Esercitazione 5

Saper usare la ricorsione

Risolvere un problema per ricorsione

1. Pensa ad un problema "piu' piccolo" (di "taglia" piu' piccola)
2. Supponi che il problema piccolo sia gia' risolto dal calcolatore: devi solo dire come mettere insieme la soluzione generale.
3. !! Caso di base:
   Pensa il caso/i casi di soluzione immediata
   (qual e' il caso "piu' semplice possibile" ?)
4. Verifica che il tuo algoritmo raggiunga un caso di base a partire da qualunque input.

1-2

2. e 3. e' tutto quello che serve al calcolatore.

Il piu' importante:

Verifica che il tuo algoritmo raggiunga un caso di base a partire da qualunque input!

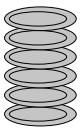
In caso di emergenza: Ctr + C

1-3

Programmare con…

un po' di magia

PROBLEMA: Voglio lavare una pila di 15 piatti



1-5

PROBLEMA: Voglio lavare una pila di 15 piatti.

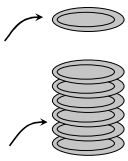
Ho un sistema che me li fa trovare lavati se gli do' una pila piu' piccola:
(se do' al calcolatore 14 piatti, e' disposto a lavarli lui)

Algoritmo ***RICORSIVO:***

- prendo un piatto,
- resta una pila di 14 piatti: il calcolatore me li lava lui
- lavo il mio piatto

un po' più semplice *del problema intero*

facilissimo



1-6

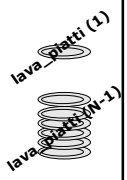
Ricorsione

PROBLEMA: Voglio lavare **N** piatti

Algoritmo ***RICORSIVO***

- lavo un piatto ← ***caso base:*** *facilissimo*
- lava_piatti (**N-1**)

passo ricorsivo: *un po' meno faticoso del problema intero*



1-7

Esempio

Condizione di terminazione

```
void lava_piatti(N)
{
   if (nessun_piatto_da_lavare)      // caso base
        riposati!

   else                              // passo ric.
     {
       Prendi un piatto
       Lavalo
       lava_piatti(N-1)
     }
}
```

Chiamata Ricorsiva

il numero di piatti da lavare decresce: la terminazione e' garantita!

1-8

Una funzione ricorsiva...

Ha una o piu' condizioni di terminazione (casi base)

Chiama se stessa ricorsivamete.

Ad ogni chiamata ci si avvicina alla condizione di terminazione

1-9

Il mio primo esercizio

- Scrivere una funzione ricorsiva che, dato un intero positivo K, stampa a video K "Salve!";.

```
void aSalve(int k){
     if (k==0) return;
     else
          { cout<<("Salve! ");
            aSalve(k-1);}

}
```

1-10

Quante biglie in questo vaso?

```
contaBiglie()

Vaso vuoto: return 0   //caso piu' semplice
                       // possibile
//else

Estraggo una biglia;
Return: 1 + contaBiglie()
```

Come avere un problema piu' piccolo?
Estraggo una biglia. Nel vaso ci sono meno biglie.
La soluzione generale e' il numero di biglie che restano +1

1-11

```
int  contaParole(ifstream & IN){
        string parola;
        if ( IN.eof()  ) return 0;

        IN>> parola;
        return (1+ contaParole(IN));}
```

la stessa idea

```
main(){ifstream IN;
  IN.open("vaso_biglie");
  if(!IN)
     {cout<<"vaso_biglie non si apre"<<endl; exit(-1);}
 cout << contaParole(IN)<< endl;
  IN.close();}
```

file vaso_biglie

```
rossa nera nera
rossa nera
```

1-12

Conta una sequenza di caratteri letta da cin e terminata da [enter]

la stessa idea

```
int contaChar (){
char ch;
ch=cin.get();

if (ch=='\n') return 0;
      else return 1+ contaChar();
}
```

1-13

Essere sicuri della propria ricorsione

Stampa dei primi N interi positivi

Per esempio, N=3.
Assumiamo che il calcolatore sappia fare la stampa di N-1 interi.

Quindi stampa(N-1) produce l'output:
1 2

Per ottenere a video
1 2 3
dobbiamo fare cout << N ***dopo*** aver chiamato stampa(N-1)

1-15

MORALE:
assumo che la chiamata ricorsiva svolge sempre il suo compito correttamente.

Devo concentrarmi solo sull'ultimo passo, quello che programmo io, per ottenere la soluzione corretta.

IMPORTANTE
Quest'idea ci permette anche di ***verificare*** che quello che abbiamo scritto e' corretto:
se la chiamata ricorsiva e' corretta, ed aggiungo il mio passo, ho il risultato giusto?

1-16

Esempio di verifica.
Per stampare (in ordine) gli interi da 1 a N, ho scritto il seguente codice:

```
void StampaN (int n)
{   if (n==0) return;
    cout << n;
    StampaN (n-1); }
```

Io stampo 3, e poi so
che la chiamata ricorsiva (se corretta) stampa a video 1 2

Quindi ottengo a video
3 1 2

so che NON VA!!

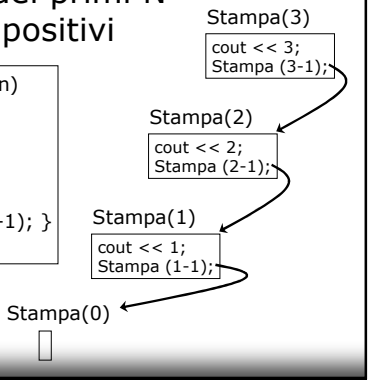
1-17

Capire la ricorsione:

tracciare il programma

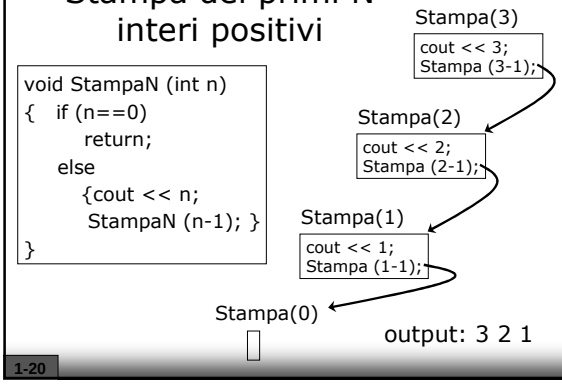
Stampa dei primi N interi positivi

```
void StampaN (int n)
{   if (n==0)
        return;
    else
       {cout << n;
        StampaN (n-1); }
}
```



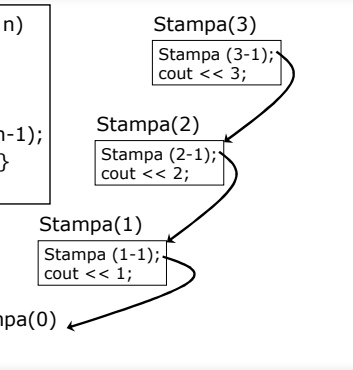
Stampa dei primi N interi positivi

```
void StampaN (int n)
{   if (n==0)
        return;
    else
       {cout << n;
        StampaN (n-1); }
}
```

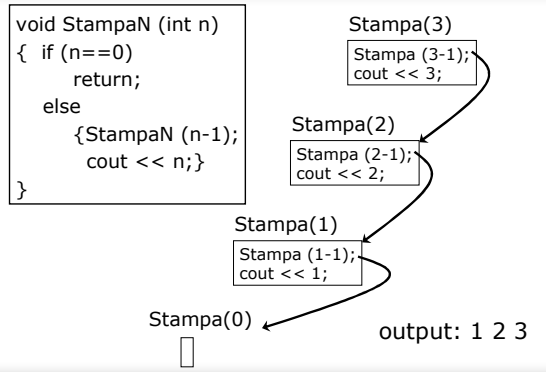


output: 3 2 1


```
void StampaN (int n)
{  if (n==0)
       return;
   else
       {StampaN (n-1);
        cout << n;}
}
```




```
void StampaN (int n)
{  if (n==0)
       return;
   else
       {StampaN (n-1);
        cout << n;}
}
```



output: 1 2 3

Ancora qualche esercizio

Conversione di base

Scrivere una funzione ricorsiva che dato un numero intero positivo **n** stampa la sua rappresentazione binaria.

ESEMPIO: Convertire il numero **35** dalla base **10** alla base **2**

35 : 2 = 17	resto	**1**
17 : 2 = 8	resto	**1**
8 : 2 = 4	resto	**0**
4 : 2 = 2	resto	**0**
2 : 2 = 1	resto	**0**
1 : 2 = 0	resto	**1**

RIDURRE IL PROBLEMA

se ho la rappresentazione binaria di 35/2 = 17 ed il resto 35%2 = 1

posso ottenere la rapp. binaria di 35 aggiungendo 1 alla rapp. di 17

La rappresentazione binaria del numero **35** é **100011**

```
void converti (int n) {
   int k=n/2;
   int resto=n%2;

   if (k>0)
        converti(k);
   cout << resto;
}
```

Prova ad eseguire –cioe' tracciare- converti(5);

1-25

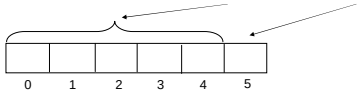
Ricorsione sugli array

Esercizio

Scrivere una procedura *ricorsiva* che, dato un array di interi di lunghezza n, calcoli la somma dei suoi elementi.

Esempio

int A [6]; somma(A, 6) = (A, 5) + a[5];



1-27

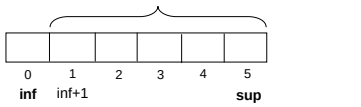
```
int somma (int a[ ], int dim)  {
        if (dim == 0) return 0;
        else
           return somma(a, dim-1) + a[dim-1]; }

main ( ) {
        const int DIM=4;
        int A[DIM]={3,2,1,2} ;
        cout << somma(A, DIM);
}
```

1-28

Un'altra soluzione

Esempio

int A[6]; somma(A, 0, 5) = A[0] + somma(A, 1, 5);



1-29

in due versioni...

```
int somma (int A[ ], int inf, int sup)  {
        if (inf == sup)
                return A[inf];
        else
                return A[inf] + somma (A, inf+1, sup); }
```

```
int somma2 (int A[ ], int inizio, int dim)  {
        if (dim==0)
                return 0;
        else
                return A[inizio] + somma2(A, inizio+1, dim-1);}
```

```
main ( ) {
        const int DIM=4;
        int A[DIM]={3,2,7,2} ;
        cout << somma(A, 0, DIM-1);
        cout << somma2(A, 0, DIM);  }
```

1-30

Esercizio

Scrivere una procedura *ricorsiva* che dato un array di interi di lunghezza dim

… raddoppia tutti gli elementi
...raddoppia tutti i valori dispari

… controlla se array e' ordinato

Anatomia di una funzione ricorsiva

Disegnare:

- Base (terminazione):
uno o piu' casi in cui la funzione svolge il suo compito in modo immediato (senza ricorsione).

- Passo ricorsivo (soluzione):
uno o piu' casi in cui otteniamo la soluzione utilizzando la chiamata di F su un problema piu' piccolo.

La terminazione e' garantita da qualcosa che decresce: ad ogni chiamata ci avviciniamo al caso base.

DEVO EVITARE

```
int CattivaRicorsione(n) {
    int x = CattivaRicorsione(n); //disastro!!
    if (n==1) return 1;
    else return n*x;
}
```


Forma generale di una funzione ricorsiva

Caso base (passo immediato)

Fa qualcosa,
chiama se stessa ricorsivamete,
fa qualcosaltro.

Ragionare per ricorsione, ancora e ancora

- svolto solo in AL -

Immaginatevi uno spiedino di carne:

pollo,pomodorino,pollo,manzo,pomodorino,cipolla,pollo…

1-37

PROBLEMA: quanti pezzi di manzo ci sono nel mio spiedino??

- *qual e' il caso piu' semplice possibile? quando termino?*
- *qual e' un problema piu' piccolo?*
  (per questo assumo di sapere la risposta)
- *come ottengo da questo il risultato finale?*

```
int ContaManzo (spiedino){
stecchino: return 0;

 Tolgo un pezzo e lo assaggio;
 if (pezzo ==manzo)
        return     1 + ContaManzo(spiedino);
   else
        return     ContaManzo(spiedino);
}
```

1-38

```
int ContaManzo (ifstream & IN){
        string parola;
        if ( IN.eof()  ) return 0;
        IN >> parola;

        if (parola=="manzo")
                return 1+ ContaManzo(IN) ;
        else  return ContaManzo(IN);}
```

la stessa idea

```
main(){ifstream IN;
  IN.open("spiedino");
  if(!IN)
     {cout<<"spiedino non si apre"<<endl; exit(-1);}

 cout << ContaManzo(IN)<< endl;

  IN.close();}
```

file spiedino:

```
manzo pomodorino manzo
cipolla manzo
```

1-39

Conta le occorrenze di 'a'
in una sequenza letta da cin
e terminata da ';'

la stessa idea

```
int contaA(){
char ch;
cin >> ch;

if (ch==';'){return 0;}
if (ch=='a') return (1+ conta() );
       else return conta();
}
```

1-40

Ci sono 'a' nella sequenza?

Da fare
entro la prossima esercitazione !

1-41

C'e' cipolla nel mio spiedino?

```
bool verif_Cipolla(spiedino){

Stecchino: return false;

Tolgo un pezzo;
if (Cipolla) return true;
else return verif_Cipolla(spiedino);
}
```

1-42

Ricorsione e liste

Del perche'
una lista e' uno spiedino

Formula magica per le liste

Una lista e':

O la lista vuota

O un nodo seguito da una lista

1-44

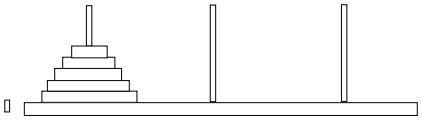
La torre di Hanoi
(un esempio interessante)

- svolto solo in MZ -

Le Torri di Hanoi

Quello delle *Torri di Hanoi* è un gioco che si svolge con tre paletti e alcuni dischi di diametro differente con un foro al centro in modo da poter essere infilati nei paletti.

Inizialmente i dischi sono tutti impilati a piramide sul primo paletto. Il disco più grande è in basso, il più piccolo in alto.



1-46

SCOPO DEL GIOCO

Lo scopo del gioco è quello di trasferire i dischi dal paletto di sinistra a quello di destra, senza mai mettere un disco su un altro di dimensione più piccola.

REGOLE DEL GIOCO

- È possibile spostare un solo disco alla volta;
- tutti i dischi devono essere sempre infilati nei paletti.

STRATEGIA

La strategia consiste nel considerare uno dei paletti come origine e un altro come destinazione. Il terzo paletto sarà utilizzato come deposito temporaneo.

1-47

STRATEGIA

Supponiamo di avere n dischi, numerati dal più piccolo al più grande. Inizialmente sono tutti impilati nel paletto di sinistra.
Il problema di spostare n dischi sul paletto di destra può essere descritto in modo ricorsivo così:

- Spostare i primi n-1 dischi dal paletto di sinistra a quello di centro.
- Spostare il disco n-esimo (il più grande) sul paletto di destra.
- Spostare i rimanenti n-1 dischi dal paletto di centro a quello di destra.

In questo modo il problema può essere risolto per qualsiasi valore di n>0 (n=0 è la condizione di stop della ricorsione).

1-48

PROGRAMMA

Vogliamo un programma che ci dia la strategia da seguire dato il numero di dischi

- il primo paletto (quello di sinistra) con S
- il secondo paletto (quello di centro) con C
- il terzo paletto (quello di destra) con R

Definiamo la procedura ricorsiva transfer, che trasferisce n dischi da un paletto all'altro.

1-49

```
void transfer (int n, char from, char to, char temp) {
/* n e' il numero di dischi, from indica il paletto di origine, to indica
   l'arrivo e temp indica il paletto usato come sosta temporanea*/
if (n=0) return;
if (n > 0) {
                /*  sposta n-1 dischi dall'origine alla sosta temporanea,
                                    usando il paletto libero come deposito*/
        transfer (n-1, from,  temp,  to);

              /* sposta il disco n-esimo dall'origine alla destinazione */
        cout <<"Sposta il disco "<< n
              << " da " << from  << "a " << to <<endl;

        /*  sposta n-1 dischi dalla sosta temporanea alla destinazione,
                                              usando l'origine come deposito*/
        transfer (n-1, temp,  to,  from);
   }
}
```

chiamata: transfer (3, 'S', 'D', 'C');

1-50

```
/* programma principale per il gioco delle TORRI DI HANOI */
/* realizzato con una procedura ricorsiva */
#include <iostream>
using namespace std;

void transfer (int n, char from, char to, char temp);

main ( )
{
   int n;
   cout <<"Benvenuto nelle TORRI DI HANOI\n\n";
   cout << "Quanti dischi ? ";
   cin >> n;

   transfer (n, 'S', 'D', 'C');
}
```

1-51

Eseguendo il programma con n=3 si otterrà il seguente output:

Benvenuto nelle TORRI DI HANOI

Quanti dischi ? 3

Sposta il disco 1 da L a R
Sposta il disco 2 da L a C
Sposta il disco 1 da R a C
Sposta il disco 3 da L a R
Sposta il disco 1 da C a L
Sposta il disco 2 da C a R
Sposta il disco 1 da L a R

1-52