16. Visibilità di una variabile

Corso di Algoritmi e Linguaggi di Programmazione Python/C

Angelo Cardellicchio
angelo.cardellicchio@stiima.cnr.it

Outline

- Ambito di una variabile
- Tempo di vita di una variabile

Ambito di una variabile

- Supponiamo di avere un programma C che definisca due funzioni (oltre al `main`).
  - Entrambe le funzioni accettano come parametro in ingresso un intero che rappresenta il lato di un quadrato.
  - Chiamiamo la prima funzione `calcola_area_quadrato`, e la seconda `calcola_perimetro_quadrato`.
- All'interno della funzione `calcola_area_quadrato`, proviamo a stampare a schermo il valore dell'area e del perimetro del quadrato.
- Proviamo ad eseguire il programma.

Ambito di una variabile

```c
#include <stdio.h>

int calcola_area_quadrato(int lato) {
    int area = lato * lato;
    printf("Il valore dell'area è: %d", area);
    printf("Il valore del perimetro è: %d", perimetro);
    return area;
}

int calcola_perimetro_quadrato(int lato) {
    int perimetro = lato * 4;
    return perimetro;
}

int main() {
    int lato = 5;
    int area = calcola_area_quadrato(lato);
    int perimetro = calcola_perimetro_quadrato(lato);
    return 0;
}
```

- Il programma ci darà un errore!

Ambito di una variabile

```c
#include <stdio.h>
#include <math.h>

int calcola_area_quadrato(int lato) {
    int area = lato * lato;
    return area;
}

int calcola_perimetro_quadrato(int lato) {
    int perimetro = lato * 4;
    return perimetro;
}

int main() {
    int lato = 5;
    int area = calcola_area_quadrato(lato);
    int perimetro = calcola_perimetro_quadrato(lato);
    printf("Il valore dell'area è: %d", area);
    printf("Il valore del perimetro è: %d", perimetro);
    return 0;
}
```

- La variabile `perimetro` non è **visibile** alla funzione `calcola_area_quadrato`.
  - *Proviamo invece a stampare a schermo questi valori dal* `main`.

Ambito di una variabile

```c
#include <stdio.h>
#include <math.h>

int lato = 5;

int calcola_area_quadrato() {
    int area = lato * lato;
    return area;
}

int calcola_perimetro_quadrato() {
    int perimetro = lato * 4;
    return perimetro;
}

int main() {
    int area = calcola_area_quadrato();
    int perimetro = calcola_perimetro_quadrato();
    printf("Il valore dell'area e': %d\n", area);
    printf("Il valore del perimetro e': %d\n", perimetro);
    return 0;
}
```

- Una variabile ha ambito **locale** quando è accessibile soltanto all'interno di una particolare porzione di codice.
- Una variabile ha ambito **globale** quando è accessibile da tutto il programma.
- In caso di concorrenza, viene data precedenza alla variabile **locale**: questo è spesso fonte di errori!

Tempo di vita di una variabile

- Supponiamo di avere un programma C che definisca una funzione `incrementa.`
- La funzione `incrementa` ha al suo interno una variabile locale di tipo intero chiamata `contatore` che viene incrementata di uno ogni volta che la funzione viene chiamata.
- Proviamo ad implementare la funzione, ed a chiamarla due volte nel `main`, stampando a schermo il risultato.

Tempo di vita di una variabile

```c
#include<stdio.h>

int incrementa() {
    int contatore = 0;
    contatore++;
    return contatore;
}

int main() {
    printf("Il valore del contatore è %d \n",  incrementa());
    printf("Il valore del contatore è %d \n", incrementa());
    return 0;
}
```

- A schermo verrà stampato due volte il valore 1.
- Questo è legato al fatto che la variabile locale contatore è **dinamica**.
- Una variabile è dinamica quando **cessa di esistere** una volta usciti dall'ambito in cui è dichiarata.
- Ciò non vale per le variabili **statiche**, contraddistinte dalla parola chiave **`static`.**

Domande?

42