Le regole di visibilità

Docente
Mario Perna
prof.perna.mario@darzo.net

A.S.
2025/2026
Materia
Informatica

Introduzione

Fino ad ora abbiamo visto come suddividere il nostro programma in sottoprogrammi più semplici che prendono il nome di funzioni o procedure.

Questo meccanismo ci permette di:

- Scrivere programmi più strutturati
- Riutilizzare porzioni di codice
- E tante altre belle cose…

Introduzione

Soffermiamoci per un attimo sul seguente esempio:

```cpp
#include <iostream>
using namespace std;

int s = 50;

int somma(int a, int b){
    s = 0;
    s = a + b;
    return s;
}

int main(){
    int n1 = 5;
    int n2 = 3;
    cout << "La variabile s vale " << s << endl;
    cout << "La somma di " << n1 << " e " << n2 << " vale " << somma(n1, n2) << endl;
    cout << "La variabile s vale " << s << endl;
    return 0;
}
```

```
La variabile s vale 50
La somma di 5 e 3 vale 8
La variabile s vale 8
```

Notate come la variabile s(esterna alla funzione) è stata modificata. Come mai?

Le variabili globali

sè una variabile **globale**, cioè una variabile visibile ovunque nel programma. E quindi modificabile da qualsiasi punto del programma.

```cpp
#include <iostream>
using namespace std;

int s = 50;

int somma(int a, int b){
    s = 0;
    s = a + b;
    return s;
}

int main(){
    int n1 = 5;
    int n2 = 3;
    cout << "La variabile s vale " << s << endl;
    cout << "La somma di " << n1 << " e " << n2 << " vale " << somma(n1, n2) << endl;
    cout << "La variabile s vale " << s << endl;
    return 0;
}
```

Una variabile globale è accessibile ovunque!

Le regole di visibilità

A tempo di **compilazione** tutti gli **identificatori** di variabili, costanti, funzioni, procedure, etc. devono essere riconosciuti.

Questi **identificatori** possono avere tre regole di visibilità in base a **DOVE** la loro dichiarazione viene posta:

- **Blocco (locale):** quando l'identificatore è dichiarato all'interno di un blocco di istruzioni
- **Funzione o procedura (locale)**: quando l'identificatore è dichiarato all'interno di una funzione o procedura
- **Programma (globale)**: quando l'identificatore è dichiarato fuori da qualsiasi blocco e funzione o procedura

Le regole di visibilità: esempio

```cpp
#include <iostream>
using namespace std;

int s = 50;

int somma(int a, int b){
    s = 0;
    s = a + b;
    return s;
}

int main(){
    int n1 = 5;
    int n2 = 3;
    cout << "La variabile s vale " << s << endl;
    cout << "La somma di " << n1 << " e " << n2 << " vale " << somma(n1, n2) << endl;
    cout << "La variabile s vale " << s << endl;
    return 0;
}
```

sè una variabile con visibilità globale e quindi leggibile e modificabile ovunque

ae bsono variabili con visibilità funzione e quindi leggibili e modificabili sono all'interno della funzione somma

n1 e n2 sono variabili con visibilità di blocco e quindi leggibili e modificabili solo all'interno del blocco main

N.B. queste regole non valgono solo per le variabili, ma anche per costanti e funzioni o procedure, ad esempio la funzione somma ha visibilità globale!

Le regole di visibilità: sovrapposizione

E' possibile che un **medesimo identificatore** possa essere **sostituito** da un'altra regola di visibilità. In generale vince sempre la regola più specifica nel seguente ordine:

1. Blocco
2. Funzione
3. Programma

Dove la prima vince su tutte le altre in caso di più identificatori con il medesimo nome.

```cpp
#include <iostream>
using namespace std;

int s = 50;

int somma(int a, int b){
    int s = 0;
    s = a + b;
    return s;
}

int main(){
    int n1 = 5;
    int n2 = 3;
    cout << "La variabile s vale " << s << endl;
    cout << "La somma di " << n1 << " e " << n2 << " vale " << somma(n1, n2) << endl;
    cout << "La variabile s vale " << s << endl;
    return 0;
}
```

s viene dichiarato sia come variabile globale e locale alla funzione somma!

```
La variabile s vale 50
La somma di 5 e 3 vale 8
La variabile s vale 50
```

Le regole di visibilità: sovrapposizione

Tutto questo casino per capire che cosa?!?

Cercate di limitare le variabili globali a quelle strettamente necessarie e preferire l’utilizzo di quelle locali.

Informazioni sull’utilizzo dei materiali didattici

Queste slides si basano su materiali originariamente elaborati dal **Prof. Andrea Melioli**, opportunamente modificati e integrati secondo specifiche esigenze riguardanti la programmazione disciplinare.

L’autore autorizza al **prof. Mario Perna** l’utilizzo, la modifica, la pubblicazione e qualsiasi altra forma di operazione sui materiali a scopo didattico e formativo.

L’uso o la diffusione di questi materiali è **vietato** senza il preventivo contatto con il proprietario del documento, Prof. Mario Perna (prof.mario.perna@darzo.net).