Fondamenti di Informatica

Ing. Biomedica

Esercitazione n.10

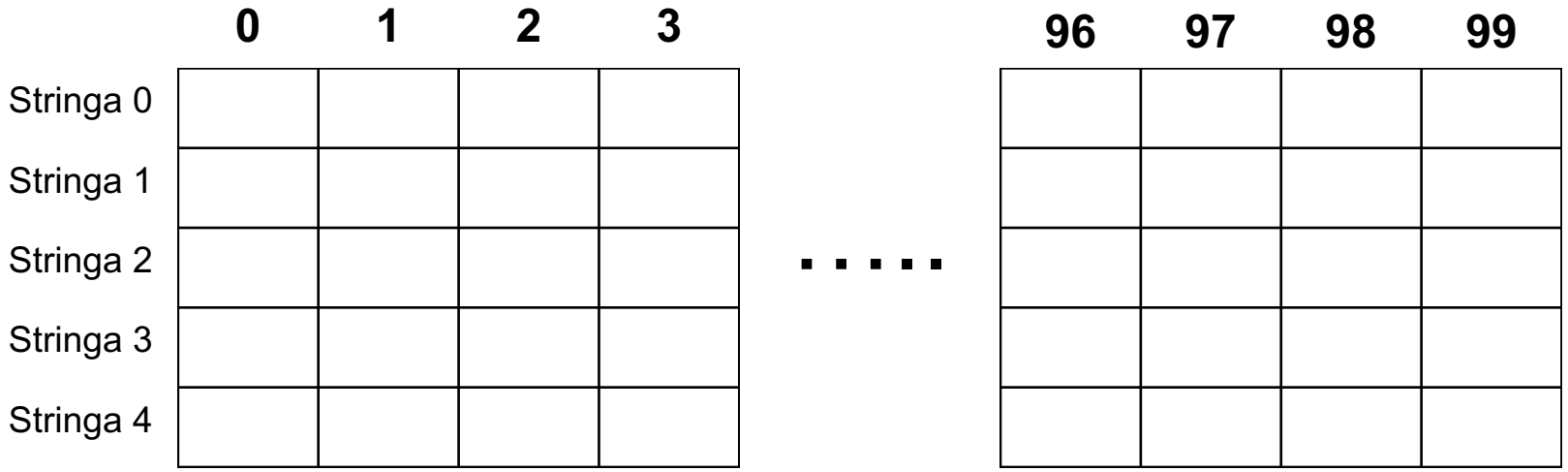
Strutture

Antonio Arena
antonio.arena@ing.unipi.it

Ripresa esercizio per casa

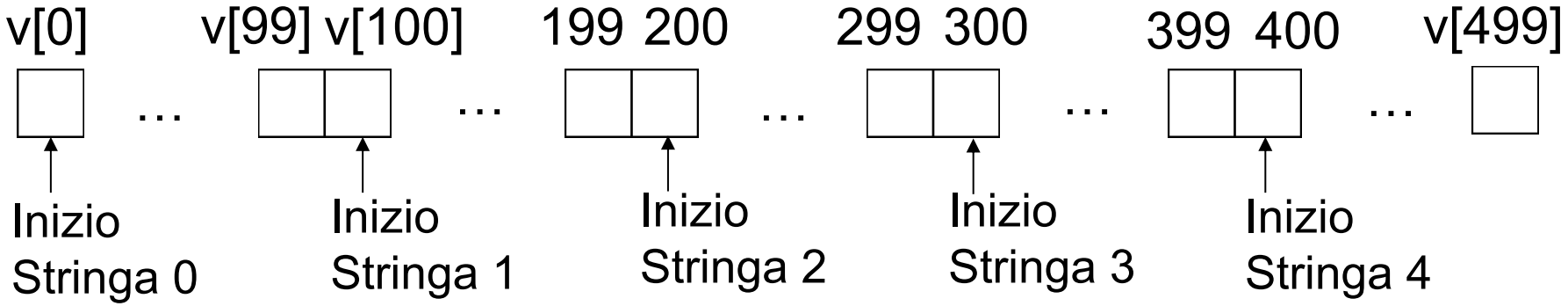
- Scrivere una funzione che prenda in ingresso un array di 5 stringhe di al più 100 caratteri e lo ordini in ordine crescente di lunghezza delle stringhe.

- Nel main per avere un array di stringhe va creata una matrice di caratteri → char vettore[5][100] : 5 array di 100 elementi



Ripresa esercizio per casa

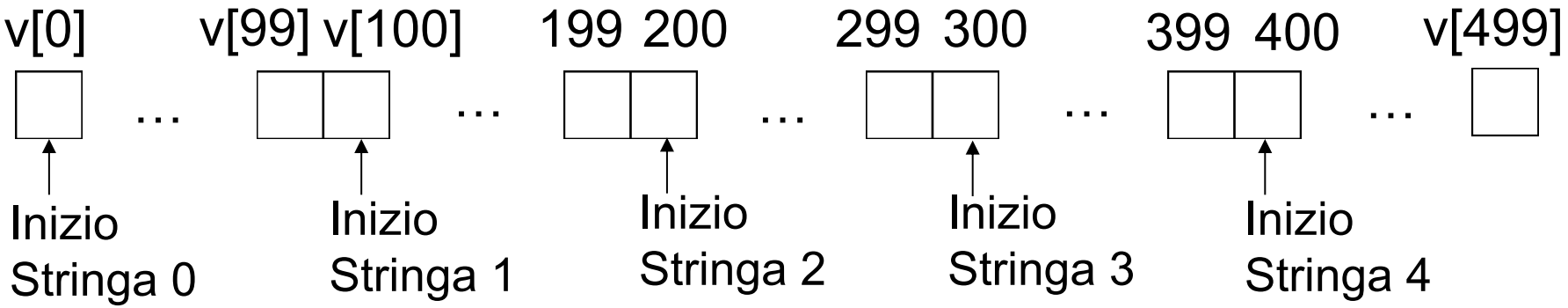
- Quando la matrice viene passata alla funzione tramite puntatore, la matrice viene considerata come *vettore monodimensionale.*
- *Esempio:*
  bubble_stringhe(char *v, int n) { … }
  n è il numero di stringhe, cioè la prima dimensione, ovvero 5



Quindi per scorrere il vettore per avere l'i-esima stringa, o meglio, il primo elemento dell'i-esima stringa, dobbiamo incrementare i di 100 ogni volta, quindi v[0], v[100], v[200], and so on…

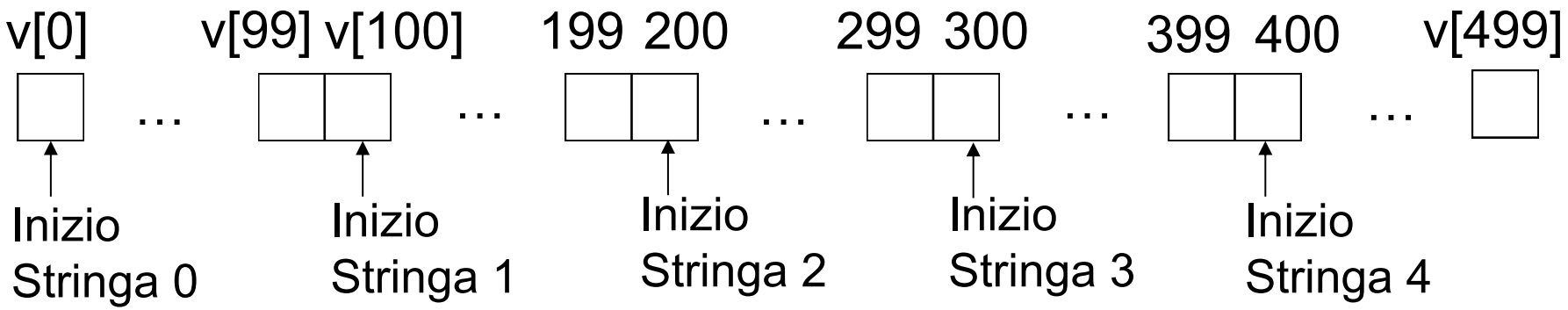
Ripresa esercizio per casa

- Con l'algoritmo bubble, confrontiamo la lunghezza delle stringhe, utilizzando la funzione di libreria *strlen.*
  La strlen ha come parametro formale un puntatore a carattere, quindi va passato l'indirizzo del primo elemento della stringa di cui si vuole sapere la lunghezza, esempio strlen(&v[300]) ci calcola l'indirizzo della stringa 3.
- Ogni volta, usando la variabile j nel bubble per interi, confrontiamo l'elemento j con l'elemento j-1. Qui dobbiamo confrontare la stringa che parte dall'elemento j con la stringa che parte dall'elemento j-100 (o LEN_MAX come nella sol.)



Ripresa esercizio per casa

- Scambio di stringhe: non posso usare l'assegnamento, ma bisogna creare una stringa temporanea di 100 caratteri, e fare 3 strcpy. Anche qui, c'è bisogno di dare l'indirizzo del primo elemento della stringa.
- Esempio: strcpy(&v[300], &v[400]) copia la stringa 4 nella stringa 3.



- *Arrivati qui, dovremmo avere tutte le informazioni necessarie per poter svolgere l'esercizio…*

Ripresa esercizio per casa

```
 1 #include <iostream>
 2 #include <cstring>
 3
 4 using namespace std;
 5
 6 const int LEN_MAX = 100;
 7 const int N_STRING = 5;
 8
 9 void scambia_stringhe(char *stringa1, char* stringa2){
10     char temp[LEN_MAX];
11     strcpy(temp,stringa1);
12     strcpy(stringa1, stringa2);
13     strcpy(stringa2, temp);
14 }
15
16 void bubble_stringhe(char *vettore, int n){
17     for(int i=0; i<n;i++){
18         for(int j=LEN_MAX*(n - 1); j > LEN_MAX*i; j-=LEN_MAX){
19             if(strlen(&vettore[j]) < strlen(&vettore[j-LEN_MAX]))
20                 scambia_stringhe(&vettore[j], &vettore[j-LEN_MAX]);
21         }
22     }
23 }
```

Strutture

- In C++ una struttura è un gruppo di variabili, anche di tipo diverso, che vengono aggregate sotto il nome di un'unica variabile.

  Esempio: vogliamo creare un database in cui vengono salvati i voti degli esami sostenuti da studenti universitari. Di quante informazioni al minimo?
  Perlomeno 3: matricola, codice insegnamento e ovviamente il voto…

- Quindi potremmo creare una struttura che aggrega come informazioni queste variabili, e chiamarla per esempio votazione

```
struct votazione{
    char matricola[10];
    char codice[10];
    int voto;
};
```

Strutture

- Come creare una variabile di tipo votazione?
  `votazione v1, v2;`
  → vengono creati i campi, ma non vengono inizializzati
- Come accedere ai campi di una struttura? `v1.<campo>`
```
strcpy(v1.matricola, ‘‘482245’’);
strcpy(v1.codice, ‘‘DII1234’’);
v1.voto = 30;
```
- Assegnamento: si può fare `v2 = v1`.
  → L’effetto di questa istruzione è che i campi di v1 vengono copiati elemento per elemento all’interno di v2.
- Si può creare un puntatore a struttura,
  `votazione *pv = &v1;`
  → pv punta alla struttura v1.
  Si può accedere agli elementi di v1 con l’operatore `->`.
```
cout << v1.matricola << pv->voto << pv->codice;
```

Struttura parametro di funzione

- Il ragionamento rimane identico ai tipi primitivi
- Si può passare una oggetto struttura *per valore*
  `void f1(votazione v3)` --> nel main: `f1(v1)`
  → viene creato alla chiamata della funzione un nuovo oggetto struttura, di nome v3, e i suoi campi vengono inizializzati con gli stessi valori dell'oggetto v1 creato nel main. Se si modificano i campi di v3 all'interno della funzione, i campi di v1 rimangono inalterati. Per accedere ai campi di v3 si usa l'operatore punto (v3.<campo>)
  Si può passare una oggetto struttura *tramite puntatore o riferimento*
  `void f2(votazione *v4)` --> nel main: `f2(&v1)`
  `void f3(votazione &v5)` --> nel main: `f3(v1)`
  → viene creato alla chiamata della funzione un puntatore/riferimento a oggetti votazione, che punta/riferisce all'oggetto v1. Per accedere ai campi di v4 si usa l'operatore ->, mentre ai campi di v5 si usa il punto. Se si modificano i campi di v4/v5 all'interno della funzione, si stanno modificando in realtà i campi di v1!

Esercizio 1

- Supponiamo di stare lavorando su un piano cartesiano.
- Ogni punto viene identificato da una coppia di valori reali (x,y) chiamate coordinate cartesiane (x:ascissa, y:ordinata).

1. Creare una tipo struttura «coordinata» che definisca la coordinata cartesiana di un punto.
2. Scrivere una funzione che prenda in ingresso **due coordinate cartesiane di due punti** e ritorni la distanza euclidea tra i due punti.

- Esempio: se si hanno i punti A(1,1) e B(2,3) la distanza euclidea vale

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} = \sqrt{(1-2)^2 + (1-3)^2} = \sqrt{5} = 2.23607$$

3. Scrivere infine un main che legga le due ascisse e le due ordinate da tastiera, chiami la funzione e stampi a video il risultato.

Esercizio 1 - Soluzione

```
 1 #include <iostream>
 2 #include <cmath>
 3
 4 using namespace std;
 5
 6 struct coordinata
 7 {
 8     double x;
 9     double y;
10 };
11
12 double distanza(coordinata A, coordinata B){
13
14     double diff_x = A.x - B.x;
15     double diff_y = A.y - B.y;
16
17     return sqrt(diff_x*diff_x + diff_y*diff_y);
18 }
19
20 int main(){
21
22     coordinata P1, P2;
23
24     cout << "Inserisci le coordinate del primo punto:"<<endl;
25     cin >> P1.x;
26     cin >> P1.y;
27
28     cout << "Inserisci le coordinate del secondo punto:"<<endl;
29     cin >> P2.x;
30     cin >> P2.y;
31
32     double d = distanza(P1, P2);
33
34     cout << "La distanza tra i due punti vale: " << d << endl;
35     return 0;
36 }
```

Esercizio 2

1. Creare una tipo struttura «coordinata» che definisca la coordinata cartesiana di un punto
2. Scrivere una funzione che prenda in ingresso **due coordinate cartesiane di due punti** e ritorni le coordinate del punto medio tra i due punti.

- Esempio: se si hanno i punti A(1,1) e B(2,3) il punto medio ha coordinate

$$M = \left(\frac{x_A + x_B}{2}, \frac{y_A + y_B}{2}\right) = \left(\frac{1+2}{2}, \frac{1+3}{2}\right) = (1.5, 2)$$

3. Scrivere infine un main che legga le due ascisse e le due ordinate da tastiera, chiami la funzione e stampi a video il risultato.

Esercizio 2 - Soluzione

```
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  struct coordinata
7  {
8      double x;
9      double y;
10 };
11
12 coordinata punto_medio(coordinata A, coordinata B){
13
14     coordinata M;
15     M.x = (A.x + B.x)/2;
16     M.y = (A.y + B.y)/2;
17
18     return M;
19 }
20
21 int main(){
22
23     coordinata P1, P2;
24
25     cout << "Inserisci le coordinate del primo punto:"<<endl;
26     cin >> P1.x;
27     cin >> P1.y;
28
29     cout << "Inserisci le coordinate del secondo punto:"<<endl;
30     cin >> P2.x;
31     cin >> P2.y;
32
33     coordinata PM = punto_medio(P1, P2);
34
35     cout << "Le coordinate del punto medio sono: ";
36     cout << '(' << PM.x << ',' << PM.y << ')' << endl;
37     return 0;
38 }
```

Esercizio per casa

- Le coordinate cartesiane non sono l'unico modo di esprimere la posizione di un punto nel piano.
- Un altro modo sono le coordinate polari, composte sempre da due numeri reali, r e theta.
  r rappresenta la distanza del punto P dall'origine, theta rappresenta l'angolo (in radianti) che il segmento OP forma con l'asse delle x.

1. Creare due tipi struttura «coordinata_c» e «coordinata_p» che definiscano rispettivamente una coordinata cartesiana e una coordinata polare di un punto.
2. Scrivere due funzioni:
   1. la prima prende in ingresso una coordinata cartesiana, e ritorna la coordinata polare del punto
   2. la seconda fa l'opposto, prende in ingresso una coordinata polare e ritorna una coordinata cartesiana
3. Scrivere infine un main che legga da tastiera una coordinata cartesiana e una polare e stampi a video le rispettive conversioni.

Esercizio per casa

Formule di conversione:

- Da polari a cartesiane: x = r*cos(theta) , y = r*sin(theta)
- Da cartesiane a polari: r = $\sqrt{x^2 + y^2}$, theta = arctan(y/x)

- Utilizzando la libreria cmath, esistono già le funzioni trigonometriche:

```
double cos(double z) {…}
double sin(double z) {…}
double atan(double z){…}
```

- Esempio:
  - Se P ha coordinate cartesiane (3,4), le coordinate polari saranno (5, 0.9273)
  - Se Q ha coordinate polari (7.28, 0.2783), le coordinate cartesiane saranno (7, 2)