Esercitazione 12 Marzo

Puntori a funzioni e le strutture

Un po' di teoria [*prova completa del 14 giugno 2019*]

Si scriva la dichiarazione delle seguenti variabili C++:

1. Un puntatore *p* a numero reale inizializzato all'indirizzo della variabile *x*:
   _____.
2. Un array q di puntatori a 50 strutture di tipo *Automobile*:
   _____.
3. Un riferimento *r* a variabile di tipo intero inizializzato alla variabile *n*:
   _____.
4. Una matrice M di 100×200 puntatori a numeri reali:
   _____.

Un po' di teoria [*prova completa del 14 giugno 2019*]

Soluzione

1. double* p = &x;
2. Automobile* q[50];
3. int& r = n;
4. double* M[100][200];

Puntatori

Un po' di teoria [*prova completa del 14 giugno 2019*]

Si scriva il valore stampato a video dal seguente programma C++:

Provateci prima di passare alle slide successive!

```
int main() {
  int a[5] = {4, 2, 6, 1, 8};
  double x = 2.5;
  for (int* p = a; p < a + 5; p += 2)
    *p -= x;
  int& r = a[3];
  int* q = &a[1] + 1;
  r = a[3] + *q;
  for (int k = 0; k < 5; k++)
     cout << a[k] << " ";
  cout << endl;
  return 0;
}
```

Puntatori

```
int a[5] = {4, 2, 6, 1, 8};
  double x = 2.5;
  for (int* p = a; p < a + 5; p += 2)
    *p -= x;
```

- Dopo aver definito ed inizializzato l'array *a* e la variabile *x*, il programma scandisce gli elementi di indice pari dell'array *a* (si parte, infatti, dal primo elemento, di indice 0, e ci si sposta di 2 posizioni) utilizzando l'aritmetica dei puntatori. Tali elementi vengono modificato sottraendo al loro valore quello di *x*.

- Attenzione: *a* è un array di numeri interi, pertanto avviene la conversione implicita al tipo *int* del risultato della sottrazione. Dopo questo passaggio, l'array *a* contiene dunque i seguenti valori: *a* = {1, 2, 3, 1, 5}.

Puntatori

```
int& r = a[3];
int* q = &a[1] + 1;
r = a[3] + *q;
for (int k = 0; k < 5; k++)
   cout << a[k] << " ";
```

- Viene quindi definito il riferimento *r*, come un altro nome dell'elemento *a*[3] e il puntatore *q* inizializzato a &*a*[1] + 1.
- Il puntatore *q* punta pertanto all'elemento successivo ad *a*[1], cioè ad *a*[2].
- L'istruzione successiva assegna ad *r* (e quindi ad *a*[3]) il valore stesso di *a*[3] (cioè 1) sommato al valore puntato da *q* (cioè al valore di a[2], ovvero 3). *a*[3] vale ora 1 + 3 = 4.
- Infine, il programma stampa a video i nuovi valori degli elementi di *a*, ottenendo la seguente stampa: 1 2 3 4 5.

Puntatori a funzione

[Esercizio C022] – Array dei puntatori a funzione

- Scrivi un il programma abbia lo scopo di gestire un vettore di 5 interi attraverso tre funzioni: *leggi_vettore, stampa_vettore e somma_vettore*:
  - leggi_vettore: Chiede all'utente di inserire gli elementi nel vettore,
  - stampa_vettore: Stampa gli elementi del vettore,
  - somma_vettore: Calcola la somma degli elementi del vettore e stampa il risultato .
- Tutte tre le funzioni non restituiscono alcun risultato.

- Il programma principale utilizza **un array di puntatori a funzione** per consentire eseguire tutte tre operazioni attraverso un Il ciclo for nel seguente ordine: leggi, stampa, somma

Puntatori a funzioni

[C024] – Array dei puntatori a funzione

- Implementino le seguenti funzioni:
  - *minimum* che riceva come parametri un array *a* di numeri reali e la sua dimensione *n* (un numero intero) e restituisca come valore di ritorno il valore del minimo elemento dell'array (un numero reale).
  - *maximum* che riceva come parametri un array *a* di numeri reali e la sua dimensione *n* (un numero intero) e restituisca come valore di ritorno il valore del massimo elemento dell'array (un numero reale).
  - *sum* che riceva come parametri un array *a* di numeri reali e la sua dimensione *n* (un numero intero) e restituisca come valore di ritorno la somma degli elementi dell'array (un numero reale).
  - *sum_square* che riceva come parametri un array *a* di numeri reali e la sua dimensione *n* (un numero intero) e restituisca come valore di ritorno la somma dei quadrati degli elementi dell'array (un numero reale).
  - *mean* che riceva come parametri un array *a* di numeri reali e la sua dimensione *n* (un numero intero) e restituisca come valore di ritorno la media degli elementi dell'array (un numero reale).
- Si scriva quindi un programma C++ che operi come segue:
  - dichiari un array *v* di 10 numeri reali.
  - dichiari un array ***pf* di cinque puntatori a funzione** che ricevano come parametri un array di numeri reali e un numero intero e restituiscano come valore di ritorno un numero reale e inizializzi gli elementi **dell'array *pf* con gli indirizzi delle cinque funzioni** precedentemente implementate (*minimum*, *maximum*, *sum*, *sum_square*, *mean*).
  - chieda all'utente di inserire da tastiera i valori degli elementi dell'array *v*.
  - chieda all'utente di scegliere quale operazione desidera applicare agli elementi dell'array *v*, tra le cinque precedentemente implementate e i cui puntatori sono disponibili in *pf*.
  - Utilizzando il puntatore contenuto nell'array *pf*, chiami la funzione corrispondente all'operazione desiderata e stampi a video il risultato.

Uso dei puntatori come parametri in una funzione

[Esercizio C021] – Puntatori e array (=C014)

- Si scriva la **funzione** C++ *prodotto_scalare* che **riceva come parametri il puntatore *px* al primo elemento** di un array di numeri reali, il puntatore *py* al primo elemento di un array di numeri reali e la dimensione comune *n* dei due array (un numero intero). Utilizzando **l'aritmetica dei puntatori**, la funzione dovrà scandire i due array e calcolarne il prodotto scalare, **restituito come valore di ritorno (un numero reale).**
- Si scriva quindi un programma C++ per verificare il corretto funzionamento della funzione. Il programma chiederà all'utente di immettere da tastiera i valori per i due array, chiamerà la funzione *prodotto_scalare* e ne stamperà a video il valore di ritorno.

Esempio: se l'array puntato da *px* vale {1.0, 3.0, 2.5, 0.0, 1.2} e l'array puntato da *py* vale {2.0, 1.0, 2.0, 3.8, 10.0} (si ha quindi $n = 5$), la funzione restituisce il valore del prodotto scalare dei due array, ovvero: $1.0 \times 2.0 + 3.0 \times 1.0 + 2.5 \times 2 + 0.0 \times 3.8 + 1.2 \times 10.0 = 22.0$.

- *Per fare di più*: calcolare anche la distanza tra gli array puntati da *px* e da *py* e restituire, come parametro di uscita (anziché come valore di ritorno) una struttura che contenga due campi, il prodotto scalare e la distanza calcolati dalla funzione (due numeri reali). Modificare poi il programma di prova in modo che stampi entrambi i valori restituiti dalla funzione.

Uso dei puntatori come parametri in una funzione

[C025] – Puntatori e matrici

- Scrivi un programma che calcola la somma di due matrici 3x3 utilizzando puntatori. La somma di due matrici A e B di dimensione 3x3 deve essere memorizzata in una matrice C, e infine il risultato deve essere stampato a video.
  - Le matrici A, B, e C devono essere dichiarate come matrici bidimensionali di dimensione 3x3 in *main()*
  - Creare una funzione che accetta tre matrici (due per l'input e una per il risultato) come puntatori e somma gli elementi corrispondenti di A e B, memorizzando il risultato in C.
  - Utilizzare i puntatori per accedere agli elementi delle matrici e calcolare la somma senza utilizzare l'indicizzazione degli array.
  - Il programma in *main()* deve chiedere all'utente di inserire i valori delle matrici A e B.
  - Una volta calcolata la somma, la matrice risultante C deve essere stampata a video.

[C026] – Puntatori e strutture

- Scrivi un programma che gestisce informazioni relative a due film, utilizzando strutture (struct) e puntatori.
- Il programma deve consentire all'utente di inserire i dati di due e determinare quale dei due film è il più recente.
- Se entrambi i film hanno lo stesso anno di uscita, il programma deve notificare l'utente di questa situazione.

  - Il programma deve utilizzare i puntatori per gestire e manipolare le strutture dei film. I puntatori vengono utilizzati per passare le strutture alla funzione e per accedere ai campi.

  - Creare una funzione che accetta i puntatori a due strutture film e confronta gli anni di uscita dei due film, restituendo il puntatore al film più recente.

  - Il programma in *main()* deve chiedere all'utente di inserire i dati di due film (titolo, durata e anno) e usare questi dati per confrontare i film.

  - Il programma deve stampare il titolo del film più recente, oppure indicare che i due film hanno lo stesso anno di uscita.

[C027] – Puntatori e strutture

- Scrivi un programma che gestisce un array di film utilizzando strutture (struct) e puntatori. L'utente dovrà inserire le informazioni di tre film, e il programma dovrà determinare e restituire il film più recente confrontando l'anno di uscita.
  - il programma utilizza un array di puntatori a strutture film. Ogni elemento dell'array punta a un film diverso.
  - La funzione *piu_recente* riceve come argomento l'array di puntatori a film e il numero di film.
  - La funzione confronta l'anno di uscita dei film e restituisce il puntatore al film che ha l'anno di uscita più recente.
  - Se ci sono più film con lo stesso anno di uscita, la funzione restituisce il primo film con quell'anno.
  - Il programma chiederà all'utente di inserire il titolo, la durata e l'anno di uscita di tre film.
  - Dopo aver ottenuto i dati, verrà invocata la funzione *piu_recente* per determinare il film più recente.
  - Infine, verrà stampato il titolo del film più recente, insieme al suo anno di uscita.

[C029] – Puntatori e strutture e l'array dei puntatori a funzione

Scrivi un programma che gestisca una lista di film, memorizzando i dati relativi a ciascun film in una struttura film. Ogni film ha un titolo, una durata, e un anno di uscita. Il programma dovrà svolgere le seguenti operazioni:

- permettere all'utente di inserire i dati di tre film.
- creare un array di puntatori a queste strutture film.
- utilizzare un array di puntatori a funzioni per eseguire le seguenti operazioni in ordine:
  - stampare le informazioni sui film.
  - aggiornare la durata di ogni film, impostando la durata a 120 minuti per i film con durata maggiore.
  - stampare nuovamente le informazioni sui film aggiornati.

- Perciò servono 2 funzioni:
  - funzione *aggiorna_durata*: prende un array di puntatori a film e un intero n, e aggiorna la durata del film a 120 minuti se la durata è maggiore di 120 minuti.
  - funzione *stampa_film*: prende un array di puntatori a film e un intero n, e stampa le informazioni di ogni film nell'array.

- In *main():*
  - viene creato un array di puntatori a film per memorizzare i dati dei film.
  - viene creato un array di puntatori a funzioni per eseguire le operazioni di stampa e aggiornamento durata.
  - un ciclo for esegue ogni funzione nell'array di puntatori a funzioni, passando l'array di film.

Puntatori a funzione

[C028] – Puntatori a funzione, il puntatore a funzione come parametro formale di un'altra funzione.

- Si modifichi la funzione C++ *bubbleSort*, che implementa l'algoritmo di ordinamento BubbleSort, facendo in modo che possa gestire ordinamenti in senso crescente e decrescente. A tale scopo, la funzione *bubbleSort* riceverà come parametri un array di numeri interi *v*, la sua dimensione *n* e un puntatore *pf* a una funzione che riceva come parametri due numeri interi e restituisca come valore di ritorno un numero intero. La funzione puntata da *pf* restituisce 1 se l'ordine degli interi è corretto rispetto all'ordinamento prescelto e 0 altrimenti. Quindi la funzione *bubbleSort* chiamerà la funzione puntata da *pf*, passandole come parametri gli elementi *v*[*j*] e *v*[*j* + 1] dell'array *v* e li scambierà se la chiamata alla funzione puntata da *pf* restituirà 0.
- Per implementare l'ordinamento nei due sensi avete le seguenti funzioni C++:
  - La funzione *maggiore* che riceva come parametri due numeri interi *a* e *b* e restituisca 1 se *a* è maggiore di *b* e zero altrimenti.
  - La funzione *minore* che riceva come parametri due numeri interi *a* e *b* e restituisca 1 se *a* è minore di *b* e zero altrimenti.
- Si scriva, infine, un programma C++ che dichiari un array *w* di 10 numeri interi, chieda all'utente di inserirne i valori da tastiera, chieda all'utente di scegliere se desidera ordinare gli elementi di *w* in senso crescente o decrescente, chiami la funzione *bubbleSort* passando a *pf* la funzione *maggiore* nel caso di ordinamento decrescente e la funzione *minore* nel caso di ordinamento crescente e stampi a video il valore degli elementi dell'array ordinato.