I VETTORI

Fondamenti:

- Riempimento di un vettore con numeri acquisiti da tastiera;
- Riempimento di un vettore lungo N con i numeri da 0 a N;
- Riempimento di un vettore lungo N con i numeri da N a 0;
- Riempimento di un vettore lungo N con numeri casuali interni ad un determinato range;
- Eliminazione dell'elemento m-esimo da un vettore lungo N;
- Unione di due vettori lunghi N e M in un unico vettore lungo N+M.

ORDINAMENTO DI UN VETTORE: Metodo ingenuo o sequenziale

L'idea è quella di confrontare il primo elemento del vettore con tutti gli altri: se il valore che trovo è inferiore a quello della prima cella, scambio i due valori. Una volta completata la serie di confronti relativa alla prima cella, ripeto il procedimento sulla seconda cella, poi sulla terza e così via…

NB: Scambio dei valori contenuti in due variabili A e B:

L'algoritmo:

Quante volte viene eseguito il confronto?

L'istruzione di confronto viene eseguita *n-1* volte sulla prima cella (quando $i=0$), *n-2* volte sulla seconda cella (quando $i=1$), *n-3* volte sulla terza cella (quando $i=2$), e così via… In tutto quindi il numero di confronti effettuati è:

$$(n-1)+(n-2)+\ldots+2+1=\frac{n\cdot(n-1)}{2}$$ ← Formula di Gauss

NB: Spiegazione della formula di Gauss per la somma dei primi n numeri naturali

Da fare:

Scrivi un programma che ordini in ordine crescente gli elementi di un vettore lungo N. Inserisci due contatori che verifichino quante volte viene effettuata l'istruzione di confronto e quante volte viene effettivamente eseguito lo scambio di due variabili.

Esegui il programma facendogli ordinare vettori poco disordinati (situazione *ottima*, ad esempio v={1, 2, 3, 5, 4}), vettori mediamente disordinati (situazione *media*, ad esempio v={2, 3, 1, 5, 4}) e vettori molto disordinati (situazione *pessima*, ad esempio v={7, 5, 4, 0, 2}) e riporta in una tabella il numero di confronti e di scambi effettuati.

LE MATRICI

Una matrice non è altro che una tabella contenente valori dello stesso tipo, che funziona più o meno come un vettore. Ricordiamo qualche istruzione utile quando si ha a che fare con una matrice:

- dichiarazione:

- riempimento della matrice:

- stampa della matrice

Scrivi i seguenti programmi:

1. **Gara ciclistica (vettore):** In una gara ciclistica a cronometro è stato utilizzato un sistema informatico per la registrazione dei tempi di arrivo dei corridori. Una fotocellula ha registrato i tempi effettuati e li ha salvati in un vettore in base all'ordine di arrivo: in questo modo il tempo del primo corridore a terminare la gara è stato salvato nella prima cella del vettore (*tempi[0]*), quello del secondo nella seconda cella (*tempi[1]*) e così via… Scrivi un programma che ordini l'array e comunichi in uscita il miglior tempo effettuato.

2. **Gara ciclistica (matrice):** il sistema di registrazione dei tempi della gara precedente è stato migliorato. Man mano che i corridori tagliano il traguardo la fotocellula ne registra il numero di maglia, il tempo effettuato e l'età (categoria di appartenenza). I dati vengono salvati in una matrice a tre colonne in modo che la prima colonna contanga i numeri di maglia, la seconda l'età e la terza il tempo effettuato.

   Scrivi un programma che:

   - ordini la matrice secondo il tempo di arrivo in modo da stilare la classifica finale;
   - ordini la matrice in base all'età del corridore, in modo da poter confrontare i tempi dei pari età;
   - ** ordini la matrice secondo l'età e, se esistono gruppi di corridori di pari età, essi devono essere elencati secondo il tempo effettuato, dal migliore al peggiore.

ORDINAMENTO DI UN VETTORE

Bubble sort

Il nome dell'algoritmo è dovuto al fatto che, durante l'applicazione del procedimento, i valori vengono spostati all'interno dell'array con una dinamica che ricorda il movimento delle bollicine in un bicchiere di champagne. In particolare, alcuni elementi attraversano l'array velocemente (come bollicine che emergono dal fondo del bicchiere), altri più lentamente.

La singola iterazione dell'algoritmo prevede che gli elementi dell'array siano confrontati a due a due, procedendo in un verso stabilito (che si scandisca l'array a partire dall'inizio in avanti, o a partire dal fondo all'indietro, è irrilevante).

Per esempio, saranno confrontati il primo e il secondo elemento, poi il secondo e il terzo, poi il terzo e il quarto, e così via fino al confronto fra penultimo e ultimo elemento. Per ogni confronto, se i due elementi confrontati non sono ordinati, essi vengono scambiati. Durante ogni iterazione, almeno un valore viene spostato rapidamente fino a raggiungere la sua collocazione definitiva; in particolare, alla prima iterazione il numero più grande raggiunge l'ultima posizione dell'array.

Array di esempio: 15 6 4 10 11 2

L'algoritmo:

Da fare:

1. Scrivi un programma in cui un vettore lungo N venga ordinato usando il bubble-sort.
2. Per avere un'idea della *complessità computazionale* dell'algoritmo, inserisci un contatore che conti quanti confronti vengono eseguiti nella situazione ottima (vettore quasi in ordine), media o pessima (vettore completamente disordinato). Modifica la lunghezza del vettore e riporta i dati ottenuti in una tabella.

APPLICAZIONE: TORNEO DI CALCIO

Vogliamo simulare un intero torneo di calcio tra 12 squadre con lo scopo di determinare la classifica finale e quindi il vincitore del torneo. Per farlo puoi procedere in questo modo:

- **Simulazione di tutte le partite:**

  se ogni squadra ha la stessa probabilità di vincere (non esiste ad esempio il fattore campo) ogni partita ha tre possibili esiti (1, 2, X) e può quindi essere simulata con l'estrazione di un numero random tra 0 e 2. Devo far giocare tutte le partite possibili: la prima squadra contro tutte le altre, la seconda contro tutte le altre, la terza contro tutte le altre, etc… A pensarci bene il tutto è molto simile al primo algoritmo di ordinamento che abbiamo costruito: lì la prima cella veniva confrontata con tutte le altre, la seconda con le altre, etc… Recupera quindi la doppia iterazione dell'ordinamento sequenziale e modificala per simulare tutte le partite (ricorda che ogni sfida è doppia: andata e ritorno). Dopo ogni partita devi assegnare il relativo punteggio al vincitore.

- **Classifica:**

  una volta terminate le partite, è necessario ordinare il vettore che contiene i punti accumulati per ogni squadra in modo da conoscere classifica e vincitore: usa l'algoritmo blubble sort.

Possibile variante: come possiamo simulare il fattore campo? Potresti ad esempio estrarre per ogni partita un numero casuale tra 0 e 3 (4 esiti possibili) e decidere che in caso di due esiti sui quattro possibili la vittoria vada alla squadra di casa, un esito corrisponda al pareggio e un esito alla vittoria della squadra fuori casa.

DA ALLEGARE AL PROGRAMMA:

Rispondi alle seguenti domande:

1. Di quali variabili hai bisogno? Elenca le variabili e spiega per quale motivo vengono dichiarate.
2. Come vengono assegnati i punti dopo ogni partita? Descrivi a parole ogni istruzione.
3. Come viene simulato il torneo? Descrivi in maniera chiara l'algoritmo che hai usato per far giocare ogni squadra contro tutte le altre. Quante sono le partite in totale?
4. Come viene ordinato il vettore per ottenere la classifica finale? Descrivi in maniera chiara l'algoritmo utilizzato.

LE FUNZIONI

Ecco tre esempi di funzioni:

```
#include<iostream.h>
void media1(int A, int B) {
   double m;
   m=(A+B)/2.0;
   printf("La media è %f\n", m);
}

double media2(int A, int B){
   double m;
   m=(A+B)/2.0;
   return m;
}

int main(){
   int ERRE, ESSE;
   printf("Inserisci due numeri\n");
   scanf("%d", &ERRE);
   scanf("%d", &ESSE);
   media1(ERRE, ESSE);
   printf("%f\n",media2(ERRE, ESSE)) ;

   system("PAUSE");
   return 0;
}
```

```
#include<iostream.h>
int somma(int v[ ], int n){
   int k, somma=0;
   for(k=0; k<n; k++){
      somma=somma+v[k]; }
  return somma;  }

void riempi(int v[ ], int n){
   int k;
   for(k=0; k<n; k++){
      v[k]=k; }
}

int main(){
   int vettore[5], S;
   riempi(vettore, 5);
   S=somma(vettore, 5);
   printf("S=%d\n", S);
   system("PAUSE");
   return 0;
}
```

Scrivi delle funzioni che eseguano le seguenti operazioni:

1. riempire un vettore con numeri acquisiti da tastiera
2. riempire un vettore con numeri random compresi tra 3 e 10
3. stampare a video un vettore
4. calcolare l'elemento minimo di un vettore
5. ordinare un vettore usando il bubble-sort
6. copiare un vettore v dentro un altro vettore w