Esercitazione 10

Array

Esercizio 1

- Scrivere un programma che chieda all'utente di inserire un numero di valori interi prefissato (a tempo di scrittura del programma), li inserisca in un array e stampi l'array risultante.

- Un possibile output è il seguente:

```
Inserisci un intero positivo (elemento 1 / 3): 2
Inserisci un intero positivo (elemento 2 / 3): 5
Inserisci un intero positivo (elemento 3 / 3): 1
Ecco l'array immesso:
Elemento 1: 2
Elemento 2: 5
Elemento 3: 1
```

Esercizio 1

- SOLUZIONE

```
#include <iostream>
using namespace std;

int main() {
  const int elems = 5;
  int array[elems];

  for (int i = 0; i < elems; i++) {
    cout<<"\nInserisci un intero positivo (elemento "<<i<<" / “
        <<elems<<"): ";
    cin>>array[i];
  }

  cout<<"Ecco l'array immesso:"<<endl;

  for (i = 0; i < elems; i++)
    cout<<"\nElemento"<<i<<": "<<array[i];
  cout<<endl;

  return 1;
}
```

Esercizio 2

- Scrivere un programma che definisca un array di interi di lunghezza prefissata (a tempo di scrittura del programma), lo inizializzi con valori casuali e lo stampi.

Esercizio 2

- la funzione di libreria rand() genera un numero casuale intero compreso tra 0 e RAND_MAX (prefissata, e quindi non modificabile).

- Essa permette di generare una sequenza di numeri pseudocasuali: fissato il primo valore della sequenza (chiamato seme), è fissata tutta la sequenza di valori che saranno generati nelle successive invocazioni della funzione rand().

- Infatti i numeri sono generati mediante una certa funzione f(x) che, dato l'ultimo valore x generato, genera il prossimo valore random.

Esercizio 2

- Eempio: supponiamo di aver dato come seme il valore 1. Allora il primo valore generato dalla funzione rand() sarà 41 (se la funzione rand() sulla vostra macchina e' basata sullo stesso algoritmo della funzione sulla mia macchina). La prossima volta che verra' invocata la funzione rand() si otterra' il valore f(41), che e' pari a 18647. Alla successiva invocazione si otterra' il valore di f(18647), che e' pari a 6334, e cosi' via.

Esercizio 2

- Per ottenere sequenze diverse, bisogna cambiare il valore del seme con la funzione srand(n), ove n e' il nuovo valore che si vuol dare al seme.

- In definitiva, la funzione srand va invocata una sola volta, per decidere il seme, mentre la funzione rand va invocata ogni volta che si vuole ottenere il prossimo valore.

- Per utilizzare le funzioni rand() ed srand(n) (e la costante RAND_MAX), bisogna includere il file di intestazione "cstdlib".

Esercizio 2

- Per ottenere sequenze 'quasi' completamente casuali, si può tentare di dare al seme un valore casuale.
- A questo scopo si può ad esempio sfruttare il valore di ritorno della funzione time (per usare tale funzione bisogna includere ctime), che è uguale al numero di secondi trascorsi dal 1 gennaio, 1970, GMT.
- La funzione time prende in ingresso un valore numerico, che poniamo uguale a 0.
- Si rimanda chi fosse interessato ad ulteriori dettagli alla documentazione su questa funzione di libreria.

Esercizio 2

- `#include <iostream>`
- `#include <cstdlib>`
- `#include <ctime>`

- `srand (n)` *`// n è il nuovo valore del seme`*
- `rand ()` *`// ritorna un numero casuale`*
- `time (0)` *`// ritorna i secondi dal 1/1/1970`*

Esercizio 2

- SOLUZIONE

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main() {
  const int elems = 5;
  int array[elems];

  srand (time (0));

  for (int i = 0; i < elems; i++)
    array[i] = rand ();

  cout<<"Ecco l'array immesso:"<<endl;

  for (i = 0; i < elems; i++)
    cout<<"\nElemento"<<i<<": "<<array[i];
  cout<<endl;

  return 1;
}
```

Esercizio 3

- Dato un vettore di *N* interi, inizializzati da *stdin* o casualmente, si determini il valore massimo tra quelli memorizzati nel vettore e lo si stampi
- Cogliamo anche quest'occasione per effettuare i passi fondamentali assieme
- Iniziamo dall'idea ...

Idea

- Assumi, come tentativo, che il “massimo momentaneo” sia il primo elemento del vettore
- Poi, confronta via via il “massimo momentaneo” con ciascuno dei successivi elementi del vettore
  - Ogni volta che trovi un elemento del vettore maggiore del “massimo momentaneo” sostituisci il “massimo momentaneo” con quell’elemento del vettore
- Dopo aver controllato tutti gli elementi del vettore, il “massimo momentaneo” corrisponderà al massimo del vettore

Verso un algoritmo

- Per trasformare la precedente idea in un algoritmo bisogna definire in modo preciso la struttura dati ed i passi da effettuare
- Come è fatta la struttura dati?

Struttura dati

- Array che realizza il vettore
- Costante **`N`** contenente la dimensione dell'array
- Variabile ausiliaria **`massimo`** destinata a contenere il massimo alla fine dell'algoritmo
- Variabile contatore ausiliaria per scandire l'array

Algoritmo

- Assegno alla variabile **`massimo`** il valore del primo elemento del vettore (quello di indice 0)
- Poi, scandisco il vettore da 1 a **`N`**-1 confrontando **`massimo`** con ciascun elemento
- Se trovo un elemento del vettore maggiore di **`massimo`** sostituisco il valore di **`massimo`** con il valore di quell'elemento del vettore
- Dopo aver controllato tutti gli elementi del vettore, il massimo del vettore sarà contenuto nella variabile ausiliaria **`massimo`**

Esercizio 3

- SOLUZIONE

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main() {
  int massimo;
  const int N = 10 ;
  int vettore[N];

  for (int i=0; i<N; i++)
    cin>>vettore[i];

  massimo=vettore[0];
  for (int i=1; i<N; i++)
    if (vettore[i]>massimo)
      massimo=vettore[i];

  cout<<"Il massimo del vettore e' "<<massimo<<endl;
  return 1;
}
```

Esercizio 4

- Dato un vettore di *N* interi, inizializzati da *stdin* o casualmente, si determini il valore massimo e si stampi sia il massimo sia la posizione del vettore in cui tale massimo compare

- Servono due variabili **`massimo`** e **`pos_massimo`**, o ne basta una?

Esercizio 4

- SOLUZIONE

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main() {
  int posizione;
  const int N = 10 ;
  int vettore[N];

  for (int i=0; i<N; i++)
    cin>>vettore[i];

  posizione = 0;
  for (int i=1; i<N; i++)
    if (vettore[i]>massimo)
      posizione = i;

  cout<< "Il massimo è in posizione « << posizione
      << " e vale " << vettore[posizione] <<endl;
  return 1;
}
```

Esercizio 5

- Scrivere un programma in cui sia definita ed utilizzata una funzione che, preso un vettore di numeri interi in ingresso, raddoppia il valore degli elementi del vettore. La funzione non legge da stdin e non scrive su stdout.
- Il vettore viene modificato dalla funzione?
- C'è concreto rischio overflow se si riempie il vettore con numeri completamente casuali?

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std ;

void raddoppia (int vett[], int N)
{
  for (int i=0; i<N; i++)
    vett[i] *= 2 ;
}

int main()
{
  const int M = 10;
  int A[M];
  // seme del generatore di numeri casuali
  srand (time(0));
  // inserimento dei numeri casuali nell'array e stampa
  for (int i=0 ; i<M ; i++) {
    A[i] = rand() % 1000 ; // per ridurre rischio overflow
                           // nella somma
    cout<<"Elemento "<<i<<": "<<A[i]<<endl ;
  }
  raddoppia(A,M);    // stampa nuovo contenuto

  cout<<endl<<"Dopo il raddoppio:"<<endl ;
  for (int i=0 ; i<M ; i++)
    cout<<"Elemento "<<i<<": "<<A[i]<<endl ;

  return 0 ;
}
```

Esercizio 6

- Scrivere un programma in cui sia definita ed utilizzata una funzione che, preso un vettore di numeri interi in ingresso, ne calcola la somma degli elementi. La funzione non legge da stdin e non scrive su stdout.
- Il vettore viene modificato dalla funzione?
- C'è concreto rischio overflow se si riempie il vettore con numeri completamente casuali?

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std ;

int somma (const int vett[], int N) {
  int acc=0;
  for (int i=0; i<N; i++)
    acc += vett[i]; // trascuriamo problemi di overflow
  return acc;
}

main()
{
  const int M = 10;
  int ris, A[M];
  // seme del generatore di numeri casuali
  srand (time(0));
  // inserimento dei numeri casuali nell'array e stampa
  for (int i=0 ; i<M ; i++) {
    A[i] = rand() % 1000 ; // per ridurre rischio overflow
                           // nella somma
    cout<<"Elemento "<<i<<": "<<A[i]<<endl ;
  }

  ris = somma (A,M);
  cout << "Somma elementi = " << ris <<endl ;
}
```

Esercizio 7

- Scrivere un programma che definisca un vettore di interi di lunghezza prefissata (a tempo di scrittura del programma) e lo inizializzi con valori casuali compresi tra due valori interi introdotti dall'utente. Il programma riversa quindi in un secondo vettore i soli elementi di valore pari contenuti nel primo, infine stampa il contenuto dei due vettori.
- Il riversamento DEVE essere effettuato da una funzione dedicata.
- La funzione deve o non deve ritornare il numero di elementi copiati?

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std ;

int riversa (const int a[], int pari[], int dim) {
  int lung_pari, i;
  for (i=0, lung_pari = 0; i<dim; i++)
    if ((a[i] % 2) == 0)
      pari[lung_pari++] = a[i];

  return lung_pari;
}

main()
{
  int i, lung_pari, a[20], pari[20];
  srand (time(0));
  cout<<"Inserire il valore minimo e massimo: ";
  cin>>min>>max;

  for (i=0; i<20; i++)
    a[i] = min +
     static_cast<int>((static_cast<double>(rand())/RAND_MAX) *
     (max – min)) ;

  lung_pari = riversa (a, pari, 20);

  for (i=0; i<20; i++)     // stampa array originale
    cout<<"Elemento "<<i+1<<": "<<a[i]<<endl ;

  for (i=0; i<lung_pari; i++) // stampa array pari
    cout<<"Elemento "<<i+1<<": "<<pari[i]<<endl ;
}
```

Ordinamento

- Come fate quando cercate una parola sul vocabolario?
  - E' necessario scorrere tutte le pagine?
  - In quanti passi più o meno trovate una parola in un vocabolario di, per esempio, 1000 pagine?
- Sareste riusciti a trovare una parola con la stessa velocità se l'ordine con cui le parole sono disposte nel vocabolario fosse stato casuale?
- Qual è quindi la proprietà che permette di trovare una parola così velocemente?

Ordinamento (2)

- Si possono efettuare operazioni di ricerca (e non solo) all'interno di un vettore di elementi in modo estremamente efficiente se gli elementi sono ordinati
- Ad esempio, si possono effettuare ricerche binarie
- Proprio quello l'approccio che usiamo quando cerchiamo una parola sul vocabolario

Bubblesort

- In un vettore ordinato (in senso ascendente), l'elemento in testa al vettore è necessariamente quello di valore minimo
- Possibile primo passo ordinamento: trovare l'elemento di valore minimo e metterlo in testa al vettore
- Consideriamo ora solo la *porzione di vettore* che va dal secondo elemento all'ultimo.
- Affinché il vettore originario sia ordinato, in testa a tale porzione è necessario che vi sia l'elemento di valore minimo tra tutti gli elementi della porzione stessa.
  - Ovviamente tale elemento non potrà essere maggiore di quello in testa al vettore.
- Passo successivo: trovare l'elemento di valore minimo nella porzione e metterlo in testa a tale porzione, scambiandolo con quello precedentemente in testa alla porzione.

Bubblesort (2)

- Ad esempio, dato il seguente vettore: 2 5 1 3
- Il primo passo del *bubblesort* è:
- 2 5 1 3 → 1 5 2 3
- il secondo:
- 1 5 2 3 → 1 2 5 3
- e il terzo:
- 1 2 5 3 → 1 2 3 5

Elemento di testa

Sottovettore da ordinare

Esercizio 8 (Bubblesort)

- Scrivere un programma che chieda all'utente di inserire un numero di valori interi prefissato (a tempo di scrittura del programma), li inserisca in un vettore, stampi il vettore risultante, lo ordini poi in senso crescente, e lo stampi nuovamente.
- Implementare l'ordinamento all'interno di una funzione a cui viene passato sia l'array che le sue dimensioni.

```
void bubblesort (int a[], int n) {
  for (int j=0; j<n-1; j++)
    for (int i=j+1; i<n; i++)
      if (a[i] < a[j])
      {
        int aux = a[i];
        a[i] = a[j];
        a[j] = aux;
      }
}

main()
{
  const int NUM_ELEM = 5;
  int i, j, a[NUM_ELEM];

  for (i=0; i<NUM_ELEM; i++)
  {
    cout<<"Inserire un intero positivo: ";
    cin>>a[i];
  }

  for (i=0; i<NUM_ELEM; i++)    // stampa array originale
    cout<<"Elemento "<<i+1<<": "<<a[i]<<endl ;

  bubblesort (a, NUM_ELEM);     // ordino l'array

  for (i=0; i<NUM_ELEM; i++) // stampa array ordinato
    cout<<"Elemento "<<i+1<<": "<<pari[i]<<endl ;
}
```

Funzioni per casa 1/2

- Implementare le seguenti funzioni (soluzioni non fornite):

1) void genera (int v[], int N, int TOT);
Creazione di un vettore di interi riempito con un numero casuale da 0 a TOT, per N elementi
Riceve: V, N, TOT - Restituisce: niente

2) void leggiord (int v[], int N);
Lettura di un vettore di interi letto da tastiera, per N elementi, valutando che il vettore sia inserito ordinatamente (cioè un dato è rifiutato se minore di quello inserito nella posizione precedente)
Riceve: V, N - Restituisce: niente

3) int pos (int v[], int N, int E);
Ricerca sequenziale di un elemento E in un vettore V di N elementi
Riceve: V, N, E - Restituisce: posizione dell'elemento (-1 se non esiste)

4) int ins (int v[], int N, int DIM, int E);
Inserimento di un elemento E nella posizione corretta in un vettore V ordinato di N elementi con al massimo DIM elementi, slittando a destra gli elementi successivi alla posizione di inserimento.
Riceve: V, N, DIM, E - Restituisce: il numero di elementi finale (N+1) se l'elemento è stato inserito (cioè se N<DIM), N altrimenti

Funzioni per casa 2/2

5) int canc (int v[], int N, int E);
Cancellazione di un elemento E in un vettore V ordinato di N elementi (slittando a sinistra gli elementi successivi alla posizione di cancellazione)
Riceve: V, N, E - Restituisce: il numero di elementi finale (N-1) se l'elemento è stato trovato e cancellato, N se l'elemento non è stato trovato, 0 se il vettore è vuoto

6) void stampa (int v[], int N);
Stampa di un vettore V di N elementi
Riceve: V, N - Restituisce: niente

7) int ord (int v[], int N);
Verifica che il vettore V sia ordinato
Riceve: V, N - Restituisce: 1 se v è ordinato, 0 altrimenti

8) void fusione(int v1[], int v2[], int v3[], int N);
Fonde i due vettori ordinati v1 e v2 di N elementi, nel vettore (vuoto) v3.
Riceve: V1, V2, N - Restituisce: nulla
Prima di chiamarla si richiami la funzione ord per accertarsi prima che il V1 e V2 sono ordinati .
Attenzione che la dimensione di V3 deve essere il doppio di quella di V1 e V2.

Altri esercizi (senza soluzione)

1)Si scriva una funzione somma() che riceve come parametri 3 vettori v1, v2, v3 e la loro dimensione N. La funzione confronta il primo elemento di v1 e il primo elemento di v2 e copia il maggiore in v3 come primo elemento; confronta il secondo elemento di v1 e il secondo elemento di v2 e copia il maggiore in v3 come secondo elemento; ... e così via.
La funzione restituisce il numero di volte in cui un elemento di v1 è risultato maggiore dell'elemento di v2 con cui è stato confrontato.

Si scriva poi un programma che definisce tre vettori vett1, vett2, vett3, chiede a tastiera i valori dei due vettori vett1 e vett2, richiama la funzione sopra descritta e stampa il vettore vett3 risultante e il numero restituito dalla funzione.

2) Realizzare un funzione conta che riceve in ingresso un vettore V di interi ed un elemento E e restituisce quante volte E è ripetuto in V. Scrivere poi un main() che riempie un vettore leggendo dei valori da tastiera e fermandosi quando viene digitato il numero sentinella 999. Poi stampa a video il numero che ha il maggior numero di ripetizioni nel vettore. Esempio:

Input: 15 3 5 3 7 15 5 21 15 6 9 15 5 999
Output: "il numero più ripetuto è il 15 con 4 ripetizioni"

3) Scrivere una funzione contavolte che conta quante volte un elemento x è presente in un vettore v di n elementi. La funzione riceve come parametri x, v, n. Utilizzare contavolte dentro ad una seconda funzione creaunici, per costruire, a partire da un vettore v1, un secondo vettore v2 che contiene solo gli elementi unici di v1, cioè presenti una sola volta in v1. Esempio:

v1: 2 4 3 2 7 1 3 5 1 8 9
v2: 4 7 5 8 9