Corso di Laurea in INFORMATICA

Algoritmi e Strutture Dati

Docenti dei corsi A, B, C

F. Esposito	M. F. Costabile	O. Altamura
esposito@di.uniba.it	costabile@di.uniba.it	altamura@di.uniba.it

MODULO 1

Algoritmi e programmi

Lo sviluppo del software e la qualità dei programmi. Principi di progettazione e tecniche di programmazione. Il ruolo delle tecniche di astrazione nel progetto di programmi

Questi lucidi sono stati preparati da Oronzo Altamura, Maria Francesca Costabile e Floriana Esposito, Università degli Studi di Bari, per uso didattico. Essi contengono materiale originale di proprietà dell'Università degli Studi di Bari e/o figure di proprietà di altri autori, società e organizzazioni di cui e' riportato il riferimento. Tutto o parte del materiale può essere fotocopiato per uso personale o didattico ma non può essere distribuito per uso commerciale. Qualunque altro uso richiede una specifica autorizzazione da parte dell'Università degli Studi di Bari e degli altri autori coinvolti.

Programma preventivo

- Algoritmi e programmi
  Lo sviluppo del software e la qualità dei programmi. Principi di progettazione e tecniche di programmazione. Il ruolo delle tecniche di astrazione nel progetto di programmi: Astrazione dei dati e astrazione funzionale.
- Algebre di dati
  Dati e rappresentazioni, requisiti delle astrazioni di dati, costrutti. Astrazioni di dati e dati primitivi. Specifica sintattica e semantica tramite pre- e post-condizioni sulle operazioni. La realizzazione

Programma preventivo (cont.)

- Strutture lineari di dati
  Liste: specifiche, realizzazioni attraverso rappresentazioni sequenziali e collegate. Pile: specifiche e realizzazioni alternative, pile e procedure ricorsive. Code: specifiche e realizzazioni alternative. Valutazione della complessità degli operatori realizzati. Scelta, implementazione e verifica di algoritmi per la ricerca, l'ordinamento e la fusione delle strutture dati proposte
- Insiemi e Dizionari
  Specifiche e confronto tra realizzazioni alternative. Dizionari: realizzazioni con vettori ordinati e tabelle Hash
- Strutture non lineari di dati
  Matrici. Alberi: alberi ordinati e relative proprietà. Alberi binari: caratteristiche, la corrispondenza con le liste, specifiche e realizzazioni . Alberi n-ari: specifiche, visita di alberi, realizzazioni alternative con liste di figli e area libera. Grafi

Programma preventivo (cont.)

- Code con priorità
  Generalità. Specifiche e realizzazioni con liste, alberi binari, heap
- Alberi bilanciati di ricerca
  Alberi AVL: generalità e specifiche. Algoritmi di inserimento, cancellazione e ricerca
- Tecniche per il progetto di algoritmi

Programma preventivo (cont.)

- Laboratorio
  - Introduzione al C++
    Struttura di un programma C++, tipi di dato, espressioni ed istruzioni semplici, strutture di controllo, funzioni con e senza parametri
  - Classi
    Definizione, campi pubblici e privati, costruttori, funzioni proprie, funzioni esterne, funzioni inline, funzioni friend, funzioni operatori
  - Ereditarietà
    Classi derivate, parte pubblica e protetta, ridefinizione di funzioni e dati, ereditarietà multipla

Programma preventivo (cont.)

- Esercitazioni
  - Implementazione di strutture dati in C++
    Realizzazione di strutture lineari dinamiche e di strutture non lineari in C++: liste, pile, code, alberi, dizionari,...
  - Implementazioni di alcuni algoritmi di gestione delle strutture dati in C++
- Libri di riferimento
  - M. Cadoli, M. Lenzerini, P. Naggar, A. Schaerf "Fondamenti della progettazione dei programmi", Citta' Studi Edizioni
  - E. Lodi, G. Pacini, Introduzione alle strutture di dati, Bollati Boringhieri, 1990, cap. 3-4-5
  - A. Bertossi "Algoritmi e Strutture di Dati", UTET
  - Un libro di C++

Prerequisiti

Il corso prevede che siano già acquisiti:

- I principi della programmazione strutturata e un linguaggio di programmazione algoritmica
- La tipizzazione forte e i principi della programmazione modulare
- I principi della astrazione funzionale (funzioni e procedure) e i meccanismi del passaggio dei parametri
- La programmazione modulare (progettazione orientata alle funzioni) è di solito top-down e basata sulla decomposizione di problemi in sotto-problemi.
- Alla fine del corso si sarà in grado di applicare una metodologia di progetto alternativa passando attraverso una tecnica formale basata sulla astrazione dei dati.

Gli esami di Programmazione e di Linguaggi di Programmazione sono propedeutici

Il ciclo di sviluppo del software

- Studio di fattibilità
- Raccolta e analisi dei requisiti
- Progettazione
- Verifica
- Manutenzione

Il ciclo di sviluppo del software (cont.)

- La fase di "studio di fattibilità" ha lo scopo di valutare costi e benefici del sistema da costruire analizzando tempi e modi di attuazione valutando le alternative possibili per lo sviluppo del sistema e le risorse necessarie
- La fase di "raccolta e analisi dei requisiti" ha l'obiettivo di definire il problema, specificare l'ambiente di sviluppo (sia HW che SW), analizzare gli utenti e i compiti che essi eseguono. Il risultato e' un documento di analisi, in cui e' specificato HW e SW scelto, la struttura concettuale dei dati da elaborare, le caratteristiche generali delle operazioni da fare, le tipologie di utenti e le loro caratteristiche

Il ciclo di sviluppo del software (cont.)

- La fase di "progettazione" ha lo scopo di individuare la soluzione "informatica" del problema, di definire l'architettura del programma, di specificarne e organizzarne la struttura, di specificare le funzioni delle componenti individuate, di scegliere le strutture di rappresentazione degli oggetti e, infine, di redigere il programma secondo il linguaggio e l'ambiente operativo scelto. Il risultato della fase e' il codice.
- La fase di "verifica" ha lo scopo di analizzare tanto la documentazione prodotta che il programma realizzato, mediante prove di correttezza sintattiche e logiche (test empirici o prove formali).
- La fase di "manutenzione" ha lo scopo di controllare che il programma, durante il periodo di esercizio, produca i risultati attesi e di aggiornare, ove necessario, il programma.

Le qualità dei programmi

Le qualità dei programmi si distinguono in:

- Esterne
  - si riferiscono a caratteristiche evidenziabili dal solo funzionamento del programma durante la fase di esercizio
- Interne
  - si riferiscono alle caratteristiche analizzabili e valutabili da esperti, attraverso uno studio delle scelte tecniche adottate.
- La fase di progettazione deve fornire un'analisi delle qualità che il programma deve possedere

Le qualità esterne

- Correttezza
  - capacità del programma di eseguire precisamente i compiti utili agli utenti, individuati durante l'analisi dei requisiti.
- Efficienza
  - capacità del programma di utilizzare in modo razionale ed economico le risorse di calcolo
- Robustezza
  - capacità del programma di funzionare in modo soddisfacente in condizioni limite o anomale rispetto a quelle previste in fase di analisi dei requisiti
- Usabilità
  - capacità di consentire una interazione semplice, naturale ed efficace con l'utente finale

Le qualità interne

- Riusabilità
  capacità del programma di essere riutilizzato, in tutto o in parte, per applicazioni diverse rispetto a quella per la quale è stato prodotto
- Modularità
  grado di organizzazione interna del programma, cioè strutturazione delle singole parti, della funzionalità e del modo in cui cooperano per l'obiettivo generale
- Estendibilità
  capacità del programma di adattarsi facilmente a modifiche nei requisiti
- Portabilità e la compatibilità
  facilità di trasferire il SW prodotto in ambiti diversi
- Leggibilità
  capacità del codice di essere autoesplicante
- Bontà della documentazione
  completezza ed efficacia dei documenti annessi

La fase di progettazione

- I princìpi
  - cioè le linee guida per produrre software di qualità
- Le tecniche
  - cioè i metodi per produrre programmi coerenti con i principi
- Gli strumenti
  - come i linguaggi di programmazione o, in genere, gli aiuti di varia natura utilizzati nella progettazione

Astrazione

- Tra i principi di progettazione un ruolo fondamentale e' giocato dall'astrazione, che dà luogo a tecniche definite di astrazione (programmazione strutturata, modularizzazione, astrazione dati, etc.) che hanno segnato la programmazione negli ultimi 30 anni.
- Astrazione deriva dal latino *abstrahere* , *abs* 'da' e *trahere* 'trarre': separare mentalmente nell'oggetto dato qualche proprietà particolare per considerarla separatamente.
  [Dizionario Zingarelli]
- È un processo mediante il quale la mente prende in considerazione alcuni attributi di un oggetto ed esclude il resto
- Enfasi selettiva sui dettagli
- Esempio delle mappe

MAPPA

Astrazione (cont.)

- In un sistema software una "astrazione" è una descrizione esemplificata, o una specifica, di un sistema: pone enfasi su alcuni dettagli o proprietà mentre ne elimina altri
- Concetto chiave della programmazione moderna
- L'astrazione fornisce un buon modo di controllare la complessità e garantire la continuità (test, manutenzione, estensione) di sistemi software complessi
- La programmazione O-O è nata con l'obiettivo di dotare i linguaggi di programmazione di meccanismi espliciti per realizzare astrazione sui dati

Astrazione e modellazione analitica

- La "astrazione" nei sistemi software corrisponde alla tecnica chiamata in altri campi: modellazione analitica
- La costruzione di un modello parte dalle osservazioni che sono subito seguite dalla formulazione di ipotesi circa i principi o assiomi che spiegano le osservazioni.
- Questi assiomi sono usati per costruire il modello. Le variabili o i parametri del modello possono essere derivati dagli assiomi o stimati dalle osservazioni. Il modello e' usato per fare previsioni

Ciclo di sviluppo dei programmi

- Il passo finale di questo procedimento consiste nel fare esperimenti in ambienti controllati per validare il modello cioè determinarne la accuratezza e robustezza
- Il procedimento è ciclico: nuove osservazioni possono mettere in crisi il modello e portare alla definizione di nuove ipotesi e nuove validazioni
- Nello sviluppo del software il processo di astrazione è simile e si parla di ciclo di sviluppo dei programmi

Requisiti di un sistema SW

- I requisiti o le funzionalità di un sistema SW giocano il ruolo di osservazioni che devono essere "spiegate". Il processo di astrazione prevede il decidere:
  - Quali caratteristiche sono rilevanti
  - Quali parametri andrebbero inclusi
  - Quale formalismo descrittivo andrebbe adottato
  - Come validare il modello
- Come in molti altri campi, spesso si definiscono gerarchie di modelli nelle quali i livelli più bassi forniscono descrizioni più dettagliate di quelle che appaiono nei livelli più alti

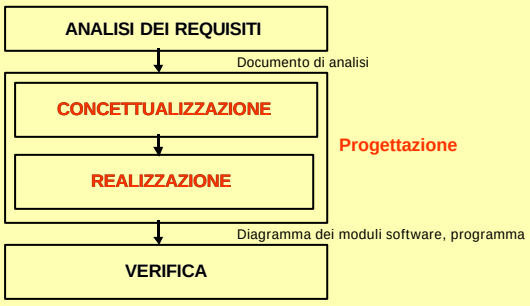
Specifica e realizzazione

- In programmazione ci riferiamo alla descrizione astratta fornita da un modello come alla specifica e al livello più basso nella gerarchia di modelli come alla realizzazione.
- Il procedimento di verificare se la realizzazione è coerente con la specifica è chiamato verifica.
- Le astrazioni usate per il software tendono ad evidenziare gli aspetti funzionali (cosa è calcolato piuttosto che come è condotto il procedimento di calcolo)

I principi di astrazione nella progettazione del sw

- Distinzione tra il livello di concettualizzazione e quello di realizzazione
- Astrarre per decomporre , rappresentare , generalizzare
- Astrarre sui dati e sulle funzioni

La fase di progettazione



Concettualizzazione

- Appartengono alla fase di **concettualizzazione** tutte le attività relative a
  - **La individuazione della soluzione al problema**
  - **La specifica degli oggetti da trattare**
  - **La ideazione delle scelte algoritmiche**
- Il risultato è lo "**schema concettuale di progetto**", ovvero la specifica di cosa deve fare il programma che ci accingiamo a costruire

Realizzazione

- La fase di **realizzazione** si riferisce a quel processo che, partendo dallo schema concettuale di progetto, permette di ottenere uno specifico programma che "realizza" gli elementi dello schema
- Questo impone:
  - la **scelta** della architettura del programma in termini di un insieme di parti indipendenti (moduli software)
  - la **realizzazione** dei moduli, tenendo conto delle caratteristiche del linguaggio di programmazione prescelto

Astrarre per decomporre, rappresentare, generalizzare

- La decomposizione di un problema in sottoproblemi è un metodo semplice e intuitivo per affrontare problemi complessi. Si astrae dalle caratteristiche peculiari dei sottoproblemi e ci si concentra sulla loro funzionalità e sulla loro interazione
- La astrazione è tipicamente usata per cogliere gli aspetti essenziali di una situazione del mondo reale allo scopo di definire opportune strutture di rappresentazione che evidenzino alcuni aspetti e ne trascurino altri. (Mappe, grafici, immagini invece che descrizioni testuali nella comunicazione)

MAPPA

Generalizzazione

- Si astrae sempre quando si generalizza: rendere generale un metodo solutivo messo a punto per un problema particolare impone di comprendere l'essenza del metodo, prescindendo dalla peculiarità del caso.
- Si tende ad individuare un modello unico che si adatti a situazioni diverse, magari attraverso l'impostazione corretta di parametri.

Tipi di astrazione fondamentali

- Astrazione sui dati

  Consente di far riferimento a strutture algebrico-matematiche, caratterizzate da valori e da operazioni su tali valori, prescindendo dal modo in cui queste strutture sono organizzate e realizzate mediante i costrutti di un linguaggio di programmazione
- Astrazione sulle funzioni

  Consente di concentrare l'attenzione su "cosa" fa una particolare operazione piuttosto che sul "come" è fatta. Si astrae dalle modalità di realizzazione, ci si concentra sul compito o sulle funzionalità di un segmento di programma.

Evoluzione delle tecniche di astrazione

- Le tecniche di astrazione si sono evolute nel corso del tempo non solo con la nostra più approfondita comprensione della programmazione ma anche con la capacità di usare sempre di più le astrazioni come specifiche formali del sistema che descrivono.
- Negli anni '60 l'attenzione, sia nelle metodologie che nei linguaggi, era posta sulle funzioni e procedure, che riassumevano un segmento di programma in termini di

  un nome e una lista di parametri
- All'epoca si era in grado di fare solo controlli di validità sintattica. Il termine "specifica" significava poco più che "intestazione di procedura"

Evoluzione delle tecniche di astrazione (cont.)

- Nei tardi anni '60 l'astrazione fu trattata come una effettiva tecnica di organizzazione dei programmi.
- I primi linguaggi consentivano tipi di dati pre-definiti (interi, reali, booleani) mentre le prime strutture di dati furono trattate in modo sistematico nel 1968. (Knuth "Fundamental Algorithms").
- I quegli anni emerse l'idea che un programmatore potesse "definire" tipi di dati adatti a un particolare problema e fu coniato il termine Software engineering
- Nei primi anni '70 fu definita una metodologia per costruire programmi partendo dalla definizione degli obiettivi: questi venivano progressivamente specificati fino ad arrivare al vero e proprio "codice" (step-wise refinement o top-down programming)

Evoluzione delle tecniche di astrazione (cont.)

- Insieme alla metodologia step-wise refinement fu sviluppata una "disciplina" dello scrivere programmi, la programmazione strutturata, che usa strutture di controllo ideali che hanno la caratteristica di un unico ingresso e una unica uscita
- Risultato è la relativa facilità di scoprire, in ogni punto del programma, quali ipotesi sono vere circa lo stato del programma, consentendo di verificare tanto l'aspetto statico che quello dinamico del programma
- Questa tecnica di verifica è basata sulla definizione di asserzioni formali sul calcolo che il programma deve fare
- Dai primi anni '70 si sono messe a punto tecniche di verifica basate sulle specifiche formali

Evoluzione delle tecniche di astrazione (cont.)

- Gli sforzi iniziali si concentrarono su cosa fosse utile mettere nelle specifiche formali
- Nacque il dibattito sui tipi di dati astratti, sulla natura dei tipi e sulle connessioni con le algebre astratte. Di pari passo furono sviluppati linguaggi di programmazione in grado di trattare tipi di dati astratti che fornissero supporto per trattare
  - tipi di dati
  - moduli
  - lo sviluppo del concetto di località
- Alcuni esempi, oltre al Pascal, sono:
  - ADA (US DOD, ref. Manual for ADA, 1980; A. N. Habermann, D. Perry, ADA for Experienced Programmers Addison Wesley, 1983)
  - MODULA (N. Wirth, Programming in MODULA-2, Springer Verlag, 1983)
  - EUCLID, GYPSY, MESA, CLU, ALPHARD, RUSSEL

Classificazione dlle tecniche di progettazione

- Tecniche di specifica

  Consentono di esprimere gli elementi essenziali dello schema concettuale mediante formalismi grafici e un linguaggio basato sulla logica e sull'algebra per descrivere gli aspetti concettuali dei tipi astratti di dato. Spesso seguono esplicitamente il principio di decomposizione

- Tecniche di programmazione

  Riguardano i metodi per la strutturazione e per la stesura dei programmi a partire dallo schema concettuale. Rientrano la programmazione strutturata (con schemi iterativi e ricorsivi), la programmazione orientata agli oggetti, la programmazione logica e funzionale

Tecniche di progettazione (cont.)

- Modularizzazione

  Consente di razionalizzare lo sviluppo del software costruendo programmi costituiti da parti indipendenti e interagenti; Un "modulo" è caratterizzato da una struttura interna, è definito con un determinato scopo, offre all'esterno un insieme prefissato di funzionalità utilizzabili da altri moduli, ha precise relazioni con altri moduli

- Tecniche di progettazione di algoritmi e strutture dati

  Riguardano la definizione degli algoritmi e la individuazione delle strutture dati più appropriate per un determinato problema (tecniche enumerative, backtracking, divide et impera etc.)

- Tecniche di progettazione per specifiche classi di applicazioni

  Esistono tecniche specificatamente definite per applicazioni particolari, quali quelle concorrenti, quelle distribuite, quelle orientate alla gestione dei dati etc.

Strumenti di progettazione

- Strumenti di specifica per la concettualizzazione, sono quei formalismi (testuali, grafici) mediante i quali si descrive lo schema concettuale di progetto)

- Strumenti di analisi per la concettualizzazione, aiutano il progettista nell'analisi e nella verifica del documento di analisi e dello schema concettuale di progetto (sistemi automatici o semiautomatici per computer assisted software engineering)

- Strumenti per la programmazione, sono quelli che supportano le fasi di scrittura del programma, compilazione e test (text editor, compilatori e interpreti, linker, debugger)

Modularizzazione

- Idea di base: strutturare un programma in parti autonome e interagenti, dette moduli.
- Facilita il processo di sviluppo e le attività di analisi e di verifica. L'uso di moduli favorisce la riusabilità
- Tra i diversi tipi di modularizzazione, grande importanza ha la modularizzazione per tipo astratto, secondo la quale vengono sviluppati moduli che realizzano tipi astratti di dati significativi per l'applicazione
- Il concetto di modularizzazione si ritrova in tutte le organizzazioni
- ESEMPI:
  - Banca organizzata in agenzie, ognuna con vari servizi alla clientela.
  - Università strutturata in facoltà, dipartimenti e settori, con determinate funzioni e compiti.

Modularizzazione (cont.)

- Un modulo è una unità di un programma con una sua struttura interna, definito per un determinato scopo, che offre all'esterno un certo insieme prefissato di servizi utilizzabili da altri moduli
- Un modulo è caratterizzato da:
- la struttura interna, cioè l'insieme dei tipi, delle variabili e delle funzioni definiti nel modulo stesso
- l'insieme dei servizi che esporta, ovvero che offre agli altri moduli
- la modalità con cui tali servizi possono essere utilizzati (interfaccia del modulo)
- l'insieme dei servizi che esso importa dagli altri moduli e che utilizza per le sue funzioni

Qualità della modularizzazione

La qualità della modularizzazione aumenta:

- all'aumentare della coesione (alta coesione di un modulo significa che esso incapsula un insieme di caratteristiche omogenee, sufficientemente indipendenti da altri moduli)
- all'aumentare dell'uso di information hiding (i dettagli interni al modulo non devono giocare alcun ruolo nell'utilizzo del modulo da parte di moduli esterni)
- al diminuire dell'accoppiamento (un basso grado di accoppiamento tra moduli significa che non si sono create indipendenze non volute o non necessarie; un esempio di questo criterio suggerisce di limitare al massimo l'uso di variabili globali, visibili e utilizzabili da più moduli, poiché creano scambi di informazioni non facilmente controllabili)
- con l'utilizzo dell'interfacciamento esplicito (questo criterio suggerisce di rappresentare mediante parametri tutti i dati che vengono scambiati tra due sottoprogrammi)

Classificazione dei linguaggi di programmazione

- Linguaggi imperativi: un programma è la specifica di un insieme di istruzioni che corrispondono a precisi comandi impartiti ad una macchina che li esegue pedissequamente
- Linguaggi di programmaz. funzionale: un programma è la specifica di una funzione che, in base a un insieme di dati in ingresso, calcola il risultato secondo una legge specificabile in modo matematico
- Linguaggi di programmaz. logica: un programma è la specifica di una relazione che sussiste tra un insieme di dati, e la specifica è costruita mediante un sistema formale basato sulla logica
- Linguaggi di programmaz. O.O.: un programma è la specifica di un insieme di oggetti che rappresentano gli elementi della situazione in gioco in un certo problema, e ciascun oggetto è specificato in termini di una struttura e di un insieme di operazioni tramite le quali si ottiene il comportamento voluto per risolvere il problema