



정렬이란?

- 정렬은 물건을 크기순으로 오름차순이
나 내림차순으로 나열하는 것
- 정렬은 컴퓨터 공학분야에서 가장 기
본적이고 중요한 알고리즘중의 하나
- 정렬은 자료 탐색에 있어서 필수적이
다.



(예) 만약 사전에서 단어들이 정렬이 안
되어 있다면?



모델	회사	모형명	요약설명	가격	입력수	출력
ROLLE	D-ticon	4102	4102	200,000	10	10년
카시오	QV-R40	4102	4102	300,000	10	10년
파나소닉	DMC-LC43	4102	4102	300,000	10	10년
현대	DC-4311	4102	4102	300,000	10	10년
삼성테크윈	Digimax400	4102	4102	300,000	10	10년
니콘	Coolpix4300	4102	4102	300,000	10	10년
삼성루스	루-20 Digital	4102	4102	300,000	10	10년
포탁	LS-4430(Dock 포함)	4102	4102	300,000	10	10년
삼성루스	C-4502	4102	4102	300,000	10	10년
삼성루스	X-1	4102	4102	300,000	10	10년
미놀타	DIMAGE-F100	4102	4102	300,000	10	10년
삼성테크윈	Digimax410	4102	4102	300,000	10	10년

© 2010 생능출판사 All rights reserved



선택정렬(selection sort)

- 선택정렬(selection sort): 정렬이 안된 숫자들중에서 최소값을 선택
하여 배열의 첫번째 요소와 교환



© 2010 생능출판사 All rights reserved



```
#include <stdio.h>
#define SIZE 6
int main(void)
{
    int a[SIZE] = { 5, 3, 8, 1, 2, 7};
    int i, j, temp, minIndex;

    for(i = 0; i < SIZE-1; i++) {
        // minIndex를 찾는다
        [red box]

        // i번째 원소와 minIndex 위치의 원소를 교환
        [red box]
    }
    for(i = 0; i < SIZE; i++)
        printf("%d ", a[i]);
    printf("\n");
    return 0;
}
```

0 1 2 3 4 5 6 7 8 9
계속하려면 아무 키나 누르십시오 ...



© 2010 생능출판사 All rights reserved

프론트 12th 수업에서 다루어지는 것들 - 배열을 함수 매개변수로 전달하기.

크기가 5인 정수형 배열의 원소들을 출력하려한다. main함수만으로 작성하면

```
int main(void)
{
    int arr[5] = {10, 20, 30, 40, 50};
    int i;

    for (i = 0; i < 5; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// 이제 위의 프로그램을 배열을 함수 매개변수로 하는
// printArray함수를 정의, 호출하여 작성해보자.

void printArray(int a[]);
int main(void)
{
    int arr[5] = {10, 20, 30, 40, 50};

    printArray(arr); // 호출시 인수로는 배열이름을 사용한다.
}
```

```
void printArray(int a[]) // 정의시 매개변수에는 배열이름[]
{
    int i;
    for (i = 0; i < 5; i++)
        printf("%d ", a[i]);
    printf("\n");
    return;
}
```

그런데, 위의 프로그램은 좋은 프로그램은 아니다. 배열을 함수 매개변수로 전달하는 경우, 대부분 그 크기를 같이 전달하는 것이 좋다. 즉 printArray 함수를 아래와 같이 수정할 수 있다. 앞으로는 이런 형식으로 사용하도록 한다.

```
// 더 좋은 코드
void printArray(int a[], int size);

int main(void)
{
    int arr[5] = {10, 20, 30, 40, 50};
    printArray(arr, 5); // 호출할 때의 인수에서는 배열의 이름과 배열의 크기
}

void printArray(int a[], int size) // 정의시 매개변수에는 배열이름[], 배열의크기를 받는 size변수
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", a[i]);
    printf("\n");
    return;
}
```

배열의 함수 매개변수 전달
✓ 호출: 인수는 배열이름, 배열크기(예: printArray(arr, 5)
이때 배열크기는 5 혹은 5로 정의한 심볼릭 상수(예: SIZE)를 써도 되고
sizeof(arr) / sizeof(int) 혹은 sizeof(arr) / sizeof(arr[0])
✓ 정의: 매개변수는 배열형식의 선언, 크기(예: void printArray(in a[], int size) {...})

■ 연습 a 위의 printArray 함수(더 좋은 코드)를 사용하고 아래의 출력이 나오게 프로그램을 완성하라.

```
#include <stdio.h>
void printArray(int a[], int size);
int main(void)
{
    int list1[5] = {10, 20, 30, 40, 50};
    int list2[3] = {100, 200, 300};

    // 위의 printArray함수를 사용하여 위의 두 배열의 요소를 출력하라
    printArray(list1, 5);
    printArray(list2, 3);
}

void printArray(int a[], int size) // 위의 코드 그대로
{ ... }
```

```
int main(void)
{
    int x;

    함수 1(x);
}

void 함수 1(int a)
{
    ...
}
```

정의부분의 매개변수인 a는
사실상 포인터(pointer)이다.
뒤에서(2 학기에!!) 배운다.

지금온 그냥 배열의 이름처럼
사용한다고 생각하자

```
C:\WINDOWS\system32\cmd.exe
10 20 30 40 50
100 200 300
계속하려면 아무 키나 누르십시오 . . .
```

■ 연습b

```
#include <stdio.h>
void printArray(int a[], int size);
void changeArray(int b[], int size);
int sumArray(int c[], int size);
int main(void)
{
    int data[5] = {10, 20, 30, 40, 50};
    printArray(data, 5); // 호출할 때의 인수에서는 배열의 이름을 사용한다.

    changeArray(data, 5); // 5대신에 sizeof(data) / sizeof(int)를 써도 된다.
    printArray(data, 5);
    printf("배열의 합은 %d\n", sumArray(data, 5));
}

void printArray(int a[], int size) // 앞에 것 그대로
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", a[i]);
    printf("\n");
    return;
}

void changeArray(int b[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        b[i] *= 10;
    return;
}

int sumArray(int c[], int size)
{
    int i, sum = 0;
    for (i = 0; i < size; i++)
        sum += c[i];
    return sum;
}
```

■ 생각해보는 예제

```
#include <stdio.h>
void printArray(int a[], int size);
void test(int arr[], int size, int num);
int main(void)
{
    int data[5] = {10, 20, 30, 40, 50};
    int number = 10;

    printf("배열 data는");
    printArray(data, 5);
    printf("number는 %d\n", number);

    test(data, 5, number);

    printf("배열 data는");
    printArray(data, 5);
    printf("number는 %d\n", number);
}
```

```
void printArray(int a[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", a[i]);
    printf("\n");
    return;
}
```

void test(int a[], int size, int num) // 매개변수 a를 변화시키는 것은 인수 data를 변화시키는 것
// 그러나 매개변수 num을 변화시켜도 인수 number에는 영향 없음

```
{
    int i;
    for (i = 0; i < size; i++)
        a[i] *= 10; // a의 원소들을 10배

    num *= 10; // num을 10배
    return;
}
```

LABHW3 배열의 함수 매개변수 연습

■ LABHW3.1(<프로그래밍 논리의 이해>복습)

아래의 프로그램을 완성하라.

```
#include <stdio.h>
int main(void)
{
    int list[10] = {10, 20, 30, 40, 50, 40, 30, 20, 10, 0};
    int value;
    int keyIndex;

    printf("합은 %d\n", sumList(list, 10));

    printf("가장 큰 수는 %d\n", maxList(list, 10));

    printf("탐색할 값은? ");
    scanf("%d", &value);

    // value 가 list 에 있는가를 판별하여 "없다" 혹은 "***째에 있다"를 출력
```

```
    return 0;
}
```

```
//sumList 의 정의
int sumList(int arr[], int size)
{
    int i, total = 0;
    for (i = 0; i < size; i++)
        total += arr[i];
    return total;
}
```

```
// 가장 큰 값을 반환한다.
int maxList(int arr[], int size)
{

```

```
//key 가 없으면 -1 을 있으면 그 인덱스를 반환한다.
int indexSearch(int arr[], int size, int key)
{

```

■ LABHW3.2(배열, 최대값 최소값 구하기)

0에서의 99까지의 난수를 넣은 정수형 배열(크기 10)에서 최대값을 가진 원소와 최대값을 가진 원소를 찾아서 인덱스와 값을 출력하는 프로그램을 위해 아래에서 printMinMax 함수를 완성하라. 아래와 같은 실행결과가 나오게 하라.

```
65 16 15 4 4 54 98 2 9 29
최대값: 인덱스 = 6, 값 = 98
최소값: 인덱스 = 7, 값 = 2
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdlib.h>
#include <time.h>
void initArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        arr[i] = rand() % 100;
    return;
}

void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

```
void printMinMax(int arr[], int size)
{
    // 최소값, 최대값을 찾아서 인덱스와 함께 출력하도록 정의하라

```

```
}
```

```
int main(void) // 변경하지 말라
{
    int a[10];
    //srand(time(NULL));
    srand(100); // 실행결과가 맞나 보기 위해서 seed를 100으로 고정
    initArray(a, 10);
    printArray(a, 10);
    printMinMax(a, 10);

    return 0;
}
```

```
C:\Windows\system32\cmd.exe
합은 250
가장 큰 수는 50
탐색할 값은? 40
값 40를 갖는 첫번째 값은 4번째에 있습니다
계속하려면 아무 키나 누르십시오 . . .
```

■ 기차표 예매

□ LABHW3_3_1 PROJECT_좌석예약을 모듈화)

왼쪽 코드는 실행되는 완성본이다. 이를 실행시켜서 실행결과를 확인하라.

오른쪽에 주어진 뼈대코드처럼 함수들을 정의해서 다시 작성하라. main함수는 변경하지 말라.

```
// 완성본
#define SIZE 10
#include <stdio.h>

int main(void)
{
    int seatChoice;
    char choice; // 계속 여부
    int seats[SIZE] = {0};
    int i;

    printf("*****좌석 예약 시스템*****\n");

    printf("\n좌석을 예약하시겠습니까?(y/n) ");
    scanf("%c", &choice);
    while (choice == 'y')
    {
        // 좌석표 출력
        printf("-----\n");
        printf(" 1 2 3 4 5 6 7 8 9 10\n");
        printf("-----\n");
        for(i = 0; i < SIZE; i++)
            printf("%2d", seats[i]);
        printf("\n");

        printf("몇번째 좌석? ");
        scanf("%d", &seatChoice);

        if (seats[seatChoice - 1] == 0) { //예약 가능이면
            seats[seatChoice-1] = 1;
            printf("예약되었습니다.\n");

            // 예약내용 반영한 좌석표 출력
            printf("-----\n");
            printf(" 1 2 3 4 5 6 7 8 9 10\n");
            printf("-----\n");
            for(i = 0; i < SIZE; i++)
                printf("%2d", seats[i]);
            printf("\n");
        }
        else // 이미 예약되었으면
            printf("이미 예약된 자리입니다.\n");

        while (getchar() != '\n'); //getchar()는 하나 문자입력

        printf("\n좌석을 예약하시겠습니까?(y/n) ");
        scanf("%c", &choice);
    }
}
```

```
//함수와 뼈대 코드를 완성하라
#define SIZE 10
#include <stdio.h>

char askReservation();
void printSeats(int s[], int size);
void processReservation(int s[], int size, int seatNumber);

int main(void) // 변경하지 마라
{
    int seatChoice;
    int seats[SIZE] = {0};

    printf("*****좌석 예약 시스템*****\n");

    while (askReservation() == 'y')
    {
        printSeats(seats, SIZE);

        printf("몇번째 좌석? ");
        scanf("%d", &seatChoice);

        processReservation(seats, SIZE, seatChoice);

        while (getchar() != '\n'); // 버퍼 비움
    }
}

char askReservation()
{
    ...
}

void processReservation(int s[], int size, int seatNumber)
{
    ...
}

void printSeats(int s[], int size)
{
    ...
}
```

- LABHW3_3.2 위의 문제와는 조금 예매 방식이 다른 기차표 예매 프로그램을 작성하려고 한다. 간단한 구현을 위해서 좌석은 모두 10 개라고 하자. 예매할 좌석수를 입력받아 빈 자리를 앞에서부터 차례로 할당한다. 0 은 예매 가능, 1 는 이미 예약되었음을 의미한다. 더 이상 예매할 자리가 남아있지 않으면 프로그램을 종료한다.

가정: 남은 자리만큼 예매자수가 정확히 입력된다고 가정하자. 즉, 3 명 4 명이 예약된 후 다음에는 남은 좌석이 3 자리 이므로 3 명이 입력된다. (실행예 A 참조)

요구사항: 가능하면 프로그램을 모듈화하라. (함수 사용!).

힌트: 위의 문제에서와 유사하게 모듈화 할 수 있다. 즉, askReservation, processReservation 는 수정하여 printSeats 는 그대로 사용한다.

<실행예 A>

```
*****좌석 예약 시스템*****
 1 2 3 4 5 6 7 8 9 10
0 0 0 0 0 0 0 0 0 0
좌석을 예약하시겠습니까?(몇명) 3
 1 2 3 4 5 6 7 8 9 10
1 1 1 0 0 0 0 0 0 0
좌석을 예약하시겠습니까?(몇명) 4
 1 2 3 4 5 6 7 8 9 10
1 1 1 1 1 1 1 0 0 0
좌석을 예약하시겠습니까?(몇명) 3
 1 2 3 4 5 6 7 8 9 10
1 1 1 1 1 1 1 1 1 1
계속하려면 아무 키나 누르십시오 . . .
```

□ LABHW3_3_challenge

위의 문제를 다음과 같이 수정하라.

가정: 남은 좌석보다 많은 자리를 예약하려고 할 때 예약을 할 수 없는 인원수를 출력한다. (실행예 B 참조)

<실행예 B>

```
*****좌석 예약 시스템*****
 1 2 3 4 5 6 7 8 9 10
0 0 0 0 0 0 0 0 0 0
좌석을 예약하시겠습니까?(몇명) 3
 1 2 3 4 5 6 7 8 9 10
1 1 1 0 0 0 0 0 0 0
좌석을 예약하시겠습니까?(몇명) 4
 1 2 3 4 5 6 7 8 9 10
1 1 1 1 1 1 1 0 0 0
좌석을 예약하시겠습니까?(몇명) 5
>>>2명은 예약 안됨
 1 2 3 4 5 6 7 8 9 10
1 1 1 1 1 1 1 1 1 1
계속하려면 아무 키나 누르십시오 . . .
```

■ 배열과 집합

□ LABHW3.4.1(다중집합) (난이도 중)

수학에서 다중집합(multiset)은 원소의 중복을 허용한다. 최대 5 개의 정수형 원소를 저장할 수 있는 다중 집합에 정수를 입력 받아 원소로 추가하고, 그 때마다 다중집합의 원소들을 출력하시오.

다중집합의 원소가 5 개가 되면 프로그램을 종료한다.

- 함수를 사용하여 모듈화 하라. (예: printSet 등등)

```
다중집합에 추가할 원소: 30
{ 30 }
다중집합에 추가할 원소: 30
{ 30, 30 }
다중집합에 추가할 원소: 20
{ 30, 30, 20 }
다중집합에 추가할 원소: 10
{ 30, 30, 20, 10 }
다중집합에 추가할 원소: 20
{ 30, 30, 20, 10, 20 }
계속하려면 아무 키나 누르십시오 . . .
```

□ LABHW3.4.2(집합) (난이도 중상)

수학에서 집합(set)은 다중집합(multiset)과 다르게 원소의 중복을 허용하지 않는다. 최대 5 개의 정수형 원소를 저장할 수 있는 집합(set)을 구현하시오. 정수를 입력받아 집합의 원소로 추가하고, 그 때마다 집합의 원소들을 출력하시오.

집합의 원소가 5 개가 되면 프로그램을 종료한다.

```
집합에 추가할 원소: 20
{ 20 }
집합에 추가할 원소: 20
20는 이미 집합에 있음
집합에 추가할 원소: 10
{ 20, 10 }
집합에 추가할 원소: 30
{ 20, 10, 30 }
집합에 추가할 원소: 10
10는 이미 집합에 있음
집합에 추가할 원소: 50
{ 20, 10, 30, 50 }
집합에 추가할 원소: 40
{ 20, 10, 30, 50, 40 }
계속하려면 아무 키나 누르십시오 .
```

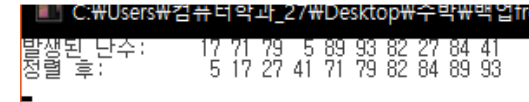
■ LABHW3_5(<프로그래밍 논리의 이해>의 논리 11: 선택 정렬 함수화)

수업시간에 배운 선택 정렬을 이용하여 다음과 같은 프로그램을 작성하라.
0 이상 100 미만의 난수를 발생시켜 이를 정렬하여 아래와 같이 출력하라.

요구사항

배열을 정렬하는 함수 selectionSort 를 이용하라. 원형은 다음과 같다.

```
void selectionSort(int list[], int size);
```



```
C:\Users\컴퓨터학과_27\Desktop\수학\백업Tr
발생된 난수:      17 71 79 5 89 93 82 27 84 41
정렬 후:          5 17 27 41 71 79 82 84 89 93
```

디버거를 이용하여

프로그램을 디버깅(에러 고치기) 하자

LAB 2 디버거 연습

■ LAB2_1(디버깅 연습)아래와 같은 프로그램을 작성해 보자.

```
#include <stdio.h>
int main(void)
{
    int x = 0;
    int y = 1;
    print_add(x, y);

    x = 10;
    y = 11;
    print_add(x, y);
}

void print_add(int a, int b)
{
    printf("%d와 %d의 합은 %d입니다. \n", a, b, a + b);
    printf("함수가 수행되었습니다.\n");
}
```

가) 위의 프로그램을 build해 보자. VS 아래 창에 두 개의 에러 메시지가 보일 것이다. 이를 해석하고 이해한다. (주의: 여러분은 빌드 에러 메시지들에 익숙해져야 하고, 메시지의 의미를 아는 것은 프로그래밍 능력과 매우 밀접한 관계가 있다. 대부분 매우 간단한 영어 문장이고 영어 단어가 의미하는 것만 알면 쉽게 이해된다. 무서워 하지 말고 항상 주의 깊게 보고 어떤 의미인지를 하나 하나 알아가도록 한다)

자, 그럼 위의 프로그램에 함수 프로토타입을 선언해 줌으로써 에러 메시지가 없어지는 것을 확인해 보자.

나) 빌드가 되었다면, F10을 눌러 한 라인씩 수행시켜 보자.
print_add() 함수의 바디 안으로 디버거 수행 라인이(노란색 화살표) 들어가는가? 어떻게 하면 바디 부분으로 들어갈 수 있을지 생각해 보자.

다) <호출된 함수의 바디로 들어가는 방법>

방법1: 함수의 바디 부분에 BP 생성

print_add() 함수의 바디 부분에 F9를 눌러 BP를 생성해 두면 된다는 것이다. 자 이제 한번 해보자.

방법2: F11의 사용

print_add() 함수를 부르는 라인에서(메인 함수 안에 있는) F10 대신 F11을 누르는 것도 한번 해보자. 그리고 이 F11 키의 사용도 꼭 기억한다.

참소리) 디버깅을 하기 위해서는 당연히 함수의 바디 부분을 한 라인씩 수행할 수 있어야 한다. 어떻게 바디 부분에서 멈춰, 한 라인씩 수행할 수 있는지 기억하도록 하자. 꼭!

■ LAB2_2(디버깅 연습 추가)아래의 프로그램을 실행시켜보라.

시도 1: F10을 이용하여 모든 라인을 진행시키면서 [Variable]창과 [Watch]창의 내용의 변화를 살펴보라. (Watch창에 변수의 이름(array, total)을 추가해본다) F5로 디버깅을 종료한다.

시도 2: A와 B에 BP를 설정(F9 사용)한 후 F5를 사용하여 BP 단위로 진행시켜보라.

```
// F10(한 라인씩 진행)
// F5(BP혹은 프로그램끝까지 진행)
// F9(BP설정/해제)를 연습합시다.
#include <stdio.h>
int main(void)
{
    int num;
    int i;
    int total;
    int array[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    printf("Enter a number:");
    scanf("%d", &num);

    for (i = 0; i < num; i++)
        array[i] *= 10;

    total = 0; // A
    i = 0;
    while (i < num)
    {
        total += array[i];
        i++; // B
    }

    printf("Total is %d\n", total);
}
```

■ LAB2.3 디버깅 연습 추가(함수 부분)

시도 1: F10만을 눌러서 디버깅 해보자. F5로 종료한다.

시도 2: 어떻게하면 print_add, multiply 등의 함수 바디 안으로 디버깅을 진행할 수 있는가?

방법1: 함수안에 BP 설정...(F9 이용)

방법2: F11이용 // F11을 사용하여 호출되는 함수의 바디로 진행하기

```
void print_add(int a, int b);
int multiply(int a, int b);
#include <stdio.h>
int main(void)
{
    int x = 10;
    int y = 20;
    int result;

    print_add(x, y);

    result = multiply(x, y);

    printf("main함수: %d와 %d의 곱은 %d입니다. \n", x, y, result);
}

void print_add(int a, int b)
{
    int sum;
    sum = a + b;
    printf("print_add함수: %d와 %d의 합은 %d입니다. \n", a, b, sum);
    printf("print_add함수가 수행되었습니다.\n\n");

    return;
}

int multiply(int a, int b)
{
    int product;
    product = a * b;
    printf("add함수가 수행되었습니다.\n");

    return product;
}
```