Dipartimento di Informatica e Sistemistica
Antonio Ruberti

“Sapienza” Università di Roma

Ricorsione

Corso di Fondamenti di Informatica

Laurea in Ingegneria Informatica

(Canale di Ingegneria delle Reti e dei Sistemi Informatici - Polo di Rieti)

Anno Accademico 2007/2008

Prof. Paolo Romano

Si ringrazia il Prof. Alberto Finzi per aver reso
disponibile il materiale didattico sul quale si basano queste slides

Ricorsione

Una funzione che contiene al suo interno una attivazione di se stessa è detta *ricorsiva*.

Esempio:

```
void f(int a){
  if (a==0)
    printf("f(%d): ho finito \n“,a);
  else {
    printf("sono f(%d) ",a);
    printf("chiamata di f(%d) \n",a-1);
    f(a-1);
    printf ("f(%d): ho finito \n",a);
  }
}
int main(){
  f(3);
}
```

Output prodotto:

```
Sono f(3) chiamata di f(2)
Sono f(2) chiamata di f(1)
Sono f(1) chiamata di f(0)
f(0):ho finito
f(1):ho finito
f(2):ho finito
f(3):ho finito
```

Ricorsione

Ricorsione utilizzata per implementare funzioni che ammettono una **definizione induttiva**

Esempio: fattoriale

- Definizione iterativa: $fatt(n) = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$
- Definizione induttiva:

$$fatt(n) = \begin{cases} 1 & \text{se } n=0 \quad \text{(base)} \\ n \cdot fatt(n-1) & \text{se } n>0 \quad \text{(passo)} \end{cases}$$

Fissata la **base** dell'induzione, il **passo induttivo** permette la definizione della funzione successiva utilizzando la funzione precedente.

Algoritmo:

```
if (n==0) return 1
else {calcola il fattoriale di n-1
      moltiplicalo per n
      restituisci il valore ottenuto}
```

```
int fatt(int x){
   if (x==0)
      return 1;
   else
      return x*fatt(x-1);
}
```

Ricorsione

Altre definizioni induttive

Somma non negativi

$$somma(x,y) = \begin{cases} x & \text{se } y=0 \quad \text{(base)} \\ 1 + somma(x,y-1) & \text{se } y>0 \quad \text{(passo)} \end{cases}$$

Prodotto non negativi

$$prod(x,y) = \begin{cases} 0 & \text{se } y=0 \quad \text{(base)} \\ somma(x, prod(x,y-1)) & \text{se } y>0 \quad \text{(passo)} \end{cases}$$

Potenza non negativi

$$espon(x,y) = \begin{cases} 1 & \text{se } y=0 \quad \text{(base)} \\ prod(x, espon(x,y-1)) & \text{se } y>0 \quad \text{(passo)} \end{cases}$$

Esercizio: implementare queste operazioni come funzioni C.

Ricorsione

Esempio: leggere sequenza di caratteri interrotte da "\n" e stampare invertite. Es. legge: "arco" e stampa "ocra"

Problema: per invertire occorre prima memorizzare tutta le lettere, occorre una struttura per la memorizzazione:
1. Vettore (vediamo più avanti)
2. Sfruttare lo stack di RDA

Algoritmo:

leggi carattere i-esimo
if carattere letto "\n" fine
else {
  stampa il carattere i+1
  stampa il carattere corrente
}

```
void stampaInv(){
   char a;

   scanf("%c",&a);
   if(a != '\n'){
      stampaInv();
      printf("%c",a);
   }
}
```

Esempio

Leggere una sequenza di caratteri con un punto centrale e decidere se è palindroma (ignorando gli spazi bianchi)

Una sequenza è detta palindroma se la seq. letta da destra a sinistra è identica a quella letta da destra (es. anna).

Caratterizzazione ricorsiva di una plaindroma:
1. La sequenza costituita solo da ' . ' è palindroma
2. Una sequenza x s y è palindorma se lo è s ed x = y

Come si può fare senza il ' . ' centrale?

Soluzione

```
void palind() {
        char x,y;

        scanf(“%c”,&x);
        if (x==‘.’)  return 1;

        if ( palind() ) {
                scanf(“%c”,&y);
                if (x!=y) return 0;
                return 1;
        }
return 0;
}
```

```
int main() {
while ( palind() )
  printf(“Stringa palindroma, prova ancora!\n”);
printf(“Stringa non palindroma”);
}
```

Ricorsione Multipla

Si ha ricorsione multipla quando un'attivazione di funzione causa **più di una attivazione ricorsiva** della stessa funzione

Esempio: n-esimo numero di Fibonacci

Sequenza Fibonacci: 0, 1, 1, 2, 3, 5, 8, …

Def. induttiva

$$F(0) = 0 \qquad n=0$$
$$F(1) = 1 \qquad n=1$$
$$F(n) = F(n-2) + F(n-1) \qquad n > 1$$

Algoritmo

if n = 0 return 0
if n = 1 return 1
if n > 1
  calcola F(n-2)
  calcola F(n-1)
  return F(n-1)+F(n-1)

```
int fib(int n){
   if (n==0) return 0;
   if (n==1) return 1;
   if (n>1)  return fib(n-2) + fib(n-1);
}
```

Esempi

MCD con funzione ricorsiva:

$$MCD(x,y) \begin{cases} x & \text{se } y = x \\ MCD(x,y-x) & \text{se } y > x \\ MCD(x-y,y) & \text{se } x > y \end{cases}$$

```
int mcd(int x, int y) {
   if (y==x) return x;
   else if (y > 0) return mcd(x,y-x);
        else return mcd(x-y,x); }
```

Verifica interi primi tra loro:

$$primi(x,y) \begin{cases} vero & \text{se } x=1 \text{oppure } y=1 \\ falso & \text{se } x \neq 1,\ y \neq 1 \text{e } x = y \\ primi(x,y-x) & \text{se } x \neq 1,\ y \neq 1 \text{e } x < y \\ primi(x-y,y) & \text{se } x \neq 1,\ y \neq 1 \text{e } x > y \end{cases}$$

```
int primi(int x,int y) {
   if(x==1 || y==1) return 1;
   if(x!=1 && y!=1 && x ==y) return 0;
   if(x!=1 && y!=1 && x > y) return primi(x,y-x);
   if(x!=1 && y!=1 && x < y) return primi(x-y,y);
}
```

Esempi

Resto:

$$\text{resto}(x,y) \begin{cases} x & \text{se } 0 \leq x < y \\ \text{resto}(x-y,y) & \text{se } x > y \\ \text{resto}(x+y,y) & \text{se } x < 0 \end{cases}$$

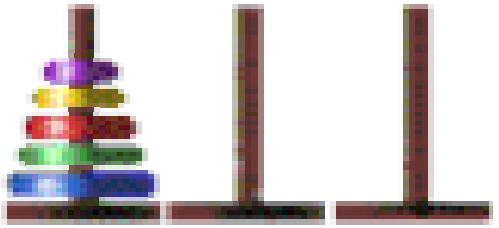
```
int resto(int x){
   if (x < y && x ≥ y) return x;
   if (x > y) return resto(x-y,y);
   if (x <0) return resto(x+y,y);
}
```

Ackerman:

$$A(m,n) \begin{cases} n+1 & \text{se } m=0 \text{ (caso base)} \\ A(m-1,1) & \text{se } n=0 \text{ (passo ricorsivo)} \\ A(m-1,A(m,n-1)) & \text{altrimenti} \end{cases}$$

Esempio Torre di Hanoi

- Tre perni 1, 2, 3
- Pila di dischi dimensione crescente su perno 1
- vincoli:
  - uno solo disco alla volta può essere spostato
  - disco più grande mai sopra uno più piccolo



Problema: come spostare m dischi dal perno 1 al perno 2?

Algoritmo:
1. Spostare n-1 dischi da 1 a 3
2. Spostare m-esimo da 1 a 2
3. Spostare n-1 dischi da 3 a 2

Esempio Torre di Hanoi

Algoritmo:
1. Spostare n-1 dischi da 1 a 3
2. Spostare m-esimo da 1 a 2
3. Spostare n-1 dischi da 3 a 2

Si utilizza una funzione muovi:

```
void muovi(int n, int s, int d, int a)
{
 if (n == 1)
     muoviUnDisco(s, d);
 else {
     muovi(n-1, s, a, d);
     muoviUnDisco(s, d);
     muovi(n - 1, a, d, s);
 }
}  /* muovi */
```

Esempio Torre di Hanoi

```
int main(void)
{
  int dischi;    /* numero di dischi */
  int s, d;      /* pali sorgente e destinazione */

  printf("Numero di dischi? ");
  scanf("%d", &dischi);
  printf("Palo sorgente?     [1, 2 o 3] ");
  scanf("%d", &s);
  printf("Palo destinazione? [1, 2 o 3] ");
  scanf("%d", &d);

  muovi(dischi, s, d, 6 - d - s);
  return 0;

}  /* main */
```

Esempio Torre di Hanoi

Quando si utilizza la ricorsione multipla, il numero di attivazioni può essere esponenziale nella profondità della chiamata ricorsiva (cioè nell'altezza della pila di attivazione)

Esempio Torre Hanoi:

att(n) = numero attivazioni di `muovi(…)` per n dischi = numero spostamento dischi

$$att(n) = \begin{cases} 1, & n=1 \\ 1+2\ att(n-1), & n>1 \end{cases}$$

$$att(n) > 2^{n-1}$$