

Corso di Programmazione I

Programma dell'A.A. 2003/2004

Ing. Marcello Esposito

1. Argomenti del corso

1.1. Programmazione procedurale: aspetti avanzati.

Le funzioni: aspetti avanzati. Funzioni con argomenti di default, funzioni in linea (inline), sovraccarico dei nomi di funzione, sovraccarico degli operatori (overload). Firma di una funzione (signature).

Dal testo adottato C++ Cap. 7 §§ da 7.2 a 7.4 (richiami), e §§ da 7.6 a 7.8.

Dal testo adottato F1, Parte II, Cap. 4 §§ 8 e 9 (richiami).

Variabili e puntatori: aspetti avanzati. Classi di memorizzazione delle variabili in C++: variabili statiche, automatiche e dinamiche. Area stack, area heap, ed area dati statici. Visibilità (scope) e tempo di vita (lifetime) di una variabile. Variabili `static`, `register` e `volatile`. Puntatori e variabili dinamiche. Allocazione dinamica della memoria: operatori `new` e `delete` del linguaggio C++. Aritmetica dei puntatori. Passaggio dei parametri per puntatore. Puntatori costanti e puntatori a dati costanti. Concatenamento di strutture dinamiche mediante puntatori.

Dal testo adottato C++ Cap.8 §§ 1 e 2 e §§ 8.3.1, 8.3.5, 8.3.6, 8.4.

Dal testo adottato F2, Parte II, Cap. 8 § 2.

La libreria `iostream` per le operazioni di I/O. Operazioni di ingresso ed uscita primarie. Collegamenti con il sistema operativo. Operazioni di I/O verso memorie di massa. Generazione ed ispezione di un file sequenziale di tipo testo o binario. Specificatori. La gerarchia delle classi per l'I/O in C++.

Dal testo adottato F2, Parte I, Cap. III

Dal testo adottato C++, Cap. 27; Cap. 28 §§ 1,2,3.

1.2. Tecniche di programmazione

Strutture dati. Tipi di dati astratti: specifica ed implementazione. Liste, pile, code, alberi, tabelle (realizzazioni statiche e dinamiche). Alberi binari ordinati. Inserimento ed eliminazione di elementi da un albero. Visita di un albero. Ordine della complessità ed analisi dell'efficienza delle realizzazioni concrete.

Dal testo adottato F2, Parte I, Cap. 2.

Dal testo adottato C++, Capp. 17, 40 (leggere) e 41 (leggere).

La ricorsione. Induzione e ricorsione. Schema degli algoritmi ricorsivi. Meccanismo interno di ricorsione. Risoluzione in forma iterativa di un algoritmo ricorsivo. Esempio: il fattoriale. Impiego degli algoritmi ricorsivi.

Dal testo adottato F1, Parte II, Cap. IX, §§ 1, 2, 4, 5, 6.

Problemi di ricerca ed ordinamento. Ricerca lineare e ricerca binaria in una lista. Ordinamento per selezione (selection sort). Ordinamento per scambi (bubble sort), Ordinamento per inserzione (insertion sort). Ordinamento con doppio indice (quick sort). Complessità degli algoritmi di ricerca ed ordinamento. Analisi comparata dell'efficienza degli algoritmi di ordinamento (esercizio).

Dal testo adottato F1, Parte II, Cap XI, §§1, 2, 3, 4, 5, 11.

Dal testo adottato C++, Cap. 39.

Programmazione modulare. Ciclo di sviluppo di un programma. Variabili extern. Programmazione modulare: file di intestazione e di implementazione. Il precompilatore: direttive #include, #ifndef, #define, #endif. Compilazione separata e strutturazione di un programma in forma modulare.

Dal testo adottato C++, Cap. 12; Cap. 13, §§1, 2, 3, 5.

1.3. Programmazione ad oggetti in C++

Introduzione alla programmazione orientata agli oggetti. Meccanismi di astrazione. Incapsulamento. Information hiding. Programmazione orientata agli oggetti. L'ereditarietà quale strumento di progettazione o di riuso. Il polimorfismo.

Dal testo adottato C++, Cap. 15.

Le classi: notazioni di base. Specifica ed implementazione della classe. La specifica come interfaccia. Funzioni membro e funzioni ordinarie operanti su oggetti. Costruttore di default, costruttore con parametri, costruttore di copia, ciclo di vita degli oggetti e funzione distruttore. Operatore di assegnazione. Copia superficiale (shallow-copy) e copia profonda (deep-copy). Metodi di accesso e di posizionamento. Metodi const. Accesso ai membri di una classe: membri private, protected e public. Membri espansi in linea. Funzioni e classi friend. Il puntatore this.

Dal testo adottato C++, Cap. 16.

Ereditarietà in C++. Le classi derivate nel C++. Meccanismi sintattici per la derivazione. La ridefinizione dei metodi nelle classi derivate (overriding). Trasmissione dei diritti di accesso. Meccanismi selettivi di derivazione. Ordine di chiamata di costruttori e distruttori in gerarchie di classi. Compatibilità tra classi antenate e classi derivate. Concetto di tipo statico e tipo dinamico.

Dal testo adottato C++, Capp. 20 e 21.

Polimorfismo in C++. Motivazioni per il polimorfismo. Meccanismi sintattici per il polimorfismo. Il legame ritardato (late binding). Funzioni virtuali. Classi puramente astratte. Costruttori e distruttori in classi polimorfe. Meccanismi per l'implementazione del polimorfismo: la tabella dei metodi virtuali (VMT).

Dal testo adottato C++, Cap. 22.

1.4. Progettazione ad oggetti in UML

Il linguaggio UML per la modellazione ad oggetti. La progettazione tradizionale. Analisi, progetto e programmazione orientata agli oggetti (OOA, OOD e OOP). Il linguaggio UML. Diagramma dei casi d'uso (use-case diagram). Diagramma delle classi (class diagram). Diagramma di sequenza (sequence diagram). Diagramma di attività (activity diagram). Diagramma di collaborazione (collaboration diagram).

Dal testo adottato C++, Cap. 32; Cap. 33 §§ 1, 2, 3.

1.5. Programmazione ad oggetti in Java

Dal testo consigliato Thinking in Java, Capp. 2, 4, 6, 7, 8.

Elementi base del linguaggio Java. Caratteristiche principali del linguaggio Java. La portabilità di Java: la macchina virtuale ed il bytecode. Architettura della macchina virtuale. I tipi primitivi.

Gestione della memoria: il garbage collector. Le classi in Java. Modalità di scambio dei parametri nelle funzioni.

L'ereditarietà in Java. La parola chiave `extends`. Modalità di inizializzazione della classe base. Specificatori di accesso alle funzioni membro.

Polimorfismo in Java. Il late binding. Polimorfismo mediante classi ordinarie. Polimorfismo mediante classi astratte. Le differenze con il C++: il concetto di interfaccia in Java. Polimorfismo mediante interfacce.

2. Parte esecutiva

2.1. Preparazione dell'esame

Al fine di conseguire sufficienti competenze di tipo tecnico-pratico, anche in vista della prova di accertamento pratica, si consiglia agli studenti lo sviluppo autonomo di programmi in linguaggio C++ relativi alla programmazione:

- di algoritmi con puntatori e variabili dinamiche;
- di algoritmi con l'impiego di strutture statiche;
- di algoritmi con l'impiego di stringhe;
- di algoritmi ricorsivi;
- di tipi di dati astratti;
- di strutture dati fondamentali (liste, pile, code, alberi) con realizzazione statica e dinamica;
- di gerarchie di classi;
- di classi polimorfe e funzioni virtuali;
- di algoritmi basati su meccanismi di I/O su stream.

2.2. Modalità di svolgimento dell'esame

L'esame è suddiviso in tre prove distinte: la prima prova consiste nello svolgimento di un test a risposta multipla al calcolatore. Il superamento di questo dà accesso alla seconda prova pratica, anch'essa al calcolatore, consistente nella realizzazione di un semplice applicativo in linguaggio C++. Infine, a valle del superamento delle suddette prove, il candidato affronterà un colloquio orale basato sugli argomenti elencati nel programma.

La prova pratica si intende superata qualora il programma passi correttamente le fasi di compilazione e collegamento e fornisca risultati corretti quando eseguito a fronte dei casi di prova prescritti dal testo, oppure autonomamente definiti dal candidato o dal docente nel rispetto delle specifiche assegnate dalla traccia. In caso di prova superata la commissione tiene in conto il corretto impiego, nella strutturazione del programma da parte del candidato, delle tecniche di programmazione illustrate durante il corso.

2.3. Prenotazione degli esami

Per la prenotazione degli esami si utilizzerà lo stesso sistema di prenotazione basato sul web messo a disposizione degli studenti per le esercitazioni in laboratorio multimediale (raggiungibile a partire dall'indirizzo <http://www.grid.unina.it>). Gli studenti che non avessero la possibilità di accedere al web possono contattare il docente durante l'orario di ricevimento previsto oppure telefonicamente al numero 081-768-3901.

3. Materiale didattico

3.1. Testi

Testi adottati

- C++) C. Savy, "Da C++ ad UML: guida alla progettazione", Mc Graw Hill, 2000
F1) B. Fadini, C. Savy, "Fondamenti di Informatica I", Liguori Editore.
F2) B. Fadini, C. Savy, "Fondamenti di Informatica II", Liguori Editore.

Testi consigliati

- H. Schildt, "Guida completa al C++", 2a edizione, McGraw Hill.
B. Eckel, "Thinking in C++", Apogeo (anche liberamente disponibile in rete al sito <http://www.bruceeckel.com>).
B. Eckel, "Thinking in Java", Apogeo (anche liberamente disponibile in rete al sito <http://www.bruceeckel.com>).

3.2. Altro materiale

- Sito web del corso raggiungibile a partire da: <http://www.grid.unina.it>
- Disponibili sul sito web del corso:
 - dispense dalle lezioni;
 - esercizi con svolgimenti;
 - testi e soluzioni delle esercitazioni in laboratorio.
- Gruppi di discussione dedicati agli studenti del corso disponibili mediante protocollo NNTP all'indirizzo <news://news.grid.unina.it>
 - p1.tlc.teoria
 - p1.tlc.discussioni
 - p1.tlc.c++
 - p1.tlc.ot
 - p1.tlc.java
- Gruppo di discussione Usenet disponibile mediante protocollo NNTP sul server <news://news.unina.it>
 - it.comp.lang.c
 - it.comp.lang.c++
 - comp.lang.c (in inglese)
 - comp.lang.c++ (in inglese)
- Motore di ricerca storico sul circuito Usenet disponibile mediante protocollo HTTP all'indirizzo <http://groups.google.com>

4. Ricevimento studenti

L'orario di ricevimento per gli studenti del corso è fissato il mercoledì dalle 15.30 alle 17.30 nella stanza 4.08 al IV piano del Dipartimento di Informatica e Sistemistica situato in via Claudio, 21. Tel.: 081-768-3901 - Fax: 081-768-3816 - E-mail: mesposit@unina.it