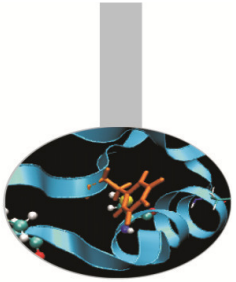
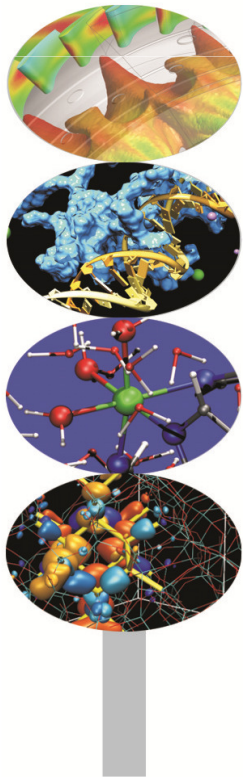
CINECA SCAI
SuperComputing Applications and Innovation

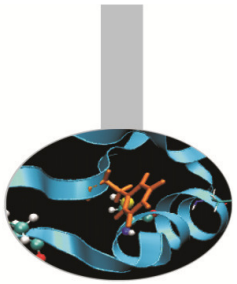


Introduzione



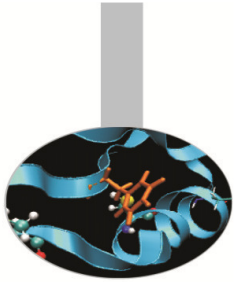
CINECA

Informazioni generali sul corso



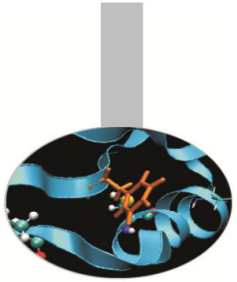
- Il corso si propone di illustrare la sintassi del C con un'insistenza maggiore sulle strutture, le potenzialità, i limiti e le utilities messe a disposizione dal linguaggio per il calcolo tecnico-scientifico.
- E' rivolto a principianti nella programmazione.
- Permetterà di scrivere semplici programmi in C e di capire il contenuto di codici più articolati.
- Vi saranno alcuni rimandi-esempi e paragoni con il C++ sempre inteso come programmazione procedurale
- Non si parlerà di oggetti/classi e altri strumenti tipici del C++ e della programmazione orientata agli oggetti.

Programma generale



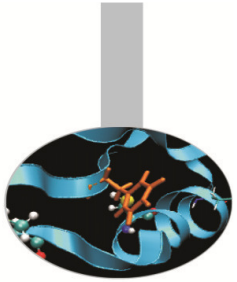
- Introduzione: breve storia del linguaggio – tipo di programmazione
- Sintassi di base del linguaggio
- La scrittura di makefile
- Le funzioni
- I tipi di dato strutturato
- Le librerie standard del C
- La gestione dei file in C

Breve storia del linguaggio



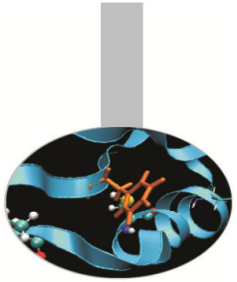
- Il C è uno tra i linguaggi più diffusi nel mondo della programmazione non solo tecnico-scientifica.
- Il linguaggio C nasce negli anni ’70 ad opera di Dennis Ritchie e si propone come linguaggio “middle-level” che integra il controllo sulle strutture tipico dei linguaggi “high-level” con la possibilità di manipolare bits, bytes e indirizzi tipico dei linguaggi “low-level”.
- Nel 1989 la American National Standard Institute (ANSI) standardizza il C (C89).

Breve storia del linguaggio



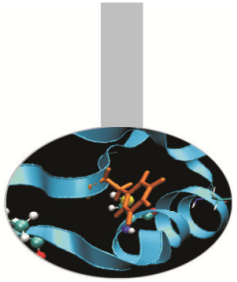
- Nel 1999 la International Standards Organisation crea un ulteriore standard (C99) che differisce dal C89 per l'introduzione di nuovi tipi numerici (long long – bool), nuove funzioni matematiche, più caratteri, inlinig di funzioni, restrict pointers ecc...
- Nel 2011 la International Standards Organisation ufficializza lo standard attuale (C11) che aggiunge ulteriori funzionalità al C99 (maggiore flessibilità, threads, caratteri Unicode) .

Filosofia generale del C



- Linguaggio semplice ed efficiente
- Solamente 44 parole riservate
- Tipi di dato di base che mappano naturalmente nella CPU
- Facilities per creare nuovi tipi di dato a partire da quelli built-in
- Strutture di controllo di flusso flessibili
- Linguaggio compilato
- Una ricca libreria standard di funzioni (funzioni matematiche, gestione della memoria, manipolazione di stringhe)
- Preprocessore per compilazione condizionale e manipolazione del sorgente prima della compilazione

C – Programmazione procedurale



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

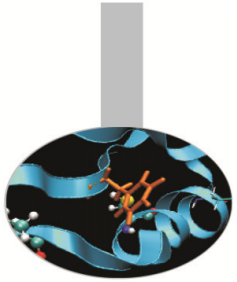
Il linguaggio supporta questo tipo di programmazione fornendo la possibilità di costruire funzioni e librerie di funzioni.

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 che permettono di scrivere codice secondo questo paradigma.



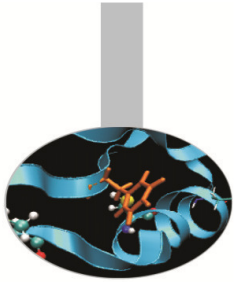
C – Programmazione modulare

Questo tipo di programmazione è particolarmente diffuso in ambito tecnico-scientifico

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.

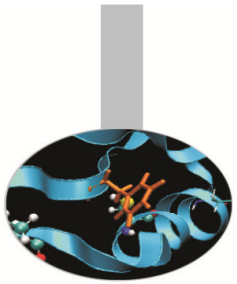


C – Programmazione modulare

Come semplice esempio si supponga di dover implementare un programma che definisca una pila di dati o stack e le operazioni che si possono effettuare sullo stack (push(), pop(), empty(), full(), ecc...).

Si potrebbe modularizzare il problema in questo modo:

- un file *stack.h (interfaccia o file dichiarativo)* che contenga i prototipi delle funzioni nelle quali sono definite le operazioni
- Un file *stack.c (implementazione dell'interfaccia o file di definizione)* che contenga le funzioni nelle quali sono definite le operazioni
- un file *main_stack.c (file di utilizzo della struttura stack)* che faccia uso dello stack e delle sue funzionalità. In questo modo i metodi di accesso ai dati sono forniti dalle funzioni descritte in stack.h.



C – Programmazione modulare

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.

Main_stack.c

Inclusione di stack.h

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