Puntatori in C++

I puntatori sono uno degli aspetti più distintivi e potenti del linguaggio di programmazione C++. Permettono un controllo dettagliato sulla memoria e consentono di lavorare direttamente con gli indirizzi di memoria. In questo capitolo, esploreremo i puntatori in C++ in modo approfondito.

Concetto di Puntatore

Un puntatore è una variabile che contiene l'indirizzo di memoria di un'altra variabile. In altre parole, un puntatore "punta" a una posizione specifica in memoria. Utilizzando puntatori, puoi accedere e manipolare dati in memoria in modo più diretto.

```
int valore = 42;     // Dichiarazione di una variabile intera
int *puntatore = &valore; // Dichiarazione di un puntatore a un intero che punta a 'valore'
```

Nell'esempio sopra, puntatore contiene l'indirizzo di memoria di valore. Possiamo accedere a valore utilizzando il puntatore.

Dichiarazione e Inizializzazione dei Puntatori

Per dichiarare e inizializzare un puntatore, è necessario specificare il tipo di dati a cui il puntatore farà riferimento. La sintassi generale è:

```
tipoDati *nomePuntatore;
```

Dove tipoDati rappresenta il tipo di dati della variabile a cui il puntatore farà riferimento, e nomePuntatore è il nome del puntatore.

```
int *ptrInt;     // Puntatore a un intero
double *ptrDouble; // Puntatore a un double
```

Un puntatore deve essere inizializzato prima di essere utilizzato. Puoi inizializzarlo con l'indirizzo di memoria di una variabile esistente:

```
int valore = 10;
int *ptrValore = &valore; // Inizializzazione con l'indirizzo di 'valore'
```

Operazioni con Puntatori

Dereferenziamento

Per accedere al valore a cui un puntatore fa riferimento, utilizziamo l'operatore di dereferenziamento *. Ad esempio:

```
int valore = 42;
int *ptr = &valore;

int valoreDereferenziato = *ptr; // valoreDereferenziato ora contiene 42
```

L'operatore * viene utilizzato sia per dereferenziare un puntatore e ottenere il valore a cui punta, sia per dichiarare un puntatore.

Operazioni Aritmetiche sui Puntatori

I puntatori possono essere utilizzati in operazioni aritmetiche, ma il loro comportamento dipende dal tipo di dati a cui puntano. Ad esempio:

```
int array[] = {10, 20, 30, 40, 50};
int *ptr = array; // Il puntatore punta al primo elemento dell'array

int primoValore = *ptr;  // primoValore contiene 10
int secondoValore = *(ptr + 1); // secondoValore contiene 20
```

In questo esempio, ptr punta al primo elemento dell'array array, e utilizzando operazioni aritmetiche, possiamo accedere agli altri elementi dell'array.

Puntatori a Funzione

In C++, è possibile utilizzare puntatori per memorizzare l'indirizzo di una funzione. Questo consente di passare funzioni come argomenti ad altre funzioni o di memorizzare un set di funzioni per essere chiamate dinamicamente.

```
#include <iostream>

void Saluta() {
   std::cout << "Ciao, mondo!" << std::endl;
}

int main() {
   void (*ptrFunzione)() = &Saluta; // Puntatore a una funzione che non accetta argomenti e non restituisce nulla
   (*ptrFunzione)(); // Chiamata della funzione attraverso il puntatore

   return 0;
}
```

Puntatori e Gestione della Memoria

I puntatori sono spesso utilizzati per gestire dinamicamente la memoria in C++. La memoria allocata dinamicamente consente di creare oggetti la cui durata non è limitata dalla loro vita all'interno di uno specifico ambito di blocco. Puoi utilizzare new per allocare memoria dinamicamente e delete per deallocarla quando non è più necessaria.

```
int *ptr = new int;     // Alloca dinamicamente un intero
*ptr = 42;              // Assegna un valore all'intero
delete ptr;             // Dealloca la memoria
```

È importante ricordare di deallocare la memoria quando non è più necessaria per evitare perdite di memoria (memory leaks).

Puntatori e Array

Gli array sono correlati ai puntatori in C++. Quando dichiari un array, in realtà stai creando un puntatore al primo elemento dell'array. Ad esempio:

int array[] = {10, 20, 30, 40, 50};
int *ptr = array; // Il puntatore punta al primo elemento dell'array

Puoi utilizzare i puntatori per accedere agli elementi dell'array come visto precedentemente.

Esercizi

Esercizi sulla gestione dei puntatori in C++ in ambito di programmazione procedurale:

1. Scambio di Valori: Scrivi un programma che scambi i valori di due variabili intere utilizzando puntatori. Chiedi all'utente di inserire due numeri, quindi crea una funzione che accetta due puntatori e scambia i valori delle variabili a cui puntano.
2. Somma di Elementi in un Array: Scrivi una funzione che accetta un array di numeri interi e restituisce la somma di tutti gli elementi. Utilizza un puntatore per scorrere l'array e calcolare la somma.
3. Copia di una Stringa: Scrivi una funzione che accetta due puntatori a stringhe (array di caratteri) e copia il contenuto della prima stringa nella seconda stringa utilizzando puntatori. Assicurati di gestire correttamente la terminazione della stringa.
4. Verifica di Palindromi: Scrivi una funzione che verifichi se una parola è un palindromo utilizzando puntatori. Chiedi all'utente di inserire una parola, quindi utilizza un puntatore per confrontare i caratteri dalla prima metà con quelli della seconda metà.
5. Ordinamento di un Array: Scrivi una funzione che accetti un array di numeri interi e utilizzi i puntatori per ordinare gli elementi in ordine crescente. Puoi utilizzare l'algoritmo di selezione o il bubble sort per l'ordinamento.