli array

int K[5][10]; tipo = int (*) [10] = int [][10]
taglia=10*4

char R[4][6][8]; tipo = char (*) [6][8] =
char[][6][8] taglia=6*8

double F[3][5][7][9]; tipo = double (*)[5][7][9]
taglia=8*5*7*9

e cosa otteniamo eseguendo:cout<< K << R << F

&K[0][0] &R[0][0][0] &F[0][0][0][0]

double F[3][5][7][9]; tipo = double (*)[5][7][9]

tipo di *F ? e R-valore di *F ?

tipo di **F ? e R-valore di **F ?

tipo di ***F ? e R-valore di ***F ?

tipo di ****F ? e R-valore di ****F ?

CAPIRE

double F[3][5][7][9];

allora ***F ha tipo double * e

***(F+4) ha anch'esso tipo double*

la dereferenziazione cambia il tipo,

ma l'aritmetica cambia il valore non il tipo !!!

CAPIRE :

float K[3][5][7][10];

K[1]= *(K+1)

K[3][2]=*(*(K+3)+2)

K[2][1][4][1]= *(*(*(*(K+2)+1)+4)+1)

esercizio

float K[3][5][7][10];

se K == L, che valore ha K[2][2]?

K[2] = L + 2*(5*7*10) *4 =L+2800 = L1
K[2][2] = L1 + 2*(7*10)*4 = L1+560

K[3][5][10] = ?

float K[3][5][7][10]; se K=L

K[-1][-2] = ?

K[-1]-K = L - (5*7*10)*4 = L1

K[-1][-2] = L1 - 2*(7*10)*4

K[-1][-2][5] = L -(5*7*10)*4 - 2*(7*10)*4 + 5*10*4

tutti gli elementi degli array sono contigui nella RAM, dato: int Z[4][5][6][10];

Gli elementi di Z partono da &Z[0][0][0][0] e sono 4*5*6*10. Se vogliamo sommarli:

```
int * prossimo=&Z[0][0][0][0], somma=0;
for(int n=0; n<4*5*6*10; n++)
somma=somma + prossimo[n];
```

invariante?
R=(0<=n<=10*10*20*30,
somma= prossimo[0..n-1])

Gli array di char si comportano diversamente dagli array di altri tipi

1) Inizializzazione:

```
int A[]={0,1,2,3,4,56,99};//ha 7 elementi

char B[]={'p','i','p','p','o'};// ha 5 elementi

char C[]="pippo";   //ha 6 elementi
```

C[5] contiene '\0' carattere nullo
codice ASCII = 0

2) STAMPA

il carattere nullo serve da sentinella che segnala la fine stringa e serve per molte operazioni sulle stringhe

infatti cout<<"pippo";

si comporta allo stesso modo di:

cout << C; // stampa pippo

C[3]='\0' ; cout << C; ??

e cosa stampa:

```
char B[]={'p','i','p','p','o'};
cout<< B;
```

???