DALLA RICORSIONE ALLA RICORSIONE TAIL

Il punto

- **Come esprimere la soluzione a un problema in modo che il processo computazionale ottenuto sia di tipo iterativo?**

ovvero:

- ***Come scrivere il programma in modo che risulti tail-ricorsivo?***

DALLA RICORSIONE ALLA RICORSIONE TAIL

- **La ricorsione tail è caratterizzata dal fatto che si computa "in avanti"**
  - **il risultato viene sintetizzato passo per passo**
  - **a ogni passo, prima si computa quel che c'è da calcolare, poi si effettua, come ultima cosa, la chiamata ricorsiva**
- **La chiamata ricorsiva ha quindi lo scopo di *portare avanti il risultato parziale* fin qui calcolato.**

DALLA RICORSIONE ALLA RICORSIONE TAIL

Dunque, per *esprimere il problema in forma ricorsiva tail* occorre:

- ***identificare il valore* iniziale da cui partire**
- ***identificare il valore* da sintetizzare passo dopo passo**
- ***identificare una regola* per ricavare il nuovo valore (relativo al passo successivo) *in funzione del valore attuale.***

Vediamo un esempio.

UN FATTORIALE… INNOVATIVO!

- **Definizione:**

  $n! = 1 * 2 * 3 * \ldots * k * \ldots * n$

- **Chiamiamo v_k il valore corrispondente al fattoriale di k**
- **Il valore di partenza è $v_1=1$**
- **La *regola per passare al fattoriale successivo* è allora $v_{k+1} = (k+1) * v_k$**

UN FATTORIALE… INNOVATIVO!

Quindi:

- **all'inizio (al passo k=1) si ha**

  $1! = v_1 = 1$

- **alla fine (al passo k=n) si ha**

  $n! = v_n$

  Perciò, dopo il passo k=n, il valore v denota il valore cercato, pari a n!

UN FATTORIALE… INNOVATIVO!

- **Le relazioni:**

  $1! = v_1 = 1$ **$(k+1)! = (k+1) * v_k$**

- **possono essere rilette dicendo:**
  - **al passo k=1, il valore del fattoriale è 1**
  - **detto v il valore del fattoriale al passo k, il valore del fattoriale al passo k+1 è v' = (k+1)*v**
  - **il valore v <u>dopo</u> il passo k=n dà il valore finale cercato.**

UN FATTORIALE… INNOVATIVO!

- **Nuova codifica con *due funzioni*:**
  - **una funzione fact(), inalterata rispetto a prima, per mascherare la differenza agli occhi del cliente**
  - **una funzione factIter() come *servitore privato di fact()*, che opera secondo il nuovo principio di "accumulo".**

UN FATTORIALE… INNOVATIVO!

```
int fact(int n){
  return factIter(n,1,1);
}

int factIter(int n, int v, int k){
  return (k==n) ? v :
       factIter(n, (k+1)*v, k+1);
}
```

Situazione iniziale: $v_{k=1} = 1$

Regola iterativa: $v_{k+1} = (k+1) * v_k$

UN FATTORIALE… INNOVATIVO!

```
int fact(
  return
}

int factIter(int n, int v, int k){
  return (k==n) ? v :
       fac ter(n, (k+1)*v, k+1);
}
```

v rappresenta il valore del fattoriale *dopo il passo k-esimo*

Quindi, il valore *dopo* il passo *n* è disponibile quando k=n.

UN FATTORIALE… INNOVATIVO?

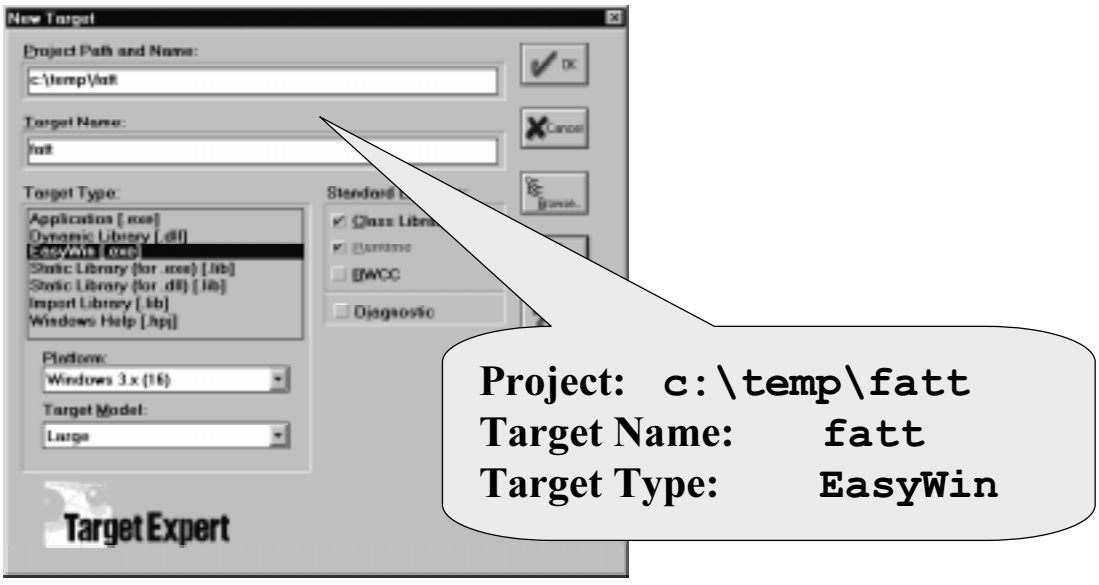
- **Perché questo esempio è "innovativo" ?**
- **il risultato viene sintetizzato via via che le chiamate si aprono, *"in avanti".***
- **È una soluzione *sintatticamente ricorsiva..***
- **… ma che dà luogo a un un processo computazionale *iterativo.***
  - **infatti, dopo k passi, abbiamo a disposizione il risultato parziale relativo a fact(k)**

UN FATTORIALE… INNOVATIVO?

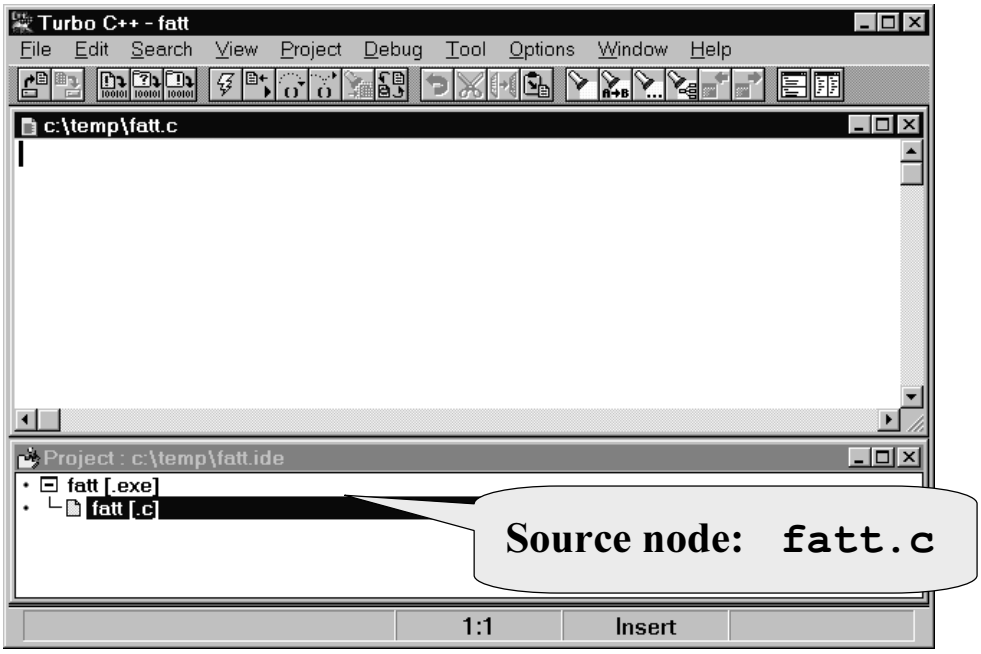
- **Perché q**
- **il risultat**
  chiamate
- **È una soluzione *sintatticamente ricorsiva..***
- **… ma che dà luogo a un un processo computazionale *iterativo.***
  - **infatti, dopo k passi, abbiamo a disposizione il risultato parziale relativo a fact(k)**

È un caso di RICORSIONE TAIL

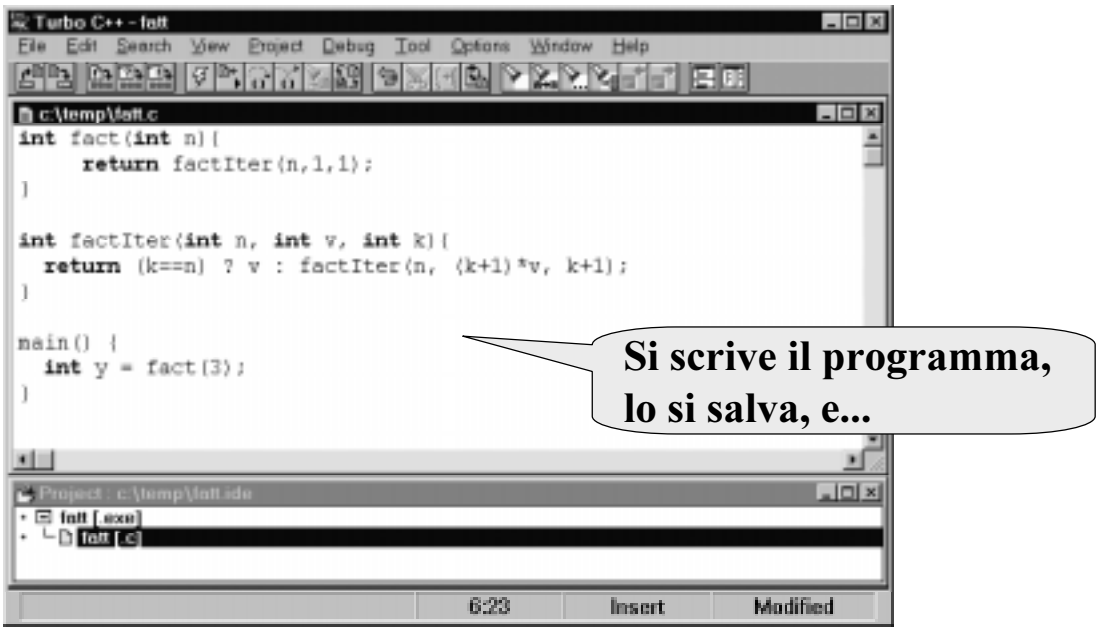
IL FATTORIALE: ESEMPIO COMPLETO



IL FATTORIALE: ESEMPIO COMPLETO



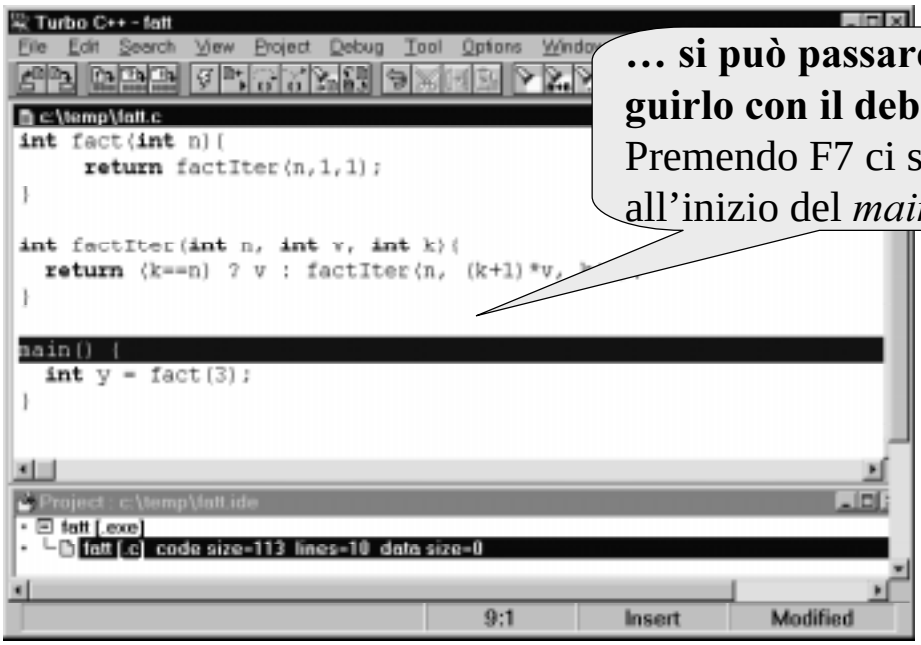
IL FATTORIALE: ESEMPIO COMPLETO



IL FATTORIALE: ESEMPIO COMPLETO

… lo si costruisce (F9). Se non ci sono errori...

IL FATTORIALE: ESEMPIO COMPLETO

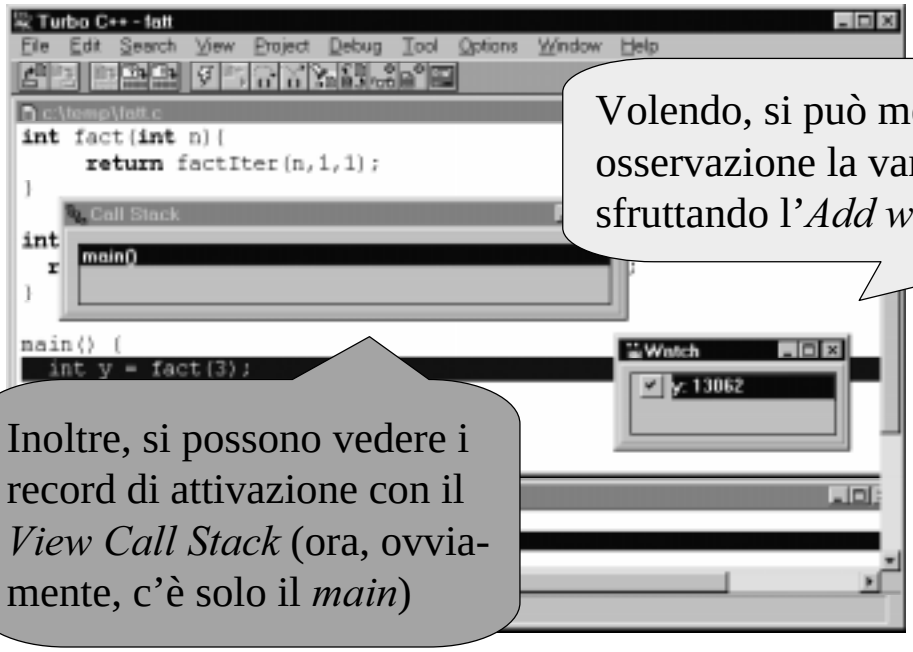


… si può passare a eseguirlo con il debugger. Premendo F7 ci si posiziona all'inizio del *main.*

IL FATTORIALE: ESEMPIO COMPLETO

Premendo ancora F7 ci si posiziona sulla prima istruzione, pronti per eseguirla.

IL FATTORIALE: ESEMPIO COMPLETO



Volendo, si può mettere sotto osservazione la variabile **y**, sfruttando l'*Add watch.*

Inoltre, si possono vedere i record di attivazione con il *View Call Stack* (ora, ovviamente, c'è solo il *main*)

ESEMPIO COMPLETO

Via via che l'esecuzione prosegue, il *Call Stack* mostra la successione dei record di attivazione che si accumulano.

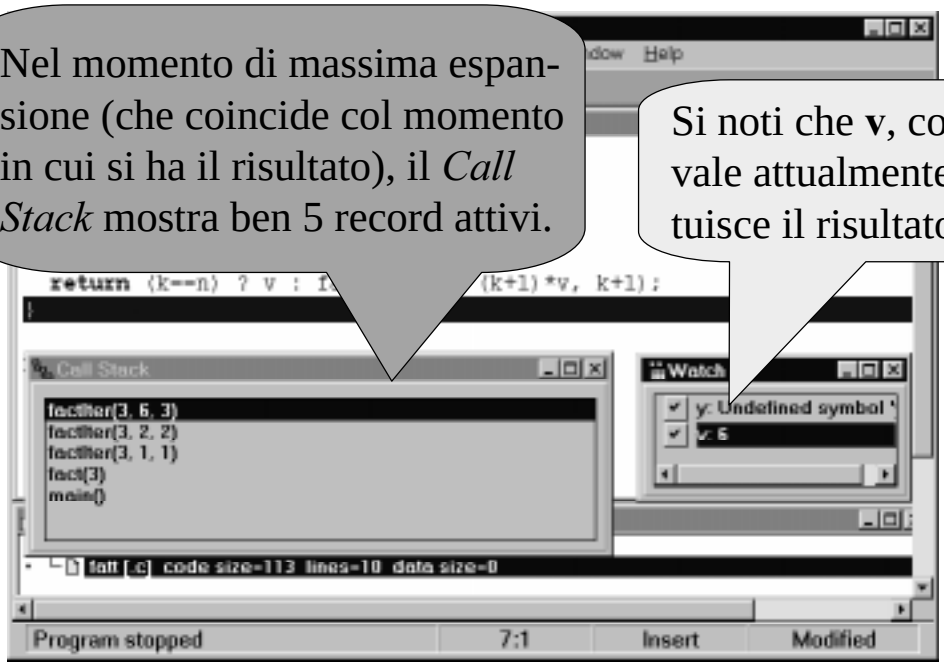
Si noti che la variabile **y** risulta ora *indefinita:* **siamo usciti dallo scope del main!**

Però, possiamo aggiungere ai *watch* la variabile **v**, che esiste **nello scope di factIter.**

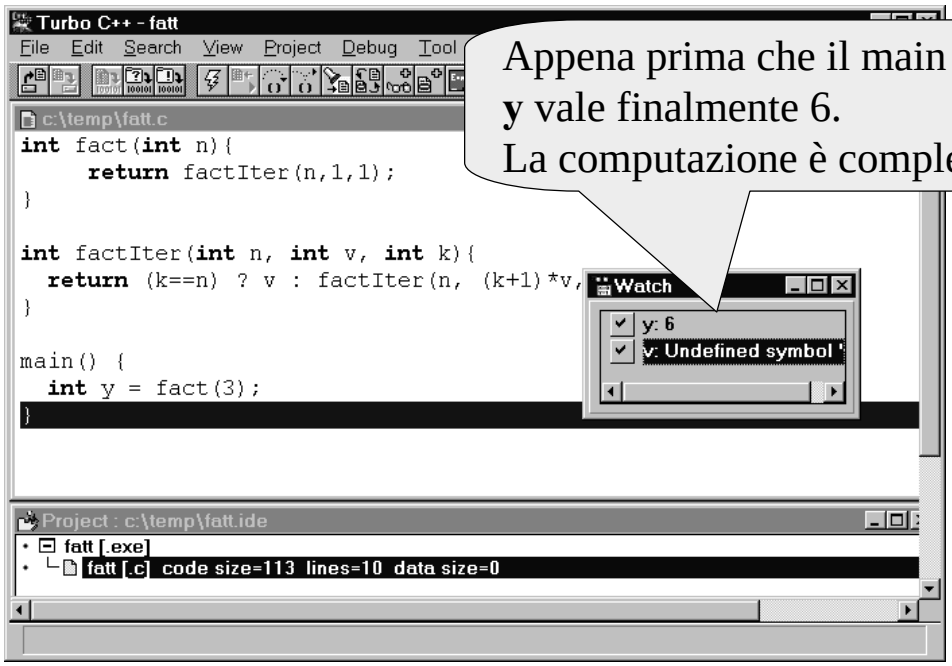
IL FATTORIALE: ESEMPIO COMPLETO

Nel momento di massima espansione (che coincide col momento in cui si ha il risultato), il *Call Stack* mostra ben 5 record attivi.

Si noti che **v**, come previsto, vale attualmente 6 (che costituisce il risultato)



IL FATTORIALE: ESEMPIO COMPLETO



Appena prima che il main termini,
y vale finalmente 6.
La computazione è completata.

IL FATTORIALE: ESEMPIO COMPLETO

Quando il main è terminato,
nessuna variabile esiste più
Tutto è scomparso.

UN ALTRO PROBLEMA: IL POLINOMIO

- **Calcolo del valore di un polinomio di grado $n \geq 0$ *a coefficienti unitari***

  $P(x,n) = x^0 + x^1 + \ldots x^n$

- **Raccogliendo x a fattore comune:**

  $P(x,n) = 1 + x^1 + \ldots x^n =$

  $= 1 + x * (1 + x^1 + x^2 + \ldots x^{n-1}) =$

  $= 1 + x * (1 + x * (1 + x + \ldots x^{n-2})) =$

  $= \ldots = 1 + x *(1 + x * (\ldots(1+(x+1)*x)\ldots))$

UN DIVERSO APPROCCIO

- **Questo suggerisce un diverso modo di procedere:**

  $P(x,n) =$

  $= 1 + x *(\overbrace{1 + x * (\ldots(1+(x+1)*x)\ldots)}^{v})$

- **Detto v il valore tra parentesi:**

  $P(x,0) = v_0 = 1$

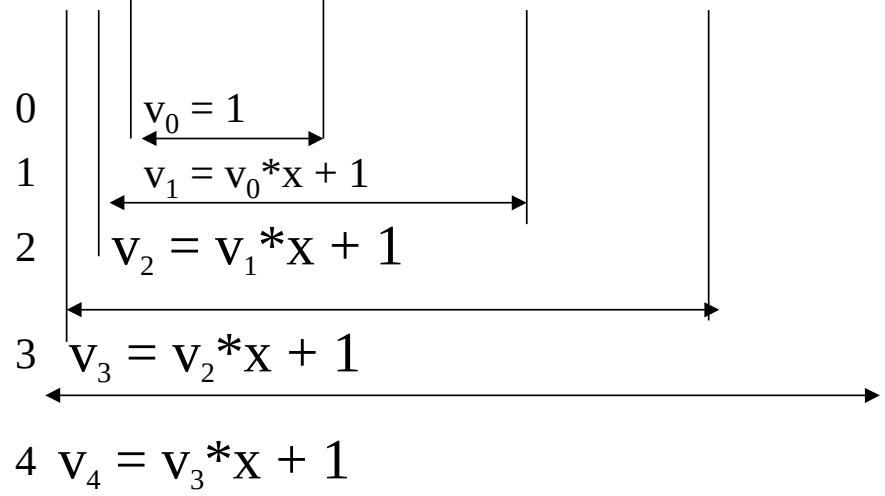
  $P(x,k+1) = v_{k+1} = 1 + x * v_k$

  $P(x,n) = v_n$

UN DIVERSO APPROCCIO - ESEMPIO

$x^4 + x^3 + x^2 + x^1 + x^0 =$

$(((x + 1) * x + 1) * x + 1) * x + 1$



UN DIVERSO APPROCCIO

- **La relazione:**

  $P(x,0) = v_0 = 1$ $\quad$ $P(x,k+1) = v_{k+1} = 1 + x * v_k$

- **può essere interpretata dicendo:**
  - **al passo k=0, il valore del polinomio è 1**
  - **detto v il valore del polinomio al passo k, il valore del polinomio al passo k+1 è v' = 1 + x * v**
  - **il valore v dopo il passo k=n dà il valore finale del polinomio.**

UN DIVERSO APPROCCIO

- **Codifica con *due funzioni*:**
  - **una funzione pol(), inalterata rispetto a prima, per mascherare la differenza agli occhi del cliente**
  - **una funzione polin() come *servitore privato di pol()*, che opera secondo il nuovo principio di "accumulo".**

UN DIVERSO APPROCCIO

- **Per passare dal valore del polinomio al passo k a quello al passo k+1 si applica la relazione v' = 1+x*v**
- **Il valore v dopo il passo k=n dà il valore finale di P(x,n)**

```
double polin(double x, int n,
                  double v, int k){
  return (k==n) ? v :
           polin(x,n,1+x*v,k+1);
}
```