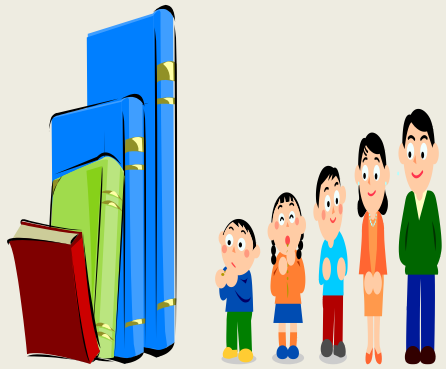
정렬(sort)이란?

- 정렬은 데이터를 크기순으로 오름차순이나 내림차순으로 나열하는 것
- 정렬은 가장 기본적이고 중요한 알고리즘중의 하나
- 정렬은 자료 탐색에 있어서 필수적이다.

  (예) 만약 사전에서 단어들이 정렬이 안되어 있다면?



비교	제조사	모델명	요약설명	최저가↓	업체수	출시
☐	ROLLEI	D-41com	410만화소(0.56")/1.8"LCD/3배줌/연사/CF카드	320,000	4	02년
☐	카시오	QV-R40	413만화소(0.56")/1.6"LCD/3배줌/동영상/히스토그램/앨범기능/SD,MMC카드	344,000	73	03년
☐	파나소닉	DMC-LC43	423만화소(0.4")/1.5"LCD/3배줌/동영상+녹음/연사/SD,MMC카드	348,000	36	03년
☐	현대	DC-4311	400만화소(0.56")/1.6"LCD/3배줌/동영상/SD,MMC카드	350,000	7	03년
☐	삼성테크윈	Digimax420	410만화소(0.56")/1.5"LCD/3배줌/동영상+녹음/음성메모/한글/SD카드	353,000	47	03년
☐	니콘	Coolpix4300	413만화소(0.56")/1.5"LCD/3배줌/동영상/연사/CF카드HOT4	356,800	79	02년
☐	올림푸스	뮤-20 Digital	423만화소(0.4")/1.5"LCD/3배줌/동영상/연사/생활방수/xD카드	359,000	63	03년
☐	코닥	LS-443(Dock포함)	420만화소/1.8"LCD/3배줌/동영상+녹음/SD,MMC카드/Dock시스템	365,000	39	02년
☐	올림푸스	C-450Z	423만화소(0.4")/1.8"LCD/3배줌/동영상/연사/xD카드	366,000	98	03년
☐	올림푸스	X-1	430만화소/1.5"LCD/3배줌/동영상/연사/xD카드	367,000	19	03년
☐	미놀타	DiMAGE-F100	413만화소(0.56")/1.5"LCD/3배줌/동영상+녹음/음성메모/동체추적AF/연사/SD,MMC카드	373,000	18	02년
☐	삼성테크윈	Digimax410	410만화소(0.56")/1.6"LCD/3배줌/동영상+녹음/음성메모/한글/CF카드	374,000	4	02년

배열 Part 2

정렬의 종류

- 선택형 정렬(Selection Sorts) : Selection sort, heap sort, …
- 삽입형 정렬(Insertion Sorts) : Insertion sort, Shell sort, Tree sort, …
- 교환형 정렬(Exchange Sorts) : Bubble sort, Quick sort, …
- 병합형 정렬(Merge Sorts) : Merge sort

선택정렬(Selection sort)

- 선택정렬(selection sort): 정렬이 안된 숫자들중에서 최소값(최대값)을 선택하여 배열의 첫 번째 요소와 교환 – Min(Max)-Selection sort

Selection-Sort-Animation.gif

```c
#include <stdio.h>
#define SIZE 10

void selection_sort(int list[], int n);
void print_list(int list[], int n);

int main(void)
{
	int grade[SIZE] = { 3, 2, 9, 7, 1, 4, 8, 0, 6, 5 };

	// 원래의 배열 출력
	printf("원래의 배열₩n");
	print_list(grade, SIZE);

	selection_sort(grade, SIZE);

	// 정렬된 배열 출력
	printf("정렬된 배열₩n");
	print_list(grade, SIZE);

	return 0;
}
```

```
void print_list(int list[], int n)
{
        int i;
        for(i = 0;i < n; i++)
                printf("%d ", list[i]);
        printf("₩n");
}
void selection_sort(int list[], int n)              // 인덱스 기반으로 최소값 선택
{
        int i, j, temp, least;

        for(i = 0; i < n-1; i++)
        {
                least = i;

                for(j = i + 1; j < n; j++)  // 최소값 탐색
                        if(list[j] < list[least])
                                least = j;
                // i번째 원소와 least 위치의 원소를 교환
                temp = list[i];
                list[i] = list[least];
                list[least] = temp;
        }
}
```

원래의 배열
3 2 9 7 1 4 8 0 6 5
정렬된 배열
0 1 2 3 4 5 6 7 8 9

```
void selection_sort1(int list[], int n) // 인덱스를 기반으로 최소값 선택
{
        int i, j, temp, min, min_index;

        for(i = 0; i < n-1; i++)
        {
                min_index = i;

                for(j = i + 1; j < n; j++)  // 최소값 탐색
                        if(list[j] < list[min_index]) {
                                min_index = j ;
                         }
                // i번째 원소와 least 위치의 원소를 교환
                temp = list[i];
                list[i] = list[min_index];
                list[min_index] = temp;
        }
```

```
void selection_sort2(int list[], int n) // 데이터 자체를 기반으로 최소값 선택
{
        int i, j, temp;

        for(i = 0; i < n; i++)
        {
                for(j = i + 1; j < n; j++)  // 최소값 탐색
                        if(list[j] < list[i]) {
                            temp = list[i];
                            list[i] = list[j];
                            list[j] = temp;
                        }
          }
}
```

◆ **selection_sort1과 selection_sort2의 차이는 무엇일까?**

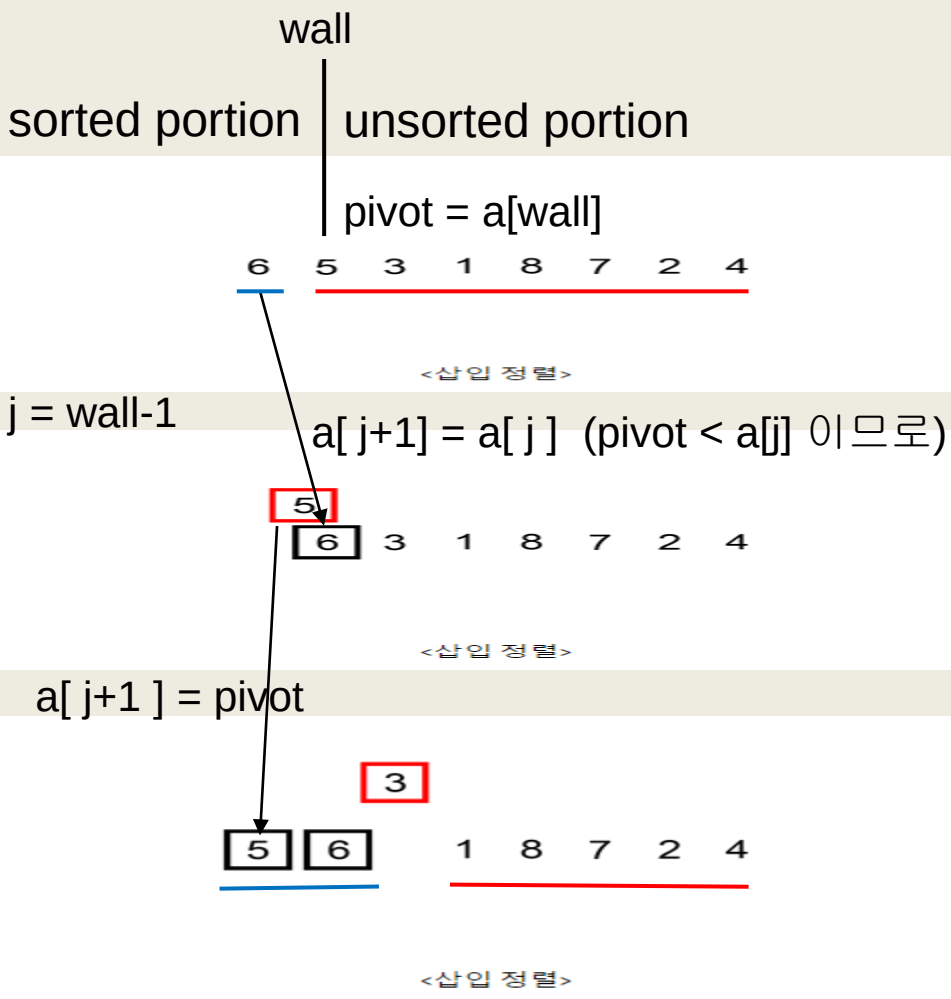
삽입정렬(Insertion Sort)

삽입정렬 예제 영상

- **Algorithm (Sorted portion/Unsorted portion)**

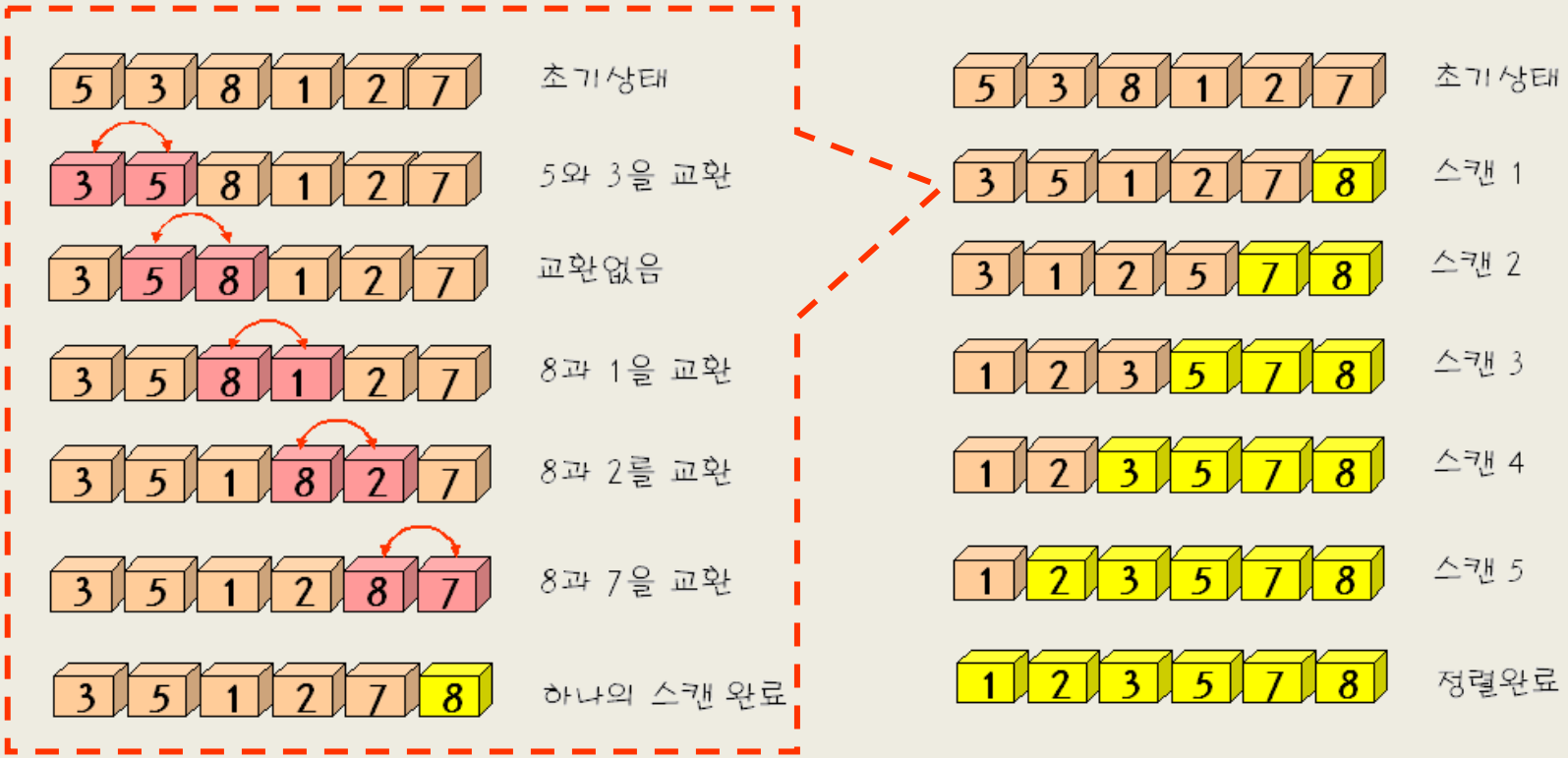
```
InsertionSort(int a[], int n) {
int wall, j, pivot;

 for(wall=1; i<n; wall++) {
   pivot = a[wall] ; // wall 미만 = sorted portion
   j = wall-1 ;
  while (j >= 0 && (pivot < a[j])) {
      a[j+1] = a[j];
      j -- ;
  }
 a[j+1] = pivot ;
 }
}
```


wall
sorted portion
unsorted portion
pivot = a[wall]
6 5 3 1 8 7 2 4
<삽입 정렬>
j = wall-1
a[j+1] = a[j] (pivot < a[j] 이므로)
5
6 3 1 8 7 2 4
<삽입 정렬>
a[j+1] = pivot
3
5 6 1 8 7 2 4
<삽입 정렬>

버블정렬(bubble sort)

- 인접한 레코드가 순서대로 되어 있지 않으면 교환
- 전체가 정렬될 때까지 비교/교환 계속



버블 정렬 프로그램

```
void BubbleSort(int a[], int n)
{
        int i, scan, temp;

        // 스캔 회수를 제어하기 위한 루프
        for(scan = 0; scan < n-1; scan++)
        {
                // 인접값 비교 회수를 제어하기 위한 루프
                for(i = 0; i < n-1; i++)
                {
                        // 인접값 비교 및 교환
                        if( a[i] > a[i+1] )
                        {
                                temp = a[i];
                                a[i] = a[i+1];
                                a[i+1] = temp;
                        }
                }
        }
}
```

- 좀 더 효율적 알고리즘

```
void BubbleSort(int a[], int n)
{
        int i, scan, temp;

        // 스캔 회수를 제어하기 위한 루프
        for(scan = 0; scan < n-1; scan++)
        {
                // 인접값 비교 회수를 제어하기 위한 루프
                for(i = 0; i < n-1-scan ; i++)
                {
                        // 인접값 비교 및 교환
                        if( a[i] > a[i+1] )
                        {
                                temp = a[i];
                                a[i] = a[i+1];
                                a[i+1] = temp;
                        }
                }
        }
}
```

- 과제 1(출석용 + 성적반영)
  - 100개의 정수 데이터를 rand()를 사용하여 생성하여 int sortData[100] 배열에 저장
  - Selection sort 함수를 만들어 실행항 결과,
  - Insertion sort 함수를 만들어 실행한 결과,
  - Bubble sort 함수를 만들어 실행한 결과를 다음과 같이 구성하여 제출하시오
  - 1: 생성된 배열을 화면에 프린트
  - 2: 해당 sort 함수 호출
  - 3: 정렬된 배열을 화면에 프린트
  - 프로그램은 하나의 main() 함수와 3개의 sort 함수 그리고 배열을 프린트하는 print_array 함수, 총 5개의 함수로 구성한다