학습목표

- 정렬의 개념을 이해하고 설명할 수 있다.
- 버블, 삽입, 선택 정렬의 알고리즘을 이해하고 프로그램을 작성할 수 있다.

1 정렬의 개요

(1) 정렬의 개념과 종류

① 정렬(sort)은 순서 없이 나열된 자료를 일정한 기준에 따라 순서에 맞게 나열하는 것이다.

예 • 출석부 : 학기 초에 학생들의 반편성이 끝난 후에 각 반별로 이름을 '가나다' 순으로 정렬하여 번호를 부여함.

• 전화번호부 : 전화번호를 쉽게 찾을 수 있도록 하기 위한 방법으로 전화번호를 이름순으로 정렬함.

② 정렬은 오름차순 또는 내림차순으로 정렬한다.

③ 오름차순은 작은 것에서 큰 것으로 정렬하는 방법이다.

④ 내림차순은 큰 것에서 작은 것으로 정렬하는 방법이다.

⑤ 정렬은 수행하는 장소에 따라 구분하면 내부 정렬과 외부 정렬로 나눌 수 있다.

⑥ 내부 정렬 방식은 주 기억 공간 내에서 한 번에 정렬하는 방식이다.

⑦ 내부 정렬 방식에는 선택 정렬, 버블 정렬, 삽입 정렬, 쉘 정렬, 퀵 정렬, 힙 정렬, 합병 정렬 등이 있다.

⑧ 외부 정렬 방식은 정렬해야 할 자료가 매우 많아 주 기억 공간 내에서 한 번에 처리가 불가능한 경우에 사용한다.

⑨ 외부 정렬은 입력 파일을 여러 개의 서브 파일로 나누어 내부 정렬 방식으로 정렬한 후 다시 보조 기억 장치에 넣어 정렬된 서브 파일을 병합하는 방식으로 정렬을 수행한다.

(2) 정렬의 활용

① 우리들은 일상생활 속에서 수많은 자료들을 접하게 되고, 이러한 자료들을 쉽게 찾아서 업무를 효율적으로 처리하기 위한 방법으로 자료들을 정렬하고 있다.

② 정렬은 프로그램에서 가장 기본적인 알고리즘이라 할 수 있을 정도로 정렬을 응용하는 프로그래밍 문제는 많다. 정렬을 이해하여 여러 가지 프로그래밍 작업을 해결할 때 어떻게 활용할 수 있는지 알아보자.

- 유일성 검사 : 여러 항목들을 모아 놓은 집합이 있을 때 거기에 있는 모든 데이터들이 서로 다른지 확인할 수 있다.
- 중복된 항목 삭제 : 집합에 중복된 항목이 있을 경우 하나만 남기고 모두 삭제하려고 할 때, 정렬한 후에 항목을 한 번씩 확인하며 겹치는 항목은 삭제한다.
- 교집합, 합집합 : 두 집합을 모두 정렬한 후, 각 집합 맨 앞에 있는 원소 중에서 더 작은 값을 골라 새 집합에 넣고, 그 원소가 들어 있는 목록에서 맨 앞에 있는 원소를 지우는 작업을 반복한다.
- 빈도 구하기 : 집합에서 가장 많이 등장하는 값, 즉 최고 빈도를 구할 때에도 정렬한 후 모든 원소를 비교하면 구할 수 있다.

EBS tip

◉ 오름차순 정렬

오름차순 정렬이란 '1, 2, 3, 4, 5'와 같이 자료들을 작은 것부터 큰 것의 순으로 배열하는 방법을 의미한다. 예를 들어, 학번 순으로 나열한다든지, 이름을 '가나다'의 순으로 나열하는 것이 오름차순 정렬이다.

◉ 내림차순 정렬

내림차순 정렬은 자료들을 '5, 4, 3, 2, 1'과 같이 큰 것에서부터 작은 것의 순으로 배열하는 방법을 의미한다. 예를 들어, 순위를 결정하기 위하여 높은 점수부터 낮은 점수로 나열하는 것이 내림차순 정렬에 해당한다.

2 정렬의 방법

(1) 버블 정렬

① 인접하는 두 항을 비교하여 순서가 반대이면 자료의 위치를 서로 교환하는 정렬 방식이다.

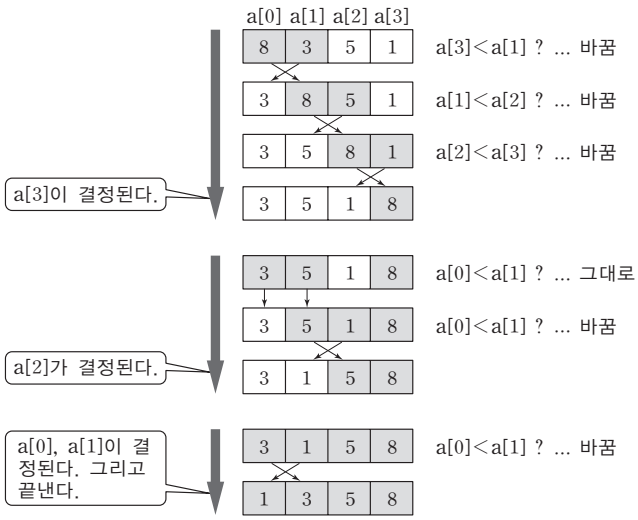
② 오름차순으로 정렬하는 과정을 보면 다음과 같다.

ⓐ 첫 번째와 두 번째의 값을 비교해서 첫 번째 값이 크면 두 값의 위치를 바꾼다.

ⓑ 두 번째와 세 번째, 세 번째와 네 번째 등 계속해서 끝까지 비교해 가면서 바꾼다.

ⓒ 이처럼 한 번 실행하면 맨 끝 값이 가장 큰 값이 된다.

ⓓ 이 방법을 계속해서 자료의 수보다 1 작은 수만큼 반복하면 전체 자료는 정렬된다.



▲ 버블 정렬 방식에 따른 자료의 정렬 과정

예 버블 정렬 프로그램

프로그램 소스 코드	실행 결과

#include <stdio.h>
void main( ) {
  int m, n, temp;
  int a[ ]={8, 3, 5, 1};
  for(m=0; m<=2; m++) {
  for(n=0; n<=2-m; n++)
  if(a[n] > a[n+1]){
  temp=a[n];
  a[n]=a[n+1];
  a[n+1]=temp; }
  printf("\n%d 회전\n", m+1);
  for(n=0; n <= 3; n++)
  printf("%d ", a[n]);
  }
}
	1 회전
	3 5 1 8
	2 회전
	3 1 5 8
	3 회전
	1 3 5 8

EBS tip

버블 정렬

- 정렬이 진행하는 모양이 비누거품(bubble)과 같다고 하여 버블 정렬(거품 정렬)이라고 한다.
- 효율성이 떨어지나 프로그램 구현이 간단하여, 속도가 중요하지 않은 경우에 널리 쓰인다.

EBS tip

[프로그램 해설]

for(m=0; m<=2; m++)
{
for(n=0; n<=2-m; n++)
....
}
	ⓐ 바깥쪽 반복 횟수를 자료의 수 – 1로 한다. 따라서 제어 변수 m은 자료가 4개이므로 0부터 2까지 3번 반복한다.
	ⓑ 안쪽 반복 횟수를 (자료의 수-1)-m으로 한다. 왜냐하면 반복 횟수가 진행될 때마다 맨 뒤부터 자료가 정렬되기 때문이다.

if(a[n]>a[n+1])
{
temp=a[n];
a[n]=a[n+1];
a[n+1]=temp;
}
	ⓒ 앞 뒤의 값을 비교해서 앞의 값이 크면 두 값의 위치를 바꾼다.(배열 a[n]의 값과 a[n+1]에 기억된 값을 서로 바꾸어 준다.)

printf("\n%d 회전\n", m+1);
for(n=0; n<=3; n++)
printf("%d ", a[n]);
	ⓓ 1회전이 실시될 때마다 그 과정을 알아보기 위해 출력한다.

(2) 선택 정렬

① 기준이 되는 위치를 설정하고, 그 위치에 들어갈 자료를 찾아 기준 위치의 자료와 찾은 자료를 서로 바꾸어 정렬하는 방식이다.

② 오름차순으로 정렬하는 과정을 보면 다음과 같다.

ⓐ N개의 자료가 있을 경우 가장 작은 원소를 찾아 첫 번째 위치의 원소와 교환한다.

ⓑ 다음에는 나머지 N-1개의 값 중 가장 작은 것을 찾아서 두 번째 위치의 원소와 교환한다.

ⓒ 이러한 과정을 반복 수행한다.

ⓓ 최종 N-1 단계에서 가장 큰 값을 갖는 자료가 N번째 위치에 오게 한다.

ⓔ 내림차순으로 정렬할 경우에는 반대로 가장 작은 자료가 N번째 위치에 오게 한다.

입력 자료	5 2 8 3 1
처음 자료는 5이다. 그 다음 자료를 하나하나 읽으면서 가장 작은 자료를 찾아 처음 자료와 위치를 바꾼다. 1과 위치를 바꾼다.	1 2 8 3 5
두 번째 자료는 2이다. 그 다음 자료를 읽으면서 비교하여 가장 작은 자료를 찾는다. 그 위치 그대로이다.	
세 번째 자료는 8이다. 그 다음 자료를 하나하나 읽으면서 가장 작은 자료를 찾아 세 번째 자료인 3과 위치를 바꾼다.	1 2 3 8 5
네 번째 자료는 8이다. 그 다음 자료를 읽으면서 비교하여 가장 작은 자료를 찾는다. 5와 위치를 바꾼다.	1 2 3 5 8
정렬이 완료되었다.	

▲ 그림 15-2 선택 정렬 방식에 따른 자료의 정렬 과정

◉ 두 변수 a, b의 자료 교환 방법

임시 변수 temp 선언

```
temp = a;
a = b;
b = temp;
```

a와 b의 값을 서로 맞바꾸기 위해 임시 변수 temp를 이용함.

EBS tip

예 선택 정렬 프로그램

프로그램 소스 코드	실행 결과

void main( ) {
  int a, b, min, temp;
  int c[5]={5, 2, 8, 3, 1};
 	
  for (a=0; a<=3; a++){
  min=a;
  for(b=a+1; b<=4; b++)
  if (c[b] < c[min])
  min=b;
 	
  temp=c[a];
  c[a]=c[min];
  c[min]=temp;
 	
  printf("\n%d 회전\n", a+1);
  for(b=0; b<=4; b++)
  printf("%3d", c[b]);
  }
}
	1 회전
	  1 2 8 3 5
	2 회전
	  1 2 8 3 5
	3 회전
	  1 2 3 8 5
	4 회전
	  1 2 3 5 8

■ 내림차순인 경우에는
if(c[b] < c[min])
대신
if(c[b] > c[min])
으로 바꾸어 준다.

[프로그램 해설]

for(a=0; a<=3; a++)
  min=a;
  for(b=a+1; b<=4; b++)
  if (c[b] < c[min])
  min=b;
	ⓐ 바깥쪽 반복 횟수를 자료의 수-1로 한다. 따라서 제어 변수 a는 자료가 5개이므로 0부터 3까지 4번 반복한다.
	ⓑ min=a;는 기준 위치를 저장한다.
	ⓒ 기준 위치와 기준 위치 뒤에 있는 자료들 중에서 가장 작은 자료가 있는 위치를 찾는다.

temp=c[a];
c[a]=c[min];
c[min]=temp;
	ⓓ 찾은 자료를 기준 위치에 있는 자료와 교환한다.

printf("\n%d 회전\n",a+1);
for(b=0; b<=4; b++)
  printf("%3d", c[b]);
	ⓔ 1회전이 반복될 때마다 그 과정을 알아보기 위해 회전 수를 출력한다.

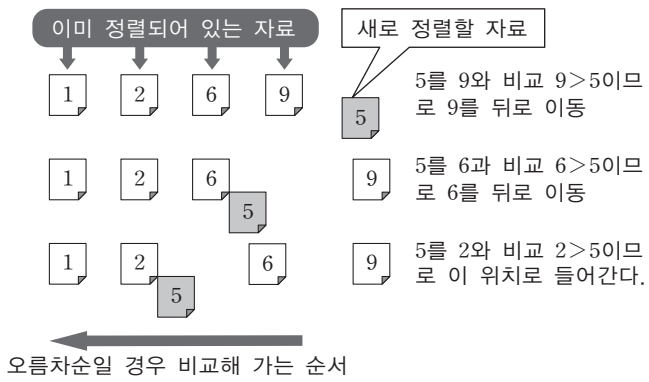
(3) 삽입 정렬

① 이미 정렬된 자료 사이에 정렬되지 않은 자료를 해당되는 위치에 삽입하여 정렬하는 방식이다.

② 오름차순으로 정렬하는 과정을 보면 다음과 같다.

ⓐ 정렬하고자하는 자료를 이미 정렬된 마지막 위치의 자료와 비교하여 자신보다 크면, 큰 자료를 다음 자리로 이동하고, 앞 자료와 다시 비교한다.

EBS tip



▲ 삽입 정렬 방식에 따른 자료의 정렬 과정

ⓑ 만일 비교하는 자료가 자신보다 크지 않으면 비교하던 자료 다음 자리에 들어간다.

ⓒ 두 번째와 세 번째, 세 번째와 네 번째와 같이 계속해서 끝까지 비교해 가면서 해당 위치를 찾아 들어간다.

ⓓ 이 방법을 계속해서 자료의 수보다 1 작은 수만큼 반복하면 모든 자료가 정렬된다.

예 삽입 정렬 프로그램

프로그램 소스 코드	실행 결과

```
#include <stdio.h>
void main() {
    int m, n, temp;
    int a[ ]={2, 6, 1, 9, 5};

    for(m=1; m<=4; m++)
    {
        temp=a[m];
        for(n=m-1; n>=0; n--)
        {
            if(a[n]>temp)
              a[n+1]=a[n];
            else
              break;
        }
        a[n+1]=temp;

        printf("\n%d 회전\n", m);
        for(n=0; n<=4; n++)
            printf("%d ", a[n]);
    }
}
```

```
1 회전
 2 6 1 9 5
2 회전
 1 2 6 9 5
3 회전
 1 2 6 9 5
4 회전
 1 2 5 6 9
```

■ 내림차순인 경우에는
if (a[n] > temp)
대신
if (a[n] < temp)
로 바꾸어 준다.

EBS tip

[프로그램 해설]

for(m=1; m<=4; m++)
  temp=a[m];
  for(n=m-1; n>=0; n--){
  if(a[n]>temp)
  a[n+1]=a[n];
	ⓐ 바깥쪽 반복 횟수를 '자료의 수-1'로 한다. 따라서 제어 변수 a는 자료가 5개이므로 1부터 4까지 4번 반복한다.
	ⓑ temp=a[m]; 정렬할 자료를 저장한다.
	ⓒ 정렬하고자하는 자료(temp)를 이미 정렬된 마지막 위치의 자료와 비교하여 자신보다 크면, 큰 자료를 다음 자리로 이동하고, 앞 자료와 다시 비교한다.

else
  break;
}
a[n+1]=temp;
	ⓓ 만일 비교하는 자료가 자신보다 크지 않으면 반복문을 빠져 나가 비교하던 자료 다음 자리에 들어간다.

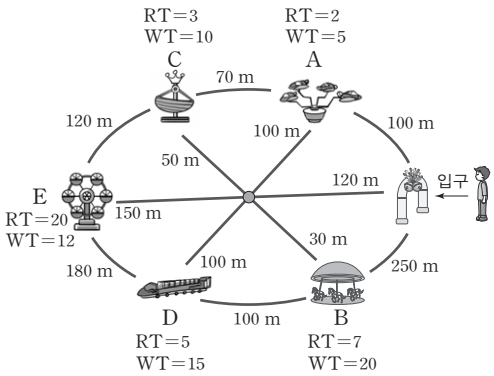
printf("\n%d 회전\n", m);
for(n=0; n<=4; n++)
  printf("%d ", a[n]);
	ⓔ 1회전이 실시될 때마다 그 과정을 알아보기 위해 출력한다.

기출 모의고사

정답 및 해설 p. 16

2007학년도 대수능

1 그림은 놀이 공원에서 각 놀이 기구의 운행 시간(RT)과 대기 시간(WT), 그리고 거리 정보를 나타낸 것이다. 물음에 답하시오.



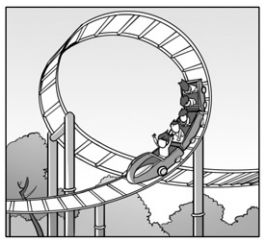
다음은 위의 놀이 기구를 타는 순서를 결정하는 프로그램이다. 프로그램 실행 결과에 따라 놀이 기구를 탈 때 그 순서는?

```
#include <stdio.h>
void main( ){
  int m, n, p, num, idx;
  char chData;
  int RT[ ]={2, 7, 3, 5, 20};
  int WT[ ]={5, 20, 10, 15, 12};
  int ST[ ]={0, 0, 0, 0, 0};
  char MN[ ]={'A', 'B', 'C', 'D', 'E'};
  for(idx=0; idx<=4; idx++)
    ST[idx]=RT[idx]+WT[idx];
  for(m=1; m<=4; m++)
  {
       num=ST[m];
       chData=MN[m];
       n=m-1;
       while((n>=0) && (ST[n]>num))
        {
            ST[n+1]=ST[n];
            MN[n+1]=MN[n];
            n--;
        }
    ST[n+1]=num;
    MN[n+1]=chData;
  }
  for(p=0; p<=4; p++)
    printf("%c", MN[p]);
}
```

① A-B-C-D-E ② A-B-C-E-D
③ A-C-D-B-E ④ E-B-D-C-A
⑤ E-D-B-C-A

2010년 6월 시행 평가원 모의평가

2 다음 프로그램을 실행하였을 때 [탑승 규칙]을 만족하는 학생들의 탑승 순서를 바르게 배열한 것은?



[탑승 규칙]

- 롤러코스터에 탑승할 학생은 표와 같다.

번호	10	15	20	5
이름	철수	영희	영수	순희

- 학생들은 프로그램의 출력 결과로 나오는 번호 순서에 따라 롤러코스터에 탑승한다.

```
#include <stdio.h>
void main( ){
  int a, b, temp;
  int num[4]={10, 15, 20, 5};
    /*num : 학생의 번호 */
  for(a=0; a<3; a++) {
    for(b=0; b<3-a; b++) {
      if (num[b] < num[b+1]) {
          temp=num[b];
          num[b]=num[b+1];
          num[b+1]=temp;
      }
    }
  }
  for(a=0; a<4; a++)
    printf("%d", num[a]);
}
```

① 영수 – 영희 – 철수 – 순희
② 영수 – 철수 – 영희 – 순희
③ 철수 – 영희 – 영수 – 순희
④ 순희 – 영희 – 철수 – 영수
⑤ 순희 – 철수 – 영희 – 영수

3 다음 부분 프로그램을 수행하고 난 뒤 출력 결과가 다음과 같을 때, 프로그램의 (가) 안에 들어갈 내용으로 옳은 것은?

```
43 39 27 8 60
39 27 8 43 60
27 8 39 43 60
8 27 39 43 60
8 27 39 43 60
```

```
#include <stdio.h>
void main( ) {
  int dat[5]={60, 43, 39, 27, 8};
  int a, b, k;
  for(k=0; k<=4; k++) {
      for(a=0; a<=3; a++) {
            if ( 가 ){
               b = dat[a];
               dat[a] = dat[a+1];
               dat[a+1] = b;
                     }
            }
      for(a=0; a<=4; a++)
         printf("%d ", dat[a]);
      printf("\n");
  }
}
```

① dat[a] > dat[a+1]
② dat[a] < dat[a+1]
③ dat[a]%4 == 0
④ dat[a]/4 == 0
⑤ dat[a] == dat[a+1]

4 다음은 배열 a와 b를 병합하여 오름차순으로 배열 c에 배정하는 프로그램이다. 실행 결과가 '1 2 3 4 5 6'일 때 (가)와 (나)에 들어갈 것으로 옳은 것은?

```
#include <stdio.h>
void main( ) {
  int a[ ]={1, 2, 3};
  int b[ ]={4, 5, 6};
  int x=0, y=0, z=0, m, c[6];
  while((x<3) && (y<3)) {
    if (a[x] < b[y]) {
       (가)
    } else {
       (나)
    }
    z++;
  }
  while(x<3) {
    c[z] = a[x];
    z++; x++;
  }
  while(y<3) {
    c[z] = b[y];
    z++; y++;
  }
  for(m=0; m<=5; m++)
    printf("%d", c[m]);
}
```

① (가) c[z] = a[x];
   (나) c[z] = b[y];
② (가) c[z] = b[y];
   (나) c[z] = a[x];
③ (가) c[z] = a[x]; x--;
   (나) c[z] = b[y]; y--;
④ (가) c[z] = a[x]; y++;
   (나) c[z] = b[y]; x++;
⑤ (가) c[z] = a[x]; x++;
   (나) c[z] = b[y]; y++;

5 〈보기 A〉와 같이 정렬되지 않은 배열 원소를 〈보기 B〉와 같이 오름차순으로 정렬된 배열로 바꾸는 프로그램이다. ☐ 안에 들어갈 내용은?

보기 A

	a[0]	a[1]	a[2]	a[3]	a[4]
배열 a	2	5	11	4	6

보기 B

	a[0]	a[1]	a[2]	a[3]	a[4]
배열 a	2	4	5	6	11

```
#include <stdio.h>
void main( ) {
  int i, j, temp;
  int a[5]={2, 5, 11, 4, 6};
  for(i=4; i>=1; i-- ){
    for(j=1; j<=i; j++)
      if (a[j-1] > a[j]) {

                ☐

            }
  }
  for(i=0; i<=4; i++)
    printf("%3d", a[i]);
}
```

① temp=a[j];
a[j]=a[j-1];
a[j-1]=temp;

② a[j]=a[j-1];
temp=a[j];
a[j-1]=temp;

③ temp=a[j-1];
a[j]=temp;

④ temp=a[j-1];
a[j]=temp;
a[j-1]=a[j];

⑤ a[j]=a[j-1];
a[j-1]=a[j];

6 다음 정렬 프로그램에 대한 설명으로 옳지 <u>않은</u> 것은?

```
void main ( ) {
   int m, n, temp, flag;
   int dat[ ]={6, 3, 2, 9, 4};
   for(m=0; m<=3; m++){
        flag=0;
        for(n=0; n<=3-m; n++)
        if(dat[n]>dat[n+1])
        {
              temp=dat[n];
              dat[n]=dat[n+1];
              dat[n+1]=temp;
              flag=1;
        }
        if(flag == 0)
          break;
   }
}
```

① 오름차순으로 정렬된다.
② 일부 정렬되어 있으면 자료의 교환 횟수가 줄어든다.
③ flag의 값이 0에서 1로 바뀌지 않으면 정렬이 완료된 것이다.
④ 인접하는 두 요소값을 비교하여 순서가 맞지 않으면 두 요소값을 교환한다.
⑤ 자료의 개수가 n이면 자료의 정렬 여부와 관계없이 (n-1) 회전을 반복한다.