Corso Programmazione 2008-2009

(docente)

Fabio Aioli

E-mail: aioli@math.unipd.it

Web: www.math.unipd.it/~aioli

(docenti laboratorio)

A. Ceccato, F. Di Palma, M. Gelain

Dipartimento di Matematica Pura ed Applicata
Torre Archimede, Via Trieste 63

Orario delle lezioni e esercitazioni

- ~36 ore di lezioni in aula
 - Martedì
 - Ore 11:30 - 13:15
 - Venerdì
 - Ore 10:30 - 12:15
- ~32 ore di esercitazioni in laboratorio
 - Solitamente.. Martedì, Mercoledì, ore 14:00 - 17:00
 - Aula informatica C dell' ex Dipartimento di Matematica Pura ed Applicata (Paolotti)
 - Prima esercitazione in laboratorio il 16/04/09

Risorse per il corso

A. Kelley, I. Pohl. "C Didattica e programmazione", IV edizione, Pearson, 2004.

Altri libri consigliati, slide ed esercizi saranno disponibili sul **sito web** del corso:

<http://www.math.unipd.it/~aiolli/corsi/0809/prgxmat/prg.html>

Il **GOOGLE GROUP** del corso:

<http://groups.google.it/group/prxmat09atunipd>

Il **GOOGLE GROUP** del corso:

<http://groups.google.it/group/prxmat09atunipd>

ISTRUZIONI x GLI STUDENTI

- **Nickname:** <Nome><InizialeCognome> (per esempio, FabioA)
- **Inserire indirizzo di posta elettronica**

Esame Scritto

Prima Parte

- Domande riguardanti la sintassi del linguaggio C e semplici programmi

Parte Pratica

- Analisi e implementazione di algoritmi in C

Colloquio (per esame da 8 crediti)

- Contenuti di Introduzione alla Programmazione

Contenuti del corso

- Panoramica sul C
- Strutture dati ed algoritmi
- Programmi x il calcolo scientifico

Molto Importante

Imparare a programmare NON e' facile e
NON si impara sui libri ma ci vuole MOLTA
pratica (e curiosita')

RENDETEMI IL COMPITO + FACILE

Qualsiasi tipo di interazione con lo studente
e' ben accetta da parte mia

Richiedete spesso chiarimenti e partecipate
attivamente alle lezioni e ai laboratori

Iniziamo..

PARTE 1

Definizioni Fondamentali

Algoritmo

DEFINIZIONE ALGEBRICA

- Insieme completo delle **regole** che permettono la soluzione di un determinato problema

DEFINIZIONE OPERATIVA

- Procedura **effettiva** che indica le istruzioni (passi) da eseguire per ottenere i risultati voluti a partire dai dati di cui si dispone

Algoritmo: Esempi

Esempio Culinario:

- Ricetta x cucinare gli spaghetti

Esempi sui numeri:

- Insieme di passi per verificare se un numero è dispari, pari, primo, ecc.

Altri esempi: MCD(a,b), mcm(a,b)

- Per esempio **l'algoritmo di Euclide** per il calcolo del MCD (che può essere usato anche per il calcolo del mcm!)

Algoritmo: Caratteristiche

- Esprimibile con un numero finito di istruzioni
- Istruzioni eseguibili da un elaboratore
- Insieme di istruzioni di cardinalità finita
- Tempo di esecuzione di ogni istruzione finito
- Elaboratore ha una memoria
- Calcolo per passi discreti
- Non esiste limite alla lunghezza dei dati di ingresso
- Non c'è un limite alla memoria disponibile
- Numero di passi esecuzione eventualmente illimitato

Linguaggi

LINGUAGGI NON FORMALI (ambigui)

- Linguaggio Naturale
- Linguaggio della musica
- Linguaggio del corpo
- Ecc.

LINGUAGGI FORMALI (di programmazione)

- Linguaggi NON ambigui, regolati da regole grammaticali precise
- Possono essere classificati in base al loro livello di astrazione

Linguaggi di Alto Livello (LAL)

ESEMPI FAMOSI

- Imperativi: PASCAL, FORTRAN, COBOL, C
- Ad oggetti: CPP, JAVA

Appositi software (**compilatori**) si occupano di tradurre le istruzioni scritte in questi linguaggi (cosiddetto **codice sorgente**, un file di testo), nell'equivalente codice eseguibile dalla macchina (cosiddetto **codice eseguibile**, binario)

Caratteristiche LAL strutturati

SEQUENZA

- Le istruzioni vengono eseguite in sequenza nell'ordine in cui compaiono in un *blocco* di istruzioni

SELEZIONE

- Strutture di controllo decisionali:
 - P.e. Se <espressione> esegui <BloccoIstruzioniV>
altrimenti esegui <BloccoIstruzioniF>

ITERAZIONE

- Strutture di controllo iterative:
 - P.e. Fintanto che <espressione> esegui <BloccoIstruzioni>
 - Oppure, Esegui <BloccoIstruzioni> fintanto che <espressione>

PARTE 2

Il processo di programmazione

Problemi e algoritmi

- La **programmazione** e' la disciplina che si occupa della risoluzione di problemi tramite programmi scritti in qualche **linguaggio di programmazione**
- Il **problema** e' una definizione sintetica di cio' che il programma deve realizzare dal punto di vista dell'utente
- La **soluzione** fornisce un'idea dell'approccio che deve essere seguito per risolvere il problema, tenendo anche in considerazione quale software gia' esistente potrebbe essere riutilizzato
- L'**algoritmo** e' una descrizione precisa e non ambigua di una sequenza di passi da seguire per risolvere un problema. L'algoritmo e' quindi un raffinamento della soluzione. Gli algoritmi possono essere espressi in molti modi
- Il **programma** e' una (possibile) descrizione dell'algoritmo espressa usando un particolare linguaggio di programmazione. Si intende che il programma debba essere completo ed eseguibile
- Per i banali problemi che vedremo, useremo direttamente il linguaggio C per descrivere gli algoritmi. Quindi, per noi, **algoritmi=programmi**

Il processo di programmazione

- Il **processo di programmazione** comprende tutte le attivita' necessarie per sviluppare dei programmi in modo che siano memorizzati e preparati per l'esecuzione
- Per sviluppare programmi su un computer e' necessario un ambiente di sviluppo che almeno comprenda, oltre al sistema operativo, un **editor** ed un **compilatore**

Editor e compilatori

- Un **programma** e' un testo che consiste in una sequenza di istruzioni scritte in un particolare linguaggio di programmazione. Tale "testo" viene creato e salvato come un file su disco tramite un **editor**.
- Una volta che e' stato scritto e memorizzato, il (testo del) programma viene dato in input al **compilatore**
- Il compilatore ha una duplice funzione:
 - Controlla la **validita' del programma**: segnala gli **errori di sintassi**. Questi errori devono essere corretti tramite l'editor ed il programma corretto deve quindi essere nuovamente compilato
 - Se il programma non contiene errori, il compilatore **traduce il programma** in linguaggio macchina

Esecuzione dei programmi

- Il compilatore naturalmente non e' in grado di scoprire gli **errori logici o di esecuzione** (i ben noti **bug**) del programma: in tali casi, il programma risulta essere sintatticamente corretto ma non si comporta come ci si aspetta
- Il programma viene **eseguito** (o "**fatto girare**") attraverso un comando al sistema operativo che lo carica in memoria e quindi diventa un processo in esecuzione
- Gli errori logici possono essere prevenuti seguendo delle **buone regole di programmazione**
- Spesso gli errori logici si individuano mandando in esecuzione il programma ed osservandone il comportamento tramite degli **insiemi di test affidabili**
- Una volta rilevato e corretto un errore logico, si dovrebbero nuovamente eseguire i test sul suo comportamento, in quanto le presunte "correzioni" potrebbero aver introdotto nuovi errori
- Il processo di rilevazione e correzione degli errori logici di un programma viene chiamato **debugging**