

한국방송통신대학교 자료포털 노우존

www.knouzone.com



자료구조

2018학년도 2학기 출석수업대체시험

핵심체크 및 출제예상문제



범위 : 교재 1 ~ 5장

제1장 자료구조란 무엇인가

1. 자료와 정보 사이의 관계

1) 자료의 정의

- ① 현실 세계에서 관찰이나 측정을 통해서 수집된 값(value)이나 사실(fact)
- ② 우리의 생활에서 실제로 만질 수 있거나 볼 수 있거나 하는 것(길이, 무게, 부피 등을 측정할 수 있는 대상)에 대해서 물리적인 단위로 표현하여 얻어낼 수 있는 내용

2) 정보의 정의

- ① 어떤 상황에 대해서 적절한 의사결정(decision)을 할 수 있게 하는 지식(knowledge)으로서 자료의 유효한 해석(interpretation)이나 자료 상호 간의 관계(relationship)를 표현하는 내용
- ② 어떠한 상황에 적절한 결정이나 판단에 사용될 수 있는 형태로 가공되거나 분류되기 위해 '처리 과정'을 거쳐서 정리되고 정돈된 '자료'의 2차 처리 결과물

3) 자료와 정보의 관계

컴퓨터

자료 → 처리 → 정보

2. 추상화의 개념

1) 추상화

- ① 공통적인 개념을 이용하여 같은 종류의 다양한 객체를 정의하는 것
- ② 추상화를 통해 간결하게 말하는 사람의 의사를 전달할 수 있게 되는 것

2) 자료의 추상화

- ① 자료의 추상화 : 다양한 객체를 컴퓨터에서 표현하고 활용하기 위해 필요한 자료의 구조에 대해서 공통의 특징만을 뽑아 정의한 것
- ② 자료의 추상화에는 컴퓨터 내부의 이진수의 표현 방법, 저장 위치 등은 포함되지 않고 단순히 개발자의 머릿속에 그림을 그리는 것처럼 개념화하는 것

3. 자료구조의 개념

1) 자료구조

- ① 추상화를 통해 자료의 논리적 관계를 구조화한 것
- ② 자료의 추상화와 구조화가 적절히 이루어지지 못하면 소프트웨어는 비효율적으로 수행되거나 소프트웨어의 확장성에 문제가 생길 수 있음

2) 자료구조와 알고리즘의 협동

자료구조는 입력값의 추상화된 상태라면, 알고리즘은 컴퓨터가 수행해야 할 명령의 추상화

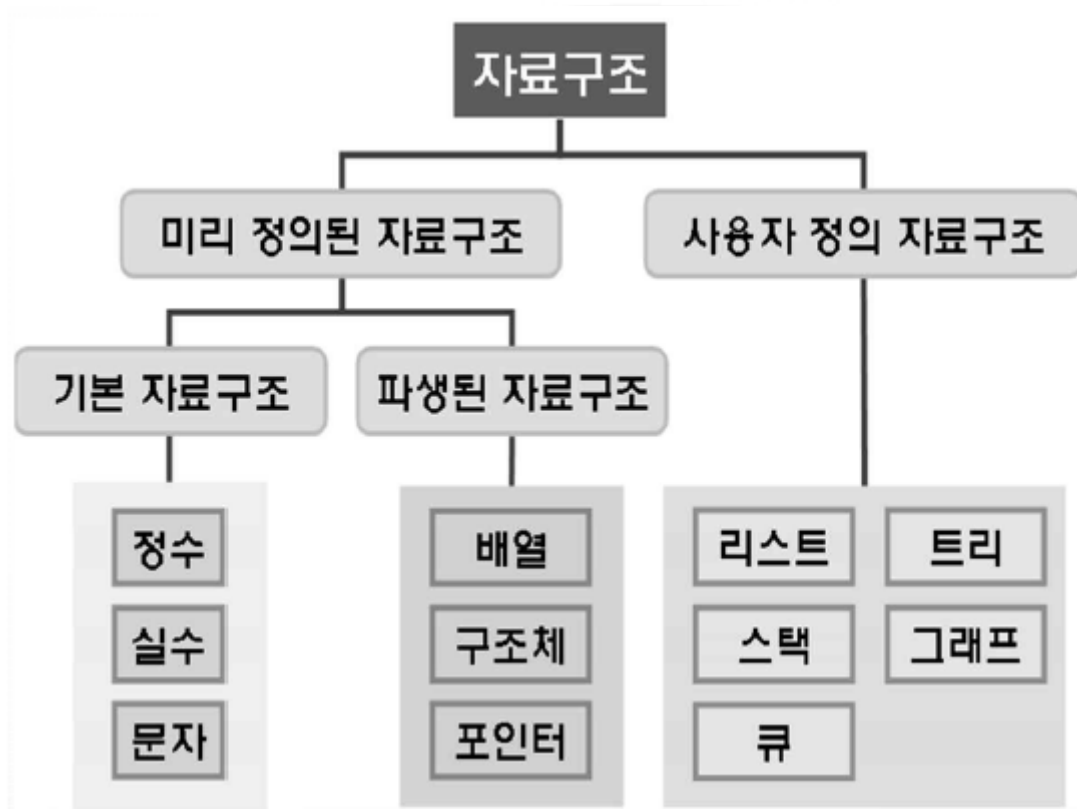
3) 자료구조의 두 가지 측면

자료구조는 입력값의 추상화된 상태라면, 알고리즘은 컴퓨터가 수행해야 할 명령의 추상화

4) 자료구조와 알고리즘의 추상화 / 구체화

입력될 값을 머릿속에서 추상화된 형태(자료구조)로 구조화하고, 수행되어야 할 명령어를 머릿속에서 추상화된 형태(알고리즘)로 체계화

5) 자료구조의 종류와 관계



4. 자료구조와 알고리즘의 관계

1) 알고리즘

- ① 컴퓨터에게 일을 시키는 명령들의 덩어리
- ② 컴퓨터에 의해 수행되기 위해 필요한 명령어들의 유한 집합이 램의 머릿속에 추상화되어 존재하는 것
- ③ 사람(개발자)이 컴퓨터에게 일을 시키기 위해서는 사람의 의도와 명령을 전달해 줄 수 있는 방법(프로그래밍언어)

2) 알고리즘의 조건

- ① 출력: 적어도 한 가지의 결과를 생성함
- ② 유효성: 원칙적으로 모든 명령들은 종이와 연필만으로 수행될 수 있도록 기본적 이어야함
- ③ 입력: 외부에서 제공되는 자료가 있을 수 있음
- ④ 명확성: 각 명령들은 명확하고, 모호하지 않아야 함
- ⑤ 유한성: 알고리즘의 명령대로 수행하면, 어떤 경우에도 한정된 수의 단계 뒤에는 반드시 종료함

5. 알고리즘 성능의 분석과 측정

1) 실행시간 분석

(1) 알고리즘을 실행하는데 필요한 실행시간을 추정하여 알고리즘의 성능을 분석(performance analysis)

(2) 실행시간의 예측

① 프로그램의 시간 복잡도는 프로그램을 실행시켜 완료하는 데 걸리는 시간을 의미. 일반적으로 어떤 프로그램 P를 실행하는 데 필요한 시간을 T_p 라 할 때 이것을 다음과 같이 표현할 수 있음

$$- T_p = T_c + T_e$$

- T_c = 컴파일 시간, T_e = 실행 시간

② 컴파일 시간은 소스 프로그램을 컴파일하는데 걸리는 시간으로서 프로그램의 실행 특성에 의존하지 않기 때문에 고정적

③ 프로그램이 일단 정확하게 수행된다는 것이 검증되면, 그 프로그램을 재 컴파일하지 않고도 계속해서 반복해 실행시킬 수 있음. 그러므로 시간 복잡도를 비교할 때는 프로그램의 실제 실행 시간 T_e 만 고려하면 됨. 프로그램의 시간 복잡도를 분석하기 위해서는 프로그램 단계의 실행 빈도수를 계산함

④ 연산 시간 표기법: Big-oh(O)

- f 와 g 를 각각 양의 정수를 갖는 함수라 하면, 만일 어떤 두 양의 상수 a 와 b 가 존재하고, 모든 $n \geq b$ 에 대하여, $f(n) \leq a \cdot g(n)$ 이면 $f(n) = O(g(n))$ 임

- 만일 $f(n)$ 이 $O(g(n))$ 의 범위에 들어가면, 우리는 $f(n)$ 의 차수가 $g(n)$ 이라고 말하거나 $f(n)$ 는 $O(g(n))$ 이라고 함. 이 Big-oh 표기법의 기본 아이디어는 만일 $f(n)$ 가 $O(g(n))$ 이라면, n 이 계속적으로 무한히 커질 때 $f(n)$ 의 값은 결국 $g(n)$ 를 상한으로 점점 가깝게 점근적으로 한정되므로, $g(n)$ 를 $f(n)$ 의 어림값으로 볼 수 있다는 것. 그런 의미에서 Big-oh(O)를 점근식 표기법(asymptotic notation)이라 함

2) 실행메모리 분석

(1) 알고리즘을 실행하는데 필요한 공간(메모리)을 추정하여 알고리즘의 성능을 분석(performance analysis)

(2) 실행 메모리의 예측

① 프로그램을 실행시켜 완료하는 데 필요한 총 저장 공간. 일반적으로 어떤 프로그램 P가 필요로 하는 총 저장 공간을 S_p 라 할 때 이것은 다음과 같이 표현할 수 있음

$$- S_p = S_c + S_e$$

② 고정 공간(S_c): 프로그램의 크기나 입출력의 횟수에 관계없이 고정적으로 필요한 저장 공간

③ 가변 공간(S_e): 여기에는 실행 과정에서 자료 구조와 변수들이 필요로 하는 저장 공간이 포함됨. 또한 함수가 순환 호출을 할 때마다 추가로 필요로 하는 런타임 스택을 위한 저장 공간도 포함함

3) 성능 측정

(1) 컴퓨터가 실제로 프로그램을 실행하는데 걸리는 시간을 측정하여 알고리즘의 성능을 측정(performance measurement)

(2) 실행 시간의 측정

① 알고리즘의 성능을 측정한다는 것은 실제로 실행시간을 시계로 잴다는 것을 의미함

② 알고리즘의 실행파일을 얻기 위해 실제 프로그램을 작성해야 함

③ 일반적으로 프로그램의 실행은 운영체제에서 제공하는 시스템 시계를 이용함

- ④ 시스템 시계에 접근하는 시스템 연산 호출을 이용하면 쉽게 프로그램의 실행을 측정할 수 있음

<1장 출제예상문제>

1. 다음 중 알고리즘의 설명으로 적합한 것은 무엇인가?

- ① 경우에 따라 처리 결과가 생각될 수 있다.
- ② 입력은 반드시 내부에서만 제공되어야 한다.
- ③ 특정한 일을 수행하는 명령어의 유한 집합이다.
- ④ 경우에 따라 무한 수행이 가능해야 한다.

[정답] 3

[해설] 알고리즘은 컴퓨터가 수행해야 할 명령의 추상화이다.

2. 어떤 알고리즘의 분석 결과가 각각 다음과 같을 때, 가장 효율적인 결과는?

- ① $O(n^2)$
- ② $O(n)$
- ③ $O(2^n)$
- ④ $O(n \log n)$

[정답] 2

[해설] $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$

3. 알고리즘의 성능 분석을 위한 공간 복잡도를 바르게 나타낸 것은?

- ① 컴파일 시간과 실행 시간의 합계
- ② 고정 공간과 명령어 공간의 합
- ③ 고정 공간과 가변 공간의 합
- ④ 명령어의 빈도수 계산

[정답] 3

[해설] $Sp(\text{공간 복잡도}) = Sc(\text{고정 공간}) + Se(\text{가변 공간})$

4. 다음에서 선형 자료 구조의 종류가 아닌 것은?

- ① 배열
- ② 큐
- ③ 스택

④ 트리

[정답] 4

[해설] 선형자료 구조에는 배열, 레코드, 스택, 큐, 연결 리스트가 있다.

5. 자료구조에서 의미하는 자료와 정보에 대한 설명으로 잘못된 것은?

- ① 자료는 컴퓨터 처리에 의해 얻어진 종합적인 결과를 의미한다.
- ② 정보를 얻기 위해서는 프로그램이 필요하다.
- ③ 자료란 현실세계로부터 얻어지는 단순한 사실이나 값을 의미한다.
- ④ 정보란 어떤 상황에 대한 적절한 의사 결정을 할 수 있는 지식이다.

[정답] 1

[해설] 자료는 현실 세계에서 관찰이나 측정을 통해서 수집된 값(value)이나 사실(fact)을 말함. 이러한 값이나 사실은 수치(number)로 표현하거나 문자(character, string)로 표현됨. 반면에 정보는 어떤 상황에 대해서 적절한 의사결정(decision)을 할 수 있게 하는 지식(knowledge)으로서 자료의 유효한 해석이나 자료 상호간의 관계를 말함. 따라서 정보는 자료를 처리(process)해서 얻어진 결과(result)라고 할 수 있음. 이것을 수식으로 표현하면 $I = P(D)$ 로 나타낼 수 있으며, 이때 I는 정보, D는 자료, P는 처리를 의미한다.

6. 다음 중 알고리즘이 갖추어야 할 조건으로 틀린 것은?

- ① 한 가지 이상의 출력이 결과를 생성해야 한다.
- ② 각 명령은 명확하고 모호하지 않아야 한다.
- ③ 명령은 한정된 단계 뒤에는 반드시 종료되어야 한다.
- ④ 입력으로 외부에서 제공되는 자료가 있을 수 없다.

[정답] 4

[해설] 알고리즘이 갖추어야 할 조건은 다음과 같다.

- 입력: 외부에서 제공되는 자료가 있을 수 있다.
- 출력: 적어도 한 가지의 결과를 생성한다.
- 명확성: 각 명령들은 명확하고, 모호하지 않아야 한다.
- 유한성: 알고리즘의 명령대로 수행하면, 어떤 경우에도 한정된 수의 단계 뒤에는 반드시 종료한다.
- 유효성: 원칙적으로 모든 명령들은 종기와 연필만으로 수행될 수 있도록 기본적인어야 함. 이것은 각 연산이 명확해서만은 안 되고 반드시 실행 가능해야 한다.

7. 다음 빈 칸에 들어갈 말로 적당한 것은?

순환은 ()의 특성을 가진 문제에 적합하다. ()는 어떤 복잡한 문제를 직접 간단하게 풀 수 있는 작은 문제로 분할하여 해결하는 방법이다.

- ① 내부처리

- ② 분할정복
- ③ 순차처리
- ④ 계승함수

[정답] 2

[해설] 순환 방식에는 분할 정복(divide and conquer)의 특성을 가진 문제가 가장 적합. 이 분할 정복은 어떤 복잡한 문제를 직접 간단하게 풀 수 있는 작은 문제로 분할하여 해결하는 방법이다.

8. 추상화와 구체화는 서로 대칭되는 표현으로 알고리즘의 구체화 표현은?

- ① 추상형자료형
- ② 데이터
- ③ 프로그램
- ④ 자료형

[정답] 3

[해설] 알고리즘의 구체화 표현은 프로그램이다.

9. 다음이 설명하는 것은 무엇인가?

현실 세계에서 관찰이나 측정을 통해서 수집된 값이나 사실이다. 일반적으로 눈으로 보거나 귀로 듣거나 코로 냄새를 맡거나 해서 얻게 된다.

- ① 자료
- ② 배열
- ③ 정보
- ④ 의사

[정답] 1

[해설] 자료에 대한 설명이다.

10. 공통적인 개념을 이용하여 같은 종류의 다양한 객체를 정의하는 것을 다음 중 무엇이라고 하는가?

- ① 자료
- ② 알고리즘
- ③ 추상화
- ④ 출력

[정답] 2

[해설] 추상화란 공통적인 개념을 이용하여 같은 종류의 다양한 객체를 정의하는 것이다.

제2장 배열

1. 배열의 정의

1) 정의

- ① 배열은 일정한 차례나 간격에 따라 벌여 놓음이라고 표준국어사전에 정의되어 있음
- ② 배열 내에서 원소의 상대적 위치를 나타내는 것이 인덱스
- ③ 인덱스가 하나의 값으로 표현되면 1차원 배열, 2개의 값으로 표현되면 2차원 배열, n개의 값으로 표현되면 n차원 배열이라 함
- ④ 인덱스가 몇 개의 값으로 구성되든지 간에 원소의 위치를 나타내는 값이라는 의미에서 배열에서는 인덱스를 하나의 논리적 단위로 취급하고 있음

2) 배열의 특성

- ① 순차 표현이 연속된 메모리 블록을 이용하여 데이터를 표현하는 대표적인 예가 배열
- ② 배열은 순차적 메모리 할당 방식으로 <인덱스(index), 원소(element)> 쌍의 집합으로, 각 쌍은 어느 한 인덱스가 주어지면 그와 연관된 원소값이 결정되는 대응 관계를 나타냄
- ③ 인덱스는 순서를 나타내는 원소의 유한 집합
- ④ 배열의 원소들은 모두 같은 타입 즉, 동질의 값으로 그 크기도 같음
- ⑤ 배열의 접근 방법은 직접 접근

2. 배열 추상 자료형

1) 배열 추상 데이터 타입 설명

- ① 행의 연산 create(n)는 n개의 원소를 저장할 수 있는 공백 배열을 생성함. 초기에는 모든 원소값이 정의되지 않은 상태임
- ② 행의 연산 retrieve(a,i)는 Array a와 Index I를 매개 변수로 전달받아 Index를 검사하여 유효하면 이 Index에 대응되는 원소 e를 반환하고, 무효이면 error를 반환함
- ③ 행의 연산 store(a,i,e)는 Array, Index, Element를 매개 변수로 전달받아 Index를 검사하여 유효하면 <인덱스, 원소>쌍이 되게 원소를 저장한 뒤 Array를 반환함

3. 배열의 연산의 구현

1) 연산의 구현

- ① 배열을 공백 배열로 생성하는 연산(create)
- ② i 번째 인덱스에 저장되어 있는 원소 값을 반환하는 연산(retrieve)
- ③ i 번째 인덱스에 e 원소값을 저장하는 연산(store)

4. 1차원 배열

1) 1차원 배열의 표현

- ① 배열 중에 가장 단순한 배열은 1차원 배열임. 이 1차원 배열을 A라고 할 때 배열 원소 $A(i \ 1, i \ 2, \dots, i \ n)$ 의 메모리 위치가 효율적으로 결정될 수 있는 표현 방법을 하나 택하여야 함
- ② 배열의 순서나 위치를 표시할 때는 첨자나 인덱스를 사용함. 위 그림의 배열 이름은 A이고 n개의 크기를 가짐. 1차원 배열의 또 다른 표현은 배열 범위, 즉 하한선(low bound)과 상한선(high bound)으로 표현하

는 것

- $A(L:U)=\{A(i)\}$

- $i=L, L+1, \dots, U-1, U$

③ 예) 1차원 배열 A를 연속적인 주소의 개념을 사용하여 기억 장치에 표현

- 배열원소: $A[0], A[1], A[2], \dots, A[i], \dots, A[u-1]$

- 주소: $a, a+1, a+2, \dots, a+i, \dots, a+u-1$

- 총 원소수 = $u-1$

④ 위에서 배열 원소 $A[0]$ 의 주소가 a 라면 임의의 $A[i]$ 의 주소는 $a+i$ 가 되지만, 원소의 크기를 L 이라 할 때 주소는 $a+i \times L$ 이 됨. C언어에서 일차원 배열만을 생각함. C언어에서 일차원 배열은 변수의 이름 끝에 중괄호를 추가 하여 묵시적으로 선언 (`int list[5], *plist[5];`)

5. 배열의 확장

1) 2차원 배열의 표현

① 2차원 배열 $A[]$ 는 $A[m, n]$ 형식으로 선언된다. 여기서 m 은 행(row)을 나타내는 행의 수, n 은 열(column)을 나타내는 열의 수임. 2차원 배열을 위의 그림과 같이 $A[3, 4]$ 로 선언하였다고 함

② 먼저, 행우선 순서로 차례대로 배열의 원소를 $A[0, 0], A[0, 1], A[0, 2], A[0, 3], A[1, 0], A[1, 1], \dots, A[2, 3]$ 이 됨. 따라서 2차원 배열을 $A[m, n]$ 으로 선언했을 때, 임의의 $A[i, j]$ 의 주소를 구하는 일반식은 $A[i, j] = \alpha \times n + j$ 가 됨

2) 3차원 배열의 표현

3차원 배열 $A[m, n, p]$ 의 경우에는, 차원이 $[n, p]$ 인 2차원 배열이 u_0 개모인 것으로 해석하면 됨. 임의의 $A[i, j, k]$ 의 주소를 찾아내기 위해서 우선 $A[0, 0, 0]$ 의 주소를 α 라 할 때, 행우선 순서에서의 $A[i, j, k]$ 의 주소 = $\alpha + i \times n \times p + j \times p + k$ 가 되고, 열우선 순서에서의 $A[i, j, k]$ 의 주소 = $\alpha + k \times m \times n + j \times m + i$ 가 됨

6. 희소행렬의 개념

1) 희소 행렬의 개요

① 일반적으로 행렬은 m 개의 행과 n 개의 열로 구성되는데 이것을 $m \times n$ (m by n 이라 읽음)으로 표현하고 행렬의 차수(dimension)라 함

② $m \times n$ 행렬의 원소 수는 $(m \times n)$ 개가 됨. 만일 m 과 n 이 같으면 정방 행렬(square matrix)이라 함

③ 행렬 B는 7개의 행과 6개의 열로 된 7×6 행렬로서 42개의 원소를 가지고 있음

2) 희소 행렬의 표현 방법

이제 이러한 연산을 구현하기 위해서는 희소 행렬의 표현 방법을 결정해야 함. 행렬은 기본적으로 행과 열로 원소를 식별할 수 있기 때문에, 희소 행렬의 0이 아닌 원소에 대한 <행, 열, 값>의 3원소 쌍을 열이 3인 2차원 배열로 표현하면 됨

3) 희소 행렬의 전치 알고리즘

① 전치 행렬(transposed matrix)은 원소의 행과 열이 서로 교환된 행렬 즉, 원소 $\langle i, j, v \rangle$ 가 $\langle j, i, v \rangle$ 로 변환되어 만들어진 행렬

② 예를 들면 원소 $\langle 0, 4, 13 \rangle$ 과 $\langle 2, 5, 3 \rangle$ 은 전치 행렬에서 $\langle 4, 0, 13 \rangle$ 과 $\langle 5, 2, 3 \rangle$ 으로 각각 변환됨

③ 만일 주어진 행렬이 일반 $m \times n$ 행렬로서 $a[m, n]$ 배열로 표현되었다면, 전치 행렬을 표현하는 배열 $b[n,$

m]을 만들 수 있음. 이 방법에서는 배열 a[]를 열 별로 처리하고 결과는 배열 b[]에 행별로 저장

```
for ( j ← 0; j ≤ n-1 ; j ← j+1) do
  for ( i ← 0; i ≤ m-1; i ← i+1) do
    b[ j, i] ← a[i, j];
```

4) C에서의 최소 행렬 구현

- ① C언어에서 함수의 매개변수 전달은 원칙적으로 값 호출(call by value)임
- ② 매개변수가 배열인 경우 transposeS() 함수 정의에서와 같이 배열로 선언된 형식 매개변수는 실제로 포인터임
- ③ 이에 대응한 함수 호출문에서 배열이 인자(argument)로 함수에 전달될 때는 그 배열의 기본 주소 (base address)가 전달되며 배열의 내용 즉, 원소들의 값은 복사되거나 전달되지 않음
- ④ 배열 매개변수는 참조 호출(call by reference)로 전달되는 것과 같음
- ⑤ 함수 호출문 sum(a, SIZE)에서 a는 배열의 시작 주소를 나타내고 sum() 함수의 형식 매개변수로 명세된 b[]는 배열을 지시하는 포인터로서 인자 배열 a[]의 시작 주소 값을 전달 받음

<2장 출제예상문제>

1. 다음에서 2차원 배열 K(4:6, -2:1)의 원소의 개수는?

- ① 9
- ② 10
- ③ 12
- ④ 15

[정답] 3

[해설] 2차원 배열의 표현은 2차원 배열 a[]는 a[n1, n2] 형식으로 선언된다. n1은 행(row)을 나타내는 행의 수, n2는 열(column)을 나타내는 열의 수이며 2차원 배열 a[]의 원소 수는 $n1 \cdot n2$ 이고 각 원소는 <행 인덱스(i), 열 인덱스(j)>를 기초로 a[i, j]로 표현된다. 2차원에서 1차원으로 사상시키는 방법으로 행우선(row major order) 방법과 열우선(column major order) 방법이 있으며 2차원 배열 B[m][n]일 경우, 처음원소 : B(0,0) 주소가 α 일 때, 임의의 원소 B(i, j)의 주소는 행우선 : $\alpha + i * n + j$, 열우선 : $\alpha + j * m + i$ 이다.

2. 다음 배열 N(1:3, 1:2, 1:4)의 열우선 순서에 의해 기억 장소 할당을 바르게 기술한 항은?

- ① N(1,1,1) → N(2,1,1) → N(3,1,1)
- ② N(1,1,1) → N(1,2,1) → N(1,1,2)
- ③ N(1,1,1) → N(1,1,2) → N(1,1,3)
- ④ N(1,1,1) → N(2,1,1) → N(1,3,1)

[정답] 1

[해설] 열우선 순서에 의해 기억장소의 할당은 $N(1,1,1) \rightarrow N(2,1,1) \rightarrow N(3,1,1)$ 으로 할당된다.

3. 3차원 배열 $A(1:2, 1:2, 1:3)$ 의 12개 원소에 행우선 순서로 저장할 때 $A(1,2,1)$ 는 몇 번째가 되겠는가?

- ① 두 번째
- ② 세 번째
- ③ 네 번째
- ④ 다섯 번째

[정답] 3

[해설] 3차원 배열 표현으로 3차원 배열 $a[n_1, n_2, n_3]$ 의 메모리 표현은 n_1 개의 2차원 배열(크기가 $n_2 \times n_3$)을 다시 차례로 1차원 메모리에 순차적으로 사상시키는 것으로 생각하면 되며 원소 수는 당연히 $n_1 \cdot n_2 \cdot n_3$ 이 되고, 첫 번째 원소 $a[0, 0, 0]$ 의 주소를 α 라 할 때 임의의 원소 $a[i_1, i_2, i_3]$ 의 주소는 $\alpha + i_1 \cdot n_2 \cdot n_3 + i_2 \cdot n_3 + i_3$ 이 된다.

4. 다음 중에 희소 행렬에 대한 내용으로 맞게 기술한 것은?

- ① 다항식의 표현 방법으로 가장 적절한 행렬을 희소 행렬이라고 한다.
- ② 희소 행렬은 정방 배열에서만 나타날 수 있다.
- ③ 희소 행렬은 원소값이 대부분 0인 경우를 말한다.
- ④ 희소 행렬의 모든 원소를 메모리에 저장하는 것이 가장 효율적인 방법이다.

[정답] ③

[해설] 희소행렬은 (b)의 배열 $b[]$ 의 원소값도 대부분 0으로 채워져 있다. 전체 원소 수에 비하여 극소수의 원소만 0이 아닌 행렬을 희소 행렬(sparse matrix)이라 한다.

5. 배열의 개념에 대한 설명은?

- ① 불연속적인 메모리 블록에 할당하고, [인덱스, 원소] 쌍으로 표현된다.
- ② 불연속적인 메모리 블록에 할당하고, [매개변수, 원소] 쌍으로 표현된다.
- ③ 연속적인 메모리 블록에 할당하고, [매개변수, 원소]쌍으로 표현된다.
- ④ 배열은 연속적인 메모리 블록에 할당하고, [인덱스, 원소] 쌍으로 표현된다.

[정답] 2

[해설] 배열의 특성으로는 순차 표현이 연속된 메모리 블록을 이용하여 데이터를 표현하는 대표적인 예가 배열이며 배열은 순차적 메모리 할당 방식으로 <인덱스(index), 원소(element)> 쌍의 집합으로, 각 쌍은 어느 한 인덱스가 주어지면 그와 연관된 원소값이 결정되는 대응 관계를 나타내고 인덱스는 순서를 나타내는 원소의 유한 집합이다. 배열의 원소들은 모두 같은 타입 즉, 동질의 값으로 그 크기도 같으며 배열의 접근 방법은 직접 접근이다.

6. 배열 A[3][4]는 A[0][0]부터 B[2][3]까지이다. B[0][1]의 열우선 위치는?

- ① 1
- ② 3
- ③ 4
- ④ 5

[정답] 3

[해설] 행우선과 열우선 방식의 주소계산 공식은 행우선 방식에서 배열 A(l1:u1,l2:u2)에서, 처음 주소 A(l1,l2)= α 라 할 때, A(l1,2)= $\alpha+1$ 이고, 임의의 주소 A(i, j)= $\alpha+[(i-l1)u2 + (j-l2)]L$ (여기에서 L은 단위 원소의 크기)이며 열우선 방식에서 배열 A(l1:u1,l2:u2)에서, 처음주소 A(l1,l2)= α 라 할 때, A(2, l2)= $\alpha+1$ 이고, 임의의 주소 A(i, j)= $\alpha+[(j-l2)u1 + (i-l1)]L$

7. 배열을 생성하는 함수의 이름은 다음 중 무엇인가?

- ① create
- ② define
- ③ index
- ④ error

[정답] 1

[해설] 배열을 생성하는 함수의 이름은 create로 정의된다.

8. 배열이 인덱스 값을 이용해서 배열의 원소값에 접근하는 방식은 다음 중 무엇인가?

- ① 인접접근
- ② 직접접근
- ③ 도구접근
- ④ 순차접근

[정답] 2

[해설] 배열에 인덱스 값을 이용해서 배열의 원소값에 접근하기 때문에 직접 접근하게 된다.

9. 다음의 빈칸에 들어갈 알맞은 말은 무엇인가>

create(n)는 n개의 원소들을 저장할 수 있는 ()을 생성한다. 배열을 생성할 때 n개의 원소들을 저장할 수 있는 공간은 만들어지지만 그 안에 채워진 원소 값 들이 아직은 없다는 것을 의미한다.

- ① 희소배열
- ② 직접배열
- ③ 인덱스
- ④ 공백배열

[정답] 4

[해설] 공백배열에 대한 설명이다.

10. 행이 모두 연속적으로 메모리 영역을 할당받고 다음 행이 메모리 영역을 연속적으로 할당받는 방식의 행 우선 저장 방식을 가지는 배열은 다음 중 무엇인가?

- ① 1차 배열
- ② 2차 배열
- ③ 다중 배열
- ④ 단일 배열

[정답] 2

[해설] 2차 배열에 대한 설명이다.

제3장 스택

1. 스택의 개념

- ① 0개 이상의 원소를 갖는 유한 순서 리스트
- ② push(add)와 pop(delete)연산이 한곳에서 발생하는 자료구조
- ③ 객체와 그 객체가 저장되는 순서를 기억하는 방법에 관한 추상 자료형
- ④ 가장 먼저 입력된 자료가 가장 나중에 출력되는 관계를 표현함

2. 스택의 추상 자료형

1) 추상 자료형

- ① 행의 createStack() 연산자는 공백 스택을 생성함
- ② 행의 push(stack,item) 연산자는 스택의 top에 item을 삽입
- ③ 행의 isEmpty(stack) 연산자는 스택이 공백이면 true를 반환하고, 아니면 false를 반환
- ④ 행의 pop(stack) 연산자는 만약 스택이 공백이면 error를 반환하고, 아니면 스택의 top에서 한 item을 삭제하고 item 값을 반환
- ⑤ 행의 delete(stack) 연산자는 스택이 공백이면 error를 반환하고, 아니면 스택의 top이 가리키는 item을 삭제
- ⑥ 행의 peek(stack) 연산자는 스택이 공백이면 error를 반환하고, 아니면 스택의 top이 가리키는 item을 반환

3. 스택의 응용

1) 응용

- ① 컴퓨터 시스템 측면에서 아주 많은 분야에서 사용되고 있고, 따라서 응용분야가 다양함
- ② 변수에 대한 메모리의 할당과 수집을 위한 시스템 스택이 있음
- ③ 스택은 서브루틴의 수행이 끝난 후에 되돌아갈 함수 주소를 저장하기 위한 서브루틴 호출 관리를 위해 사용됨
- ④ 연산자들 간의 우선순위에 의해 계산 순서가 결정되는 수식계산이 있음
- ⑤ 프로그램이 수행 도중 발생오디는 인터럽트의 처리와 인터럽트 처리가 끝난 후에 되돌아갈 명령 수행지점을 정하기 위해 스택이 사용됨
- ⑥ 한 문자 한 문자씩 입력 받은 후에 명령어의 문법이 옳은지 검사하기 위한, 컴파일러가 있음
- ⑦ 함수가 자신을 함수로 되부르는 순환 호출의 실행이 끝나고 되돌아갈 실행 주소를 저장하기 위한 순환호출 관리 등에 스택은 매우 유용하게 사용됨

4. 스택의 연산

1) 스택의 삭제 연산

- ① pop 연산자는 스택에서 원소를 하나 제거하는 연산인데, 만일 스택이 비어 있는 상태이면 stackEmpty 메시지를 출력한다는 것을 의미함
- ② 스택이 비어있는 상태가 아니라면, 스택에는 어떤 원소가 있다는 것을 의미함
- ③ *top이 가리키는 주소 값을 하나 감소(-1)시키는 것임
- ④ *(top)--에서 사용된 '--' 연산자는 C/C++ 프로그래밍 언어에서만 사요오디는 독특한 단항연산자임

2) 스택의 삽입 연산

- ① 메인프로그램에서 Push를 호출하면 *top에 해당하는 실 매개변수는 주소를 나타내는 포인터 변수이고, 함수 push의 형식 매개변수에는 주소 값이 전달됨
- ② 스택에서 *top이 가리키는 주소 값을 하나 증가(+1)시키고, top이 가리키는 메모리 주소에 item으로 전달된 값을 저장함
- ③ ‘++’연산자가 사용되며 ++(*top)에서 사용된 ‘++’연산자는 C/C++ 프로그래밍 언어에서만 사용되는 독특한 단항 연산자임
- ④ *top이 가리키는 주소 값을 1만큼 증가(+1) 시킨 후에 top이 가리키는 주소 위치에 가서 값을 저장한다고 해석하면 됨

5. 배열을 이용한 스택의 구현

1) 스택 삭제 연산

- ① 스택에서 데이터를 삭제하고 반환하는 연산함수를 pop으로 정의함
- ② 스택의 공백 여부를 확인하기 위해 top 변수의 값이 -1인지 비교함
- ③ top의 값이 -1인 경우에는 스택이 비어 있다는 내용을 출력함, 제대로 연산이 이루어지지 않았음을 나타내는 0의 값을 반환하고 연산을 종료함
- ④ top의 값이 -1이 아닌 경우에는 데이터가 들어 있는 경우이므로 top 위치에 있는 데이터의 값을 반환하고 top 변수의 값을 하나 감소시킴

2) 스택 삽입 연산

- ① 스택에서 데이터를 삽입하는 연산 함수를 push로 정의함
- ② 스택에 데이터를 저장할 공간이 있는지를 확인하기 위해서 top 변수의 값이 스택의 사이즈보다 1 작은 값과 비교하여 크거나 같은지를 비교함
- ③ top의 값이 스택의 사이즈보다 1 작은 값과 비교하여 크거나 같은 경우에는 스택이 full 상태로 꽉 차 있다는 내용을 출력하고 연산을 종료함
- ④ 스택에 빈 공간이 있는 경우에는 top의 위치를 하나 증가시킨 후에 매개변수로 전달받은 데이터 item을 변경된 top의 위치에 저장함

6. 사칙연산식의 전위/후위/중위 표현

1) 전위 표기법

- ① ‘A+B’를 +AB와 같이 연산자 ‘+’를 피연산자인 A와 B앞에 놓고 식을 표현하는 방법으로 폴란드 수학자 루카지비치에 의해서 처음 소개됨
- ② polish notation이라고 함

2) 후위 표기법

- ① A+B를 AB+와 같이 표현하는 기법임
- ② ‘+’를 피연산자인 A와 B 뒤에 놓고 식을 표현하는 방법으로 우리가 일반적으로 사용하는 수식표현 방법인 중위표기법과 달리 후위 표기법이 컴퓨터가 해석하기에 훨씬 빠르고 간결한 표현법임

3) 스택을 이용한 후위 표기식의 계산

- ① 수식 계산을 하려면 일상생활 속에서 사용하는 중위 표기식을 후위 표기식으로 변화해야 함

- ② 후위 표기식으로 변환하는 과정에서 기억해야 할 것은 연산자(+,-,/,*)와 피연산자가 언제나 하나의 뭉치로 움직임
- ③ 스택을 이용한 후위 표기식의 생성과정에서 명심해야 할 것은 새로 입력되는 연산자와 스택의 가장 위에 저장되어 있는 연산자의 우선순위를 비교해야 한다는 것임
- ④ 스택의 가장 위에 저장되어 있는 연산자의 우선순위가 저장하려는 연산자의 우선순위보다 높거나 같으면, 스택의 가장 위에 저장된 연산자를 스택에서 삭제하고 출력해야 함
- ⑤ 입력된 연산자가 현재 스택의 가장 위에 저장되어 있는 연산자의 우선순위보다 높은 경우에만 새롭게 입력된 연산자가 스택에 저장됨

4) 후위 표기식의 계산 알고리즘과 설명

① 알고리즘

```

- element evalPostfix(char *exp) {    // 후위 표기식(369**+)을 계산하는 연산
-     int oper1, oper2, value, i=0
-     int length = strlen(exp);
-     char symbol;
-     top = -1
-     for(i=0; i<length; i++) {
-         symbol = exp[i];
-         if(symbol != '+' && symbol != '-' && symbol != '*' && symbol != '/') {
-             value = symbol - '0';
-             push(value);
-         }
-         else {
-             oper2 = pop( );
-             oper1= pop( );
-             switch(symbol) {
-                 case '+': push(oper1 + oper2); break;
-                 case '-': push(oper1 - oper2); break;
-                 case '*': push(oper1 * oper2); break;
-                 case '/': push(oper1 / oper2); break;
-             }
-         }
-     }
-     return pop( );
- }

```

② 알고리즘 설명

- 스택을 사용하여 후위 표기식을 계산하는 함수를 evalPostfix라고 정의함
- 피연산자의 변수를 정의하며 사칙연산을 하기 위해서는 연산자 두 개가 필요함
- char형 포인터 매개변수로 전달받은 후위 표기식의 저장된 배열의 길이를 계산하여 length 변수에 저장함
- 배열에서 값을 가져와 저장할 변수 symbol을 정의함

- 스택의 위치를 가리키는 top 변수를 -1로 초기화함
- 반복문이 시작됨
- 처음 i가 0이므로 exp[0]의 값 '3'을 symbol 변수에 저장함
- symbol 변수의 값이 연산자가 아닌 피연산자인지 비교함
- 피연산자를 스택에 저장함

<3장 출제예상문제>

1. 다음 중 LIFO 알고리즘을 갖는 순서 리스트는?

- ① 큐
- ② 스택
- ③ 배열
- ④ 연결 리스트

[정답] ②

[해설] 스택은 제일 나중에 삽입된 원소가 제일 먼저 삭제되므로 후입 선출(LIFO:Last-In-First-Out) 리스트라고 하며 스택에서의 삽입은 push() 삭제는 pop()이라고 한다.

2. 다음 중위 표기식을 후위 표기식으로 바르게 나타낸 항은?

$$A*(B+C)/D-P$$

- ① $ABC+D/*P-$
- ② $BC+D/A*P-$
- ③ $ABC+*D/P-$
- ④ $BC+A*D/P-$

[정답] ③

[해설] 괄호에 싸여있는 수식을 가장 먼저 계산하고 +보다는 우선순위가 높은 *나 /가 계산되어야 한다.

3. 다음 중위 표기식을 전위 표기식으로 바르게 나타낸 항은?

$$A-B*C/D$$

- ① $-A/*BCD$
- ② $-*AB/CD$
- ③ $-*BC/DA$
- ④ $*/-ABCD$

[정답] ①

[해설] 가장 먼저 괄호로 둘러싸인 부분을 먼저 계산하고 그러고 난 후에 나누기 연산자가 빼기 연산자보다 우선순위가 높기 때문에 나누기 연산자를 먼저 계산해야 한다.

4. 다음 중 스택의 응용 분야로서 부적합한 항은?

- ① 순환 호출
- ② 인터럽트 처리
- ③ 서브루틴 호출
- ④ 작업의 스케줄링

[정답] ④

[해설] 스택의 응용 분야로는 우리 주위의 일상생활에서 택시 기사들이 사용하는 동전 보관통 외에 뷔페식당에 쌓아 둔 접시들이 좋은 예이며 컴퓨터 시스템 측면에서는 응용 분야가 다양한데, 운영 체제가 사용하는 시스템 스택(system stack), 서브루틴 호출(subroutine call), 수식 계산(evaluation of expression), 인터럽트(interrupt) 처리, 컴파일러(compiler), 순환 호출(recursion call) 등에 매우 유용하게 사용되고 있다.

5. 스택이 저장할 수 있는 최대개수의 element를 의미하는 것은 무엇인가?

- ① maxStack
- ② stack
- ③ IsFull
- ④ Boolean

[정답] 1

[해설] 매개변수인 maxStack은 스택이 저장할 수 있는 최대 개수의 element를 의미한다.

6. 다음이 설명하는 것은 무엇인가?

스택이 빈 상태라면 삭제 할 원소가 없으므로 'stackEmpty'를 출력한다. 하지만 빈 상태가 아니라면 삭제 할 원소가 있으므로, 스택의 top이 가리키는 원소를 삭제하고 그 원소를 반환한다.

- ① maxStack
- ② Stack push
- ③ Element Pop
- ④ infix notation

[정답] 3

[해설] Element Pop에 대한 설명이다.

7. 연산자를 연산자 앞에 표기하는 방법은 다음 중 무엇인가?

- ① 중의 표기법
- ② 후위 표기법
- ③ 전위 표기법
- ④ 선행 표기법

[정답] 3

[해설] 전위표기법은 연산자를 피연산자 앞에 표기하는 방법(+AB)이다.

8. 다음의 용어에 대해서 올바르게 설명하지 않은 것은?

- ① 스택: 객체와 그 객체가 저장되는 순서를 기억하는 방법에 관한 추상자료형
- ② 중위표기법: 변수에 대한 메모리의 할당과 수집을 위해 운영체제가 관리하는 스택
- ③ 전위표기법: 연산자를 피연산자의 앞에 표기하는 방법
- ④ 후기표기법: 연산자를 피연산자의 뒤에 표기하는 방법

[정답] 2

[해설] ②는 시스템 스택에 대한 설명이다.

9. 땅속에 박혀 있는 관에서 공을 넣을 수 있는 입구 부분이라고 정의하는 스택의 부분은 다음 중 무엇인가?

- ① 시스템 스택
- ② 전위표기법
- ③ XML
- ④ 스택의 톱

[정답] 4

[해설] 스택의 톱에 대한 설명이다.

10. 다음 중 스택을 생성하는 연산은 무엇인가?

- ① CreateS
- ② element
- ③ maxStracksize
- ④ IsFull

[정답] 1

[해설] CreateS은 스택을 생성하는 연산이다.

제4장 큐

1. 큐의 개념

1) 개념

- ① 큐의 응용분야는 우리의 일상생활에서 많이 볼 수 있음
- ② 큐는 가장 처음에 제출되어 작업대기 줄에 들어간 작업이 가장 처음에 처리되어 작업 스케줄이 만들어짐
- ③ 큐는 양쪽이 터진 관이라고 생각하면 됨

2. 큐의 추상 자료형

- ① 큐(queue)는 한쪽 끝에서는 항목들(items)이 삭제되고 다른 한쪽 끝에서는 항목들이 삽입되는 1차원 배열의 선형 리스트(linear list)를 의미
- ② 항목들이 삭제되는 끝을 앞(front)이라 하고 삽입되는 끝을 뒤(rear)라고 함
- ③ 큐에 먼저 삽입된 항목이 먼저 삭제되므로 선입 선출(FIFO : First-In-First-Out) 또는 선착순 서브(FCFS : first-come-first-serve) 알고리즘을 갖는 순서 리스트임
- ④ 큐를 컴퓨터 시스템에 표현하는 가장 간단한 방법은 1차원 배열을 사용하는 것
- ⑤ 큐 추상 데이터 타입

ADT Queue

데이터 : 0개 이상의 원소를 가진 유한 순서 리스트

연산 :

$queue \in Queue; item \in Element;$

① createQ() ::= create an empty queue;

② enqueue(queue, item) ::= insert item at the rear of queue; //큐의 rear에 item을 삽입//

③ isEmpty(queue) ::= if (queue is empty) then return true else return false;

//큐가 비었으면 true를 반환하고, 아니면 false를 반환//

④ dequeue(queue) ::= if (isEmpty(queue)) then return error

else {delete and return the front item of queue}; //만약 큐가 공백(isEmpty(queue))

이면 error를 반환하고, 아니면 큐의 front가 가리키는 item을 삭제하고 그 값을 반환//

⑤ delete(queue) ::= if (isEmpty(queue)) then return error

else {delete the front item of queue}; //만약 큐가 공백(isEmpty(queue))이면 error

을 반환하고, 아니면 큐의 front item을 삭제//

⑥ peek(queue) ::= if (isEmpty(queue)) then return error

else {return the front item of queue}; //만약 큐가 공백(isEmpty(queue))이면 error

을 반환하고, 아니면 큐의 front item을 반환//

End Queue

3. 큐의 응용

1) 응용

- ① 중앙처리장치(CPU)는 컴퓨팅 자원 중에서 아주 중요한 자원임
- ② FCFS스케줄링은 작업이 준비 큐에 도착한 순서대로 CPU를 할당받도록 해주는 기법임

③ RR 스케줄링 기법은 대화형 시스템에 사용되는 스케줄링 방식임

4. 배열을 이용한 큐의 구현

1) 큐의 생성

변수 rear의 초기값은 큐의 공백 상태를 나타내는 '-1'로 시작함

2) 큐의 삽입 연산

- ① 삽입 연산이 발생하면 rear 변수만 오른쪽으로 이동함
- ② 메인 프로그램에서 Add_q(int *rear, element item)를 호출하면 '*rear'에 해당하는 실 매개변수는 메모리에서 큐의 처음을 나타내는 주소를 가리키는 포인터 변수를 전달하고, 함수 Add_q(int *rear, element item)의 형식 매개변수에서는 rear 변수의 주소로 전달함
- ③ 함수 Add_q는 먼저 *rear변수의 상태를 검사함
- ④ 큐가 full 상태가 아니라면 큐의 원소를 삽입할 수 있는 공간이 있음 따라서 원소를 삽입하기 위해 *rear 값을 하나 증가시킨 위치에 item 값을 저장하게 됨

3) 큐의 삭제 연산

- ① 메인 프로그램에서 delete_q(int *front, in rear)를 호출하면, '*front'에 해당하는 실 매개변수는 메모리에서 큐의 삭제가 발생하는 위치를 가리키는 포인터 변수로 전달됨
- ② 함수 Delete_q는 먼저 큐의 상태를 검사함
- ③ 큐가 빈 상태가 아니라면 큐에서 원소를 삭제할 수 있다는 것을 의미함

5. 원형 큐

1) 정의

- ① 원형 큐는 파이프의 입구와 출구 부분을 연결시킨 형태임, 연결된 부분의 데이터 공간을 연속적으로 사용하기 위해 '나머지 연산자'를 사용함
- ② 원형 큐는 mod 연산자를 사용하여 n 개의 공간을 (0 : n-1)로 운용함
- ③ 원형 큐에서의 삽입(add)과 삭제(delete) 알고리즘은 일반 큐의 삽입과 삭제 알고리즘과 약간 다른데 새로운 항목의 삽입을 위해서는 rear를 시계 방향으로 하나 이동시켜야 함
- ④ 즉, rear의 값을 1 증가시켜야 하는데 rear를 Q(n-1)에서 시계 방향으로 하나 이동시키면 Q(0)가 되어야 하므로, 이를 수식으로 표현하기 위해 나머지를 계산하는 mod(modulus) 연산자를 사용함
- ⑤ 원형 큐에 항목을 삽입하는 경우 : $rear \leftarrow (rear+1) \bmod n$
- ⑥ 원형 큐에 항목을 삭제하는 경우 : $front \leftarrow (front+1) \bmod n$

<4장 출제예상문제>

1. 큐가 저장할 수 있는 최대 개수의 원소를 의미하는 것은 무엇인가?

- ① MaxQueue
- ② FCFS
- ③ rear

④ queueFull

[정답] 1

[해설] 매개변수인 maxQueue는 큐가 저장할 수 있는 최대 개수의 원소를 의미한다.

2. 큐의 상태가 빈 상태인지 확인하는 연산은 다음 중 무엇인가?

- ① MaxQueue
- ② FCFS
- ③ rear
- ④ Boolean IsFull_q

[정답] 4

[해설] Boolean IsFull_q는 큐의 상태가 빈 상태인지를 우선적으로 확인하는 연산이다.

3. 다음이 설명하는 것은 무엇인가?

작업이 도착한 순서대로 CPU가 할당되지만, CPU의 시간 할당량 또는 시간 간격에 의해 제한을 받으며, 일정한 크기의 시간 할당량을 모든 작업에 주고 그 시간 동안 작업이 완료되지 못하면 준비 큐의 맨 뒤에 다시 배치되는 기법이다.

- ① RR 스케줄링 기법
- ② FCFS 스케줄링 기법
- ③ 큐
- ④ 원형 큐

[정답] 1

[해설] RR 스케줄링 기법에 대한 설명이다.

4. 큐에 원소를 삽입하는 기본연산은 다음 중 무엇인가?

- ① FCFS
- ② rear
- ③ Boolean IsFull_q
- ④ Add_q

[정답] 4

[해설] Add_q연산자는 큐에 원소를 하나 삽입하는 연산자이다.

5. 원형 큐의 특징은?

- ① 빈 공간을 없애기 위해 front, rear를 재설정한다.
- ② 빈 공간을 없애기 위해 앞으로의 이동이 필요하다.
- ③ 만원 큐이지만 빈 공간이 있을 수 있다.
- ④ mod 연산자를 이용하여 삽입 삭제가 이루어진다.

[정답] 4

[해설] 원형 큐는 mod 연산자를 사용하여 n개의 공간을 (0 : n-1)로 운용하며 원형 큐에서의 삽입(add)과 삭제(delete) 알고리즘은 일반 큐의 삽입과 삭제 알고리즘과 약간 다른데 새로운 항목의 삽입을 위해서는 rear를 시계 방향으로 하나 이동시켜야 한다. 즉, rear의 값을 1 증가시켜야 하는데 rear를 Q(n-1)에서 시계 방향으로 하나 이동시키면 Q(0)가 되어야 하므로, 이를 수식으로 표현하기 위해 나머지를 계산하는 mod(modulus) 연산자를 사용한다.

6. 큐의 논리 구조는?

- ① 후입선출
- ② 선입후출
- ③ 선입선출
- ④ 임의입출

[정답] 3

[해설] 큐에 먼저 삽입된 항목이 먼저 삭제되므로 선입 선출(FIFO : First-In-First-Out) 또는 선착순 서브(FCFS : first-come-first-serve) 알고리즘을 갖는 순서 리스트이다.

7. 다음 중 데이터의 삽입 삭제가 양쪽 끝에서 이루어지는 자료구조는?

- ① front
- ② pointer
- ③ array
- ④ deque

[정답] 4

[해설] 데크(deque)은 스택과 큐의 동작을 복합시킨 방식으로 수행하는 선형 리스트이다. 'double ended queue'의 약자로서 deque라고 하며, 데크의 새로운 변수로는 큐의 front 대신에 left-pointer와 rear 대신에 right-pointer 변수가 있다. 어느 한쪽으로부터 삽입하고 양쪽으로 삭제할 수 있는 자료 구조를 스크롤(scroll) 리스트라 하고, 반대로 삽입은 양쪽으로 하고 삭제는 어느 한쪽으로 할 수 있는 자료 구조를 쉘프(shelf) 리스트라고 한다.

8. 다음이 설명하는 용어에 대해 옳바르지 않은 것은 무엇인가?

- ① FCFS 스케줄링: 작업이 준비 큐에 도착한 순서대로 CPU를 할당받도록 해 주는 기법

- ② RR 스케줄링: 작업이 도착한 순서대로 CPU가 할당되지만, CPU의 시간 할당량 또는 시간 간격에 의해 제한을 받으며, 일정한 크기의 시간 할당량을 모든 작업에 주고 그 시간 동안 작업이 완료되지 못하면 준비 큐의 맨 뒤에 다시 배치되는 기법
- ③ 원형큐: 파이프의 입구와 출구 부분을 연결시킨 형태이며, 큐의 양 끝을 연결시켜서 원으로 만든 형태의 큐
- ④ 큐: 원소의 삭제 연산이 이루어지는 곳

[정답] 4

[해설] 큐는 한쪽에서 삽입이 발생하고 다른 한쪽에서 삭제가 발생하도록 정의되었으며, 먼저 삽입된 원소가 먼저 삭제되므로 선입선출 또는 선착순 서브 알고리즘을 갖는 순서 리스트이다.

9. 다음 중 큐에서 원소를 하나 삭제하는 연산자는 무엇인가?

- ① Delete_q
- ② rear
- ③ Boolean IsFull_q
- ④ Add_q

[정답] 1

[해설] Delete_q에 대한 설명이다.

10. 작업이 준비 큐에 도착한 순서대로 CPU를 할당받도록 해 주는 기법은 무엇인가?

- ① FCFS 스케줄링
- ② RR 스케줄링
- ③ 원형큐
- ④ 큐

[정답] 1

[해설] FCFS 스케줄링에 대한 설명이다.

제5장 연결 리스트

1. 리스트의 개념

1) 개념

- ① ‘일정한 순서’의 나열
- ② 어떤 정의에 의해서 결정된 ‘논리적인 순서’의 나열
- ③ 리스트의 ‘순서’는 데이터가 저장되는 물리적인 위치와 상관없이 사람들의 머릿속에 인식되는 ‘논리적인 순서’, 혹은 리스트에 나타나는 원소들 간의 ‘의미적인 순서’를 의미함
- ④ 물품이나 사람의 이름 따위를 일정한 순서로 적어 놓은 것
- ⑤ 배열은 인덱스로 표현되는 ‘순서’가 배열 원소의 메모리 공간(주기억 장치, DDR)에서의 물리적인 위치를 의미함
- ⑥ 하지만 리스트의 ‘순서’ 개념은 어떤 정의에 의해서 결정된 ‘논리적인 순서’임
- ⑦ 원소들의 물리적인 저장 순서나 위치와는 무관하게 원소들 간의 논리적인 순서만 유지함

2) 리스트의 구현 방법

- ① 포인터를 이용한 리스트의 구현 방법 : 원소값을 저장하는 공간과 다음 원소를 가리키는 위치 정보를 저장하는 공간을 함께 구현하는 방법
- ② 배열을 이용하여 리스트의 구현 방법

2. 배열을 이용한 리스트의 구현

1) 개념

배열을 이용한 리스트를 구현하는 방법은 리스트의 원소값을 순서대로 배열에 저장함

2) 배열을 이용한 리스트의 원소 삽입

배열의 확장 : 초기 배열 선언에서 충분히 크게 하면 어느 정도는 피할 수 있겠지만, 원소를 리스트의 중간에 삽입하기 위해서는 리스트의 원소값을 하나씩 뒤로 밀어야 하는 상황이 발생함

3) 배열을 이용한 리스트의 원소 삽입 / 삭제

‘배열로 구현된 리스트’는 원소의 순서가 연속적인 물리적 주소에 저장됨

- ⇒ 원소를 삽입하거나 삭제하기 위해서는 해당 원소의 위치 뒤에 있는 모든 원소를 뒤로 물리거나 앞으로 당겨야만 됨
- ⇒ 리스트 원소값의 이동은 원소수가 많을수록 프로그램의 수행시간을 증가시킴

4) 배열을 이용한 리스트의 원소 삽입 / 삭제 시 발생하는 문제

- ① 리스트의 원소 삽입은 프로그램의 실행 중에 메모리 할당을 필요로 하는 경우도 발생시킴
- ② 배열을 이용한 리스트의 구현은 실제 IT 서비스 환경에서는 자주 사용되지 않고 있음
- ③ 자료의 삽입과 삭제가 빈번히 발생하는 상황에서 리스트를 배열로 구현하는 것은 빈번한 자료 이동으로 인한 비효율적인 컴퓨팅 성능을 유발함

3. 포인터를 이용한 리스트의 구현

1) 개념

- ① 원소의 자리에는 원소값을 저장하고, 다음 원소를 가리키는 정보의 자리에는 다음 원소가 저장될 위치의 주소 값을 저장함
- ② 리스트의 원소의 자리와 다음 원소를 가리키는 정보의 자리를 합쳐서 노드라고 함
- ③ 노드에는 데이터 요소와 리스트의 다음 원소를 지시하는 포인터가 있다고 생각하면 됨
- ④ 연결 리스트는 리스트 원소들의 논리적 순서만을 지원하며 실제로 리스트 원소의 저장 위치는 순차적으로 이루어지지 않음
- ⑤ 실제로 연결 리스트는 컴퓨터 메모리에서 저장되고, 원래 프로그래머가 추상적으로 생각하는 리스트 원소의 순서와는 다르게 컴퓨터 메모리에 저장됨

4. 포인터 변수

1) 구조체 포인터 타입

- ① 다양한 데이터형 변수를 하나의 상자 안에 넣어서 선언하거나 사용하는 C프로그래밍 문법이 구조체임
- ② 구조체는 다양한 데이터형을 하나의 단위로 다룰 수 있어서 하나의 객체에 대한 다양한 정보를 모아서 사용할 경우에 유용함
- ③ 연결리스트의 원소를 하나의 구조체로 정의하고, 여러 개의 구조체를 링크로 연결하여 연결리스트를 구현함
- ④ struct의 멤버는 같은 타입의 또 다른 struct를 지시하는 포인터도 될 수 있음. 이것을 자체 참조 구조라 하는데 이것은 리스트의 노드를 정의하는 데 유용함

```
struct char_list_node {
    char letter;
    struct char_list_node *next;
};
struct char_list_node *p;
```

- ⑤ 이 선언문은 struct 형의 char_list_node를 선언하는 동시에 p를 이 struct char_list_node형에 대한 포인터로 선언하고 있음
- ⑥ 이 선언은 char_list_node 형의 노드로 구성된 연결 리스트를 형성할 수 있게 하고 포인터 p는 이 리스트의 노드를 지시할 수 있게 함
- ⑦ 이 포인터 p가 가리키는 노드에 연결된 다음 노드를 접근하기 위해서는 다음과 같이 포인터를 순회시키면 됨

```
p = p->next;
```

- ⑧ 이렇게 하면 포인터 p는 다음 노드를 지시하게 되는데 이 struct는 멤버인 next에 다음 노드의 주소를 저장하게 되어 있기 때문임
- ⑨ 리스트 처리를 위해 노드와 포인터를 정의할 때 typedef를 이용하면 간결해짐
- ⑩ 포인터 형 list_pointer는 아직 정의 되지 않은 char_list_node라는 struct 형을 이용하여 정의하였음. C언어는 아직 정의되지 않은 자료형에 대한 포인터를 미리 선언할 수 있게 허용함
- ⑪ 노드 구조가 정의된 뒤에 포인터 p는 NULL 값으로 초기화 되었는데 이는 p가 지시하는 주소가 없다는 것을 명시적으로 표현하는 것

2) 프로그램 실행 중의 구조체 메모리 할당

- ① 배열을 이용한 리스트의 구현과 포인터를 이용한 리스트의 구현은 확장성에서도 차이가 있을 수 있음
- ② 프로그램 실행 중에 동적으로 메모리 공간을 할당받아 리스트 원소를 새롭게 할당받은 메모리 공간에 저장하고 새롭게 할당받은 메모리 공간을 연결할 수 있는 구현방법이 C 프로그래밍 언어에서 제공함
- ① 포인터 변수를 이용한 연결 리스트는 프로그램 실행 도중에 값을 저장할 메모리가 필요하게 될 때 메모리를 할당받아 데이터를 처리하고, 사용이 끝난 메모리를 반환함
- ② C언어에서는 malloc()이란 함수를 이용하여, 메모리를 할당받고, 더 이상 필요 없게 되면 free()라는 함수를 이용해 이 메모리 영역을 시스템에 반환함
- ③ malloc() 함수를 호출할 때는 int나 float와 같은 자료형을 인자로 갖는 sizeof() 함수를 사용하여 필요한 기억장소의 크기에 대한 정보를 제공함
- ④ malloc() 함수의 반환 값은 요청한 크기의 메모리 영역에 대한 첫 번째 주소가 되므로 이를 요청한 자료형과 일치하도록 형 변환(type cast)을 하고 포인터 변수에 지정. 함수 free()는 이전에 malloc()으로 할당된 메모리 영역을 더 이상 사용하지 않을 때 자유 공간 리스트(free space list)에 반환

5. 연결 리스트에서 노드의 삽입과 삭제

1) 개념

- ① 연결리스트의 원소에 대한 대표적인 연산은 원소(노드)의 삽입과 삭제임
- ② 리스트 원소 삭제는 원하는 리스트의 원소를 리스트로부터 제거하는 것임
- ③ 리스트 원소 삽입은 원하는 리스트의 원소를 리스트의 적절한 위치에 추가하는 것임

6. 연결리스트의 여러 가지 연산프로그램

1) 연결리스트의 생성

- ① 연결리스트에서 원소는 연결될 다음 원소에 대한 주소를 저장해야 함
- ② 노드는 원소의 값을 저장하는 데이터 필드와 다음 노드의 주소를 저장할 수 있는 링크 필드로 구성됨

2) 연결리스트의 노드 삽입

- ① 노드를 추가하는 함수의 이름을 addNode라고 정의함
- ② 새롭게 만든 노드를 NewNode라고 정의함
- ③ 연결 리스트의 노드들이 많은 경우 마지막 노드를 LastNode라고 정의함
- ④ Mallic 함수를 사용하여 NewNode의 데이터 공간을 메모리에 할당받음

3) 연결리스트의 노드 삭제

- ① 레드 노드에서 시작하여 링크 노드의 값을 따라가면서 마지막 노드를 찾아서 삭제를 해주어야 함
- ② 맨 뒤의 노드가 삭제되면서 삭제되는 노드 앞의 선행 노드의 링크 필드 값은 NULL이 되어야 함
- ③ 삭제할 때는 연결리스트의 노드가 존재하는지, 노드가 한 개만 존재하는지, 노드가 여러 개 존재하는지를 고려해야 함

4) 연결리스트의 특정 노드 위에 삽입

- ① 특정 노드의 위치는 prevNode라고 정의하고, 삽입할 노드의 데이터 값과 함께 매개변수로 전달받음
- ② 특정노드의 삽입연산일 경우도 연결 리스트의 공백 여부 및 삽입할 노드의 위치를 찾기 위한 탐색과정이 필요함

5) 연결리스트의 특정 노드 검색

- ① 특정노드를 검색하려면 연결 리스트의 헤드 노드로부터 출발함
- ② 링크를 따라가면서 노드의 데이터 값을 원하는 데이터 값과 비교하여 같을 경우 존재 여부를 출력함
- ③ 간단하게 연결리스트에는 노드들이 여러 개 존재하며 찾고자 하는 노드가 반드시 있는 것으로 가정 함

<5장 출제예상문제>

1. 연결 리스트에 관한 설명으로 틀리게 기술된 항은?

- ① 연결 리스트는 자료 필드와 링크 필드로 구성된다.
- ② 자료 필드에 기억 장소의 주소가 기억될 수 있다.
- ③ 연결 리스트는 기억 공간상의 불연속한 기억 장소를 활용하는 자료 구조이다.
- ④ 연결 리스트의 끝은 일반적으로 NULL값(혹은 0값)이 기억된다.

[정답] ②

[해설] 주소 값은 링크 필드에서 저장한다.

2. 이중 연결 리스트의 가장 큰 장점은?

- ① 다음 노드로의 이동을 쉽게 할 수 있다.
- ② 자료의 삭제를 쉽게 할 수 있다.
- ③ 기억 장소의 낭비가 적다.
- ④ 현재 노드의 선행 노드를 쉽게 찾을 수 있다.

[정답] ④

[해설] 이중 연결 리스트는 단순 연결 리스트의 장점은 포인터가 현재 어떤 노드 p를 가리키고 있을 때 링크의 방향, 즉 다음 노드로의 이동은 쉽게 이루어질 수 있으며 노드 p의 선행(좌측) 노드를 찾아내려면 리스트의 처음부터 다시 찾거나 원형 연결 리스트의 경우, 리스트를 한 바퀴 순회해야만 하는 단점이 있다. 데이터 값이 오름차순으로 정렬되어 있는 단순 연결 리스트 t에서 삽입할 데이터(x)가 어떤 노드 p의 데이터 값보다 작고 p의 앞쪽 노드 값보다는 클 때, 노드 p 앞에 삽입하기 위해서 노드 p의 선행 노드의 링크 필드를 변경시켜야 하므로 p의 선행 노드를 처음부터 찾아야 한다.

3. 연결 리스트 구성을 위한 노드 구조를 바르게 설명한 항은?

- ① 자료 부분과 링크 부분으로 구성된다.
- ② 자료 부분으로만 구성된다.
- ③ 링크 부분으로만 구성된다.
- ④ storage-pool로 구성된다.

[정답] ①

[해설] 노드 구조의 논리적 표현은 데이터-링크 이다.

4. 다음은 이중 연결 리스트의 노드 x를 삭제하는 프로그램의 일부이다. 프로그램 7번째 행에 있는 괄호 (A) 속에 들어가야 할 적합한 내용은?

```
void delete(list_pointer x, list_pointer L){
    if(x == L){
        no_more_nodes();
        return;
    }

    x -> llink -> rlink = x -> rlink;
    (                A                )

    free(x);
}
```

- ① x -> llink -> rlink = x -> rlink;
- ② x -> llink -> rlink = x -> llink;
- ③ x -> rlink -> llink = x -> llink;
- ④ x -> rlink -> llink = x -> rlink;

[정답] ③

[해설] 이중 연결 원형 리스트 L에서 임의의 노드 x를 삭제하는 알고리즘으로서 다음 과 같이 풀이된다.

```
typedef struct list_node *list_pointer;
typedef struct list_node {
    char data;
    struct list_node *rlink;
    struct list_node *llink;
} list_node;
```

- ① void delete(list_pointer x, list_pointer L) {
- ② if (x == L) {
 no_more_nodes();
 return;
 }
- ③ x->llink->rlink=x->rlink;
- ④ x->rlink->llink=x->llink;
- ⑤ free(x);
- ⑥ }

5. 다음 중 연결리스트에 대한 설명으로 잘못된 것은?

- ① 원소는 자료이고, 주소는 링크이다.
- ② 연결리스트의 노드구조는 [원소, 주소] 쌍이다.
- ③ 링크 필드는 노드의 자료(data) 값을 저장한다.
- ④ 연결 리스트는 배열과는 다른 비순차적 표현이다.

[정답] 3

[해설] ③ 일반적으로 노드는 데이터 필드와 링크 필드로 구분된다. 데이터 필드(data field)는 리스트의 원소 즉, 데이터 값을 저장하는 곳이고, 링크 필드(link field)는 포인터(pointer) 즉, 다음 노드의 주소 값을 저장하는 곳이다.

6. 다음 빈 칸에 들어갈 말로 알맞은 것은?

연결리스트에서 노드의 사용을 위한 할당과 사용이 끝난 노드의 반환을 위해 () 리스트가 사용된다.

- ① 이중연결
- ② 단순연결
- ③ 원형연결
- ④ 자유공간

[정답] 4

[해설] 자유공간리스트는 자유 공간(free space)은 필요에 따라 요구하는 노드를 할당할 수 있도록 준비해둔 자유 메모리 풀(memory pool)이며 자유공간을 연결 리스트 구조를 이용하여 관리할 수 있는데 이를 위해서 초기에 사용할 수 있는 모든 메모리를 자유 공간 리스트(free space list)로 만들어 놓는다고 가정한다. 노드 할당 요청이 들어오면 자유 공간 리스트 앞에서부터 공백 노드를 할당하고 프로그램을 실행하는 기간 동안 메모리를 일종의 용품으로 취급한다.

7. 리스트의 원소의 자리와 다음 원소를 가리키는 정보의 자리는 다음 중 무엇인가?

- ① 포인터
- ② 노드
- ③ 리스트
- ④ 링크

[정답] 2

[해설] 노드는 리스트의 원소의 자리와 다음 원소를 가리키는 정보의 자리를 합쳐 이르는 말이다.

8. 다음 용어의 설명으로 적절하지 않은 것은 무엇인가?

- ① 리스트: 원소값과 다음 원소를 가리키는 위치의 주소 값으로 구성된 자료 단위
- ② 포인터: 메모리에 저장되는 값의 저장 위치에 대한 주소를 가리키는 데이터 형

- ③ 단항연산자: 피연산자 하나만 갖는 연산자
- ④ 구조체: 다양한 데이터형의 변수를 하나의 상자 안에 넣어서 선언하거나 사용하는 C프로그래밍 문법

[정답] 1

[해설] 리스트는 원소들 간의 순서가 지켜지며 유지되는 자료구조이다.

9. 다음이 설명하는 것은 무엇인가?

메모리에 저장되는 값의 위치이다. 메모리에 저장되는 값은 저장위치에 대한 주소를 가지며, 이 저장 위치를 이용해서 리스트의 원소값을 찾아갈 수 있다.

- ① 리스트
- ② 포인터
- ③ 메모리 주소 값
- ④ 구조체

[정답] 3

[해설] 메모리 주소 값에 대한 설명이다.

10. 다양한 데이터형의 변수를 하나의 상자 안에 넣어서 선언하거나 사용하는 C프로그래밍 문법은 다음 중 무엇인가?

- ① 리스트
- ② 포인터
- ③ 메모리 주소 값
- ④ 구조체

[정답] 4

[해설] 구조체에 대한 설명이다.