Programmazione Avanzata

struct vs class

Pierluigi Roberti

pierluigi.roberti@unitn.it

Struct

Tipo di dati composito personalizzato, che può essere costruito dai tipi di dati incorporati esistenti [**int** , **char** , ecc.], campi di bit [interi di bit specificato size] e altri **struct** s. Questo esempio mostra un semplice esempio di definizione di **struct** , insieme alla dichiarazione di una variabile di quel tipo e all'accesso a uno dei campi. Uno **struct** è un modo pratico e flessibile per rappresentare i dati. Strutture simili esistono nella maggior parte dei linguaggi di programmazione moderni [1].

```
struct mystruct
{
    void fun();
    myclass();
    ~myclass();
private:
    char  a;
    int   b;
};

mystruct myobj;

myobj.fun();
```

Può anche contenere funzioni [chiamate metodi], costruttori e un distruttore.

[1] The real difference between struct and class - Fluent C++ (fluentcpp.com)

Class

- Le variabili membro ei metodi sono nascosti al mondo esterno, a meno che la loro dichiarazione non segua un'etichetta pubblica .
- Possono esistere una coppia di metodi speciali, il costruttore e il distruttore, che vengono eseguiti automaticamente quando un'istanza della classe [un oggetto] viene creata e distrutta.
- Gli operatori per lavorare sul nuovo tipo di dati possono essere definiti utilizzando metodi speciali [funzioni membro].
- Una classe può essere utilizzata come base per la definizione di un'altra [ereditarietà].
- Dichiarazione di una variabile del nuovo tipo [un'istanza della classe; un oggetto] richiede solo il nome della classe – la parola chiave class non è richiesta.

```
class myclass
{
    char  a;
    int   b;
public:
    void fun();
    myclass();
    ~myclass();
};

myclass myobj;

myobj.fun();
```

Struct vs Class

- In termini di lingua, tranne un piccolo dettaglio, non c'è differenza tra struct e class. Contrariamente a quanto credono inizialmente gli sviluppatori più giovani, o le persone provenienti da C, una struct può avere costruttori, metodi (anche virtuali), membri pubblici, privati e protetti, utilizzare l'ereditarietà, essere modellato... proprio come un class.

- L'unica **differenza** è che se non specifichi la visibilità (pubblica, privata o protetta) dei membri, questi saranno pubblici nel struct file class. E la visibilità per impostazione predefinita va solo un po' oltre i membri: per l'ereditarietà se non specifichi nulla struct , erediterà pubblicamente dalla sua classe base.

Struct vs Class

- La differenza che conta davvero tra struct e class si riduce a una cosa: le **convenzioni** . Ci sono alcune convenzioni là fuori che sono abbastanza diffuse e che seguono una certa logica. Seguire queste convenzioni ti dà un modo per esprimere le tue intenzioni nel codice durante la progettazione di un tipo, perché come vedremo tra poco, implementarlo come a struct non trasmette lo stesso messaggio dell'implementarlo come class.

- Contrariamente a una struct, una classe è fatta per offrire un'interfaccia, che ha un certo grado di separazione dalla sua implementazione. La class non è solo lì per memorizzare i dati. In effetti, un utente di una classe non dovrebbe sapere quali dati sta memorizzando la classe o se contiene dati per quella materia. Tutto ciò che gli interessa sono le sue responsabilità, espresse attraverso la sua interfaccia.

Struct o Class

Una semplice regola pratica per la scelta tra struct o class è quella da **adottare class** ogni volta che nella struttura è presente almeno un **membro privato**.
In effetti, questo suggerisce che ci sono dettagli di implementazione che devono essere nascosti da un'interfaccia, che è lo scopo di una classe.

Una classe ha delle responsabilità. Queste responsabilità possono essere abbastanza semplici, come il recupero di dati che la classe potrebbe anche contenere se stessa. Per questo motivo, si desidera utilizzare il termine class quando si modella un concetto (che esiste o meno nel dominio aziendale), il concetto di un oggetto in grado di eseguire azioni.