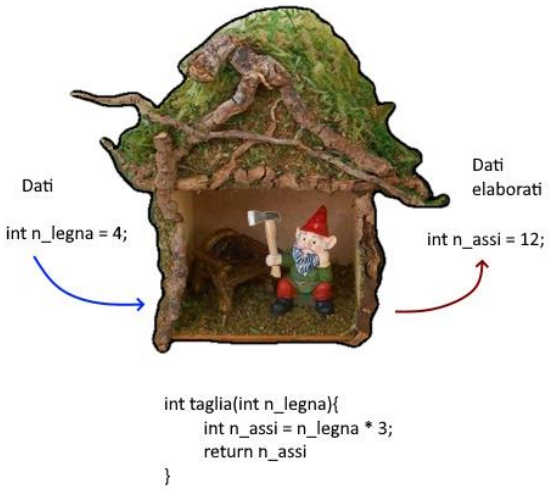
Funzioni

In C++, le funzioni sono degli insiemi di istruzioni che eseguono compiti specifici, le funzioni permettono di dividere il programma in parti più piccole, con responsabilità precise, che possono essere composte e riutilizzate all'occorrenza.

Definizione

Una funzione è una sequenza di istruzioni che può prendere dei valori in input, eseguire un compito specifico ed eventualmente ritornare un valore in output.

Possiamo vedere le funzioni come delle scatole che contengono un piccolo gnomo: i dati (la legna) entrano nella funzione taglia, vengono processati ed escono dalla funzione in forma di dati elaborati (assi).



Uno gnomo armato di ascia

Sintassi

Per utilizzare le funzioni nel nostro programma per prima cosa dobbiamo dichiararne la struttura della funzione al di fuori della funzione principale main() e successivamente richiamarla all'interno del nostro programma.

La sintassi è la seguente:

```
#include <iostream>

 //Dichiaro la struttura della funzione

tipo_ritorno  identificatoreDellaFunzione( parametri ){

        return valore;

}


int main(){

        identificatoreDellaFunzione( parametri );     //Utilizzo la funzione all'interno del programma

}
```

In modo più specifico abbiamo:

- tipo_ritorno: è il tipo di dato riportato dalla funzione alla sezione di programma che l'ha richiamata, oltre ai tipi principali (int, float, bool, void) possono essere riportati anche puntatori, riferimenti e classi(Programmazione ad oggetti).
  In particolare, il tipo **void** permette di non riportare alcun valore.
- identificatoreDellaFunzione: nome che identifica la funzione all'interno del programma
- parametri: parametri iniziali passati alla funzione, normalmente il passaggio dei parametri avviene per copia.
- valore: risultato delle operazioni eseguite all'interno della funzione, il tipo di dati deve essere uguale al tipo_ritorno prima definito.

Un esempio più concreto è il seguente:

```
#include <iostream>
int incremento(int valore_iniziale, int valore_incremento){
        int somma = valore_iniziale + valore_incremento;
        return somma;
}
int main(){
        int n = 0;
        n = incremento( n, 1 );
        std::cout << "il valore della variabile n e' " << n << '\n';
}
```

Questa funzione prende in input un valore_iniziale (in questo caso la variabile n), lo incrementa del valore specificato dal parametro valore_incremento (in questo caso 1) e riporta la somma in output.

Risorse utili:

https://www.youtube.com/watch?v=IsB44VZuHyQ

Function overloading

Nel precedente capitolo abbiamo visto come dichiarare ed utilizzare una funzione ed abbiamo visto come passare dei parametri di un tipo specifico, elaborarli ed emettere il risultato in output, ma come ci comportiamo se nel caso volessimo utilizzare la stessa funzione per tipi differenti?

Per spiegare il concetto di function overloading prendiamo in esempio la seguente funzione:

```
#include <iostream>
int sommaInteri (int valore1, int valore2){
        return valore1 + valore2;
}
int main(){
        int v1 = 5;
        int v2 = 6;
        std::cout << sommaInteri (v1, v2) << '\n';          //Visualizzo il risultato della funzione sommaInteri() in output
}
```

Questa funzione ci permette di sommare due **valori interi** ed emettere il risultato in output, mettiamo ora di voler eseguire la stessa operazione su due valori decimali, possiamo procedere nel seguente modo:

```
#include <iostream>
int sommaInteri(int valore1, int valore2){
	return valore1 + valore2;
}
double sommaDecimali(double valore1, double valore2){
	return valore1 + valore2;
}

int main(){
	int n1 = 5;
	int n2 = 6;
	double d1 = 3.2;
	double d2 = 2.3;
	std::cout << sommaInteri(v1, v2) << '\n';	//Visualizzo il risultato della funzione sommaInteri() in output
	std::cout << sommaDecimali(d1, d2) << '\n';  //Visualizzo il risultato della funzione sommaDecimali() in output
}
```

In questo modo abbiamo creato due funzioni che eseguono esattamente la stessa operazione ma siamo costretti ad utilizzare due nomi diversi per richiamarle creando così confusione ed è qui che entra in gioco l'argomento di questo capitolo: il function overloading del C++ ci permette di dichiarare due funzioni con lo stesso identico nome e parametri differenti, richiamando semplicemente un'unica funzione e semplificandone l'utilizzo.

```
#include <iostream>
int somma(int valore1, int valore2){
	return valore1 + valore2;
}
double somma(double valore1, double valore2){
	return valore1 + valore2;
}

int main(){
	int n1 = 5;
	int n2 = 6;
	double d1 = 3.2;
	double d2 = 2.3;
	std::cout << somma(v1, v2) << '\n';	//Visualizzo il risultato della funzione somma() con parametri interi in output
	std::cout << somma(d1, d2) << '\n';	//Visualizzo il risultato della funzione somma() con parametri decimali in output
}
```

Attenzione: il function overloading ci permette di dichiarare funzioni con parametri differenti sia nel tipo che nel numero.