La programmazione *Object Oriented* e le sue applicazioni in Fisica.

Gabriella Cataldi, INFN Lecce

Daniele Martello, dip. Fisica & INFN Lecce

Classi e Oggetti: *Costruttori e Distruttori*

Una classe dovrebbe contenere un costruttore in grado di gestire ogni modo in cui un oggetto puo' essere creato ...

```
#include <iostream>
class BugsLife{
 public:
   BugsLife(int s){
     _zanzara = new int(s);
    cout << " ZZZ….. La zanzara numero''
        << s << " e'  stata creata!"<< endl;
    };
   int * punta_il_Bug() {return _zanzara;};
 private:
 int * _zanzara;
};
void main () {
  int * p;
  BugsLife zizi(1);
  p =zizi.punta_il_Bug();
  cout << " la zanzara " << *p
          <<" e' puntata da"<< p << endl;
};
```

?

E se qualcuno scrive?
BugsLife zizi;
…….

Classi e Oggetti: *Costruttori e Distruttori*

E' buona regola inserire un costruttore di default!...

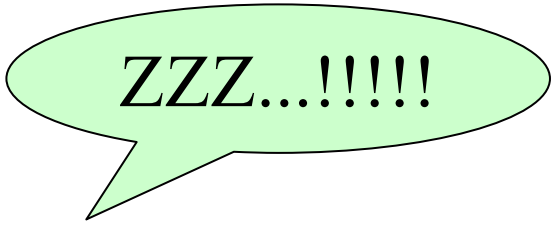
```
#include <iostream>
class BugsLife{
 public:
   BugsLife(int s){
     _zanzara = new int(s);
    cout << " ZZZ….. La zanzara numero''
        << s << " e'  stata creata!"<< endl;
    };
BugsLife(){
     _zanzara = new int(0);
    cout << " ZZZ….. La zanzara di default ''
        << " e'  stata creata!"<< endl;
    };

   ……..
```

E se voglio clonare?

Classi e Oggetti: *Costruttori e Distruttori*

Costruttore copia!

```
#include <iostream>
class BugsLife{
 public:
   BugsLife(int s){
     ……};
BugsLife(){
    …… };
BugsLife( const BugsLife& ref){
   int*  tmp = ref.punta_il_Bug()
     _zanzara = new int(*tmp);
    cout << " ZZZ….. Il clone della  zanzara''
        << " e'  stato creato!"<< endl;
    };
……..
```



Classi e Oggetti: *Costruttori e Distruttori*

Attenzione all' allocazione di memoria!!!!

Stiamo creando oggetti allocando dinamicamente la memoria!

In un loop abbiamo creato 10 zanzare ….

```
void main(){
 int *p[10];
 for(int i=0;i<10;i++) {
  BugsLife zzz(i);
  p[i]=zzz.punta_il_Bug();
 }
// sono fuori dal loop … cosa c'e' in memoria …?
// le mie zanzare sono morte …?
}
```

Classi e Oggetti: *Costruttori e Distruttori*

Attenzione all' allocazione di memoria!!!!


```
for(int i=0;i<10;i++) {
  BugsLife zzz(i);
  p[i]=zzz.punta_il_Bug();
 }
 for(int i=0;i<10;i++) {
 cout << "la zanzara "<< p[i];
       << " e' puntata  da "<< *(p[i])<<endl;
 }
}
```

Classi e Oggetti: *Costruttori e Distruttori*

Il Distruttore


```
public:
.......
~BugsLife(){
  cout << "RAID! Li ammazza stecchiti!"<<endl;
 delete _zanzara;
  };
```

Classi e Oggetti: *Esercizio1*

Data.h

```
class Data {
private:
        ErrorPoint * _data;
        int _size;
public:
// default constructor. Read the array of data from keyboard
    Data();
// constructor the array of data from file
    Data (const char * filename);
// copy constructor
    Data( const Data & data);
// destructor
   ~Data();
// return the number of error points in the data sample
    int size() const;
// return the array of data
    const ErrorPoint * data() const;
};
```

Classi e Oggetti: *Esercizio1*

Data.cxx

```
// default constructor. Read the array of data from keyboard
Data::Data() {
   cout<< "Di quante misure sperimentali disponiamo? (almeno 2)“ <<endl;
   cin>>_size;
   _data = new ErrorPoint[_size];
   for (int i=0; i<_size; i++) {
      cout<< "inserisci la misura numero “ <<i+1<< " nel formato: valore "
            << "(spazio) errore“ <<endl;
          cin>>_data[i];
   }
};
//*****************************************************************
// Step 1:
// insert the destructor and the two methods those return
// the array and the size of the array.
//*****************************************************************
```

Classi e Oggetti: *Esercizio1*

Main.cpp

```
//*********************************************************************
// Step 1:
// read data from keyboard and calculate the MediaAritmetica and
// DeviazioneStandardDati
//*********************************************************************
  Data mydata();
  cout << "Media Aritmetica        =“ <<
    MediaAritmetica(mydata.data(),mydata.size())<<endl;
  cout << "Deviazione Standard     =“ <<
    DeviazioneStandardDati(mydata.data(),mydata.size())<<endl;
```

Classi e Oggetti: *Esercizio1*

Main.cpp

```
//*********************************************************************
// Step 2:
// read data from file and calculate the MediaAritmetica and
// DeviazioneStandardDati
//*********************************************************************

Data mydata("Input.t");

cout << "Media Aritmetica        =“ <<
  MediaAritmetica(mydata.data(),mydata.size())<<endl;
cout << "Deviazione Standard     =“ <<
  DeviazioneStandardDati(mydata.data(),mydata.size())<<endl;
```

Data.cxx

```
//*********************************************************************
// Step 2:
// insert the constructor to read the data from file
//*********************************************************************
// constructor the array of data from file  ---missing ----
```

Classi e Oggetti: *Esercizio1*

Main.cpp

```
//*********************************************************************
// Step 3:
// modify the ErroreMediaAritmetica, MediaPesata and ErroreMediaPesata
// functions
//*********************************************************************
  cout << "Errore Media Aritmetica  =“ <<
    ErroreMediaAritmetica(mydata.data(),mydata.size())<<endl;
  cout << "Media Pesata             =“ <<
    MediaPesata(mydata.data(),mydata.size())<<endl;
  cout << "Errore Media Pesata      =“ <<
    ErroreMediaPesata(mydata.data(),mydata.size())<<endl;
```

Classi e Oggetti: *Esercizio1*

Data.cxx

```
//*****************************************************************
// Step 4:
// insert the copy constructor
//*****************************************************************
// copy constructor  ---missing ----
```

Classi e Oggetti: *Esercizio2*

Main.cpp

```
//*****************************************************************
// modify  all the Statistica functions to accept Data objects
// See as example MediaAritmetica.
// NOTE:  you have to insert the
// overloading of the operator [] in the Data class.
//*****************************************************************
```

Data.cxx

```
// operator overloading  --- missing  ---
const ErrorPoint & Data::operator[] (int k) const {
  // insert the code.....
};
```

uggerimento: notate che una chiamata all'operatore [] di un oggetto di classe Data mydata[k]) deve ritornare l'ErrorPoint dell'array di posto k-esimo. Ricordate inoltre he una chiamata del tipo: mydata[k].value() comporta nell'ordine prima una chiamata ll'operatore [] di Data e successivamente la chiamata di value() di cio' che ritorna 'operatore [] ……