Funzioni e Passaggio di Parametri in C++

Nella programmazione procedurale, le funzioni svolgono un ruolo cruciale nel suddividere il codice in unità più gestibili e riutilizzabili. In questo capitolo, esploreremo le funzioni in C++ e come passare i parametri ad esse.

Funzioni in C++

Una funzione è un blocco di codice che esegue un'attività specifica quando viene chiamato. In C++, una funzione è definita dai seguenti elementi principali:

Prototipo di Funzione:

Un prototipo di funzione è una dichiarazione anticipata della funzione che specifica il suo nome, il tipo di ritorno e i parametri. Ad esempio:

```
int somma(int a, int b);
```

Definizione di Funzione:

La definizione di una funzione fornisce il corpo della funzione, che contiene le istruzioni per svolgere una specifica operazione. Ad esempio:

```
int somma(int a, int b) {
   return a + b;
}
```

Chiamata di Funzione:

Una funzione viene chiamata specificando il suo nome e fornendo i valori degli argomenti necessari. Ad esempio:

```
int risultato = somma(5, 3);
```

Passaggio di Parametri

Quando si chiamano le funzioni, è comune passare loro dei parametri, ovvero dei valori o delle variabili che la funzione userà durante la sua esecuzione. Ci sono due modi principali per passare parametri alle funzioni in C++:

Passaggio per Valore

Nel passaggio per valore, la funzione riceve una copia dei valori passati come argomenti. Questo significa che le modifiche ai parametri all'interno della funzione non influenzeranno le variabili originali all'esterno della funzione. Ecco un esempio:

```
void incrementaPerValore(int x) {
   x++;
}

int numero = 5;
incrementaPerValore(numero);
// 'numero' rimane uguale a 5
```

Passaggio per Riferimento

Nel passaggio per riferimento, la funzione riceve un riferimento o un'istanza di una variabile come parametro. Questo consente alla funzione di accedere e modificare direttamente la variabile originale. Ecco un esempio:

```
void incrementaPerRiferimento(int &x) {
   x++;
}

int numero = 5;
incrementaPerRiferimento(numero);
// 'numero' ora è uguale a 6
```

Overloading delle Funzioni

L'overloading delle funzioni è una caratteristica potente in C++ che consente di definire più funzioni con lo stesso nome ma con differenti parametri. Questo permette di utilizzare lo stesso nome di funzione per operazioni simili o correlate, semplificando così la struttura del codice e migliorando la chiarezza. L'overloading delle funzioni è uno dei concetti chiave della programmazione orientata agli oggetti ed è ampiamente utilizzato in C++.

Regole per l'overloading delle funzioni

Per definire correttamente una serie di funzioni sovraccaricate, è necessario rispettare alcune regole:

Stessa Nome: Tutte le funzioni sovraccaricate devono avere lo stesso nome.
Differenza nei Parametri: Le funzioni sovraccaricate devono differire nei tipi o nel numero dei parametri. Questo è ciò che consente al compilatore di distinguere quale funzione chiamare in base agli argomenti passati.
Differenza nei Tipi di Ritorno: I tipi di ritorno delle funzioni sovraccaricate possono essere diversi, ma non è sufficiente per il compilatore per distinguere tra di loro. La differenza principale deve essere nei parametri.
Esempio di Overloading delle Funzioni
Ecco un esempio che mostra come definire e utilizzare funzioni sovraccaricate in C++:

```
#include <iostream>
class Calcolatrice {
public:
   int somma(int a, int b) {
      return a + b;
   }

   double somma(double a, double b) {
      return a + b;
   }

   int somma(int a, int b, int c) {
      return a + b + c;
   }
};
```

```
int main() {
    Calcolatrice calc;

    int risultato1 = calc.somma(5, 3);
    double risultato2 = calc.somma(3.14, 2.0);
    int risultato3 = calc.somma(2, 3, 4);

    std::cout << "Risultato 1: " << risultato1 << std::endl;
    std::cout << "Risultato 2: " << risultato2 << std::endl;
    std::cout << "Risultato 3: " << risultato3 << std::endl;

    return 0;
}
```

In questo esempio, abbiamo una classe Calcolatrice con tre diverse funzioni somma, ognuna accetta un numero diverso di argomenti o tipi di argomenti. Quando chiamiamo la funzione somma, il compilatore determina quale versione della funzione chiamare in base ai parametri passati.

Nel main(), chiamiamo la funzione somma tre volte con differenti tipi di argomenti e il compilatore seleziona automaticamente la versione corretta di somma da utilizzare.

Vantaggi dell'Overloading delle Funzioni

L'overloading delle funzioni offre diversi vantaggi:

Chiarezza del Codice: Utilizzando nomi di funzioni simili per operazioni simili, il codice diventa più chiaro e leggibile.
Riutilizzo del Codice: Puoi riutilizzare lo stesso nome di funzione per operazioni simili su diversi tipi di dati.
Flessibilità: Puoi definire funzioni che si comportano in modo diverso a seconda dei parametri passati, il che offre maggiore flessibilità nell'uso delle funzioni.

In sintesi, l'overloading delle funzioni in C++ è una potente tecnica di programmazione che semplifica la gestione delle funzioni con nomi simili ma comportamenti diversi. Consentendo di utilizzare lo stesso nome di funzione per casi simili, contribuisce a rendere il codice più chiaro, leggibile e manutenibile.

Conclusioni

Le funzioni sono uno strumento fondamentale nella programmazione procedurale in C++. Il passaggio di parametri alle funzioni può avvenire per valore o per riferimento, a seconda della necessità. Il concetto di overloading consente di definire funzioni con lo stesso nome ma con comportamenti diversi. La comprensione di queste nozioni è essenziale per scrivere codice C++ efficace e modulare.

Esercizi sulle funzioni in C++

Ecco dieci esercizi sull'uso delle funzioni in ambito procedurale con C++:

1. Calcolatrice Semplice: Scrivi un programma C++ che simuli una calcolatrice. Crea una funzione per ciascuna operazione matematica (somma, sottrazione, moltiplicazione, divisione) e permetti all'utente di scegliere quale operazione eseguire.
2. Conversione di Unità: Scrivi una funzione che converta una temperatura da Celsius a Fahrenheit e un'altra funzione che effettui la conversione opposta da Fahrenheit a Celsius. Chiedi all'utente di inserire una temperatura e visualizza entrambe le conversioni.
3. Controlla Palindromi: Scrivi una funzione che accetti una stringa e determini se è un palindromo (una stringa che si legge allo stesso modo da sinistra a destra e da destra a sinistra).
4. Calcola il Massimo Comun Divisore: Crea una funzione che calcoli il Massimo Comun Divisore (MCD) di due numeri interi utilizzando l'algoritmo di Euclide.
5. Fattoriale: Scrivi una funzione che calcoli il fattoriale di un numero intero non negativo. Chiedi all'utente di inserire un numero e visualizza il suo fattoriale.
6. Stampa Numeri Primi: Scrivi una funzione che determini se un numero è primo o no. Quindi, utilizza questa funzione per stampare tutti i numeri primi fino a un limite specificato.
7. Calcola la Potenza: Crea una funzione che calcoli una base elevata a una potenza specifica. Chiedi all'utente di inserire la base e l'esponente.
8. Calcola l'Indice di Massa Corporea (IMC): Scrivi un programma che calcoli l'IMC di una persona utilizzando una funzione. L'IMC si calcola come peso (kg) diviso per il quadrato dell'altezza (m^2).
9. Giochi d'azzardo: Crea un gioco d'azzardo in cui il giocatore lancia due dadi. Scrivi una funzione che calcoli la somma dei due dadi e determini se il giocatore ha vinto o perso.
10. Elabora una Lista di Numeri: Chiedi all'utente di inserire una serie di numeri interi e memorizzali in un array. Scrivi una funzione che calcoli la media, la somma e il massimo tra questi numeri.