Funzioni C++

Funzione

Una funzione è un blocco di codice riutilizzabile che esegue un compito specifico. Le funzioni consentono di organizzare il codice in modo più leggibile e modulare, facilitando la manutenzione e il riutilizzo.

Dichiarazione di una funzione

Prima di utilizzare una funzione, è necessario dichiararla. La dichiarazione di una funzione specifica il tipo di valore restituito, il nome della funzione e i tipi dei parametri.

```
tipo_di_ritorno nome_funzione(parametri);
```

Ad esempio, per dichiarare una funzione chiamata `somma` che accetta due interi come parametri e restituisce un intero:

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

Definizione di una funzione

Dopo aver dichiarato una funzione, è necessario definirla, ovvero fornire il codice che essa esegue. La definizione di una funzione include il corpo della funzione racchiuso tra parentesi graffe `{}`.

```
tipo_di_ritorno nome_funzione(parametri) {
    // codice della funzione
    return valore;
}
```

Ad esempio, ecco la definizione della funzione `somma`:

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

Chiamata di una funzione

Per utilizzare una funzione, è necessario chiamarla nel codice. Una chiamata a una funzione può essere effettuata passando gli argomenti appropriati tra parentesi.

```
nome_funzione(argomenti);
```

Ad esempio, per chiamare la funzione `somma` e assegnare il risultato a una variabile:

```
int x = 3, y = 4;
int risultato = somma(x, y);
```

Passaggio per valore

Quando un parametro viene passato per valore a una funzione, viene creata una copia locale del valore all'interno della funzione. Eventuali modifiche apportate al parametro all'interno della funzione non influiscono sulla variabile originale nella funzione chiamante.

```
void incrementa(int x) {
    x++; // Modifica la copia locale di x
}

int main() {
    int y = 5;
    incrementa(y); // Passa una copia di y alla funzione
    cout << y; // Stampa 5 (y non è stato modificato)
    return 0;
}
```

Il passaggio per valore è utile quando non si desidera modificare il valore originale della variabile nella funzione chiamante.

Passaggio per riferimento

Quando un parametro viene passato per riferimento a una funzione, viene passato l'indirizzo di memoria della variabile stessa. Eventuali modifiche apportate al parametro all'interno della funzione influiscono direttamente sulla variabile originale nella funzione chiamante.

```
void incrementa(int &x) {
    x++; // Modifica direttamente la variabile originale
}

int main() {
    int y = 5;
    incrementa(y); // Passa l'indirizzo di memoria di y alla funzione
    cout << y; // Stampa 6 (y è stato modificato)
    return 0;
}
```

Il passaggio per riferimento è utile quando si desidera modificare il valore originale della variabile nella funzione chiamante.

Funzioni con valori di ritorno

Una funzione può restituire un valore utilizzando l'istruzione `return`. Il tipo di valore restituito deve corrispondere al tipo di ritorno dichiarato nella firma della funzione.

```
int somma(int a, int b) {
    int risultato = a + b;
    return risultato; // Restituisce il valore di risultato
}

int main() {
    int x = 3, y = 4;
    int risultato = somma(x, y); // Assegna il valore restituito a
risultato
    cout << risultato; // Stampa 7
    return 0;
}
```

Funzioni senza valori di ritorno (procedure)

Una funzione che non restituisce alcun valore è chiamata procedura. In questo caso, il tipo di ritorno nella dichiarazione della funzione è `void`.

```
void stampaMessaggio() {
    cout << "Ciao, mondo!" << endl;
}
```

```
int main() {
    stampaMessaggio(); // Chiama la procedura per stampare il messaggio
    return 0;
}
```

Passaggio di array a funzioni

Gli array possono essere passati a funzioni sia per valore che per riferimento. Quando un array viene passato per valore, viene effettivamente passata una copia dell'indirizzo di memoria del primo elemento dell'array.

```
void stampaArray(int arr[], int dimensione) {
    for (int i = 0; i < dimensione; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int numeri[] = {1, 2, 3, 4, 5};
    int dimensione = sizeof(numeri) / sizeof(numeri[0]);
    stampaArray(numeri, dimensione); // Passa l'array e la sua dimensione
    return 0;
}
```

Tuttavia, è più comune passare gli array per riferimento utilizzando i puntatori.

```
void modificaArray(int *arr, int dimensione) {
    for (int i = 0; i < dimensione; i++) {
        arr[i] *= 2; // Modifica direttamente gli elementi dell'array
originale
    }
}

int main() {
    int numeri[] = {1, 2, 3, 4, 5};
    int dimensione = sizeof(numeri) / sizeof(numeri[0]);
    modificaArray(numeri, dimensione); // Passa l'indirizzo del primo
elemento dell'array
    // Stampa l'array modificato
    for (int i = 0; i < dimensione; i++) {
        cout << numeri[i] << " ";
```

```
    }
    cout << endl;
    return 0;
}
```

Questi appunti coprono i concetti fondamentali delle funzioni in C++, inclusi la dichiarazione, la definizione, la chiamata, il passaggio per valore e per riferimento, le funzioni con e senza valori di ritorno, e il passaggio di array a funzioni. Spero che questi appunti siano utili per preparare il ragazzo delle superiori in Informatica su questi argomenti.