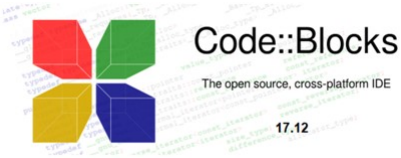
ITIS-LS "Francesco Giordani" Caserta

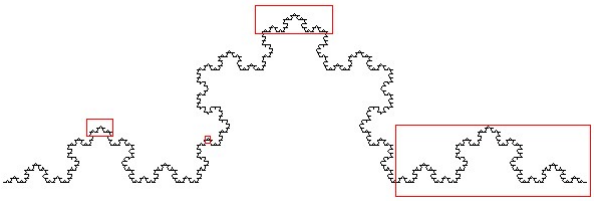
prof. Ennio Ranucci

a.s. 2019-2020

Semplici funzioni ricorsive

Esercitazioni C++





La ricorsione è un metodo per definire funzioni in modo tale che la funzione includa sé stessa nella propria definizione. Si tratta di una tecnica di programmazione che consente di suddividere il problema da risolvere in sottoproblemi analoghi all'originale ma più semplici, perché agenti su dati di ingresso ridotti. Un algoritmo ricorsivo è definito in due fasi: dapprima si definisce la risoluzione di un problema simile a quello di partenza ma di dimensione ridotta (passo base); quindi si definisce il passo di risoluzione generale come combinazione di problemi di dimensione inferiore

Nella definizione induttiva di una funzione è possibile identificare:

- *un caso base (o più di uno) detto tappo induttivo perchè è la terminazione della chiamata ricorsiva*
- *un caso induttivo (o più di uno)*

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es1*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *code::blocks ver 17.12*
Obiettivo didattico:
Utilizzare funzioni ricorsive
Obiettivo del programma:
Scrivere una funzione ricorsiva (in C) che calcoli la somma degli elementi di un array di interi.

```
#include <iostream>

using namespace std;
int vet[10]={1,2,3,4,5};
int somma(int vetPar[],int dimLogPar);
int main()
{
   cout << "Somma degli elementi del vettore: " <<somma(vet,5);
   return 0;
}
int somma(int vetPar[],int dimLogPar)
{
  if (dimLogPar==0) return 0;
  else return vetPar[dimLogPar-1] + somma(vetPar, dimLogPar-1);
}
```

Somma(vet,4)->5+somma(vet,3)->5+4+somma(vet,2)->5+4+3+somma(vet,1)->
->5+4+3+2+somma(vet,0)->5+4+3+2+1+0->15

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es2*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *code::blocks ver 17.12*
Obiettivo didattico:
Utilizzare funzioni ricorsive
Obiettivo del programma:
Calcolo del massimo di un vettore con procedimento ricorsivo

Assegnare il primo elemento dell'array alla variabile "massimo" e confrontarlo con tutti gli altri cambiando il valore del massimo se questo è minore della cella corrente del vettore.
Detta N la lunghezza del vettore "array"
– Se N = 1 (caso base)
• max = array[0]
– Se N >= 2 (passo induttivo)
• il massimo del vettore di N elementi (max) sarà uguale al risultato del confronto tra array[0] e il massimo degli elementi del sotto-vettore che che va da array[1] a array[N] (lungo N-1).

```
include <iostream>
using namespace std;

int vet[10]={1,2,3,7,5,4};
int trovaMassimo(int vetPar[], int dimPar)
{
  if (dimPar==0) return vetPar[dimPar];
    else
     if (vetPar[dimPar] > trovaMassimo(vetPar, dimPar-1)) return vetPar[dimPar];
     else return trovaMassimo(vetPar, dimPar-1);
}

int main()
{
   cout << "Massimo: " <<trovaMassimo(vet,5);
   return 0;
}
```

trovaMassimo(vet,5)) ->
-> 4>trovaMassimo(vet,4) ->
-> 4>5>trovaMassimo(vet,3) ->
-> 4>5>7>trovaMassimo(vet,2) ->
-> 4>5>7>3>trovaMassimo(vet,1) ->
-> 4>5>7>3>2>trovaMassimo(vet,0) ->
-> 4>5>7>3>2>1 ->
-> 4>5>7>3>2 ->
-> 4>5>7>3 ->
-> 4>5>7 ->

-> 4> trovaMassimo(vet,3) ->
-> 4>7>trovaMassimo(vet,2) ->
-> 4>7>3>2>trovaMassimo(vet,1) ->
-> 4>7>3>2>trovaMassimo(vet,0) ->
-> 4>7>3>2>1 ->
-> 4>7>3>2 ->
-> 4>7>3 ->
-> 4>7 ->

-> trovaMssimo(vet,4) ->
-> 5>trovaMassimo(vet,3) ->
-> 5>7>trovaMassimo(vet,2) ->
-> 5>7>3>trovaMassimo(vet,1) ->
-> 5>7>3>2>trovaMassimo(vet,0) ->
-> 5>7>3>2>1 ->
-> 5>7>3>2 ->
-> 5>7>3 ->
-> 5>7 ->

-> trovaMssimo(vet,3) ->
-> 7>trovaMassimo(vet,2) ->
-> 7>trovaMassimo(vet,1) ->
-> 7>3>2>trovaMassimo(vet,0) ->
-> 7>3>2>1 ->
-> 7>3>2 ->
-> 7>3 ->
-> 7

Per verificare:

```
#include <iostream>
using namespace std;
int i=0;
int vet[10]={1,2,3,7,5,4};
int trovaMassimo(int vetPar[], int dimPar)
{
  i++;cout <<i<<") ";
  if (dimPar==0) {cout<<"$"<< vetPar[dimPar]<<"$ "<<endl;return vetPar[dimPar];}
    else
     if (vetPar[dimPar] > trovaMassimo(vetPar, dimPar-1)) { cout<<"*"<< vetPar[dimPar]<<"*
";return vetPar[dimPar]; }
     else { cout<<endl<<"-" <<vetPar[dimPar]<<"-"<<endl;return trovaMassimo(vetPar, dimPar-1);};
}

int main()
{
   cout << endl<<"Massimo: " <<trovaMassimo(vet,5);
   return 0;
}
```

Caso migliore: vettore ordinato
Caso peggiore: vettore ordinato in modo decrescente