실습 8. 정렬

한국외국어대학교
컴퓨터.전자시스템공학전공
2016년 1학기
고 석 훈

실습 8-1: 정렬

- 강의 시간에 설명한 정렬 알고리즘을 참고하여 5가지 정렬 함수를 구현하여라.
  - `void selectionSort(int *a, int n);` // Selection sort
  - `void bubbleSort(int *a, int n);` // Bubble sort
  - `void quickSort(int *a, int n);` // Quick sort
  - `void insertionSort(int *a, int n);` // Insertion sort
  - `void shellSort(int *a, int n);` // Shell sort

소스코드 템플릿

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

void makeRandomData(int *a, int n)
{
    int i;

    srand(time(NULL));
    for (i = 0; i < n; i++)
        a[i] = rand()%1000;
}

void printData(char *title, int *a, int n)
{
    int i;

    printf("%s = {\n%3d", title, a[0]);
    for (i = 1; i < n; i++)
        printf(", %3d", a[i]);
    printf("\n}\n");
}
```

```c
void selectionSort(int *a, int n)
{
}

void bubbleSort(int *a, int n)
{
}

void quickSort(int *a, int n)
{
}

void insertionSort(int *a, int n)
{
}

void shellSort(int *a, int n)
{
}
```

```c
int main(int argc, char* argv[])
{
#define SIZE    15

    int data[SIZE], work[SIZE];

    makeRandomData(data, SIZE);
    printData("Unsorted Data", data, SIZE);

    memcpy(work, data, sizeof(data));
    selectionSort(work, SIZE);
    printData("Selection Sort", work, SIZE);

    memcpy(work, data, sizeof(data));
    bubbleSort(work, SIZE);
    printData("Bubble Sort", work, SIZE);

    memcpy(work, data, sizeof(data));
    quickSort(work, SIZE);
    printData("Quick Sort", work, SIZE);

    memcpy(work, data, sizeof(data));
    insertionSort(work, SIZE);
    printData("Insertion Sort", work, SIZE);

    memcpy(work, data, sizeof(data));
    shellSort(work, SIZE);
    printData("Shell Sort", work, SIZE);

    getchar();
    return 0;
}
```

실행 예

```
Unsorted Data = {
218, 936, 148,  53, 858, 138, 601, 946, 403, 829, 241, 472, 925, 187, 747
}
Selection Sort = {
 53, 138, 148, 187, 218, 241, 403, 472, 601, 747, 829, 858, 925, 936, 946
}
Bubble Sort = {
 53, 138, 148, 187, 218, 241, 403, 472, 601, 747, 829, 858, 925, 936, 946
}
Quick Sort = {
 53, 138, 148, 187, 218, 241, 403, 472, 601, 747, 829, 858, 925, 936, 946
}
Insertion Sort = {
 53, 138, 148, 187, 218, 241, 403, 472, 601, 747, 829, 858, 925, 936, 946
}
Shell Sort = {
 53, 138, 148, 187, 218, 241, 403, 472, 601, 747, 829, 858, 925, 936, 946
}
```

선택 정렬 알고리즘

```
selectionSort(a[ ], n)
    for (i ← 0; i < n - 1; i++) do {
        min ← i;
        for (j ← i; j < n; j++) do
            if (a[j] < a[min]) min ← j;
        swap(a[i], a[min]);
    }
end selectionSort()
```

버블 정렬 알고리즘

```
bubbleSort(a[ ], n)
    for (i ← n - 1; i ≥ 0; i--) do
        for (j ← 0; j < i; j++) do
            if (a[j] > a[j+1]) then
                swap(a[j], a[j + 1]);
end bubbleSort()
```

퀵 정렬 알고리즘 [1/2]

```
quickSort(a[ ], begin, end)
    if (begin < end) then {
        pivot ← partition(a, begin, end);
        quickSort(a[ ], begin, pivot - 1);
        quickSort(a[ ], pivot + 1, end);
    }
end quickSort()
```

퀵 정렬 알고리즘 [2/2]

```
partiton(a[ ], begin, end)
        p ← begin;
        L ← begin;
        R ← end;
        while (L < R) do {
                while (a[L] ≤ a[p] and L < end) do L++;
                while (a[R] > a[p]) do R--;
                if (L < R) then
                        swap(a[L], a[R]);        // L의 원소와 R의 원소 교환
        }
        swap(a[p], a[R]);                        // R의 원소와 피봇 원소 교환
        return R;
end partition()
```

삽입 정렬 알고리즘

```
insertionSort(a[ ], n)
    for (i ← 1; i < n; i++) do {
        val ← a[i];
        for (pos ← i; pos > 0; pos--) do {
            if (val < a[pos – 1])
                a[pos] ← a[pos – 1];
            else
                break;
        }
        a[pos] ← val;
    }
end insertionSort()
```

쉘 정렬 알고리즘

```
shellSort(a[ ], n)
    for (interval ← n/2; interval > 0; interval = interval/2) do {
        for (i ← interval; I < n; i++) do {
            val ← a[i];
            for (pos ← i; pos >= interval; pos -= interval) do {
                if (val < a[pos – interval])
                    a[pos] ← a[pos – interval];
                else
                    break;
            }
            a[pos] ← val;
        }
    }
end shellSort()
```

실습 7-2: 정렬 성능비교 [1/2]

- 5가지 정렬 함수의 성능을 측정하고 분석하여라.
  - 각 정렬 함수에서 데이터 비교횟수 측정 루틴을 추가한다.
    - int nCompare;
    - 함수 호출 전에 0으로 초기화
    - 함수 내에서 비교문을 실행할 때마다 1 증가
    - 함수 실행이 끝나면 비교횟수 확인
  - 각 정렬 함수에 데이터 교환횟수 측정 루틴을 추가한다.
    - int nMove;
    - 사용 방법은 nCompare와 동일

실습 7-2: 정렬 성능비교 [2/2]

- 5가지 정렬 함수의 성능을 측정하고 분석하여라.
  - 데이터 크기를 100, 200, 300 ~ 1000으로 증가시키면서 각 정렬 함수의 데이터 비교횟수와 교환횟수를 출력한다.
  - 출력된 데이터 비교횟수와 교환횟수를 **보고서에 표와 그래프로 표시한다.**
  - 각 정렬 알고리즘 별로 측정 결과를 분석하여 **분석 결과를 보고서에 기술한다.**

성능 측정 코드의 예

```
void swap(int *a, int *b)
{
    int t = *a;
    *a = *b;
    *b = t;

    nMove += 3; // 데이터 이동 카운터 증가
}

void selectionSort(int *a, int n)
{
    int i, j;
    int min;

    for (i = 0; i < n - 1; i++) {
        min = i;
        for (j = i; j < n; j++) {
            nCompare++; // 데이터 비교 카운터 증가
            if (a[j] < a[min])
                min = j;
        }
        swap(&a[i], &a[min]);
    }
}
```

실행 예

Selection Sort

size	compare	move	comp+move
100	5[illegible]	2[illegible]	5[illegible]
200	20[illegible]	5[illegible]	20[illegible]
300	45[illegible]	8[illegible]	46[illegible]
400	80[illegible]	11[illegible]	81[illegible]
500	125[illegible]	14[illegible]	126[illegible]
600	180[illegible]	17[illegible]	182[illegible]
700	245[illegible]	20[illegible]	247[illegible]
800	320[illegible]	23[illegible]	322[illegible]
900	405[illegible]	26[illegible]	408[illegible]
1000	500[illegible]	29[illegible]	503[illegible]

Bubble Sort

size	compare	move	comp+move
100	4[illegible]	6[illegible]	11[illegible]
200	19[illegible]	30[illegible]	50[illegible]
300	44[illegible]	67[illegible]	112[illegible]
400	79[illegible]	118[illegible]	197[illegible]
500	124[illegible]	183[illegible]	308[illegible]
600	179[illegible]	268[illegible]	447[illegible]
700	244[illegible]	358[illegible]	603[illegible]
800	319[illegible]	476[illegible]	796[illegible]
900	404[illegible]	598[illegible]	1003[illegible]
1000	499[illegible]	743[illegible]	1243[illegible]

Quick Sort

size	compare	move	comp+move
100	[illegible]	[illegible]	1[illegible]
200	1[illegible]	1[illegible]	2[illegible]
300	2[illegible]	1[illegible]	4[illegible]
400	3[illegible]	2[illegible]	6[illegible]
500	4[illegible]	3[illegible]	7[illegible]
600	6[illegible]	3[illegible]	10[illegible]
700	7[illegible]	4[illegible]	11[illegible]
800	9[illegible]	5[illegible]	14[illegible]
900	9[illegible]	6[illegible]	16[illegible]
1000	11[illegible]	7[illegible]	18[illegible]

Insertion Sort

size	compare	move	comp+move
100	2[illegible]	2[illegible]	4[illegible]
200	10[illegible]	10[illegible]	20[illegible]
300	22[illegible]	22[illegible]	45[illegible]
400	39[illegible]	39[illegible]	79[illegible]
500	61[illegible]	61[illegible]	123[illegible]
600	90[illegible]	90[illegible]	180[illegible]
700	120[illegible]	120[illegible]	240[illegible]
800	159[illegible]	159[illegible]	319[illegible]
900	200[illegible]	200[illegible]	400[illegible]
1000	248[illegible]	248[illegible]	497[illegible]

Shell Sort

size	compare	move	comp+move
100	[illegible]	[illegible]	1[illegible]
200	2[illegible]	2[illegible]	4[illegible]
300	3[illegible]	3[illegible]	6[illegible]
400	5[illegible]	5[illegible]	10[illegible]
500	6[illegible]	6[illegible]	12[illegible]
600	8[illegible]	8[illegible]	16[illegible]
700	9[illegible]	9[illegible]	19[illegible]
800	12[illegible]	12[illegible]	25[illegible]
900	14[illegible]	15[illegible]	29[illegible]
1000	14[illegible]	15[illegible]	29[illegible]

실습 보고서 작성 요령 [1/2]

- 표지 기재 내용
  - 우측 이미지 참고
- 보고서 목차
  1. 실습 문제 소개
  2. 소스코드
     - 본인이 구현한 함수의 코드만 삽입
  3. 테스트 결과
  4. 작성자 코멘트

자료구조 실습보고서

실습 8. 정렬

2016년 5월 13일
학번: 201512345
이름: 홍길동

실습 보고서 작성 요령 [2/2]

- 보고서 작성 요령
  - 소스코드 삽입 요령
    - 주석을 이용하여 프로그램 코드를 설명한다.
    - 보고서에는 8~10pt 크기로 읽기 쉬운 형태로 삽입한다.
  - 작성자 코멘트
    - 실습 과정에서 느낀점, 새롭게 배운것 등을 자유롭게 기술한다.

실습 보고서 제출 요령

- e-class 제출
  - 실습보고서 파일과 프로그램 소스코드(*.c)를 하나의 압축파일로 만들어 e-class에 제출한다.
  - 돌아오는 강의 시간 전날 자정까지 제출한다.

- 인쇄물 제출
  - 돌아오는 강의 시간에 실습보고서를 인쇄하여 제출한다.