객체지향프로그래밍과 자료구조

과목 총정리

정보통신공학과

교수 김 영 탁

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

- 대규모 소프트웨어 시스템 개발에서의 필수적인 고려사항
- 객체 지향형 프로그래밍 (Object-Oriented Programming)
- 자료구조과 알고리즘
- 이 과목에서 배운 중요 주제

대규모 시스템 개발에서의 필수적인 고려 사항

◆ 기본적으로 대규모 시스템의 개발에서 요구되는 사항

- **System stability** (시스템 안정성): 시스템이 항상 정상적인 범위/상태에 존재하도록 유지.
- **Software re-usability** (소프트웨어 재사용성): 기존에 개발된 소프트웨어 모듈을 재사용함으로써 쉽게 새로운 시스템 개발. 시스템 개발 기간과 비용을 절감.
- **User-friendly Interface** (사용자 친화형 접속): 사용자가 쉽게 이해하고, 사용할 수 있는 인터페이스
- **System expandability** (시스템 확장성): 새로운 최신 기술이 개발되었을 때, 이를 쉽게 수용할 수 있는 확장성.
- **System Manageability** (시스템 관리 용이성): 시스템을 쉽게 관리할 수 있는 구조

왜 객체 지향형 프로그래밍인가 ?

◆ 시스템 안정성 (system stability)

- all data members should contain appropriate data values
- e.g.)
  - human age: 0 ~ 150 years
  - people's height: 30 ~ 250 cm
  - water temperature: 0 ~ 100 °C
- **get() and set() member function** for each data members

◆ 소프트웨어 재사용성(Software re-use)

- usage of the existing / well-managed software modules
- reduced development cost
- minimized development period
- better reliability
- **상속(inheritance)**

◆ 사용자 친화형 접속 (Use-friendly interface)

- any system should be easy to use
- easily understood user interface
- common sense
- internationally accepted notations (e.g., mathematical operators)
- **operator overloading**

◆ 시스템 확장성 (System expandability)

- any new technology should be easily adapted without too much modification
- well defined/standardized inter-module interface
- **information hiding**

◆ 시스템 관리 유지의 용이성 (System manageability)

- without too much modification in the existing management system, new product should be easily integrated
- **polymorphism** is necessary: virtual function, late binding

자료구조와 알고리즘

◆ Applications require

- Appropriate **Data Processing Pattern (algorithm)**
- Appropriate **Data Structure**

E.g.) Customer service at bank, ticket booth

- first come first served (FCFS) rule
- FIFO (first in first out) queue

E.g.) Information search from 100 Tera-bytes data base

- linear search vs. binary search
- considerations: request pattern of update and search
- hash, binary search tree, multi-way tries

◆ Appropriate data structure and algorithm enhance the system performance

- the most appropriate data structure and algorithm should be used
  - appropriate data structure: (e.g., array of struct vs. binary tree)
  - appropriate algorithm: (e.g., linear search vs. binary search)
- major considerations:
  - CPU performance
  - special processing block
  - system memory size
  - required processing time (e.g., within 1 ms)
  - required memory size
  - communication link capacity

대표적인 자료구조와 특성

자료구조	특성
단순 배열	컴파일 단계에서 크기가 지정되어 변경되지 않는 크기의 배열.
동적 배열	프로그램 실행 단계에서 크기가 지정되며, 프로그램 실행 중에 크기를 변경할 수 있는 배열.
구조체 배열	구조체로 배열원소가 지정되는 배열.
연결형 리스트	확장성이 있으며, 단일 연결형 또는 이중 연결형으로 구성 가능. 다양한 컨테이너 자료구조의 내부적인 자료구조로도 활용됨.
스택	Last In First Out (LIFO) 특성을 가짐. 배열 또는 연결형 리스트로 구현할 수 있음.
큐	First In First Out (FIFO) 특성을 가짐. 배열 또는 연결형 리스트로 구현할 수 있음.
우선순위 큐	컨테이너 내부에 가장 우선순위가 높은 데이터 항목을 추출할 수 있도록 관리하며, 배열 또는 자기 참조 구조체로 구현할 수 있음.
이진 탐색 트리	컨테이너 내부에 포함된 데이터 항목들을 정렬된 상태로 관리하여야 할 때 매우 효율적임. 단순 이진 탐색 트리의 경우 편중될 수 있으며, 편중된 경우 검색 성능이 저하되기 때문에, 밸런싱이 필요함.
해시테이블 (Hash Table)	컨테이너 자료구조에 포함된 항목들을 문자열 (string) 또는 긴숫자를 키 (key)로 사용하여 관리하여야 하는 경우, key로부터 해시값을 구하고, 이 해시값을 배열의 인덱스로 사용함.
Map	key와 항목 간에 1:1 관계가 유지되어야 하는 경우에 사용되며, 해쉬 테이블을 기반으로 구현할 수 있음.
Dictionary	key와 항목 간에 1:N 관계가 유지되어야 하는 경우에 사용되며, 해쉬 테이블을 기반으로 구현할 수 있음.
trie	텍스트 검색을 신속하게 처리하며, 예측 구문 (predictive text) 제시, longest prefix matching 등에 활용됨.
그래프	정점 (vertex)/노드 (node)로 개체 (object)가 표현되고, 간선 (edge)/링크(link)들을 사용하여 개체 간의 관계를 표현하는 경우에 적합함. 그래프를 기반으로 경로 탐색, 최단 거리 경로 탐색, 신장트리 (spanning tree) 탐색 등에 활용됨.

대표적인 알고리즘

알고리즘	주요 기능	비고
순차탐색	배열이나 연결형리스트에 포함되어 있는 항목을 차례대로 탐색	
이진탐색	배열에 포함되어 있는 항목의 탐색 구간을 절반씩 줄여가며 탐색	
선택정렬	배열에 포함되어 있는 항목의 탐색 구간을 하나씩 줄여가며, 가장 작은(또는 큰) 항목을 찾아 그 구간의 맨 앞에 두는 동작을 반복하여 전체 항목들을 정렬. 배열에	
퀵정렬	배열에 포함되어 있는 항목의 탐색 구간을 중간 지점의 피봇 (pivot)을 중심으로 큰 항목들과 작은 항목들로 구분하여 정돈하며, (평균적으로) 절반씩 줄어든 탐색 구간에 대하여 재귀 함수 실행	
해시	문자열과 같이 길이가 일정하지 않는 키 (key)를 사용하여 해시값을 생성하고, 이를 배열의 인덱스나 데이터 항목이 저장되어 있는 그룹(버킷)을 찾는데 사용	해시코드 종류와 충돌 해결 방법에 따라 달라짐
그래프 깊이우선 탐색 (DFS)	그래프에서 지정된 시작 정점으로부터 목적지 정점까지의 경로를 깊이 우선 방식으로 탐색. 목적지까지의 탐색 시간은 상대적으로 빠를 수 있으나, 탐색된 경로가 최단 거리를 보장하지는 않음	
그래프 넓이우선탐색 (BFS)	그래프에서 지정된 시작 정점으로부터 목적지 정점까지의 경로를 넓이 우선 방식으로 탐색. 탐색된 경로는 경유하는 간선수가 최소인 것을 보장함	
최단거리 경로 탐색	간선 (edge)에 가중치 (weight)이 지정된 그래프에서 시작 정점으로부터 목적지 정점까지의 경로 중에서 전체 거리가 가장 짧은 경로를 탐색. 탐색된 경로는 최단거리인 것을 보장함	
최소비용 신장 트리 (MST) 탐색	주어진 그래프에서 사용되는 간선들의 총 거리 (또는 가중치)가 가장 작으면서 모든 노드 (정점)들을 연결할 수 있는 최소비용 신장트리 (minimum spanning tree)를 탐색	

이 과목에서 배운 핵심 주제 (1)

< C 기초내용 (복습) >

1) C 프로그램 기본 구조, 데이터 유형, 변수와 상수, 함수, 콘솔 입력 출력, 파일 입력 출력, 디버깅 및 실행
2) 포인터, 동적메모리할당, 동적 배열
3) 함수호출, 파라메터 전달
4) 구조체, 자기참조 구조체
5) C 구조체 기반 자료구조: linked list, stack, queue, binary tree

이 과목에서 배운 핵심 주제 (2)

<Part 1: C++ 기반 객체 지향형 프로그래밍>

1) class and object: class definition and implementation (private data member, member function), abstraction, encapsulation, information hiding
2) operator overloading: +, -, *, /, ==, !=, ++, --, !, ~
3) inheritance of classes
4) virtual function and polymorphism
5) standard template library (STL)

<Part 2: 자료 구조>

1) Template 기반 Linked Lists, Stacks, Queues
2) Priority Queues, Heaps
3) Trees, Binary Search Tree, Balancing of Binary Search Tree
4) Hashing, Hash Map, Hash Dictionary
5) trie
6) Graphs, DFS, BFS, Shortest Path (Dijkstra), Minimum Spanning Tree (Prim-Jarnik)

수업을 마치며

◆ 컴퓨팅 사고 (Computational Thinking)

- 컴퓨터를 활용하여 문제 해결 방안을 효율적으로 생각하고 설계 및 구현하는 방법 습득
- 문제 해결을 위한 알고리즘과 자료구조 설계에 더 많은 시간을 투자할 것

◆ 큰 규모의 소프트웨어 개발 전략

- 하드웨어를 제어하여야 하는 경우 해당 모듈 (예: device driver, 장치 구동자)을 C 프로그래밍으로 구현
- 전체 시스템을 큰 단위로 구분하여 정의되는 서브 시스템은 객체 지향형으로 설계 및 구현: 사용자 접속과 내부 구현을 구분하는 정보 은닉, 안정적인 시스템 상태 유지, 소프트웨어 재사용을 통한 개발기간/비용 축소

◆ C++ 기반 객체 지향형 프로그래밍

- 객체 지향형 프로그래밍의 장점과 성능 최적화 장점 제공
- Python에서도 핵심 모듈 구현은 C/C++로 구현

수고 많았습니다.

감사합니다.