

Esercitazione 19 Marzo

Allocazione dinamica, funzioni recursive

Uso dei puntatori (SETTIMANA precedente)

[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.
- Più difficile: scrivi la funzione con solo un ciclo *for* (*senza cicli annidati*)

```

void somma_matrici(int (*A)[dim], int (*B)[dim], int (*C)[dim], int dim) {
    for (int i = 0; i < dim; i++) {
        for (int j = 0; j < dim; j++) {
            *(*C + i) + j) = *(*A + i) + j) + *(*B + i) + j);
        }
    }
}

```

```

void somma_matrici(int (*A)[dim], int (*B)[dim], int (*C)[dim], int dim) {
    for (int i = 0; i < dim*dim; i++) {
        *(*C + i) = *(*A + i) + *(*B + i));
    }
}

```

- Un array a due dimensioni è un array monodimensionale i cui elementi sono a loro volta degli array.
- Gli elementi degli array vengono memorizzati riga per riga.



Allocazione dinamica della memoria

Un po' di teoria [*prova completa del 22 luglio 2019*]

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

Provateci prima di passare alle slide successive!

```
int main() {  
    int* p = new int[3];  
    int x = 3;  
    for (int* q = p; q < p + 3; q++)  
        *q = x++;  
    int& a = p[2];  
    a -= (p[0] - 1);  
    *p += 2;  
    for (int k = 0; k < 3; k++)  
        cout << p[k] << " ";  
    cout << endl;  
    delete [] p;  
    return 0;  
}
```

Allocazione dinamica della memoria

- Il programma inizia allocando dinamicamente un array di tre numeri interi e assegnando a *p* un puntatore a tale array.
- I valori degli elementi dell'array puntato da *p* vengono quindi inizializzati attraverso un ciclo *for* che, utilizzando l'aritmetica dei puntatori, assegna loro il valore della variabile *x*. Questa è inizializzata al valore 3 e viene incrementata ad ogni iterazione del ciclo *for*.
- L'array puntato da *p* è pertanto inizializzato con i seguenti valori: {3, 4, 5}.

```
int main() {  
    int* p = new int[3];  
    int x = 3;  
    for (int* q = p; q < p + 3; q++)  
        *q = x++;  
}
```

Allocazione dinamica della memoria

- Le istruzioni successive modificano il contenuto dell'array puntato da p in questo modo:
 - viene dichiarato il riferimento a come un altro nome dell'elemento $p[2]$;
 - attraverso il riferimento a , tale elemento viene decrementato di $p[0] - 1$ (cioè $3 - 1 = 2$) e assume quindi il valore $5 - 2 = 3$;
 - il valore puntato da p (cioè il primo elemento dell'array, $p[0]$) viene incrementato di 2 e assume pertanto il valore $3 + 2 = 5$.
- Il ciclo for successivo stampa il contenuto dell'array puntato da p dopo tali modifiche. Si ottiene quindi la seguente stampa: 5 4 3. Infine, l'array viene deallocato e il programma termina.

```
int& a = p[2];  
a -= (p[0] - 1);  
*p += 2;  
for (int k = 0; k < 3; k++)  
    cout << p[k] << " ";  
cout << endl;  
delete [] p;  
return 0;
```

Un po' di teoria

Scrivere le istruzioni C++ necessarie per allocare *dinamicamente* le seguenti variabili:

1. Un numero intero:

_____.

1. Un array di 150 numeri reali:

_____.

2. Una stringa di 64 caratteri alfanumerici:

_____.

3. Un array di 50 puntatori a numeri interi:

_____.

Soluzione

1. `int* p = new int;`
2. `double* p = new double[150];`
3. `char* p = new char[65];`
4. `int** pp = new int*[50];`

Allocazione dinamica della memoria

[Esercizio C031] – Concatenamento

- Si scriva la funzione C++ *append* che concateni due array di numeri. La funzione riceve come parametri i puntatori a due array di numeri interi *pa* e *pb* e le loro dimensioni *na* e *nb* (due numeri interi) e restituisce come valore di ritorno il puntatore a un array di numeri interi.
- La funzione allocherà dinamicamente un array di $(na + nb)$ elementi e vi copierà gli elementi degli array puntati da *pa* e *pb*. La funzione restituirà infine il puntatore all'array risultante. Sarà responsabilità del programma chiamante deallocare la memoria allocata dalla funzione.
- Si scriva quindi un programma C++ per verificare il corretto funzionamento della funzione. Il programma chiederà all'utente di immettere da tastiera le dimensioni *nx* e *ny* di due array di numeri interi, allocherà dinamicamente gli array e chiederà all'utente di inserire i valori per entrambi gli array. Il programma, chiamerà la funzione *append* e stamperà a video l'array concatenato risultante.

Esempio: se l'array puntato da *pa* è {1, 3, 5, 6, 8} ($na = 5$) e l'array puntato da *pb* è {2, 4, 10} ($nb = 3$), l'array concatenato risultante sarà {1, 3, 5, 6, 8, 2, 4, 10}.

Allocazione dinamica della memoria

[Esercizio C032] – Vettore di dimensioni variabili

Scrivere un programma in C++ che gestisca un array dinamico di caratteri, permettendo all'utente di eseguire le seguenti operazioni in seguente ordine:

1. **Inserire caratteri nell'array:** Il programma deve chiedere all'utente quanti caratteri vuole inserire, allocare dinamicamente un array per questi caratteri e poi consentire all'utente di inserire i caratteri uno alla volta.
2. **Visualizzare l'array:** Dopo aver inserito i caratteri, il programma deve stampare l'array creato dall'utente.
3. **Inserire un nuovo carattere:** Successivamente, il programma deve chiedere all'utente la posizione in cui inserire un nuovo carattere e il carattere stesso. **La funzione** dovrà gestire l'inserimento e restituire il puntatore all'array aggiornato.
4. **Cancellare un carattere:** Il programma deve poi chiedere all'utente di indicare la posizione dell'elemento che vuole cancellare. Una funzione dovrà gestire la cancellazione del carattere nella posizione specificata e restituire il puntatore all'array modificato.
5. **Visualizzare l'array finale:** Infine, il programma deve stampare l'array dopo ogni operazione di inserimento e cancellazione, in modo da permettere all'utente di vedere l'array aggiornato.
6. **Deallocazione della memoria:** Al termine, il programma deve deallocare la memoria dinamica utilizzata per gli array.

Allocazione dinamica della memoria e le strutture

[Esercizio C034] – Inventario di macchine

Scrivere un programma in C++ per gestire un inventario di macchine in un magazzino. L'utente deve poter inserire i dati di un numero prefissato di macchine (ad esempio, n=3) e il programma dovrà memorizzare e stampare questi dati. Si definisce

- una **struttura Macchina** che contiene informazioni relative a ciascun veicolo, come:
 - Tipo (stringa di caratteri): Tipo di macchina (es. SUV, berlina, ecc.)
 - Marca (stringa di caratteri): Marca della macchina (es. Fiat, BMW, ecc.)
 - Targa (stringa di caratteri di lunghezza 7): Targa della macchina.
 - Anno (intero): Anno di fabbricazione della macchina.
- **Funzione inseriscidati:** Permette all'utente di inserire i dati relativi a ciascuna macchina (marca, tipo, targa, anno). I dati vengono memorizzati in una struttura di tipo Macchina, il cui indirizzo viene passato come parametro.
- **Funzione stampa:** Stampa i dati di ciascuna macchina contenuti nell'inventario. Mostra le informazioni come la marca, il tipo, la targa e l'anno della macchina. La funzione prende come parametro l'indirizzo della struttura.

parametro ingresso = puntatore a Macchina
puntatore a Macchina

Utilizzando la memoria dinamica, viene allocato un **array di puntatori a Macchina**. Ogni puntatore dell'array viene utilizzato per memorizzare una macchina, in modo che i dati possano essere gestiti in modo flessibile. Dopo che tutte le macchine sono state inserite, il programma stampa l'inventario.

Allocazione dinamica della memoria e le strutture

[Esercizio C036] – Inventario di macchine 2

Dopo aver fatto inserire un numero iniziale di macchine, date la possibilità all'utente di aggiungere ulteriori macchine nell'inventario oppure di rimuoverle, scegliendo la targa della macchina.

- Per fare ciò, sarà necessario riallocare la memoria (creare una nuova struttura dati più grande o più piccola, copiando i dati dal precedente array e facendo inserire i nuovi dati dall'utente).

Un po' di teoria

[Prova pratica del 22 luglio 2019]

- Si scrivano i valori stampati a video
dal seguente programma C++:

```
int test(int n) {  
    if (n <= 0)  
        return 0;  
    else if (n%2 == 0)  
        return n + test(n - 1);  
    else  
        return test(n - 1) - n;  
}  
  
int main() {  
    int r = test(5);  
    cout << r << endl;  
    return 0;  
}
```

Un po' di teoria

La funzione *main* chiama la funzione *test*, passando 5 come parametro (*test*(5)).

Chiamata *test*(5): 5 è maggiore di zero, 5 non è divisibile per 2, quindi viene chiamata ricorsivamente la funzione *test* (*test*(4)) e restituito ***test*(4) – 5**.

Chiamata *test*(4): 4 è maggiore di zero. 4 è divisibile per 2, quindi viene chiamata ricorsivamente la funzione *test* (*test*(3)) e restituito **4 + *test*(3)**.

Chiamata *test*(3): 3 è maggiore di zero, 3 non è divisibile per 2, quindi viene chiamata ricorsivamente la funzione *test* (*test*(2)) e restituito ***test*(2) – 3**.

Chiamata *test*(2): 2 è maggiore di 0, 2 è divisibile per 2, quindi viene chiamata ricorsivamente la funzione *test* (*test*(1)) e restituito **2 + *test*(1)**.

Chiamata *test*(1): 1 è maggiore di 0, 1 non è divisibile per 2, quindi viene chiamata ricorsivamente la funzione *test* (*test*(0)) e restituito ***test*(0) – 1**.

Chiamata *test*(0): la condizione di terminazione $n \leq 0$ è verificata e viene restituito 0.

```
int test(int n) {
    if (n <= 0)
        return 0;
    else if (n%2 == 0)
        return n + test(n - 1);
    else
        return test(n - 1) - n;
}

int main() {
    int r = test(5);
    cout << r << endl;
    return 0;
}
```

Un po' di teoria

A ritroso si ha pertanto:

La chiamata a *test*(0) restituisce 0

La chiamata a *test*(1) restituisce $test(0) - 1 = 0 - 1 = -1$

La chiamata a *test*(2) restituisce $2 + test(1) = 2 + (-1) = 2 - 1 = 1$

La chiamata a *test*(3) restituisce $test(2) - 1 = 1 - 3 = -2$

La chiamata a *test*(4) restituisce $4 + test(3) = 4 + (-2) = 4 - 2 = 2$

La chiamata a *test*(5) restituisce $test(4) - 5 = 2 - 5 = -3$

```
int test(int n) {  
    if (n <= 0)  
        return 0;  
    else if (n%2 == 0)  
        return n + test(n - 1);  
    else  
        return test(n - 1) - n;  
}  
  
int main() {  
    int r = test(5);  
    cout << r << endl;  
    return 0;  
}
```

Funzioni ricorsive

[Esercizio C037] Potenza

- Si scriva in C++ una funzione ricorsiva che, dati due numeri interi M ed N ricevuti come parametri, verifichi che N sia nullo (se N è nullo, la funzione termina restituendo 1), calcoli e restituisca come valore di ritorno la potenza M^N (un numero intero).
- Si scriva quindi un programma C++ per verificare il corretto funzionamento della funzione.
- Il programma chiederà all'utente di inserire una tastiera due numeri interi, chiamerà la funzione e stamperà a video il suo valore di ritorno. Le operazioni si ripeteranno finché l'utente lo desidera.

Per fare di piu: Estendere la funzione in modo che possa ricevere un esponente negativo.

In tal caso, la funzione restituirà il valore $M^{(-N)} = 1 / M^N$.

Funzioni ricorsive

[Esercizio C038] Palindroma

Scrivere un programma in C++ che utilizzi una funzione ricorsiva per determinare se una parola inserita dall'utente è palindroma. Una parola si dice palindroma se può essere letta nello stesso modo da sinistra verso destra e da destra verso sinistra (ad esempio, "radar", "amoroma" sono parole palindromi).

- La funzione ricorsiva dovrà confrontare i caratteri alle estremità della parola (primo e ultimo, secondo e penultimo, ecc.) e, se sono uguali, proseguire il confronto per la parte centrale della parola.
- Se la parola è palindroma, la funzione restituirà *true*; altrimenti, restituirà *false*.
- Scrivere il programma che:
 - acquisisce una parola dall'utente,
 - chiama la funzione ricorsiva verifica se la parola è palindroma,
 - stampa il risultato (palindroma o non palindroma) a schermo.