UNIVERSITA' DEGLI STUDI ROMA TRE

DIPARTIMENTO DI FISICA "E. AMALDI"

Laboratorio di Programmazione e Calcolo

6 crediti

a cura di

Severino Bussino

Anno Accademico 2019-20

0) Struttura del Corso

1) Trattamento dell'informazione
Elementi di Architettura di un Computer

2) Sistemi operativi

3) Introduzione alla
Programmazione ad oggetti (OO)

4) Simulazione del Sistema Solare

5) Introduzione al linguaggio C/C++

6) Elementi di linguaggio C/C++

A 1 - istruzioni e operatori booleani

2 - iterazioni (`for`, `while`, `do ... while`)

B - istruzioni di selezione (`if`, `switch`, `else`)

C - funzioni predefinite. La classe `math`.

7) Puntatori

8) Vettori (Array)

9) Classe SistemaSolare (prima parte)

10) Gestione dinamica della memoria

11) Classe SistemaSolare

12) Programma di Simulazione (main)

13) Ereditarieta'

14) Classe Sonda

15) "per referenza" e "per valore"

16) Output su file

17) Polimorfismo

- Concetto
- Funzione
- Realizzazione in C++
- Esempi

18) Richiami sulle funzioni

19) L'algebra di una classe (cenni)

20) Cenni alle classi Template

21) La libreria STL
(Standard Template Library)

22) Applicazioni

23) Un esempio: la gestione dei dati nello studio delle oscillazioni del pendolo semplice

Le lezioni in aula si concludono il 13 dicembre

Le Esercitazioni di Laboratorio proseguono
senza variazioni con l'orario gia' comunicato
(e pubblicato sul sito)

3 - 4 dicembre Classe Sistema Solare + Sonda (es. 6 + es. 8a)

10 - 11 dicembre Un container STL per i pianeti (es. 8b)
[Polimorfismo. La Classe Shape (es. 7)]

17 - 18 dicembre Simulazione Prova Individuale 3 gruppi

7 gennaio non ci sara' Esercitazione

8-14-15 gennaio Prova Individuale 3 gruppi

Date degli Esami

I appello - Prova Pratica: 21 gennaio - Ore 14:30-17:30
Prova Orale: 22 gennaio - (Ore 14:30) stanza 36

II appello - Prova Pratica: 3 febbraio - Ore 14:30-17:30
Prova Orale: 4 febbraio - (Ore 14:30) stanza 36

Chi non supera gli "esoneri" puo' sostenere un esame **pratico** e **orale**

Sostenendo l'esame si rinuncia pero' al voto ottenuto "in itinere"

Verbalizzazione
(per chi ha superato gli esoneri)

E' INDISPENSABILE iscriversi su GOMP all'esame del Corso per la prova Pratica e per la prova Orale

"Laboratorio di Programmazione e Calcolo"

(e **non** Laboratorio di Calcolo , ecc)
nella data in cui ci si presentera' per la verbalizzazione

La verbalizzazione avverra' nelle stesse date in cui e' prevista la prova Orale (stanza 36)

22 gennaio Ore 14:30-16:30

4 febbraio Ore 14:30-16:30

Se non potete verbalizzare in queste date e in questi Orari, per favore avvisate per E-mail

(bussino@fis.uniroma3.it)

Applicazioni

- esempi di applicazioni sviluppate nel testo di Barone et al.:
  - Ricerca degli 0 di una funzione mediante il metodo di bisezione e mediante il metodo di Newton (4.3.2)
  - Ricerca dei numeri primi (4.3.3)
  - Problemi di arrotondamento (4.5)
  - Riordinamento degli elementi di un vettore: Bubblesort (5.2.1)
  - Ricerca binaria (5.2.2)
  - Soluzione di sistemi di equazioni lineari (5.4, 7.3.2)
  - Generazione di numeri casuali (5.5)
  - Manipolazione di testi (5.6.3, 6.5.1)
  - Istogrammi (7.4.1) – Visto
  - Calcolo del χ^2 di una distribuzione (7.4.2)
  - Calcolo di una derivata (7.6)
  - Interpolazione e integrazione numerica (cap. 8)

23) Un esempio: la gestione dei dati nello studio delle oscillazioni del pendolo semplice

ripasso!!!

Template, STL ...

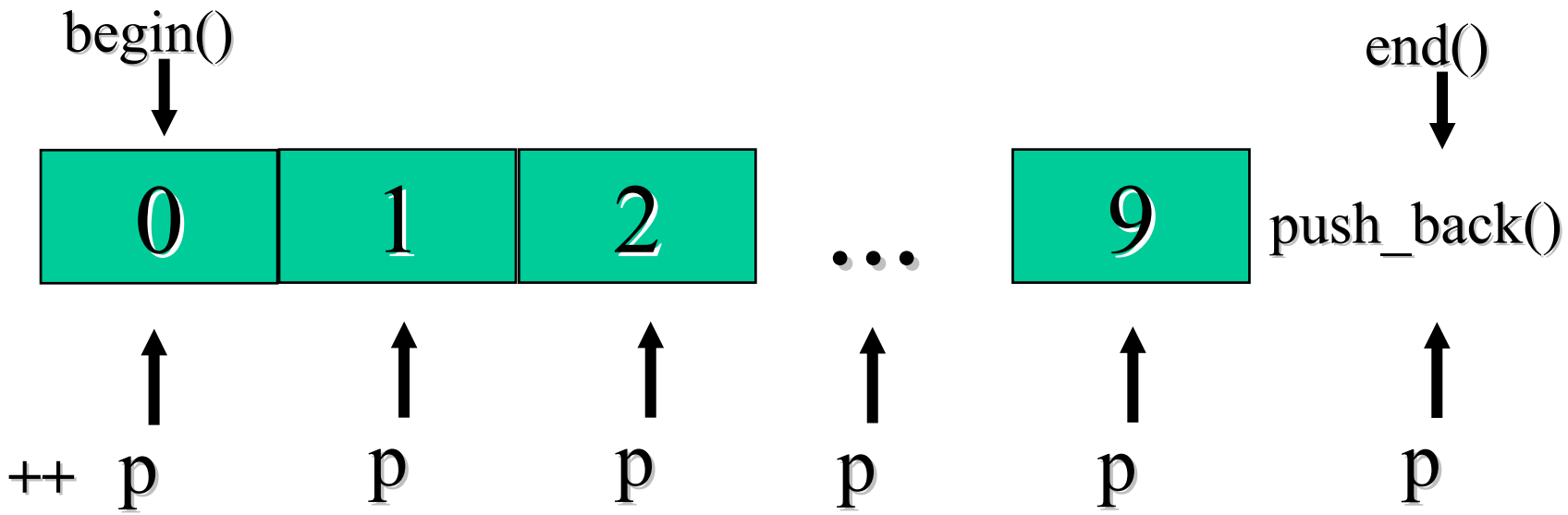
... i container STL

---

vector list map

vector: 1 - Come funziona

vector e' un array contiguo di oggetti



`push_back(value)` e' un metodo della classe vector per:

- aggiungere alla fine del vettore un nuovo elemento
- assegnare al nuovo elemento il valore `value`

vector: 2 - gli iteratori

Gli iteratori permettono di scorrere il container e si comportano come i puntatori

Se v e' un container di tipo vector

`v.begin()` - e' un iteratore
punta all'inizio del container

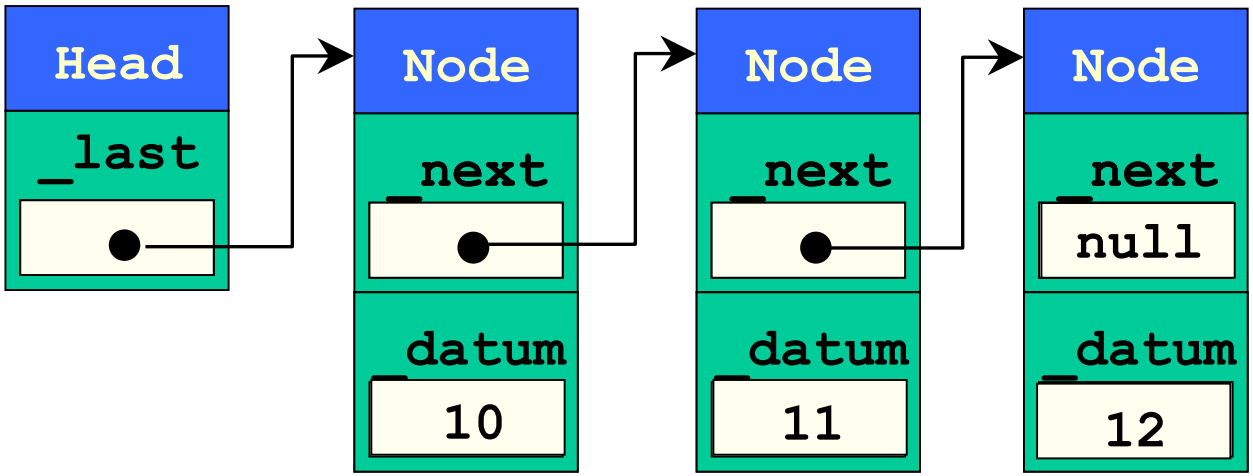
`v.end()` - e' un iteratore
punta alla prima locazione **dopo** la fine

```
vector<T>::const_iterator p ;
```

Si puo' usare il typedef

```
typedef vector<T>::const_iterator viter ;
```

List



Si utilizza come vector (sostituendo a `vector` la parola `list`)

Piu' efficiente nel gestire le inserzioni di elementi

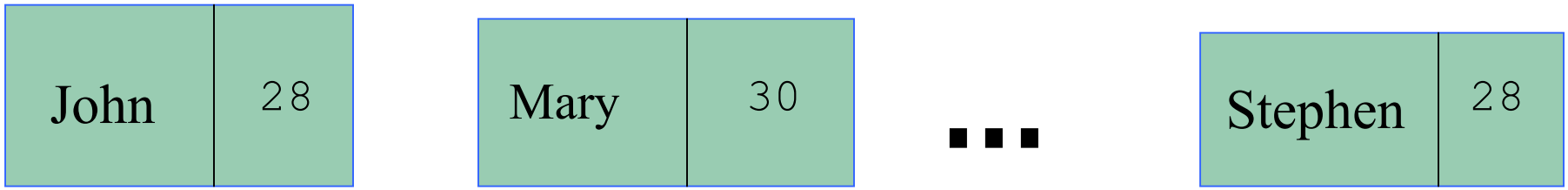
Map

Una "map" e' un container associativo costituito da coppie chiave-valore (key - T)

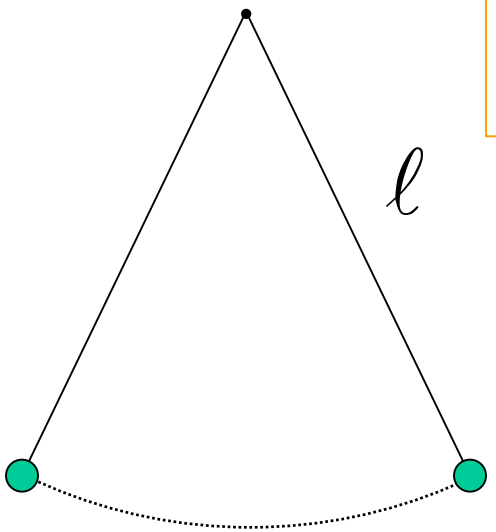
La map e' indicizzata rispetto alla chiave

Il valore della chiave deve essere univoco

Esempio: `map<string, int>`



Un esempio: il pendolo



- Seconda legge dinamica
- Forza peso
- Equazioni differenziali
- Sviluppo in serie di Taylor (Mac L
- ……

$$T = 2\pi\sqrt{\frac{\ell}{g}}$$

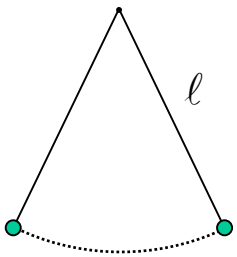
- Sara' vero?
- Posso misurare g

$$g = 4\pi^2 \frac{\ell}{T^2}$$

Come effettuo le misure ? (1)

- Con un metro **misuro** la lunghezza ℓ del pendolo
- Con un cronometro **misuro** il periodo T

- Per migliorare la qualita' delle misure del periodo T
  - lascio oscillare il pendolo per np periodi
  - misuro il tempo di oscillazione t per n periodi
  - il periodo T sara' dato da $T = \frac{t}{np}$



Per ciascun valore della lunghezza ℓ effettuo una serie n_ℓ di misure del periodo T

Come effettuo le misure ? (2)

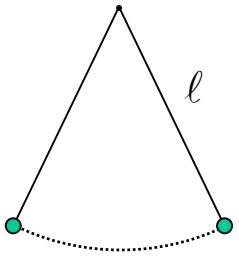
- Per verificare la relazione tra T ed ℓ

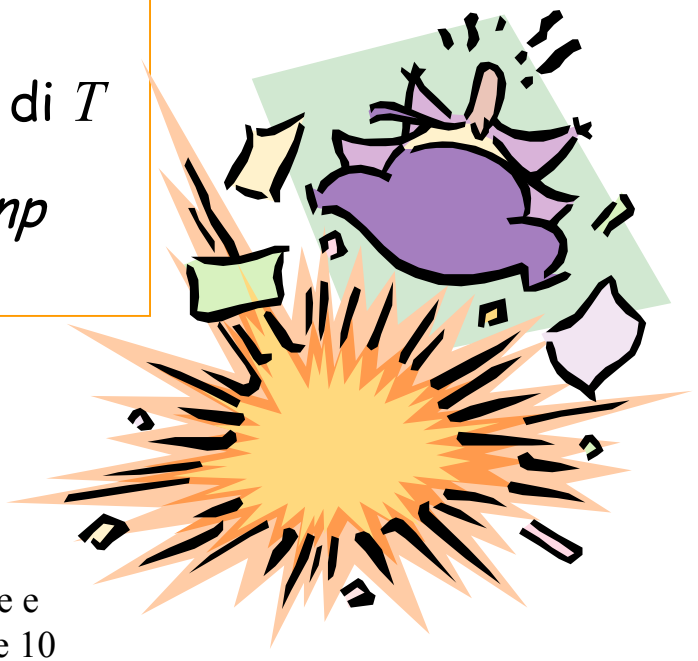
- Per migliorare la qualita' della misura di g

Effettuo una serie di misure per diversi valori di ℓ

Quindi ho:

- Una serie di misure a diversi valori di ℓ
- Per ogni valore di ℓ ho una serie di misure di T
- Ciascuna misura di T deriva dalla misura di *np* oscillazioni

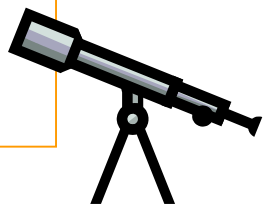




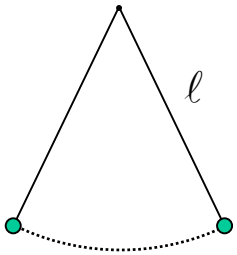
L'analisi delle misure

- Ho raccolto una grande quantita' di dato

- Voglio analizzarli!



- Un grafico di T in funzione di ℓ
- Un grafico di T^2 in funzione di ℓ
- Un valore di g che derivi dall'analisi di tutti i dati
- Se cambio il materiale... (massa inerziale e massa gravitazionale!)
- Se le oscillazioni non sono piccole (Taylor - Mac Laurin ...)



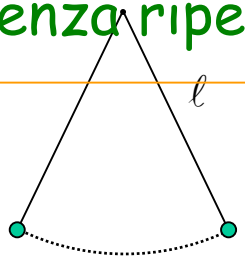
I/O (Input Output)

Input

- Dove scrivo i dati che ho raccolto?
- Dove li memorizzo per poterli analizzare?
- Se li voglio analizzare in un'altra seduta di laboratorio?
- Se voglio aggiungere dati raccolti in due esercitazioni diverse?

- Dove scrivo i risultati dell'analisi?
- Come mi interfaccio con un programma di grafica (ad esempio gnuplot) ?
- Posso combinare insieme i risultati di due analisi (senza ripetere l'analisi) ?

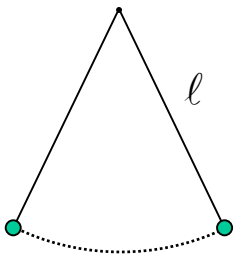
Output



Un approccio OO

- Quali sono gli oggetti?
- Come posso organizzare i dati?
- Di chi sono le responsabilita'?

- Chi sono gli attori?
- In che relazione stanno tra loro?
- Chi fa cosa?

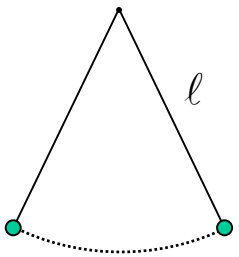


Quali sono gli oggetti?

I **dati** sperimentali!

Cioe' le misure che avete annotato sul quaderno!

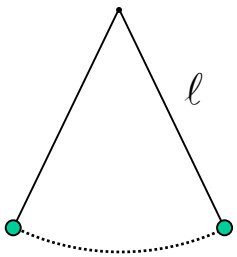
- `datiT` - un oggetto (e quindi una classe) con le misure relative al tempo impiegato a compiere np oscillazioni
- `SetdatiL` - un oggetto (e quindi una classe) con l'insieme delle misure relative a una o piu' lunghezze



La classe `datiT` (attributi)

datiT.h

```
#ifndef DATIT_H
#define DATIT_H

#include <iostream>
using namespace std ;

class datiT {

   protected:

    double misT   ;
    double d_misT ;
    int nmis      ;

                                  // continua
```



La classe datiT (metodi)

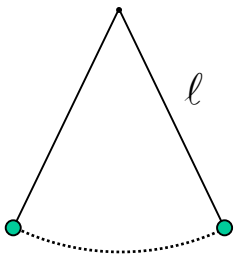
datiT.h

```
public:

    // costruttori
    datiT()  {  } ;
    datiT(double Tn, double dTn, int n) ;

    // distruttore
    ~datiT() {  } ;

    // metodi di tipo Get
    double Tmis()  const  { return misT   ; } ;
    double dTmis() const  { return d_misT ; } ;
    int Nmis()     const  { return nmis   ; } ;

    // altri metodi
    double  T() const ;
    double dT() const ;
                              // continua
```

La classe `datiT` (operatori)

datiT.h

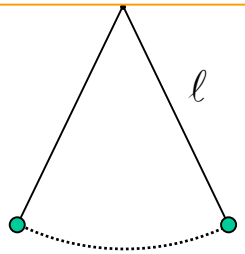
```
    // operatori booleani di confronto
    bool operator == (const datiT &d2) const ;
    bool operator != (const datiT &d2) const ;

    bool operator > (const datiT &d2)  const ;
    bool operator < (const datiT &d2)  const ;

    bool operator >= (const datiT &d2) const ;
    bool operator <= (const datiT &d2) const ;


} ;

#endif
```



L'implementazione della classe `datiT` (costruttore)

datiT.cc

```
#include "datiT.h"

    // costruttore
    datiT::datiT(double Tn, double dTn, int n)  {

      misT   =  Tn   ;
      d_misT =  dTn  ;
      nmis   =  n    ;

      if(nmis<=0) {
          cerr << "  datiT::datiT(...) Attenzione! nmis = "
               << nmis << endl ;
      }

    }
                                  // continua
```



L'implementazione della classe `datiT` (altri metodi)

datiT.cc

```
// altri metodi
double datiT::T() const  {

  if(nmis<=0) {
    cerr << " datiT::T() Attenzione! nmis = " << nmis << endl ;
  } else {
    return  misT/nmis ;
  }

}

double datiT::dT() const {

  if(nmis<=0) {
    cerr << " datiT::dT() Attenzione! nmis= " << nmis << endl ;
  } else {
    return  d_misT/nmis ;
  }

}
                    // continua
```

L'implementazione della classe datiT (operatori - 1)

datiT.cc

```
// operatori booleani di confronto
    bool datiT::operator == (const datiT &d2) const  {

      if( T() == d2.T() ) {
          return true  ;
      } else {
          return false ;
      }

    }

    bool datiT::operator != (const datiT &d2) const  {

      if( T() != d2.T() ) {
          return true  ;
      } else {
          return false ;
      }

    }

    bool datiT::operator > (const datiT &d2) const  {

      if( T() > d2.T() ) {
          return true  ;
      } else {
          return false ;
      }

    }                                          // continua
```

L'implementazione della classe `datiT` (operatori - 2)

```
bool datiT::operator < (const datiT &d2) const  {

  if( T() != d2.T() ) {
      return true  ;
  } else {
      return false ;
  }

}

bool datiT::operator >= (const datiT &d2) const  {

  if( T() >= d2.T() ) {
      return true  ;
  } else {
      return false ;
  }

}

bool datiT::operator <= (const datiT &d2) const {

  if( T() <= d2.T() ) {
      return true  ;
  } else {
      return false ;
  }

}                          // fine del file di implementazione datiT.cc
```

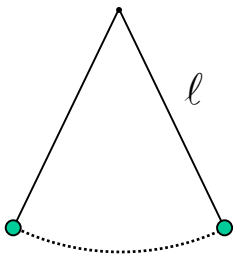
Come posso organizzare i dati?

Posso usare un container STL !

- per ogni lunghezza L ho una serie di misure di T

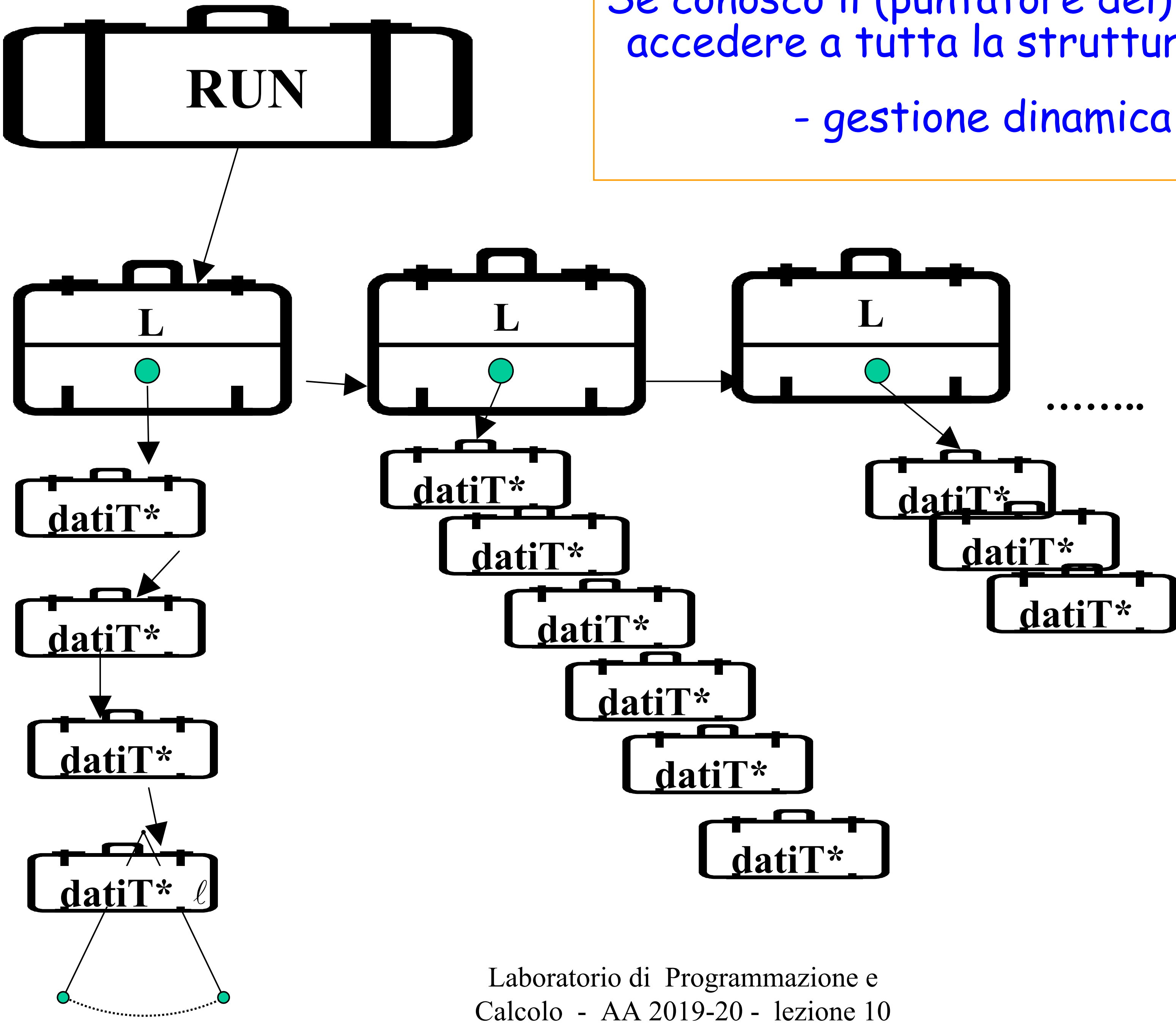
- quindi posso usare

  - un **vector** per l'insieme delle misure di T corrispondenti ad un L fissato
  - una **map** per associare ciascuna lunghezza alla serie di misure di T

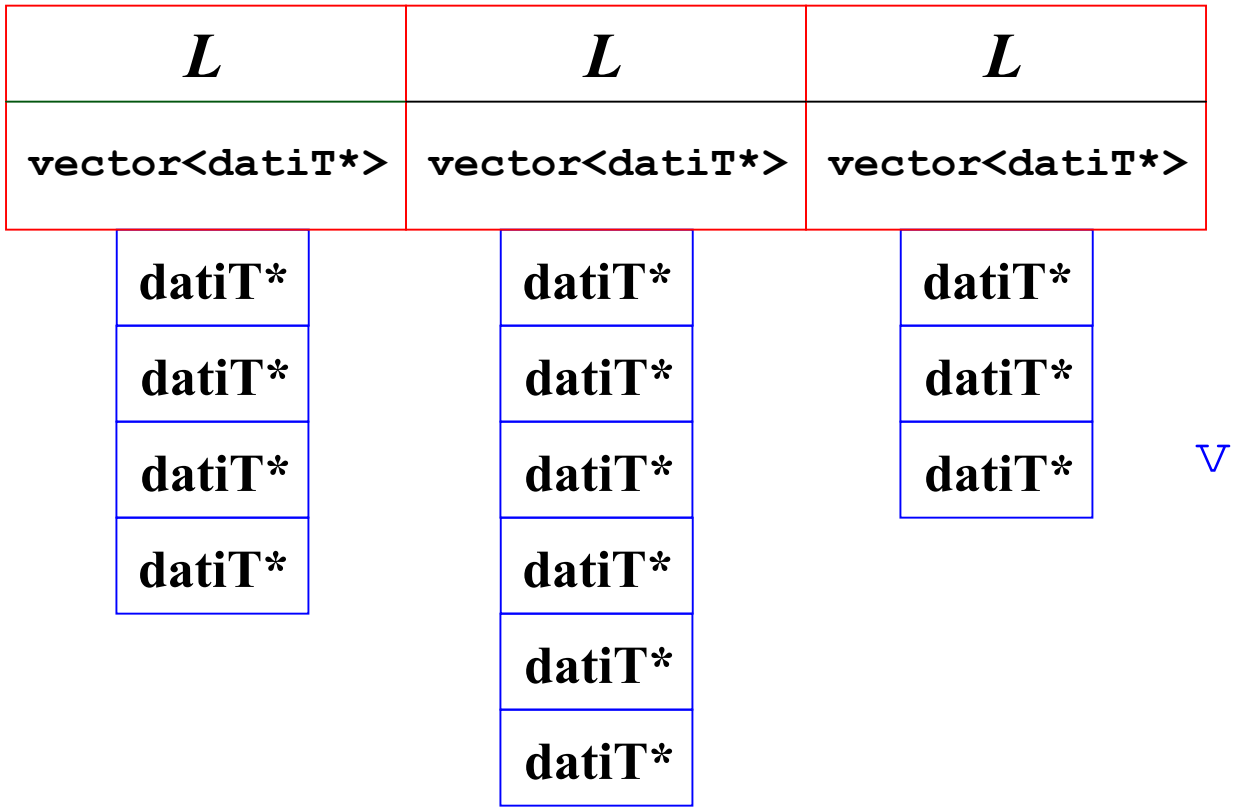


Se conosco il (puntatore del) Run, posso accedere a tutta la struttura dei dati

- gestione dinamica -



map<double, vector<datiT*> >



Se ho un problema semplice, posso usare questa struttura nel `main()` ...

```
#include "datiT.h"

#include <iostream>

#include <cstdlib>
#include <map>
#include <vector>

using namespace std ;

int main() {

  int nlmax = 10   ;
  int nTmax[nlmax] ;

  map< double, vector<datiT*> > setDati ;
  vector<datiT*> vlt ;

  typedef map< double, vector<datiT*> >::const_iterator mapiter ;
  typedef vector<datiT*>::const_iterator viter   ;

  typedef map< double, vector<datiT*> >::iterator nc_mapiter    ;
  typedef vector<datiT*>::iterator nc_viter      ;

                                        // continua
```

```
for (int i =0; i<nlmax/2; i++ ) {

   nTmax[i]          = i+1 ;
   nTmax[nlmax-i-1] = i+1 ;

}
cout << endl << endl ;

for (int il=0; il<nlmax; il++) {

   vlt.clear() ;
   cout<< vlt.size()<< endl ;

   for (int iT=0; iT<nTmax[il]; iT++) {

      vlt.push_back( new datiT(il+1,iT+1,iT+2) ) ;

   }

   setDati[il+1] = vlt ;

   cout << " mapsize=" << setDati.size() << " il+1=" << il+1 << endl;
   cout << endl ;

 }

 cout << endl ;
```

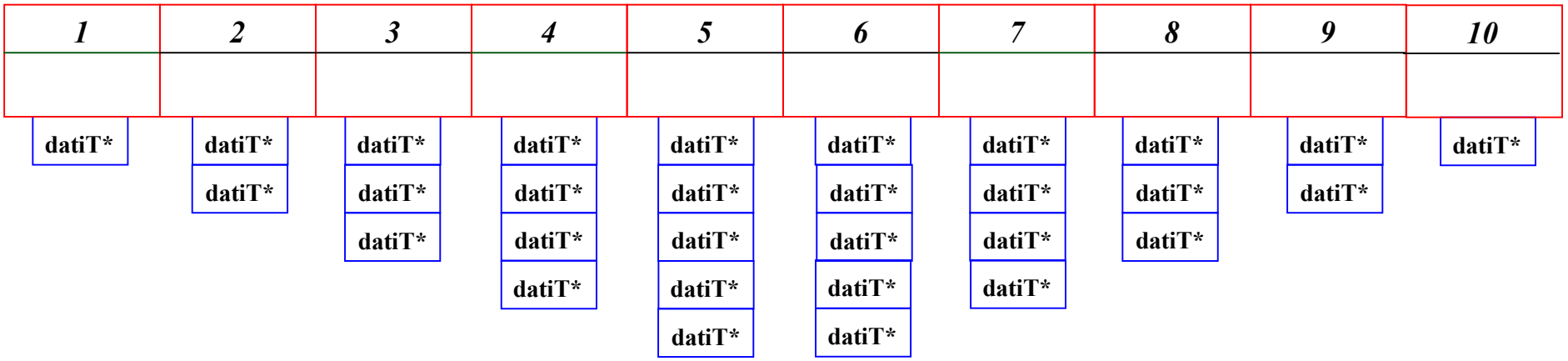
```
// output


for(mapiter pm=(setDati.begin()); pm!=setDati.end(); pm++) {

  cout << pm->first << " " << (pm->second).size() << endl ;

  for(viter pv=(pm->second).begin(); pv!=(pm->second).end(); pv++) {

   cout << "   " << (*pv)->Tmis() << "  " << (*pv)->Nmis()
                 << "  " << (*pv)->T() << endl ;

  }
  cout << endl ;

}
```

```
  // delete
  // ( oggetti datiT - vector<datiT*> - map<double, vector<dati*> )

  for(nc_mapiter pm=(setDati.begin()); pm!=setDati.end(); pm++) {

    for(nc_viter pv=(pm->second).begin();
                                    pv!=(pm->second).end(); pv++) {

      delete *pv ;

    }

    (pm->second).clear() ;

  }

  setDati.clear() ;


 cout << endl ;

 return 1;

}
```

```
nbseve(~/lez_10)>prvmap

0
 mapsize = 1   il+1 = 1

0
 mapsize = 2   il+1 = 2

0
 mapsize = 3   il+1 = 3

0
 mapsize = 4   il+1 = 4

0
 mapsize = 5   il+1 = 5

0
 mapsize = 6   il+1 = 6

0
 mapsize = 7   il+1 = 7

0
 mapsize = 8   il+1 = 8

0
 mapsize = 9   il+1 = 9

0
 mapsize = 10   il+1 = 10
```

```
1 1
    1   2   0.5

2 2
    2   2   1
    2   3   0.666667

3 3
    3   2   1.5
    3   3   1
    3   4   0.75

4 4
    4   2   2
    4   3   1.33333
    4   4   1
    4   5   0.8

5 5
    5   2   2.5
    5   3   1.66667
    5   4   1.25
    5   5   1
    5   6   0.833333
```

```
6 5
    6   2   3
    6   3   2
    6   4   1.5
    6   5   1.2
    6   6   1

7 4
    7   2   3.5
    7   3   2.33333
    7   4   1.75
    7   5   1.4

8 3
    8   2   4
    8   3   2.66667
    8   4   2

9 2
    9   2   4.5
    9   3   3

10 1
    10   2   5


nbseve(~/lez_10)>
```


1
2
3
4
5
6
7
8
9
10
datiT*

… oppure posso implementare una classe apposita …

SetdatiL.h

```
#ifndef SETDATIL_H
#define SETDATIL_H

#include "datiT.h"

#include <iostream>

#include <cstdlib>
#include <map>
#include <vector>

using namespace std ;

class SetdatiL {

   protected:

    double LMin   ;
    double LMax   ;

    map< double, vector<datiT*> > setDati ;

                                          // continua
```

```
public:
  // costruttori
  SetdatiL()  {  } ;
  SetdatiL(double minL, double maxL) ;

  // distruttore
  ~SetdatiL()  ;

  // metodi di tipo Get
  double Lmin()  const  { return LMin  ; } ;
  double Lmax()  const  { return LMax  ; } ;
  int Ndati()    const  { return setDati.size()  ; } ;
  const map< double, vector<datiT*> > * pSetDati() const
                                          { return &setDati ; }

  // altri metodi
  void  addT(double Lnow, datiT* & pTnow ) ;

  // eventualmente 3 metodi per scorrere il container (...)
    ......

} ;

#endif
```

L'implementazione della classe SetdatiL

SetdatiL.cc

```
#include "SetdatiL.h"

   // costruttore
   SetdatiL::SetdatiL(double minL, double maxL) {
         LMin = minL ;
         LMax = maxL ;
   }

   // distruttore
   SetdatiL::~SetdatiL() {

   // qui dovrei inserire i distruttori
   // lo stesso codice dell'esempio con solo main

   }

                                        //continua
```

```
   // altri metodi
   void  SetdatiL::addT(double Lnow, datiT* & pTnow ) {

        if(Lnow < LMin) { LMin = Lnow ; }
        if(Lnow > LMax) { LMax = Lnow ; }

        if(!setDati.count(Lnow)) {
          // il valore Lnow non esiste ancora
          // creo il container di tipo vector

          vector<datiT*>  vlc  ;
          vlc.push_back(pTnow) ;

          setDati[Lnow] = vlc ;

      } else {
          // il valore Lnow gia' esiste

          map< double, vector<datiT*> >::iterator mapiter ;

          mapiter = setDati.find(Lnow);
          (mapiter->second).push_back(pTnow) ;

      }

   }
// eventualmente + 3 metodi per scorrere il container (restituisce un
// iteratore a setDati, restituisce begin() e restituisce  end()  )
```

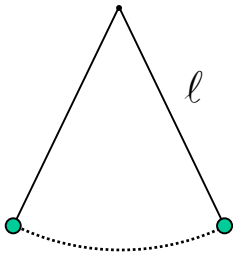
Di chi sono le responsabilita'?

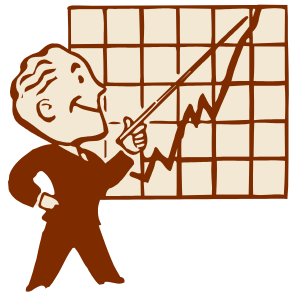
Input (da file)

Dati

Analisi Dati

Output (su file)





Input da file

Il file di input deve essere ben strutturato. Ad esempio:

```
nbseve(~)>sim_pend
0.20 1.10 10
0.20 5
0.8975 0.50 1
0.8975 0.50 1
0.8975 0.50 1
0.8975 0.50 1
0.8975 0.50 1
0.30 6
1.0992 0.50 1
1.0992 0.50 1
1.0992 0.50 1
1.0992 0.50 1
1.0992 0.50 1
0.40 4
1.2692 0.50 1
1.2692 0.50 1
1.2692 0.50 1
1.2692 0.50 1
0.50 8
1.4190 0.50 1
1.4190 0.50 1
1.4190 0.50 1
1.4190 0.50 1
1.4190 0.50 1
1.4190 0.50 1
1.4190 0.50 1
0.60 7
1.5545 0.50 1
1.5545 0.50 1
1.5545 0.50 1
1.5545 0.50 1
1.5545 0.50 1
1.5545 0.50 1
1.5545 0.50 1
0.70 9
1.6790 0.50 1
1.6790 0.50 1
1.6790 0.50 1
1.6790 0.50 1
1.6790 0.50 1
1.6790 0.50 1
1.6790 0.50 1
1.6790 0.50 1
1.6790 0.50 1
0.80 6
1.7950 0.50 1
1.7950 0.50 1
1.7950 0.50 1
1.7950 0.50 1
1.7950 0.50 1
1.7950 0.50 1
0.90 10
1.9038 0.50 1
1.9038 0.50 1
1.9038 0.50 1
1.9038 0.50 1
1.9038 0.50 1
1.9038 0.50 1
1.9038 0.50 1
1.9038 0.50 1
1.9038 0.50 1
1.9038 0.50 1
1.00 7
2.0068 0.50 1
2.0068 0.50 1
2.0068 0.50 1
2.0068 0.50 1
2.0068 0.50 1
2.0068 0.50 1
2.0068 0.50 1
1.10 4
2.1048 0.50 1
2.1048 0.50 1
2.1048 0.50 1
2.1048 0.50 1

nbseve(~/lez_10)>
```

Lettura del file e uso delle classi `SetdatiL` e `datiT`

```
#include "datiT.h"
#include "SetdatiL.h"

#include <iostream>
#include <fstream>
#include <string

using namespace std ;

int main() {

   double LMinf, LMaxf, LNow ;
   double T, dT ;
   int nL, nTL, nT ;
   datiT* datnow ;

   SetdatiL pendolo ;
   string filename = "sim_pend.dat" ;

   ifstream inFile(filename.c_str()) ;


                                                    //continua
```

Lettura del file e uso delle classi `SetdatiL` e `datiT`

```
    if(!inFile.eof()) {
      inFile >> LMinf >>  LMaxf >>  nL ;
    }

    for(int iL=0; iL<nL; iL++) {
      if(!inFile.eof()) {
        inFile >> LNow >> nTL ;
      }
      if (inFile.eof()) { cerr << " errore lettura file" << endl;}

      for(int it=0; it<nTL; it++) {
        inFile >> T >> dT >> nT ;

        datnow= new datiT(T, dT, nT) ;
        pendolo.addT(LNow, datnow) ;

      }

    }

inFile.close();
return 1;

}
```

Il Manager di Analisi (1)

Anche ora, posso scorrere l'oggetto di tipo SetDatiL nel `main()` ....

... e calcolarmi tutte le quantita' che voglio!

- valori medi di T per ciascun valore di L
- l'errore associato a ciascun valore di T

  (varianza e varianza della media)
- il valore del χ^2 assumendo la relazione $T = 2\pi\sqrt{\frac{\ell}{g}}$
- il valore di g che si ottiene dall'insieme dei dati

  e l'errore ad esso associato
- stampare delle tabelle con L Δl T ΔT T^2 Δt^2

  (ad es. per gnuplot) (magari anche con i valori previsti)

Oppure mi costruisco una classe Manager

... in analogia alla classe `Organizer` utilizzata in laboratorio

Chiamiamo questa classe Analysis

Proviamo a scriverne la struttura

Potrete implementare gran parte dei metodi utilizzando cio' che avete appreso nel corso di Esperimentazioni di Fisica I !

Il Manager di Analisi (2)

Analysis.h

```
#ifndef ANALYSIS_H
#define ANALYSIS_H

#include "SetdatiL.h"

#include <iostream>
#include <string>

using namespace std;


class Analysis {

protected:

    SetdatiL* pDati                              ;
    map< double, vector<double> > mapTmedi ;

    double gvalue ;
    double deltag ;

  // ... altri eventualmente utili
                                                 //continua
```

```
public:

  // costruttori
  Analysis() {} ;
  Analysis( SetdatiL * dati) {  pDati = dati ; } ;

  // distruttore
  ~Analysis()  ;

  // metodi specifici
  void calcolaMedieVar()    ;
  void calcola_g()          ;
  double  g()         ;
  double  errg()      ;

  void stampaTmedi()      ;

  void tabellepergrafici()                  ;
  void tabellepergrafici(string filename)   ;

  double chi2()  ;

} ;

#endif
```

L'implementazione del Manager di Analisi

Classe `Analysis`

```
#include "Analysis.h"

  // distruttore
  Analysis::~Analysis() {
        // eventuali istruzioni per cancellare le mappe
  }

  // metodi specifici
  void Analysis::calcolaMedieVar() { }
  void Analysis::calcola_g()       { }

  double Analysis::g()      { }
  double Analysis::errg()   { }

  void Analysis::stampaTmedi()  { }

  void Analysis::tabellepergrafici()                  { }
  void Analysis::tabellepergrafici(string filename)   { }

  double Analysis::chi2() { }
```

Un esempio: il metodo `calcolaMedieVar()` della Classe `Analysis`

Analysis.cc

```
void Analysis::calcolaMedieVar() {

   mapTmedi.clear() ;


  // scorro il container come nell'esempio a pag. 31

  // riempio la nuova mappa con i valori di L e di <T>
  // e di <T^2>

  // scorro la nuova mappa e mi calcolo la varianza (?)

  // aggiorno la nuova mappa che avra' come elementi
  //   primo elemento    L
  //   secondo elemento  un vector a tre componenti
  //                     T medio   Var T   errore sulla media


}
```

Buon Natale!