학습목표

- 자료 구조의 개념을 이해할 수 있다.
- 스택 구조와 스택의 연산을 이용하는 프로그램을 작성할 수 있다.
- 큐의 구조와 큐의 연산을 이용하는 프로그램을 작성할 수 있다.
- 연결 리스트의 구조와 연결 리스트의 연산을 이용하는 프로그램을 작성할 수 있다.

1 자료 구조의 개요

(1) 자료 구조의 개념

① 자료를 컴퓨터에 저장하는 방법을 결정하는 것은 프로그램의 작성에서 매우 중요한 의미를 갖는다. 왜냐하면, 저장된 자료의 구조에 따라 컴퓨터의 작업 방법과 작업 수행 시간이 달라지기 때문이다.

② 책을 쉽게 찾기 위해 정리하는 것처럼 컴퓨터는 자료를 처리할 때 일정한 형태의 자료 구조를 이용한다.

③ 자료 구조는 프로그램에서 자료를 효율적으로 기억할 수 있도록 기억 장소에 자료를 저장하는 방법이다.

④ 처리할 자료가 많아지면 자료 구조에 따라 그 처리 속도가 차이날 수 있다. 그러므로 처리할 작업에 맞는 적합한 자료 구조를 이용하여야 한다.

⑤ 효율적인 자료 구조는 같은 양의 자료를 가지고 같은 작업을 수행할 때, 보다 짧은 시간에 보다 적은 양의 기억 장소를 사용하여 원하는 작업을 수행할 수 있는 자료 구조이다.

(2) 자료 구조의 분류

① 컴퓨터에서 사용되는 자료 구조는 크게 선형 구조와 비선형 구조로 나눌 수 있다.

② 선형 구조는 앞과 뒤의 자료가 서로 일렬로 연결되어 있는 단순하고 보편적인 형태로 만들어진 자료 구조를 의미한다. 선형 구조에는 배열, 스택, 큐, 연결 리스트가 있다.

③ 비선형 구조는 선형 구조 이외의 자료 구조로 각 자료 간에 일렬로 연결되어 있지 않고 복잡한 구조로 연결된 자료 구조로, 트리와 그래프 등이 있다.

(3) 선형 구조

① 배열 : 연속적인 기억 장소에 동일한 형식의 자료를 순서적으로 저장하는 자료 구조로, 여러 개의 동일한 자료를 연속적으로 저장하여 사용할 경우에 활용되며, 첨자를 이용하여 자료를 쉽게 처리할 수 있다.

② 스택 : 한쪽 방향에서만 자료의 삽입과 삭제가 일어나는 자료 구조로, 제일 나중에 들어간 자료가 제일 먼저 나오는 LIFO 동작의 특징이 있다. 스택은 자료를 순서대로 저장한 후 저장된 순서와 반대로 자료를 처리하는 데 적합하다.

③ 큐 : 한쪽 끝에서 자료의 삽입이 이루어지고, 반대쪽 끝에서는 자료의 삭제가 일어나는 자료 구조로, 제일 먼저 들어간 자료가 제일 먼저 나오는 FIFO 동작의 특징이 있다. 그러므로 큐는 자료를 순서대로 저장한 후 저장한 순서대로 자료를 처리하는 작업에 적합한 자료 구조이다.

④ 연결 리스트 : 기차는 각 객차들이 서로 연결되어 있다. 이와 같이 연결 리스트는 순서에 관계없이 임의의 위치에 흩어져 저장되어 있는 각 자료들을 일렬로 연결한 자료 구조로, 처리할 자료의 크기가 일정하지 않고 자료의 이동과 자료의 추가 및 삭제가 자주 일어나는 작업에 적합하다.

EBS tip

자료
관찰이나 측정을 통해 얻어진 사실이나 값들의 집합

정보
자료를 처리하여 생성된 의미 있는 자료

배열
미리 할당받은 기억 장소에 동일한 형식의 자료를 순서적으로 저장하는 자료 구조

스택
한쪽 면이 막힌 구조로, 막히지 않은 한쪽 방향에서만 자료의 삽입과 삭제가 일어나는 자료 구조

큐
한쪽 끝에서는 자료의 삽입이, 반대쪽 끝에서는 자료의 삭제가 일어나는 자료 구조

연결 리스트
순서에 관계없이 임의 위치에 흩어져 저장되어 있는 각 자료들을 일렬로 연결한 자료 구조

EBS tip

(4) 비선형 구조

① 트리 : 나무의 줄기나 가지 모양을 보면 하나의 줄기에서 뻗어나간 가지가 점차 여러 개의 줄기로 나누어지는 것을 볼 수 있듯이 한 지점에서 여러 갈래로 나누어져 연결되어 있는 자료 구조이다.

② 그래프 : 지하철 노선도를 살펴보면 각 환승역과 환승역 사이가 서로 다른 노선들로 연결되어 있는 것을 볼 수 있듯이 복잡한 관계로 연결되어 있는 자료 구조이다.

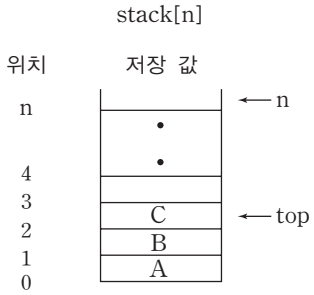
2 스택(stack)

(1) 스택 구조

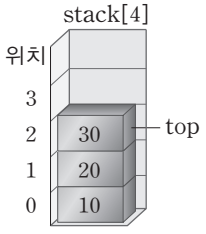
① 스택 : 일정한 순서로 나열된 자료 구조로 자료의 삽입과 삭제가 한쪽 방향에서만 일어나는 선형 구조

② 후입선출(LIFO : Last-In First-Out) 동작을 수행한다.

③ 크기가 n인 스택의 구조는 다음과 같이 표현할 수 있다.



▲ 스택 구조



▲ 스택 구조 예

n : 스택의 크기를 의미한다. n개의 자료를 입력할 수 있다.

top : 가장 최근에 입력된 자료의 저장 위치를 나타낸다.

- top=0이면 스택에 자료가 없는 상태이다.
- 가장 최근에 들어온 자료는 stack[top]에 저장된다.
- top=n이면 스택에 자료가 가득 찬 상태가 된다.
- top>n이면 스택이 오버플로(overflow)가 되어 자료를 더 이상 삽입할 수 없다.
- top<0이면 스택이 언더플로(underflow)가 되어 자료를 더 이상 삭제할 수 없다.

(2) 스택의 연산

① 자료 삽입 : 스택에 자료를 삽입하는 것을 푸시(push)라고 한다.

- 스택에 새로운 자료를 삽입할 때에는 자료를 저장할 빈 공간이 있어야 한다.
- 스택에 자료를 삽입할 공간이 없는 경우 새로운 자료를 삽입하면 오버플로가 발생한다.

② 자료 삭제 : 스택에서 자료를 삭제하는 것을 팝(pop)이라 한다.

- 스택에 있는 자료를 삭제할 때에는 스택에 자료가 저장되어 있어야 한다.
- 스택에 저장된 자료가 없는 경우 자료를 삭제하면 스택에 언더플로가 발생한다.
- 스택에서의 자료 삭제는 나중에 삽입된 자료부터 이루어진다.

EBS tip

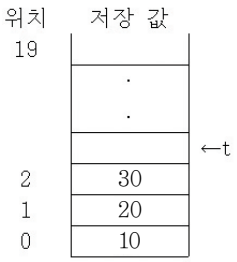
예 스택 연산 프로그램

프로그램 소스 코드	실행 결과

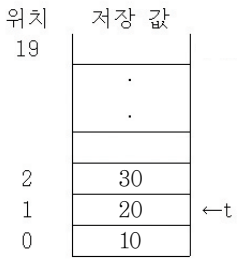
#include <stdio.h>
int s[20];
int t = 0;
void push(int a){
  s[t] = a;
  t = t + 1;
}
int pop( ){
  t = t-1;
  return s[t];
}
void main( ){
  push(10);
  push(20);
  push(30);
  printf("%3d", pop( ));
  printf("%3d", pop( ));
  push(40);
  printf("%3d", pop( ));
  printf("%3d", pop( ));
}
	30 20 40 10

[프로그램 해설]

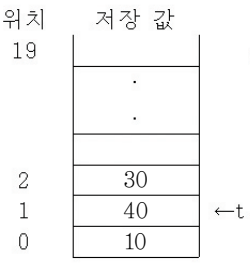
ⓐ 10 20 30이 차례대로 배열에 삽입된다.



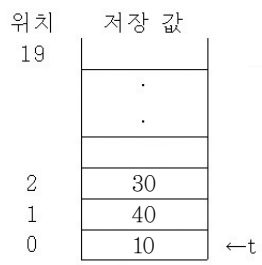
ⓑ pop( ) → pop( ) 동작에 의해 30과 20이 출력된다.



ⓒ push(40)에 의해 현재 t의 위치에 새로운 값 40이 삽입된다.



ⓓ pop( ) → pop( ) 동작에 의해 40과 10이 출력된다.



EBS tip

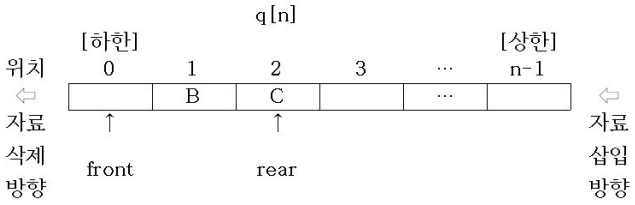
3 큐(Queue)

(1) 큐의 구조

① 큐 : 여러 개의 데이터 항목들이 일정한 순서로 나열된 자료 구조로 스택과 달리 한쪽 끝에서는 자료의 삽입이, 반대쪽 끝에서는 자료의 삭제만이 일어나는 선형 구조

② 선입선출(FIFO : First-In First-Out)형태로 동작한다.

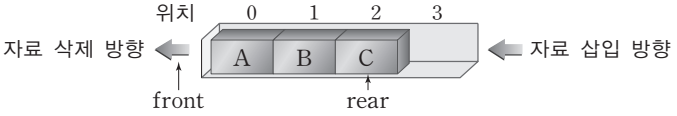
③ 삽입이 일어나는 곳을 후단(rear), 삭제가 일어나는 곳을 전단(front)이라 한다.



▲ 큐q[n]의 구조

n : 큐의 크기를 의미하며, n개의 자료를 입력할 수 있다.

- 하한 : 큐의 제일 처음 자료 저장 위치이다.
- 상한 : 큐의 제일 마지막 자료 저장 위치이다.
- front : 가장 마지막에 삭제된 자료의 위치를 나타낸다.
- rear : 가장 마지막에 입력된 자료의 위치를 나타낸다.
- front=rear=-1이면 큐가 빈 상태로, 큐의 초기조건이다.
- front=rear이면 큐에 자료가 없는 상태가 된다.
- rear>=n이면 오버플로(overflow)가 발생한다.



▲ 큐의 예

(2) 큐의 연산

① 자료 삽입

- 큐에 새로운 자료를 삽입하기 위해서는 자료를 저장할 공간이 있어야 한다.
- 큐에서 자료를 삽입할 공간이 없는 경우 새로운 자료를 삽입하면 큐에 오버플로가 발생한다.

② 자료 삭제

- 큐에서 자료를 삭제하기 위해서는 기존에 저장되어 있는 자료가 있어야 한다.
- 큐에 저장된 자료가 없는 경우 자료를 삭제하면 큐에 언더플로가 발생한다.
- 큐에서 자료 삭제는 먼저 삽입된 자료부터 이루어진다.

예 큐 연산 프로그램

프로그램 소스 코드	실행 결과

#include <stdio.h>
int s[20];
int rear= −1, front = −1 ;
void q_insert(int a){
rear = rear + 1;
s[rear] = a;
}
int q_delete( ){
front = front + 1;
return s[front];
}
void main(void)
{ q_insert(10);
q_insert(20);
q_insert(30);
printf("%3d", q_delete( ));
printf("%3d", q_delete( ));
q_insert (40);
printf("%3d", q_delete( ));
printf("%3d", q_delete( ));
}
	10 20 30 40

[프로그램 해설]

- front = rear = −1인 상태이므로 현재 큐는 비어있다.
- q_insert(10), q_insert(20), q_insert(30)에 의해 큐에 10, 20, 30이 삽입된다.
- q_delete( )에 의해 자료 10이 출력된다.
- q_delete( )에 의해 자료 20이 출력된다.
- q_insert (40)에 의해 큐에 40이 삽입된다.
- q_delete( )에 의해 차례대로 30과 40이 출력된다.

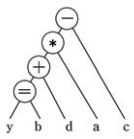
기출 모의고사

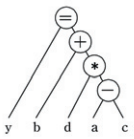
정답 및 해설 p. 21

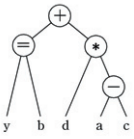
2008년 9월 시행 평가원 모의평가

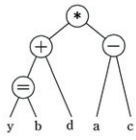
1 다음 연산식을 우선순위에 의한 트리 구조로 바르게 나타낸 것은?

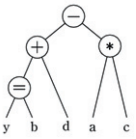
y = b + d * (a − c)

① 

② 

③ 

④ 

⑤ 

[2~3] 다음은 자료를 관리하는 프로그램이다. 물음에 답하시오.

```
#include <stdio.h>
int k[10];
int Fr, Re;
void fun_P(int m) {
    k[Re] = m;
    Re = Re + 1;
}
int fun_G( ) {
    int b;
    b = k[Fr];
    Fr = Fr + 1;
    return b;
}
void main( ){
  int a;
  Fr = 0; Re = 0;
  fun_P(7); fun_P(5);
  a = fun_G( );
  fun_P(a); fun_P(3);
  a = Fr;
  while (a! = Re) {
     printf("%d", k[a]);
     a = a + 1;
  }
}
```

2 위 프로그램에 대한 설명으로 옳은 것만을 〈보기〉에서 있는 대로 고른 것은?

보기

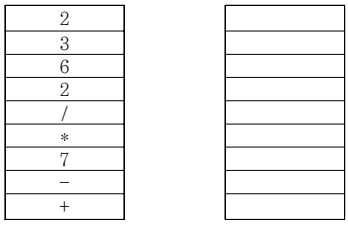
ㄱ. fun_P( )는 반환값이 없다.
ㄴ. fun_G( )는 정수형의 연산 결과를 반환한다.
ㄷ. 변수 Fr은 fun_G( )에서 사용되는 지역 변수이다.

① ㄱ ② ㄷ ③ ㄱ, ㄴ
④ ㄴ, ㄷ ⑤ ㄱ, ㄴ, ㄷ

3 위 프로그램의 실행 결과는?

① 3 5 7 ② 5 7 3 ③ 7 5 3
④ 7 5 5 3 ⑤ 7 5 7 3

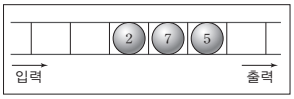
4 0~9 사이의 정수와 사칙 연산자(+, −, *, /)를 포함하는 자료가 그림과 같이 스택 A에 저장되어 있다. 스택 A의 자료를 하나씩 꺼내어 아래 [조건]에 따라 처리하였을 때, 최종적으로 스택 B에 남아 있는 값은?



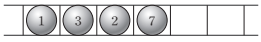
[조건]
(1) 스택 A의 자료를 1개 꺼낸다(pop).
(2) 꺼낸 자료가 숫자이면 스택 B에 넣고(push), 연산자이면 스택 B에서 2개의 자료를 꺼내어 연산을 수행한 후 결과값을 다시 스택 B에 넣는다.(꺼낸 자료가 순서에 따라 P1, P2이고 뺄셈 연산일 경우 연산식은 'P2−P1', 나눗셈 연산일 경우 연산식은 'P2/P1')
(3) 스택 A에 남은 자료가 1보다 크면 단계 (1)로 이동하고 그렇지 않으면 종료한다.

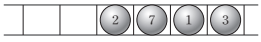
① 2 ② 3 ③ 4 ④ 5 ⑤ 6

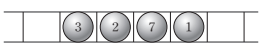
5 그림의 선형 큐에서 주어진 프로그램을 수행하고 난 후, 큐의 상태로 옳은 것은? (단, 빈 곳은 0으로 데이터를 입력한다.)

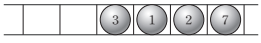


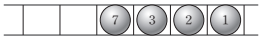

```
#include <stdio.h>
int s[6]={ 0,5,7,2 };
int rear= 3, front =0;
void q_insert(int a) {
        rear = rear + 1;
        s[rear] = a;
}
void q_delete( ) {
        front = front + 1;
        printf("%3d",s[front]);
        s[front]=0;
}
void main( ) {
        int k;
        q_insert (3);
        q_delete( );
        q_insert (1);
        printf("\n");
        for(k=0;k<=5;k++)
        printf("%3d", s[k]);
}
```

① 

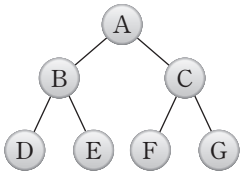
② 

③ 

④ 

⑤ 

6 그림과 같이 2진 트리 구조의 자료를 1차원 배열 a에 저장하였다. 이와 같은 자료 구조를 응용한 〈보기〉와 같은 프로그램을 실행한 결과 출력되는 내용으로 옳은 것은?



배열 a

첨자	1	2	3	4	5	6	7
내용	A	B	C	D	E	F	G

보기

```
#include <stdio.h>
void main( ) {
  int i, j, n, t;
  int a[8]={' ', 'A', 'B', 'C', 'D', 'E', 'F', 'G'};
  n=7; i=1;
  t=a[i]; j=2*i;
  while(j<=n){
    if(t>=a[j]) break;
    a[j/2]=a[j]; j=j*2;
  }
  a[j/2]=t;
  for(i=1; i<=n; i++)
    printf("%c", a[i]);
}
```

① A B C D E F G
② G F E D C B A
③ B D C A E F G
④ A B D E C F G
⑤ D B E A F C G