DIPARTIMENTO DI SCIENZE E TECNOLOGIE
Università degli Studi di Napoli "Parthenope"
Centro Direzionale di Napoli - Isola C4
80143 Napoli (I)

prof. Mariarosaria Rizzardi
stanza n. 423 - IV piano, lato NORD
tel.: 081 547 6545
emai: mariarosaria.rizzardi@uniparthenope.it

PROGRAMMA D'ESAME di
Programmazione II e Laboratorio di Programmaz. II
dall'A.A. 2019-2020

I due moduli integrati di *Programmazione II e Laboratorio di Programmazione II* (esame unico) rappresentano la naturale prosecuzione del percorso didattico iniziato con gli omologhi di primo livello e pertanto i prerequisiti necessari consistono prevalentemente nella conoscenza degli argomenti trattati nel corso di *Programmazione I e Laboratorio di Prog. I.*

L'obiettivo principale del corso consiste nell'approfondire alcuni aspetti fondamentali legati alle metodologie di sviluppo ed analisi degli algoritmi, all'organizzazione logica dei dati e alla relativa implementazione nel linguaggio C.

Sono trattati alcuni approfondimenti del linguaggio C e viene altresì introdotto il linguaggio C++ soprattutto per quanto riguarda i suoi aspetti innovativi rispetto al C. Sono introdotte le strutture dati dinamiche (lineari, gerarchiche e reticolari) e descritte le relative implementazioni in C e mediante la libreria STL del C++. Infine sono richiamati i concetti principali relativi alla programmazione ricorsiva e sono trattati alcuni algoritmi di ordinamento della classe "*Divide et Impera*".

Materiale didattico

Servizio di *web Learning*: http://e-scienzeetecnologie.uniparthenope.it/.
Documentazione online per C/C++: http//www.cpluplus.com/.

Testi consigliati

Kim.N. King – *Programmazione in C* – Apogeo

J. Soulié – *C++ Language Tutorial* – cplusplus.com (2007) [http://www.cplusplus.com/files/tutorial.pdf]

H. Schildt – *C++: The Complete Reference* – McGraw-Hill, 4th Ed. (2003)
[http://160592857366.free.fr/joe/ebooks/ShareData/C++ - The Complete Reference 4e.pdf]

Testi di consultazione

R. Sedgewick – *Algoritmi in C++* – Addison-Wesley

B. Stroustrup – *The C++ Programming Language* – Addison-Wesley, 4th Ed., (2013)
[http://home.agh.edu.pl/~sul5/php/Straustrup4th.pdfa]

T.H. Cormen, C.E. Leiserson, R.L. Rivest – *Introduzione agli algoritmi* – Jackson Libri

E. Horowitz, S. Sahni, S. Anderson-Freed – *Strutture dati in C*. McGraw-Hill Libri Italia.

ARGOMENTI TRATTATI

Compilare da riga di comando

Installare la libreria MinGW dei compilatori GNU gcc e g++ per Windows. Principali comandi DOS e Linux. Compilatori gcc e g++. Confronto versione compilatore e libreria di MinGW e Code::Blocks. Compilare da riga di comando. Utility make e Makefile; uso delle dipendenze. Redirezione dell'input standard (<) e dell'output standard (>, >>). Parametri alla funzione main(). Il primo programma in C e il primo programma in C++. Passaggio dei parametri *per reference* in C++ e variabili *reference*.

C/C++

Rappresentazione in memoria dei tipi di dato scalari. Tipo logico in C: variabili booleane e operatori booleani. Operatori *bitwise*. Operazioni sugli interi mediante *operatori bitwise*. Campi di bit di una `struct`. Precedenza fra operatori. **C++**: il tipo `bool` e la classe template `bitset<N>`. Esempi d'uso.
Unioni ed enumerazioni.
Richiami sulla rappresentazione posizionale dei numeri. Algoritmi di cambiamento di base. Tipi numerici: `signed` ed `unsigned char`, `short`, `int`, `long` e `long long` per gli interi, `float`, `double` e `long double` per i reali. Header file: `limits.h` e `float.h`. Rappresentazione in memoria dei numeri interi: rappresentazione per segno e modulo, per complemento a 2, rappresentazione *biased*. Il *Sistema aritmetico degli Interi* e suo *range*; overflow d'intero. Il *Sistema Aritmetico Binario Floating-point Standard IEEE 754* e sua parametrizzazione. Rappresentazione della mantissa per *bit implicito*. Numeri normalizzati, denormalizzati, *Nan*, underflow, overflow. Visualizzazione dei bit di una variabile numerica. Range dei numeri floating-point. Schemi di arrotondamento. Massima accuratezza statica e massima accuratezza dinamica. Errori di *roundoff* ed esempi relativi.
Variabili di tipo carattere e di tipo stringa: array di caratteri e array frastagliati di stringhe (mediante puntatori). Funzioni in `<string.h>`. Input di stringhe: `gets()`, deprecata, e `fgets()`. **C++**: classe `string`, principali metodi e operazioni su stringhe. Input di stringhe (`getline`). *Array frastagliati* di stringhe in C e in C++.
Allocazione dinamica in C: `malloc()`, `calloc()`, `realloc()`, `free()`.Il compilatore GNU gcc (dal C99) e i *Variable Length Array* (VLA). Allocazione dinamica in **C++**: operatori `new` e `delete`, array dinamici: il contenitore sequenziale `vector`. Iteratori. Funzioni per copiare o spostare blocchi di memoria (`memset`, `memcpy`, `memmove`). Allocazione dinamica di matrici: gestione tramite puntatori delle matrici allocate per righe o per colonne. Differenza tra allocazione (per righe o per colonne) di una matrice ed accesso (per righe o per colonne) ai suoi elementi. Gestione delle matrici (statiche e dinamiche) nel passaggio dei parametri.
Gestione di file testo e file binari. Input/Output su file in C. Gestione di file testo e binario in **C++**: stream e file. I/O formattato, manipolatori. Classi `ifstream`, `ofstream`, `fstream`.
Puntatori a `void`. Funzioni ricorsive in C/C++.
C++: Classi e oggetti. Metodi e attributi. Specificatori di accesso (`private`, `protected`, `public`). Diagramma delle classi. Costruttori e distruttori di classe: costruttore default, costruttore con parametri, costruttore di copia. Ereditarietà tra *classe base* e *classe derivata*. Tipo di ereditarietà. *Upcasting*. Diagramma di classi derivate. Polimorfismo e *funzioni virtuali*. *Function overloading* e *function overriding*. *Funzioni virtuali pure* e classi astratte. Diagramma di classi astratte. Cenni ai *template*: funzione generica e classe generica. Cenni ai *namespace* e alla clausola `using`. Cenni all'*overload di operatori*. La *lista di inizializzazione dei dati membro* nel costruttore. Quando va usata la lista di inizializzazione. Esempi.

Tipi di dati strutturati

Richiami sui tipi di dati strutturati. Strutture statiche (array, record) e *strutture dinamiche lineari* (lista, coda, pila). Rappresentazione in *C* di liste, code e pile. Operazioni sulle strutture lineari: visita, inserimento, eliminazione di nodi. Particolari liste: circolari, bidirezionali, multiple. Esempi. Rappresentazione in memoria di matrici sparse mediante liste multiple. **C++**: classi template `stack`, `queue`, `forward_list`, `list`. Esempi.
Strutture gerarchiche (alberi). Visita *per livelli* e visita *in ordine anticipato* di un albero qualsiasi. Alberi binari. Alberi binari completi. Algoritmi di visita su alberi binari: visita anticipata, simmetrica e differita. Rappresentazione di un albero binario mediante array oppure mediante lista multipla. Alberi Binari di Ricerca

(BST – Binary Search Trees): algoritmo di costruzione e di ricerca binaria. Struttura dati *heap* e sua rappresentazione mediante array. Code con priorità rappresentate come heap (max o min). **C++**: esempio di classe che implementa un Albero Binario di Ricerca; la classe template `priority_queue` (per un max-heap). Esempi.
Strutture reticolari (grafi). Grafi orientati e non orientati, pesati e non pesati. Rappresentazione in memoria di grafi mediante matrice di adiacenze e mediante lista di adiacenze. Algoritmi di *visita in profondità (DFS)* e di *visita in ampiezza (BFS)* di un grafo non orientato. Esempi.

Ricorsione

Funzioni ricorsive ed algoritmi ricorsivi. Struttura della ricorsione: ricorsione lineare, binaria, non lineare, mutua ricorsione. Analisi della profondità di ricorsione. Esempi di algoritmi ricorsivi in *C*: fattoriale, MCD, ricerca binaria, costruzione di una lista da un array, visita di alberi binari.

Algoritmi di ordinamento

Algoritmi di ordinamento ed analisi di complessità. Richiami sugli algoritmi a complessità quadratica: *SelectionSort*, *ExchangeSort* (*BubbleSort*), *InsertionSort*. Algoritmi della classe “Divide et Impera”: *MergeSort*, *QuickSort*, *HeapSort*. Complessità di tempo e di spazio.