Capitolo 5 - Funzioni

Introduzione

- Divide and conquer
  - Costruire un programma da pezzi più piccoli o da singole componenti
    - Questi pezzi più piccoli sono chiamati moduli
  - Ogni singolo pezzo è più facilmente gestibile rispetto al programma principale

Moduli in C

- Functions
  - Moduli in C
  - Il programma combina funzioni user-defined con funzioni di libreria
    - La libreria standard del C ha una grande varietà di funzioni
- Chiamata di funzioni
  - Invocazione
    - Si fornisce il nome della funzione e gli argomenti (dati)
    - La funzione esegue le operazioni opportune
    - La funzione restituisce un risultato

Funzioni

- Funzioni
  - Modularizzano un programma
  - Tutte le variabili definite all'interno delle funzioni sono locali
    - Conosciute solo nella funzione in cui sono state definite
  - Parametri
    - Permettono la comunicazione tra funzioni
    - Sono anch'essi variabili locali
- Benefici delle funzioni
  - Divide and conquer
  - Software reusability
    - Uso di funzioni esistenti come building blocks per nuovi programmi
    - Astrazione – nascondono i dettagli interni
  - Evitano la ripetizione del codice

Definizione di funzioni

- Formato per la definizione di funzioni

  return-value-type function-name(parameter-list)
  {
  declarations and statements
  }

  – Function-name: ogni identificatore valido
  – Return-value-type: tipo del risultato (default `int`)
    - `void` – indica che la funzione non restituisce nulla
  – Parameter-list: lista separata da virgole che permette di dichiarare i parametri
    - Specificare il tipo per ogni parametro

Definizione di funzioni

- Formato per la definizione di funzioni (continua)

  return-value-type function-name(parameter-list)
  {
  declarations and statements
  }

  – Declarations and statements: corpo della funzione (block)
    - Le variabili possono essere definite all'interno del blocco (possono essere innestate)
    - Le funzioni non possono essere definite all'interno di altre funzioni (non è permesso l'annidamento)
  – Ritorno del controllo
    - Se non deve essere restituito nulla all'esterno
      – `return;`
      – oppure si raggiunge la parentesi di chiusura }
    - Se qualcosa deve essere restituito
      – `return` *expression*`;`

```c
/* Fig. 5.3: fig05_03.c
   Creating and using a programmer-defined function */
#include <stdio.h>

int square( int y );   /* function prototype */

/* function main begins program execution */
int main()
{
   int x; /* counter */

   /* loop 10 times and calculate and output square of x each time */
   for ( x = 1; x <= 10; x++ ) {
      printf( "%d ", square( x ) ); /* function call */
   } /* end for */

   printf( "\n" );

   return 0; /* indicates successful termination */

} /* end main */
```

Outline

fig05_03.c (Part 1 of 2)

Calcola i quadrati degli interi compresi tra 1 e 10


```c
/* square function definition returns square of an integer */
int square( int y ) /* y is a copy of argument to function */
{
   return y * y; /* returns square of y as an int */

} /* end function square */
```

```
1  4  9  16  25  36  49  64  81  100
```

Outline

fig05_03.c (Part 2 of 2)

Program Output

```c
/* Fig. 5.4: fig05_04.c
   Finding the maximum of three integers */
#include <stdio.h>

int maximum( int x, int y, int z ); /* function prototype */

/* function main begins program execution */
int main()
{
   int number1; /* first integer */
   int number2; /* second integer */
   int number3; /* third integer */

   printf( "Enter three integers: " );
   scanf( "%d%d%d", &number1, &number2, &number3 );

   /* number1, number2 and number3 are arguments
      to the maximum function call */
   printf( "Maximum is: %d\n", maximum( number1, number2, number3 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

Outline

fig05_04.c (Part 1 of 2)


```c
/* Function maximum definition */
/* x, y and z are parameters */
int maximum( int x, int y, int z )
{
   int max = x;      /* assume x is largest */

   if ( y > max ) { /* if y is larger than max, assign y to max */
      max = y;
   } /* end if */

   if ( z > max ) { /* if z is larger than max, assign z to max */
      max = z;
   } /* end if */

   return max;       /* max is largest value */

} /* end function maximum */
```

```
Enter three integers: 22 85 17
Maximum is: 85
Enter three integers: 85 22 17
Maximum is: 85
Enter three integers: 22 17 85
Maximum is: 85
```

Outline

fig05_04.c (Part 2 of 2)

Program Output

Prototipi di funzione

- Prototipi di funzione
  - Tipo di ritorno (default `int`)
  - Nome della funzione
  - Parametri di input
  - Usati per validare le funzioni
  - I prototipi sono necessari solo quando la definizione delle funzioni avviene dopo l'uso nel programma
    - Altrimenti, il compilatore creerà un suo prototipo utilizzando la prima occorrenza della funzione
  - La funzione con il seguente prototipo

```
int maximum( int x, int y, int z );
```

    - Ha tre parametri di input interi
    - Restituisce un `int`

Chiamata di funzioni per valore e referenza

- Chiamata per valore
  - Crea una copia degli argomenti passati ad una funzione
  - I cambiamenti nella funzione non hanno effetto sui parametri originali
  - Usare quando la funzione non ha bisogno di modificare gli argomenti
    - Evita modifiche accidentali
- Chiamata per referenza o riferimento o indirizzo
  - Passa gli "argomenti originali"
    - L'indirizzo della locazione di memoria contenente gli argomenti
  - I cambiamenti nella funzione hanno effetto sui parametri originali
- Per ora focalizziamoci solo sulle chiamate per valore

Regole di Visibilità (Scope Rules)

- Visibilità nel File
  - Gli identificatori definiti fuori dalla funzione sono conosciuti in tutte le funzioni
  - Usato per le variabili globali, la definizione di funzioni, e i prototipi di funzione
- Visibilità nella Funzione
  - Possono essere referenziati solo all'interno del corpo di una funzione

Regole di Visibilità (Scope Rules)

- Visibilità nel Blocco
  - Identificatori dichiarati all'interno di un blocco
    - La visibilità nel blocco comincia dalla definizione e termina con la parentesi destra di chiusura del blocco
  - Usato per variabili, parametri di funzioni
  - Blocchi esterni "nascosti" dai blocchi interni se esiste una variabile con lo stesso nome nel blocco interno
- Visibilità nel prototipo di Funzione
  - Usato per gli identificatori nella lista dei parametri