



CLASSI ED OGGETTI

Object Oriented Programming

Roberto Nardone, Luigi Romano

- ☐ Principi di Programmazione ad Oggetti
- ☐ Classi
 - Specificatori di accesso
- ☐ Oggetti
- ☐ Metodi e Attributi
- ☐ Costruttori e Distruttori
- ☐ Puntatori ad oggetti e puntatore THIS
- ☐ Esempio C++

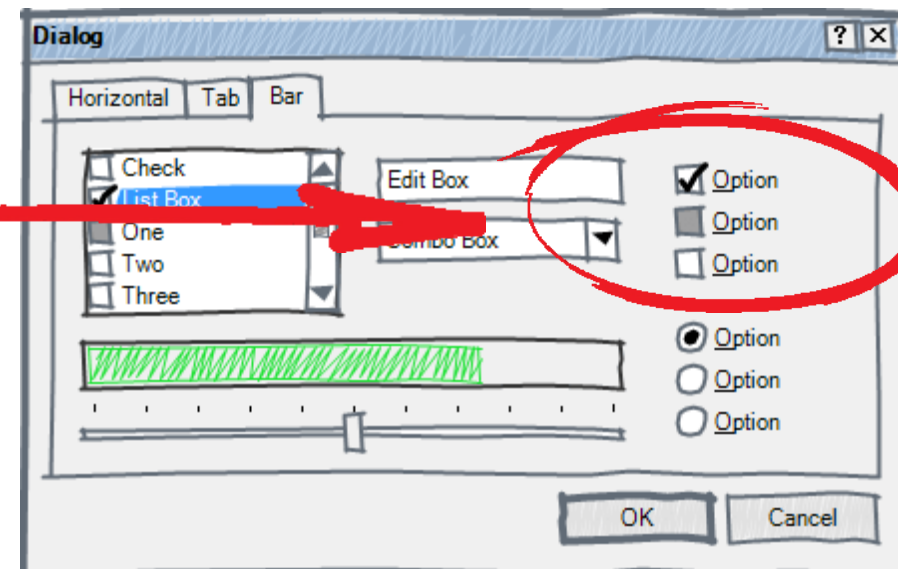


- Ci siamo concentrati sulle «funzioni» da svolgere per risolvere un problema
- Punto di vista alternativo: **da cosa è composto il programma che vogliamo realizzare?** Quali sono i «mattoncini» con cui costruiamo questo programma? E come questi mattoncini sono «incastrati» tra di loro? Sono pezzi atomici o composti da altri mattoncini più semplici?



□ Una finestra

- Tre tab
- Un elenco
- Due inputBox
- **Tre Chexbox**
 - Checkbox 1)
 - Bianca e spuntata
 - Checkbox 2)
 - Grigia e senza spunta
 - Checkbox 2)
 - Bianca e senza spunta



- Una **checkbox**
 - Ha un nome
 - Uno stato (colore, stato spunta)
 - Delle operazioni (aggiungi/togli spunta)
- Chi usa la checkbox ignora come sia fatta se non per questi aspetti
- Nella finestra sono stati aggiunti tre **oggetti** di **classe** (tipo) **CheckBox**



- Definiscono una classe di oggetti in termini di
 - Stato
 - Operazioni
- Sono tipi di dato astratto
Consentono di separare
 - Definizione
 - elenco di «attributi» che ne definisce lo stato e operazioni applicabili sugli oggetti della classe
 - Implementazione

Nome Classe: CheckBox

Attributi:

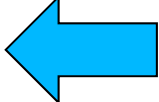
- Colore
- Spunta
- Posizione{x,y}

Metodi:

- aggiungiSpunta()
- toglispunta()



```
class NomeClasse
{
    // Lista Attributi
    // Lista Metodi
};
```



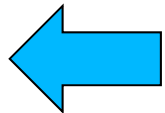
```
class NomeClasse
```

```
{
```

```
    // Lista attributi
```

```
    // Lista
```

```
};
```



```
class Persona
```

```
{ // attributi
```

```
    string nome;
```

```
    string cognome;
```

```
    int eta;
```

```
    //metodi
```

```
    void invecchia(); //eta++
```

```
    void presentati(); //Ciao mi
```

```
        //chiamo nome cognome
```

```
        // ed ho eta anni
```

```
};
```



- Per default i **membri** di una classe (metodi ed attributi) sono nascosti all'esterno (**Information Hiding** o **incapsulamento**) e dunque non accessibili
 - Sono cioè: **private**
- E' possibile specificare quali membri esporre e quali no mediante gli **specificatori di accesso**:
 - **public**: indica che i membri che seguono sono visibili
 - **private**: indica che i membri che seguono NON sono visibili
- Un terzo specificatore consente di distinguere tra utilizzatori della classe e classi derivate (vedremo in seguito) :
 - **protected**: i membri protected possono essere acceduti solo da classi derivate



```
class Persona
{
    public:
        // membri(attributi/metodi)pubblici
        void invecchia(); //eta++
        void presentati(); //Ciao mi
                                //chiamo nome cognome
                                // ed ho eta anni

    private:
        // membri(attributi/metodi)privati
        string nome;
        string cognome;
        int eta;
};
```



```
class Persona
{
    public:
        // membri (accessibili, modificabili, pubblici)
        void invecchia(); //eta++
        void presentati(); //Ciao mi
                                //chiamo nome cognome
                                //età è età anni
    private:
        // membri (accessibili, modificabili, privati)
        string nome;
        string cognome;
        int eta;
};
```

Tutto ciò che segue «public:» è pubblico

Tutto ciò che segue «private:» è privato;
l'ordine degli specificatori è indifferente.



```
class Persona
```

```
{
```

```
public
```

```
//
```

```
void
```

```
void
```

Se ci fosse qualche membro prima degli
specificatori, varrebbe la regola di default:

«privato»

pubblico

cognome

```
private:
```

```
// membri (accessori, metodi, privati)
```

```
string nome;
```

```
string cognome;
```

```
int eta;
```

Tutto ciò che segue «private:» è privato;
l'ordine degli specificatori è indifferente.

```
};
```



- Una volta definita una Classe, possiamo creare (**istanziare**) infiniti oggetti (**istanze**) di quella Classe

`Nome_classe nome_oggetto;`

- Es.

`Persona p1,p2; // Istanza due oggetti della classe Persona`

- L'accesso ai membri della classe avviene grazie all'operatore (.)

- Es.

`p1.presentati(); // invoca il metodo pubblico presentati`

- La dichiarazione di una CLASSE non alloca memoria, l'istanziamento di un oggetto alloca la memoria necessaria per tutti i suoi attributi
 - Di conseguenza non è possibile inizializzare attributi all'atto della dichiarazione della CLASSE (non sono stati ancora riservati spazi in memoria)



METODI (FUNZIONE MEMBRO) DI UNA CLASSE

Possono essere definiti insieme alla dichiarazione della classe o separatamente

```
class Persona {  
    string nome;  
    string cognome;  
    int eta;  
public:  
    void presentati(){  
        cout << "Ciao sono "  
        << nome << " " << cognome  
        << " di anni " << eta <<  
        endl;  
    }  
    void invecchia(){...}  
};
```

```
class Persona {  
    string nome;  
    string cognome;  
    int eta;  
public:  
    void presentati();  
    void invecchia();  
};  
void Persona::presentati(){  
    cout << "Ciao sono " <<  
    nome << " " << cognome <<  
    " di anni " << eta << endl;  
}  
void Persona::invecchia(){...}
```



METODI (FUNZIONE MEMBRO) DI UNA CLASSE

Possono essere definiti insieme alla dichiarazione della classe o separatamente

```
class Persona {
    string nome;
    string cognome;
    int eta;
public:
    void presentati(){
        cout << "Ciao sono "
              << nome << " " << cognome
              << " di anni " << eta <<
              endl;
    }
    void invecchia(){...}
};
```

Funzioni in linea

```
class Persona {
    string nome;
    string cognome;
    int eta;
public:
    void presentati();
    void invecchia();
};

void Persona::presentati(){
    cout << "Ciao sono " <<
         nome << " " << cognome <<
         " di anni " << eta << endl;
}

void Persona::invecchia(){...}
```

Funzioni fuori linea



- Sono metodi invocati automaticamente quando viene istanziato un oggetto
- Hanno il nome che coincide con il nome della classe e non hanno argomento di ritorno
- Hanno lo scopo di inizializzare lo stato dell'oggetto
- Es.

```
class Persona {  
    ...  
    public:  
        Persona() { // costruttore di Default  
            nome="Giuseppe"; cognome="Verdi";  
            eta=40; } //inline ma può essere fuori linea  
};
```

- Se non è definito un costruttore di Default, il compilatore ne crea uno vuoto



COSTRUTTORI CON ARGOMENTI

- Il costruttore di Default è un costruttore privo di argomenti
- E' possibile «sovraccaricare» il costruttore definendo altri costruttori con argomenti come per una normale funzione
- Es.

```
class Persona {  
    ...  
    public:  
        Persona() { // costruttore di Default  
            nome="Giuseppe"; cognome="Verdi";  
            eta=40; }  
        Persona(string pnome, string pcognome,  
            int peta) {  
            nome=pnome;  
            cognome=pcognome;  
            eta=peta; }  
};
```



- Creare la Classe Persona, istanziare un oggetto della classe assegnare un nome, un cognome ed un'età. Usare il metodo invecchia e quello presentati.

IMPLEMENTAZIONE CON STRINGHE C-STYLE

19

```
class Persona {
    char *nome;
    char *cognome;
    int eta;

public:
    Persona() { // costruttore di Default
        nome =
            new char(strlen("Giuseppe") + 1);
        strcpy(nome, "Giuseppe");
        cognome =
            new char(strlen("Verdi") + 1);
        strcpy(cognome, "Verdi");
        eta=40;
    } //inline ma puo' essere fuori linea

    void presentati();

    void invecchia() {
        eta++;
    }
};

void Persona::presentati() { //fuori linea
    cout <<"Ciao sono "<<nome<<
        " "<<cognome<<", di anni "<<eta<<endl;
}
```

```
int main() {

    Persona p;
    p.presentati();

    p.invecchia();

    cout<<"Dopo aver invecchiato p
        (inizializzato per default)...\n";
    p.presentati();

    return 0;
}
```



IMPLEMENTAZIONE CON STRING

20

```
class Persona {
    string nome;
    string cognome;
    int eta;

public:
    Persona() { // costruttore di Default
        nome = "Giuseppe";
        cognome = "Verdi";
        eta=40;
    } //inline ma puo' essere fuori linea

    void presentati();

    void invecchia() {
        eta++;
    }
};

void Persona::presentati() { //fuori linea
    cout <<"Ciao sono "<<nome<<"
    "<<cognome<<"", di anni "<<eta<<endl;
}
```

```
int main() {

    Persona p;
    p.presentati();

    p.invecchia();

    cout<<"Dopo aver invecchiato p
        (inizializzato per default)...\n";
    p.presentati();

    return 0;
}
```



```
class Persona {
    string nome;
    string cognome;
    int eta;

public:

    void presentati();

    void invecchia() {
        eta++;
    }
};

void Persona::presentati() { //fuori linea
    cout <<"Ciao sono "<<nome<<"
    "<<cognome<<" , di anni "<<eta<<endl;
}
```

```
int main() {

    Persona p;

    p.nome = "Giuseppe";
    p.cognome = "Verdi";
    p.eta=40;
    p.presentati();

    p.invecchia();

    cout<<"Dopo aver invecchiato p...\n";
    p.presentati();

    return 0;
}
```



```
class Persona {
    string nome;
    string cognome;
    int eta;

public:

    void presentati();

    void invecchia() {
        eta++;
    }
};

void Persona::presentati() { //fuori linea
    cout <<"Ciao sono "<<nome<<"
    "<<cognome<<" , di anni "<<eta<<endl;
}
```

```
int main() {

    Persona p;

    p.nome = "Giuseppe";
    p.cognome = "Verdi";
    p.eta=40;
```

Gli attributi sono tenuti privati per «information hiding» per cui non è possibile accedervi direttamente da fuori la classe, mentre si può fare da dentro la classe. Come risolvere?

- 1) Costruttori che consentono l'inizializzazione degli attributi alla creazione dell'oggetto
- 2) Metodi pubblici per inizializzare i valori



IMPLEMENTAZIONE CON STRING E METODI SET

23

```
class Persona {
    string nome;
    string cognome;
    int eta;

public:

    void setName(string pname) {
        nome = pname;
    }

    void setCognome(string pcognome) {
        cognome= pcognome;
    }

    void setEta(int peta) {
        eta = peta;
    }

    void presentati();

    void invecchia() {
        eta++;
    }
};

void Persona::presentati() { //fuori linea
    cout <<"Ciao sono "<<nome<<"
    "<<cognome<<", di anni "<<eta<<endl;
}

int main() {

    Persona p;

    p.setName("Giuseppe");
    p.setCognome("Verdi");
    p.setEta(40);
    p.presentati();

    p.invecchia();

    cout<<"Dopo aver invecchiato p...\n";
    p.presentati();

    return 0;
}
```

