Aula V

Dip. di Matematica “G. Castelnuovo”

Univ. di Roma “La Sapienza”

Soluzioni esonero e array

Igor Melatti

Slides disponibili (assieme ad altro materiale) in:

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

Traccia A

- Siano dati due interi a, b $\geq$ 0. L'intero c ottenuto dal valore assoluto della differenza delle cifre dello stesso peso di a e b, si definisce differenza assoluta di a e b e si denota con c = diffass(a, b). Ad esempio:
  - se a = 5446 e b = 3438, diffass(a, b) = 2012;
  - se a = 23 e b = 45678, diffass(a, b) = 45655.
- Implementare in C la funzione diffass (sia iterativamente che ricorsivamente)

Traccia B

- Siano dati due interi (in notazione decimale) a, b $\geq$ 0. La distanza binaria di a da b, distbin(a, b) si definisce come il numero di posizioni, aventi valori differenti nella rappresentazione binaria di a e b. Ad esempio:
  - se a = 7 e b = 4, distbin(a, b) = 2, perché la rappresentazione binaria di 4 è 100 e quella di 7 è 111;
  - se a = 27 e b = 3, distbin(a, b) = 2 perché la rappresentazione binaria di 3 è 11 e quella di 27 è 11011.
- Implementare in C la funzione distbin (sia iterativamente che ricorsivamente)

Traccia C

- Siano dati un intero a $>$ 0 e due interi 0 $\leq$ b, c $<$ 10. L'intero d ottenuto sostituendo in a ogni cifra b con la cifra c è denotato come sost(a, b, c). Ad esempio,
  - se a = 5446, b = 6 e c = 4, sost(a, b, c) = 5444;
  - se a = 2343221, b = 2 e c = 1, sost(a, b, c) = 1343111.
- Implementare in C la funzione sost (sia iterativamente che ricorsivamente)

Traccia D

- Siano dati un intero a > 0 e due interi $0 \leq$ b, c < 10. L'intero d ottenuto sostituendo in a ogni cifra b con la cifra (c + b) mod 10 è denotato come sost_10(a, b, c). Ad esempio,
  - se a = 5446, b = 6 e c = 4, sost_10(a, b, c) = 5440;
  - se a = 7343771, b = 7 e c = 5, sost_10(a, b, c) = 23431221.
- Implementare in C la funzione sost_10 (sia iterativamente che ricorsivamente)

Traccia A (secondo problema):

La seguente funzione vorrebbe calcolare il quoziente di due interi:

```
unsigned quoziente(unsigned n, unsigned m)
{
 /* pre: m > 0, n >= 0 */
 /* post: quoziente(n,m) = k t.c. km <= n e (k + 1)m > n */
 unsigned i;
 for (i = 0; m*(i + 1) >= n; i++);
 return n;
}
```

Cosa restituisce invece questa funzione? Come occorre correggerla perché invece calcoli il quoziente? Giustificare la risposta.

Traccia B (secondo problema):

Descrivere l’esecuzione della seguente procedura ricorsiva.

```
void f1(unsigned n)
{
 /* pre: n >= 0 */
 if (n > 9) f(n/10);
 printf("%d", n%10);
}
```

Aiutatevi mostrando l’output per n = 345 e facendo vedere come evolve lo stack delle chiamate ad f.

Traccia C (secondo problema):

Si consideri la seguente funzione in C.

```
int f2(int n){
 /* Pre: n>0 */
 /* Post ??? */
 int k=2;
 while (k <= n && n%k != 0) k++;
 return k <= n;
}
```

Scrivere la post-condizione, ovvero descrivere cosa restituisce la funzione, e giustificare la risposta.

Traccia D (secondo problema):

Quale dei seguenti frammenti C puo' essere un prototipo di funzione, quale (l'inizio di) una definizione di funzione, quale una chiamata di funzione e quale risulterebbe sicuramente in un errore? Motivare le risposte:

```
unsigned ciao(int);

unsigned ciao(int)
{

unsigned ciao(float);
{

unsigned (int x, int);

ciao(4.56);

ciao(5; 4);

unsigned ciao(int; float);
```

Traccia A Formulazione matematica:

- supponiamo che a e b si scrivano in notazione decimale come $a = a_{n-1}a_{n-2}\ldots a_0, b = b_{m-1}b_{m-2}\ldots b_0$ e che $m \leq n$ (senza perdita di generalità)
  - ad esempio, se $a = 5446, b = 438$, allora $a_3 = 5, a_2 = 4, a_1 = 4, a_0 = 6, b_2 = 4, b_1 = 3, b_0 = 8, n = 4, m = 3$
- allora $c = c_{n-1}c_{n-2}\ldots c_0$ t.c. $c_i = |a_i - b_i|$ per ogni $0 \leq i < n$
- N.B.: $b_i = 0$ per $i \geq m$
  - nell'esempio di sopra, $c_3 = |a_3 - b_3| = |5 - 0| = 5, c_2 = |4 - 4| = 0, a_1 = |4 - 3| = 1, c_0 = |6 - 8| = 2$
  - quindi $c = 5012$ (da notare che $5012 \neq |5446 - 438| = 5008$)

Traccia B Formulazione matematica:

- supponiamo che a e b si scrivano in notazione binaria come $a = a_{n-1}a_{n-2} \ldots a_0, b = b_{m-1}b_{m-2} \ldots b_0$ e che $m \leq n$ (senza perdita di generalità)
  - ad esempio, se $a = 27, b = 3$, allora $a_4 = 1, a_3 = 1, a_2 = 0, a_1 = 1, a_0 = 1, b_1 = 1, b_0 = 1, n = 5, m = 2$
- allora $c = |\{i$ s.t. $0 \leq i \leq n \wedge a_i \neq b_i\}|$
- N.B.: $b_i = 0$ per $i \geq m$
  - nell'esempio di sopra, $c = |\{3, 4\}| = 2$, dal momento che $a_0 = b_0 = 1, a_1 = b_1 = 1$ e $a_2 = b_2 = 0$, mentre $a_3 \neq b_3$ e $a_4 \neq b_4$

Traccia C Formulazione matematica:

- supponiamo che a si scriva in notazione decimale come $a = a_{n-1}a_{n-2}\ldots a_0$
- ovviamente, b e c sono costituiti da una singola cifra decimale
- allora $d = d_{n-1}d_{n-2}\ldots d_0$ t.c. per ogni $0 \leq i < n$ si ha che $d_i = a_i$ se $a_i \neq b$, altrimenti $d_i = c$

Traccia D Formulazione matematica:

- stesse supposizioni di cui sopra
- allora $d = d_{n-1}d_{n-2}\ldots d_0$ t.c. per ogni $0 \leq i < n$ si ha che $d_i = a_i$ se $a_i \neq b$, altrimenti $d_i = (b + c)\%10$

```
unsigned diffass(unsigned a, unsigned b)
{
 unsigned i;
 unsigned res = 0, pot = 1;
 for (i = 0; a != 0 || b != 0; i++) {
  res += abs(a%10, b%10)*pot;
  pot *= 10;
  a /= 10;
  b /= 10;
 }
 return res;
}
```

```
unsigned diffass(unsigned a, unsigned b)
{
 if (a == 0 && b == 0)
  return 0;
 return abs(a%10, b%10) + 10*diffass(a/10, b/10);
}
```

```
unsigned distbin(unsigned a, unsigned b)
{
 unsigned res = 0;
 while (a != 0 || b != 0) {
  res += (a%2 != b%2);
  a /= 2;
  b /= 2;
 }
 return res;
}
```

```
unsigned distbin(unsigned a, unsigned b)
{
 if (a == 0 && b == 0)
  return 0;
 return (a%2 != b%2) + distbin(a/2, b/2);
}
```

```
unsigned sost(unsigned a, unsigned b, unsigned c)
{
 unsigned pot = 1;
 unsigned cifra_i;
 for (pot = 1; pot <= a; pot *= 10) {
  cifra_i = a/pot%10;
  if (cifra_i == b)
   a += (c - cifra_i)*pot;
 }
 return a;
}
```

```
unsigned sost(unsigned a, unsigned b, unsigned c)
{
 if (a == 0)
  return 0;
 return a%10 == b? c : a%10 + 10*sost(a/10, b, c);
}
```

```
unsigned sost_10(unsigned a, unsigned b, unsigned c)
{
 unsigned pot = 1;
 unsigned cifra_i;
 for (pot = 1; pot <= a; pot *= 10) {
  cifra_i = a/pot%10;
  if (cifra_i == b)
   a += ((c + b)%10 - cifra_i)*pot;
 }
 return a;
}
```

```
unsigned sost_10(unsigned a, unsigned b, unsigned c)
{
 if (a == 0)
  return 0;
 return a%10 == b? (c + b)%10 : a%10 + 10*sost_10(a/10, b, c);
}
```

- In sost e sost_10, la funzione fa ancora il suo dovere se gli argomenti sono $a = 0$, $b = 0$ e $c \neq 0$?
  - Correggere le 4 funzioni in modo che anche questo caso sia considerato
- Come occorre correggere la funzione `f1` affinché stampi il numero al contrario?
- Come può essere resa più efficiente la funzione `f2`?
- Rifare tutte le funzioni (per ora iterativamente, se ci riuscite anche ricorsivamente) supponendo che l'input sia dato sotto forma di array