(Per una visualizzazione ottimale dei colori attivare il tema scuro)

In **C++**

La struttura tipica di una classe è

class NomeClasse{

private:

…variabili/metodi

protected:

…variabili/metodi

public:

…variabili/metodi

};

Private, Protected e Public sono detti “modificatori”

* Non è possibile accedere a una variabile o a una funzione membro Private, né visualizzarla dall'esterno della classe. Solo la classe e le funzioni friend possono accedere ai membri privati.
* Una variabile o funzione membro Protected è molto simile a un membro privato, ma offre un vantaggio aggiuntivo: è possibile accedervi nelle classi figlie, chiamate classi derivate.
* Un membro Public è accessibile da qualsiasi punto al di fuori della classe, ma all'interno di un programma.

**Modificatore FRIEND**

Una classe friend può accedere ai membri privati e protetti di altre classi in cui è dichiarata come amica.

class Leone{

private:

int Variabile\_Privata;

protected:

int Variabile\_Protetta;

public:

Leone (){

Variabile\_Privata = 10;

Variabile\_Protetta = 99;

}

friend class Patata;

};

class Patata{

public:

void Stampa (Leone){

cout << Variabile\_Privata<<” ”;

Variabile\_Protetta = 99;

}

};

int main(){

Leone leo;

Patata pat;

pat.Stampa(leo); // 10 99

return 0;

}

**Modificatore VIRTUAL**

Il compito delle funzioni virtual è assicurarsi che venga richiamata la funzione corretta per un oggetto.

Le funzioni virtual sono sempre definite nella classe base (*aggiungendo la keyword virtual subito prima di specificare il type del metodo*) e sovrascritte in una classe derivata. Non è obbligatorio per la classe derivata sovrascrivere (override) la funzione virtuale; in tal caso, viene utilizzata la versione della classe base della funzione.

*Esempio*

class Leone{

public:

virtual void Stampa (){

cout << “Stampa classe Leone”;

}

};

class Patata: public Leone{

public:

void Stampa (){

cout << “Stampa classe Patata”;

}

};

int main(){

Leone \*leo;

Patata pat;

leo = &pat

leo-> Stampa(); // “Stampa classe Patata”

return 0;

}