Aula V

Dip. di Matematica “G. Castelnuovo”

Univ. di Roma “La Sapienza”

Le Funzioni e la Ricorsione

Igor Melatti

Slides disponibili (assieme ad altro materiale) in:

`http://www.dsi.uniroma1.it/~melatti/programmazione1.2007.2008.html`

- Da adesso in poi, basta esercizi che chiedono di “scrivere un programma”
- Si chiederà di scrivere una o più *funzioni*
- Inoltre, non si chiederà (quasi) più di leggere un numero da tastiera o scrivere un numero su schermo
  - la lettura da tastiera non servirà (quasi) più perché basta dare i numeri come *parametri* della funzione
  - la scrittura del risultato su schermo non servirà (quasi) più perché basta dare il numero come *valore di ritorno* (o *output*) della funzione

Scrivere un programma che legge da tastiera un numero non negativo x e stampa su schermo la somma delle cifre decimali di x

```
#include <stdio.h>

int main()
{
 unsigned x, sum = 0;
 printf("Immettere il numero: ");
 scanf("%u", &x);
 printf("La somma delle cifre di %u e': ", x);
 while (x != 0) {
  sum += x%10;
  x /= 10;
 }
 printf("%u\n", sum);
}
```

Scrivere una funzione che prende in input un numero non negativo x e restituisce la somma delle cifre decimali di x

```
unsigned SommaCifreDec(unsigned n)
{
 unsigned sum = 0;
 while (n != 0) {
  sum += n%10;
  n /= 10;
 }
 return sum;
}
```

- Per completare il programma (ma solo per vedere qui in laboratorio se “gira”):

```
#include <stdio.h>

int main()
{
 unsigned x;
 printf("Immettere il numero: ");
 scanf("%u", &x);
 printf("La somma delle cifre di %u e': %u\n", x, SommaCifreDec(x));
}
```

- Riscrivere usando le funzioni come nel caso precedente tutti gli esercizi di giovedì scorso (modulo vacanze)

Le funzioni ricorsive: ripasso

- Riscrivere in maniera ricorsiva la funzione `SommaCifreDec`
- Vediamo come si ragiona (repetita iuvant)

  Parametro ricorsivo in questo caso è uno solo (chiamiamolo n), ed è il numero che contiene le cifre *ancora da sommare*

  – detto in questa maniera, è chiaro che all'inizio n è il numero "originario" (tutte le cifre devono ancora essere sommate), e alla fine n ha una sola cifra

  Caso base per quanto detto sopra, si ha quando n ha una sola cifra, ovvero quando $n \leq 9$

  – qui la somma delle cifre di n e proprio n, quindi si *restituisce* (o *ritorna*) n

- Vediamo come si ragiona (continua)

  Caso ricorsivo si ha quando n ha più cifre, ovvero quando $n > 9$ (quindi è esattamente la negazione del caso base)

  – qui la somma delle cifre di n può essere vista come la somma dell'ultima cifra di n con la somma delle altre cifre di n

  – matematicamente: supponiamo che n si scriva in decimale come $n_k n_{k-1} \ldots n_1 n_0$

  ∗ esempio: se n =349, allora $k = 2$ e n_2 =3, n_1 =4, n_0 =9

  – allora SommaCifreDec$(n) = n_0$+SommaCifreDec$(n_k n_{k-1} \ldots n_1)$

  – infatti, SommaCifreDec$(n) = \sum_{i=0}^{k} n_i =$

$$n_0 + n_1 + \ldots + n_{k-1} + n_k = n_0 + (n_1 + \ldots + n_{k-1} + n_k) =$$

$$n_0 + \sum_{i=1}^{k} n_i = n_0 + \text{SommaCifreDec}(n_k n_{k-1} \ldots n_1)$$

Le funzioni ricorsive: ripasso

Caso	Espressione da restituire
Base ($n \leq 9$)	n
Ricorsivo ($n > 9$)	$n_0 + \text{SommaCifreDec}(n_k n_{k-1} \ldots n_1)$

- Da qui all'implementazione basta notare 2 cose:
  1. $n_0 = n\%10$
  2. $n_k n_{k-1} \ldots n_1 = n/10$

Implementazione finale

```
unsigned SommaCifreDec(unsigned n)
{
 /* pre: n >= 0 */
 /* post: SommaCifreDec(n) = SommaCifreDec(n_k n_{k-1} ... n_0) = sum_{i=0}^{k} n_i */
 if (n <= 9)
  return n;
 else /* questo else non e' necessario, perche'? */
  return n%10 + SommaCifreDec(n/10);
}
```

Verifica

- Vi si potrebbe chiedere di “verificare” che la funzione termina sempre
- E anche che calcola il valore esatto
- Quest’ultima cosa è vera per l’analisi matematica fatta sopra
- La terminazione è data dal fatto che ogni volta a SommaCifreDec gli si passa lo stesso numero privato di una cifra
- Prima o poi, a forza di togliere, si arriverà ad una cifra sola
- Ma se si arriva ad una cifra sola allora viene eseguito il caso base

Altro esempio

- Un numero naturale è divisibile per 3 se e solo se la somma delle sue cifre decimali è divisibile per 3. Scrivere una funzione ricorsiva per decidere se un naturale dato è divisibile per 3.
  - N.B.: è consentito l'uso dell'operatore modulo % *solo* per dividere per 10.
- La funzione che scriveremo (chiamiamola Div3) prenderà in input un naturale n e ritornerà 1 se n è divisibile per 3 e 0 altrimenti

```
char Div3(unsigned n)
/* perche' char? */
{
 /* pre: n >= 0 */
 /* post: Div3(n) = 1 se e solo se esiste k ∈ ℕ tale che 3k = n */
 return n%3 == 0;
}
```

Altro esempio

- Proviamo a ragionare nella stessa maniera di sopra

  Parametro ricorsivo uno solo (chiamiamolo n), è il numero di cui occorre decidere se è divisibile per 3 o no

  – ancora una volta, all'inizio n è il numero "originario", e alla fine n ha una sola cifra, ma questa volta si procederà diversamente

  Caso base si ha quando n ha una sola cifra, ovvero quando $n \leq 9$

  – qui n è divisibile per 3 se e solo se $n \in \{0, 3, 6, 9\}$, quindi si *restituisce* (o *ritorna*) 1 se e solo se $n \in \{0, 3, 6, 9\}$

  Caso ricorsivo si ha quando n ha più cifre, ovvero quando $n > 9$ (di nuovo, è la negazione del caso base)

  – qui n è divisibile per 3 se e solo se la somma delle cifre di n è divisibile per 3

  – matematicamente, si tratta della proprietà fornita con il testo dell'esercizio

Altro esempio

Caso	Espressione da restituire
Base ($n \leq 9$)	$n \in \{0, 3, 6, 9\}$
Ricorsivo ($n > 9$)	Div3(SommaCifreDec(n))

- Da qui all'implementazione basta notare 2 cose:
  1. $n \in \{0, 3, 6, 9\} = (n = 0 \vee n = 3 \vee n = 6 \vee n = 9)$
  2. SommaCifreDec è già implementata

Implementazione finale

```
char Div3(unsigned n)
{
 /* pre: n >= 0 */
 /* post: Div3(n) = 1 se e solo se esiste k ∈ ℕ tale che 3k = n */
 if (n <= 9)
  return n == 0 || n == 3 || n == 6 || n == 9;
 else /* di nuovo, questo else non e' necessario */
  return Div3(SommaCifreDec(n));
}
```

- La funzione Div3 calcola il giusto valore perché si basa sulla proprietà data
- La terminazione è data dal fatto che ogni volta SommaCifreDec ritorna un valore *strettamente* più piccolo del suo argomento
  - matematicamente:
    $n = 10^k n_k + \ldots + 10^0 n_0 > n_k + \ldots + n_0 =$SommaCifreDec($n$),
    quindi $n >$SommaCifreDec(n)
  - N.B.: quanto detto sopra vale solo per numeri composti da almeno 2 cifre decimali, che in effetti è proprio quello che serve (se n ha una sola cifra decimale, la chiamata ricorsiva non viene fatta)
- Prima o poi, a forza di di passare valori sempre più piccoli, si arriverà ad una cifra sola
- Ma se si arriva ad una cifra sola allora viene eseguito il caso base

Tutti gli esercizi: funzioni iterative

- Scrivere una funzione che prende in input un numero non negativo x e stampa su schermo a quanti giorni, ore, minuti e secondi corrispondono x secondi
- Scrivere una procedura (cioè una funzione che ritorna `void`) che prende in input 4 numeri non negativi d, h, m, s e dice quanti secondi ci sono in d giorni, h ore, m minuti e s secondi
- Scrivere una procedura che stampi a video quanti bit e quanti bytes occorrano per i tipi predefiniti del C: int, unsigned, long, float, char...
- Scrivere 3 funzioni (iterative) che prendano tutte in input 2 interi x e y e poi restituiscano:
  1. il prodotto $x \times y$, ma *senza usare* l'operatore *
  2. la potenza x^y, ma *senza usare* la funzione `pow`
  3. il resto e il quoziente di x/y, ma *senza usare* né l'operatore /,

né l'operatore %

- Riscrivere le funzioni 1 e 2 in maniera che uno tra x e y sia reale (nel caso 2, deve essere per forza la base)
- Scrivere una funzione che prende in input un intero x e restituisce la somma delle cifre (decimali) di x
- Riscrivere la funzione di cui sopra facendo sì che prenda in input, oltre ad x, anche un intero b, e che poi consideri le cifre di x rappresentato in base b

Tutti gli esercizi: funzioni iterative

- Scrivere una funzione iterativa che dato in input un naturale x calcoli la radice digitale di x. La radice digitale di un numero si ottiene sommando le sue cifre (decimali) fino a che non si riducono ad un numero di una sola cifra (decimale)
- Riscrivere la funzione di cui sopra facendo sì che prenda in input, oltre ad x, anche un intero b, e che poi consideri le cifre di x rappresentato in base b
- Un numero naturale è divisibile per 3 se la somma delle sue cifre decimali è divisibile per 3. Scrivere una funzione iterativa per decidere se un naturale dato è divisibile per 3. È consentito l'uso dell'operatore modulo % *solo* per dividere per 10.

Tutti gli esercizi: funzioni iterative

- Dato un naturale n, siano si(n) e sp(n) rispettivamente la somma delle cifre di ordine pari e dispari in n. (dove la 0-esima cifra è quella di minor peso). n è divisibile per 11 se il valore assoluto della differenza tra sp(n) e si(n) cioè abs(sp(n)-si(n))) è divisibile per 11. Scrivere una funzione iterativa per decidere se un numero naturale dato è divisibile per 11. È consentito l'uso dell'operatore modulo % *solo* per dividere per 10.

Tutti gli esercizi: funzioni ricorsive

- Scrivere 3 funzioni ricorsive che prendano tutte in input 2 interi x e y e poi restituiscano:
  1. il prodotto $x \times y$, ma *senza usare* l'operatore *
  2. la potenza x^y, ma *senza usare* la funzione pow
  3. il resto e il quoziente di x/y, ma *senza usare* né l'operatore /, né l'operatore %
- Riscrivere le funzioni 1 e 2 in maniera che uno tra x e y sia reale (nel caso 2, deve essere per forza la base)
- Scrivere una funzione ricorsiva che prende in input un intero x e restituisce la somma delle cifre (decimali) di x
- Riscrivere la funzione di cui sopra facendo sì che prenda in input, oltre ad x, anche un intero b, e che poi consideri le cifre di x rappresentato in base b

Tutti gli esercizi: funzioni ricorsive

- Scrivere una funzione ricorsiva che dato in input un naturale x calcoli la radice digitale di x. La radice digitale di un numero si ottiene sommando le sue cifre (decimali) fino a che non si riducono ad un numero di una sola cifra (decimale)
- Riscrivere la funzione di cui sopra facendo sì che prenda in input, oltre ad x, anche un intero b, e che poi consideri le cifre di x rappresentato in base b
- Un numero naturale è divisibile per 3 se la somma delle sue cifre decimali è divisibile per 3. Scrivere una funzione ricorsiva per decidere se un naturale dato è divisibile per 3. È consentito l'uso dell'operatore modulo % *solo* per dividere per 10.

Tutti gli esercizi: funzioni ricorsive

- (***Difficile***) Dato un naturale n, siano si(n) e sp(n) rispettivamente la somma delle cifre di ordine pari e dispari in n. (dove la 0-esima cifra è quella di minor peso). n è divisibile per 11 se il valore assoluto della differenza tra sp(n) e si(n) cioè abs(sp(n)-si(n))) è divisibile per 11. Scrivere una funzione ricorsiva per decidere se un numero naturale dato è divisibile per 11. È consentito l'uso dell'operatore modulo % *solo* per dividere per 10.

- Scrivere una funzione ricorsiva che, dato un intero n, restituisca l'n-esimo numero di Fibonacci, definito da:

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_n &= f_{n-1} + f_{n-2} \end{aligned}$$

- (***Difficile***) In una catena di venditori $v_1, \ldots, v_n$, ogni v_i compra da v_{i-1} al prezzo x e vende a v_{i+1} al prezzo px (con $p > 1$...). Il primo venditore vende a y. Scrivere una funzione ricorsiva che dica a quanto vende l'n-esimo venditore (v_n).