campo d’azione (scope)

Sistemi Operativi

A.A. 2020-2021

Marco Tessarotto

campo d’azione (scope)

- Si parla di “scope” di variabili, di funzioni
- campo d’azione (scope): è la parte di programma in cui la variabile o la funzione può essere utilizzata

Variabili locali

- Le variabili dichiarate all'interno di una funzione o di un blocco di codice { } sono chiamate **variabili locali**.
- Possono essere utilizzate solo da istruzioni all'interno di tale funzione o blocco di codice.
- Le variabili locali non sono note e quindi utilizzabili da funzioni al di fuori della funzione o blocco dove sono dichiarate

Variabili locali

- Variabili locali: sono le variabili dichiarate all'interno di una funzione o di un blocco { … }
- il loro campo d'azione è la funzione o il blocco (precisamente: dal punto dove è dichiarata la variabile fino alla fine della funzione o del blocco)

```
int my_function(int param) {
   int result;   // dichiarazione: result è una variabile locale di my_function
   ….
   return result;
} // finisce la funzione: result non è più utilizzabile
```

Variabili locali

```
int my_function(int param) {
  int result;
  ….
  if (…) {  // inizio blocco
       int local_var; // questa variabile esiste da qui fino alla fine del blocco

       ….
    }  // fine blocco, local_var non esiste più

  return result;
}
```

Parametri formali di funzione

- I parametri formali di una funzione hanno come campo d'azione tutta la funzione (di cui sono parametri); sono da considerarsi come variabili locali

int my_function(int **param**) { // **param** esiste all'interno della funzione

…

} // fine funzione: **param** cessa di esistere

Variabili globali

- Le variabili globali sono definite al di fuori di ogni funzione, generalmente in cima al programma.
- Le variabili globali mantengono i loro valori per tutta la durata del programma e sono accessibili all'interno di una qualsiasi delle funzioni definite nel programma.

- Una variabile globale è accessibile da qualsiasi funzione. Una variabile globale è disponibile per l'uso in tutto il programma dopo la sua dichiarazione.

Variabili globali

```
// counter è definita fuori da ogni funzione
int counter = 0; // variabile globale usabile da qua fino alla file del file


int main() {
   …
   counter = counter + 1; // utilizzo la variabile globale da dentro una funzione
   …
}
```

Funzioni - dichiarazione

- Il campo d’azione (scope) di una funzione va dal punto in cui è dichiarata fino alla fine del file che viene compilato.

```
// dichiaro la funzione
void my_algorithm(int * arr); // utilizzabile da qua in poi, fino alla fine del file
…
int main() {
   …
   my_algorithm(my_arr);
   …
}
```

Funzioni - definizione

```
// definisco la funzione (vale anche come dichiarazione)
void my_algorithm(int * arr) { // utilizzabile da qua in poi, fino alla fine del file
...
}

int main() {
   …
   my_algorithm(my_arr);
   …
}
```

Funzioni – definizione e dichiarazione

- File utils.c

// **definisco** la funzione (vale anche come dichiarazione nel file)

void **my_algorithm**(int * arr) { // utilizzabile da qua in poi, fino alla fine del file

…

}

- File main.c

// **dichiaro** la funzione
// che è definita altrove (in utils.c)
void **my_algorithm**(int * arr);

int main() {
  …
  my_algorithm(my_arr)**;**
  …

utils.c e main.c fanno parte di un unico programma

extern

- Se si deve fare riferimento a una variabile esterna prima che sia definita, oppure se è definita in un file sorgente diverso da quello in cui viene utilizzata, è obbligatoria una dichiarazione esterna (extern)

- È importante distinguere tra la dichiarazione di una variabile esterna e la sua definizione. Una dichiarazione annuncia le proprietà di una variabile (principalmente il suo tipo); una definizione causa anche l'allocazione della memoria per la variabile.

extern e variabili globali

- File utils.c

```
// definisco la variabile
// e viene allocata la memoria
int error_code; // var. globale
// vale anche come dichiarazione nel file

int my_func() {
   …
   error_code = 0;
   …
```

- File main.c

```
// dichiaro la variabile
// che è definita altrove (in utils.c)
extern int error_code; // var. globale

int main() {
   …
   printf(«%d», error_code);
   …
```

utils.c e main.c fanno parte di un unico programma

Precedenza variabili locali su variabili globali

- Un programma può avere lo stesso nome per le variabili locali e globali, ma il valore della variabile locale all'interno di una funzione avrà la precedenza.

```
#include <stdio.h>

/* global variable declaration */
int g = 20;

int main () {
  /* local variable declaration */
  int g = 10;
  printf ("g = %d\n",  g);  // output:    g = 10
  return 0;
}
```

Inizializzazione di variabili

- Le **variabili locali** NON sono inizializzate automaticamente, vanno esplicitati i valori di inizializzazione
- Le **variabili globali** sono inizializzate automaticamente, ad esempio:
- int → 0
- char → ‘\0’
- double → 0.0
- puntatori → NULL