UNIVERSITÀ PARTHENOPE
DiST
Dipartimento di Scienze e Tecnologie

Programmazione II e Lab Prog II

Classi concrete

Angelo Ciaramella

Classe concreta

- Classe concreta
  - la sua rappresentazione è parte della sua definizione
  - classe che permette di istanziare oggetti
  - da distinguere dalle classi astratte
  - da distinguere dalle classi derivate all'interno di gerarchie

class Date

```
namespace Chrono {

  enum class Month { jan=1, feb, mar, apr, may, jun, jul,
  aug, sep, oct, nov, dec };

class Date {
public: // public interface:
  class Bad_date { }; // classe eccezione

  explicit Date(int dd ={}, Month mm ={}, int yy ={});
  // {} significa ``scegli un valore di default''

  . . .
```

Esempio namespace Chrono e classe Date concreta

Class Date

```
// membri per esaminare la data:
  int day() const;
  Month month() const;
  int year() const;
  string string_rep() const;

  void char_rep(char s[], int max) const;
  // rapp. stringhe stile C

// membri per cambiare la data:
  Date& add_year(int n); // aggiunge n anni
  Date& add_month(int n); // aggiunge n mesi
  Date& add_day(int n); // aggiunge n giorni

private:
  bool is_valid(); // varifica se Date rapp. una data
  int d, m, y; // reppresentazione
};
. . .
```

Esempio namespace Chrono e classe Date concreta

Class Date

```
  bool is_date(int d, Month m, int y);
  // true per data valida
  bool is_leapyear(int y); // true per anno bisestile

  bool operator==(const Date& a, const Date& b);
  bool operator!=(const Date& a, const Date& b);
  const Date& default_date(); // data di default

  ostream& operator<<(ostream& os, const Date& d);
  // stampa d su os
  istream& operator>>(istream& is, Date& d);
 // legge Date da is a d

} // Chrono
```

Esempio namespace Chrono e classe Date concreta

Costruttore explicit

- **costruttore** `explicit`
  - Il costruttore può essere usato soltanto per l'inizializzazione e le conversioni esplicite
  - non viene usato come conversione implicita

```
void my_fct(Date d);

void f()
{
  Date d {15}; // plausibile: {15,today.m,today.y}
  // ...
  my_fct(15); // oscuro
  d = 15; //oscuro
  // ...
}
```

Esempio di utilizzo di un costruttore non explicit

Costruttore explicit

```
struct A {
   A(int) { } // converting constructor
   A(int, int) { } // converting constructor (C++11)
};

struct B {
explicit B(int) { }
explicit B(int, int) { }
};

int main() {
A a1 = 1;
A a2(2);

B b1 = 1;
B b2(2);
b3 {4, 5};

}
```


Inizializzatori

```
class Date {
        int d, m, y;

public:
        Date(int, int, int);
        Date(int, int);
        Date(int);
        Date();
        Date(const char*);

// ...
};

Date::Date(int dd, int mm, int yy)
:d{dd}, m{mm}, y{yy}
{
// check that the Date is valid
}
```

Esempio di utilizzo di inizializzazioni

inline

```
class Date {
public:

void add_month(int n) { m+=n; } // inline

private:
 int d, m, y;
};
```

```
class Date {
public:

void add_month(int n); // non inline

private:
 int d, m, y;
};

inline void Date::add_month(int n) // metodo inline
{
 m+=n;
}
```


Esempio di utilizzo di metodi inline

Funzioni membro costanti

```
class Date {
        int d, m, y;
public:
        int day() const { return d; }
        int month() const { return m; }
        int year() const;
        void add_year(int n); // aggiunge n anni
// ...
};
```

Funzioni che non modificano lo stato di `Date`

Autoriferimenti

```
class Date {
        // ...
public:
       Date& add_year(int n); // aggiunge n anni
       Date& add_month(int n); // aggiunge n mesi
       Date& add_day(int n); // aggiunge n giorni
// ...
};

Date& Date::add_year(int n)
{
  if (d==29 && m==2 && !leapyear(y+n)) {
  d = 1;
  m = 3;
}
  y += n;
return *this;
}
```

`*this` si riferisce all'oggetto per il quale viene invocata una funzione membro

```
void f(Date& d)
{
// ...
d.add_day(1).add_month(1).add_year(1);
// ...
}
```


Autoriferimenti

```
Date& Date::add_year(int n)
{
  if (this->d==29 && this->m==2 && !leapyear(this->y+n)) {
  this->d = 1;
  this->m = 3;
}
  this->y += n;

return *this;
}
```

Uso implicito di `this`

Esercizio

```
Date& Date::add_year(int d)
{
  if (d==29 && m==2 && !leapyear(y+d)) {
  d = 1;
  m = 3;
}
  y += d;

return *this;
}
```

Risolvere il conflitto dei nomi del codice