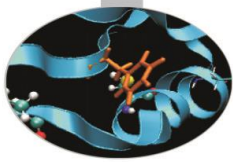
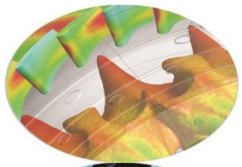
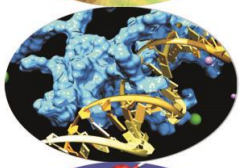
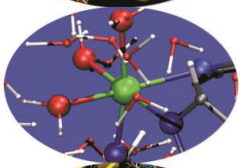
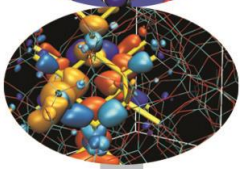
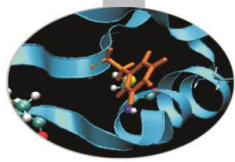


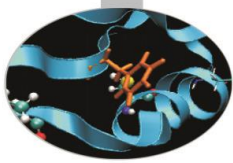
Introduzione

Informazioni generali sul corso

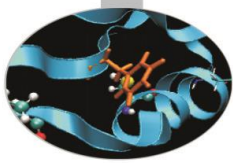


- Il corso si propone di illustrare la sintassi di base del C++, inteso qui come C migliorato, e le strutture tipiche della programmazione procedurale così come sono implementate in C ed in C++.
- All'interno del corso le differenze salienti tra questi due linguaggi verranno messe in evidenza.
- E' rivolto a principianti nella programmazione.
- Permetterà di scrivere semplici programmi in C++ e di capire il contenuto di codici più articolati.
- Non si parlerà di oggetti/classi etc.



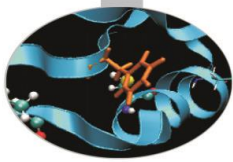
Programma generale

- Introduzione: i paradigmi di programmazione
- Sintassi di base del linguaggio C/C++
- La scrittura di makefile
- Le funzioni
- I tipi di dato strutturato
- Le librerie standard del C
- La gestione dei file in C



C++ come C migliorato

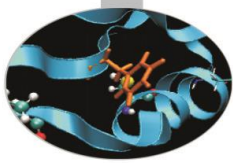
- C e C++ sono tra i linguaggi più diffusi nel mondo della programmazione
- Il linguaggio C nasce negli anni ’70 ad opera di D. Ritchie e si propone come linguaggio “middle-level” che integra il controllo sulle strutture tipico dei linguaggi “high-level” con la possibilità di manipolare bits, bytes e indirizzi tipico dei linguaggi “low-level”.
- Nel 1989 la American National Standard Institute (ANSI) standardizza il C (C89).
- Nel 1999 la International Standards Organisation crea un ulteriore standard (C99) che differisce dal C89.
- Nel 2011 la International Standards Organisation ufficializza lo standard attuale (C11) che aggiunge ulteriori funzionalità al C99.



C++ come C migliorato

- Nel 1979 B. Stroustroup inventa il C++ come estensione ad oggetti del linguaggio C.
- Nel 1998, dopo un lavoro ventennale, viene creato lo standard ANSI/ISO del C++
- Il C++ è stato originariamente pensato come sovrainsieme (grande circa il doppio) del C89 ma non del C99.
- **Lo standard attuale è il C++11 anche se sono già in fase di sviluppo i futuri standard C++14 e C++17.**

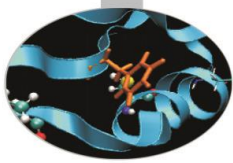
C89	**ANSI/ISO standard per il C; di fatto quando si parla di C si intende questo**
C++	**Versione ad oggetti del C89**
C99	Nuova versione che contiene tutte le innovazioni del C89 più altre che però non sono supportate dal C++



C/C++ commenti

In questo corso si farà uso di questi linguaggi per un tipo di programmazione detta procedurale. A questo livello una differenza sostanziale tra i due linguaggi non esiste per quanto l'entusiasmo che accompagna il C++ fa in generale pensare che quest'ultimo sia preferibile al C in quanto:

- Permette di sviluppare programmi nuovi in minor tempo grazie al riutilizzo del codice
- Permette di creare e gestire nuovi tipi di dato in maniera più semplice
- Permette un uso più trasparente della memoria
- I programmi scritti sono meno bugs-prone grazie alla sintassi e al controllo sui tipi di dato
- L'information hiding è semplificata e garantita da solidi strumenti (classe)

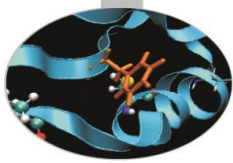


C/C++ commenti

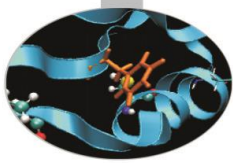
Le considerazioni sopramenzionate sono in parte vere ed in parte sono frutto dell'entusiasmo che accompagna la OOP (Object Oriented Programming) in generale.

Quello che certamente è vero è che:

- Il riutilizzo del codice è sempre possibile anche in C
- Nuovi tipi di dato sono allocabili e gestibili anche in C per quanto con meno strumenti e potenzialità
- L'accesso alla memoria è equivalente in C
- Il compilatore C++ è più stringente rispetto ai compilatori C, per quanto ad oggi siano comunque presenti dei *warning* equivalenti nei compilatori C
- L'information hiding è possibile anche in C anche se in maniera meno naturale



Paradigmi di programmazione commenti



Introduzione

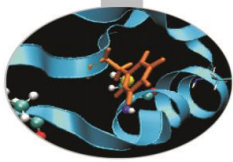
- Procedurale.
- Modulare o strutturato.
- Data abstraction.
- Object oriented.
- Generico.

Il C++ supporta questi paradigmi di programmazione nel senso che esistono degli strumenti che permettono di scrivere programmi secondo i concetti base che definiscono questi tipi.

Tuttavia nessuno di questi è a priori auspicabile o forzatamente preferibile rispetto agli altri in C++.

Il C supporta in maniera naturale la programmazione procedurale, quella modulare e l'astrazione dei dati anche se in maniera meno forte del C++.

Programmazione Procedurale



Decide which procedures you want; use the best algorithms you can find.

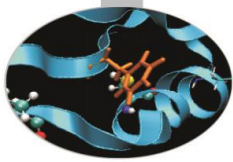
Il linguaggio supporta questo tipo di programmazione fornendo la possibilità di costruire procedure, routines e funzioni.

Un tipico esempio di questo è la creazione di una funzione che risolva un particolare task matematico.(esempio: sqr(); invert();…)

Strumenti forniti dal linguaggio:

- Procedure.
- Test e cicli.
- Librerie di funzioni.

Questo corso tratterà in maniera completa gli aspetti del linguaggio C/C++ che permettono di scrivere codice secondo questo paradigma.



Esempio: stack procedurale

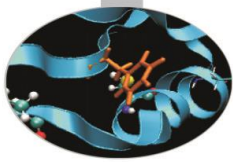
- È possibile implementare uno stack procedurale tramite un array e un indice all'ultimo dato.

- Possono essere implementate delle procedure apposite per manipolare la pila (push( ), pop( ) ed empty( )).

- Rimangono essenzialmente due grandi limiti:
  1. I dati contenuti nella pila sono accessibili e modificabili anche fuori dall'interfaccia delle funzioni scritte.
  2. Per più stack e per tipi di dati diversi servono più repliche di codice ad hoc.

Di seguito tramite questo esempio, si vedrà come i diversi paradigmi di programmazione permettono di affrontare questi problemi.

Programmazione modulare o strutturata

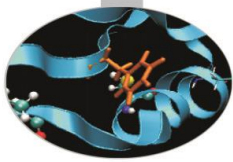


Decide which modules you want; partition the program so that data is hidden within modules. (data-hiding principle)

Riflettendo lo sviluppo storico di nuove necessità nella scrittura di codici di maggiori dimensioni e che trattassero sempre più dati, l'enfasi è stata posta sul design del programma più che sulle singole procedure.

Il C++ supporta questa programmazione permettendo di definire e compilare separatamente una interfaccia, una implementazione dell'interfaccia e un codice di utilizzo dell'interfaccia stesso. In linea di principio l'utilizzatore non è tenuto a conoscere i dettagli implementativi delle funzionalità che sono offerte dall'interfaccia.

Programmazione modulare o strutturata(2)

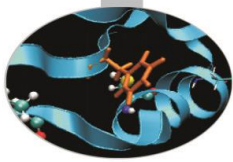


Tornando all'esempio dello stack si potrebbe avere un file *stack.h (interfaccia o file dichiarativo)*, uno *stack.c (implementazione dell'interfaccia o file di definizione)* ed uno *user.c (file di utilizzo della struttura stack).*In questo modo i metodi di accesso ai dati sono forniti dalle funzioni descritte in stack.h.

I limiti che persistono sono:

- Non c'è ancora un modo naturale che limiti l'accesso diretto ai dati nè ad una loro modifica incontrollata.
- Per più stack e per tipi di dati diversi questa struttura va ridefinita ex novo

Programmazione modulare o strutturata(3)



Stack.h

Interfaccia o file di descrizione delle funzioni d'accesso (**push ( ), pop ( ), empty ( )**).

Stack.c

Implementazione dell' interfaccia.

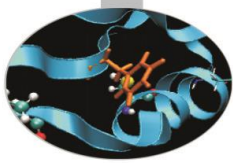
Questo file contiene le definizioni (implementazioni) delle funzioni dichiarate nel file stack.h.

User.c

inclusione di stack.h

In questo file sarà contenuto il main e le chiamate alle funzioni di interfaccia per gestire una pila.

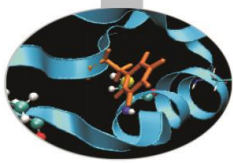
Data abstraction



Decide which types you want; provide a full set of operations for each type.

Questo tipo di programmazione, essendo di un ordine di complessità maggiore, risulta inutile per un unico oggetto di un solo tipo.

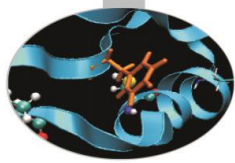
- Consiste nel definire nuovi TIPI (*abstract data* o *user-defined-type)* aventi un loro comportamento e stato dipendente in parte da come vengono definite le funzioni d'interfaccia e in parte da come abbiamo presentato il tipo rappresentativo.



Data abstraction (2)

- Tornando all'esempio dello stack , con questo paradigma di programmazione verrà creato il nuovo tipo di dato stack.
- Utilizzando la definizione di un nuovo tipo di dato *stack,* inteso come dati ed operazioni per manipolare gli stessi, si attua automaticamente e in maniera "pulita" l'incapsulamento dei dati.
- Il grosso limite di questo strumento, peraltro basilare per una buona programmazione ed un buona strutturazione del programma, è la potenziale mancanza di flessibilità. La black box che viene definita non interagisce veramente con il resto del codice e non c'è modo di adattarla per nuovi usi se non modificandone la definizione stessa.

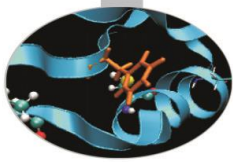
Programmazione ad Oggetti



Decide which classes you want; provide a full set of operations for each class; make commonality explicit by using inheritance.

- La programmazione ad oggetti nasce dalla volontà di distinguere i tratti e le proprietà che caratterizzano un problema e sintetizzarlo in un particolare nuovo tipo di dato detto **classe** traendone vantaggio da questa caratterizzazione.
- Non esistono le classi in C.
- Esistono e ne sono anzi la caratteristica principale in C++.
- In questo corso non verranno trattati altri aspetti riguardo alle classi.

Programmazione Generica



Decide which algorithms you want; parametrize them so that they work for a variety of suitable types and data structures.

Per ampliare ulteriormente la generalità dei concetti espressi dal codice e permetterne quindi un più ampio riutilizzo modificandone il meno possibile, viene fornito come strumento la programmazione generica (*templates).*

Il concetto generale è che se un algoritmo (es. ordinamento) può essere espresso indipendentemente dai dettagli rappresentativi e questo può essere fatto con naturalezza, allora può essere scritto così.

Esempio: pila fatta non da caratteri o da interi ma da elementi qualsiasi. Algoritmi di ordinamento……