Programming Methodology
Instructor: Kyuseok Shim

Project #2: Sorting
Due Date: 0:0 a.m. between 2008-5-5 & 2008-5-6

Introduction

이번 프로젝트에서는 클래스 상속을 이용하여 몇 가지 소팅 알고리즘을 구현할 것이다. 구현할 소팅 알고리즘은 Insertion sort, Merge sort, Quick sort, merge sort를 수정한 Modified-Merge sort 그리고 Stooge sort이다

프로젝트 구현에서는 5개의 소팅에 대하여 각각의 클래스를 만들 것이다. 각각의 클래스는 자신에게 해당되는 Sort()함수를 가지고 있어서 소팅을 수행한다. 이번에 구현하는 4가지의 소팅 알고리즘에서는 같은 유형의 데이터(int형의 어레이)를 사용하기 때문에 같은 방식으로 데이터를 관리 혹은 사용을 할 수 있다. 따라서 데이터만을 관리(입력, 저장, 출력)하는 클래스를 따로 만들어 줘서 각각의 소팅 클래스에서 이 데이터 클래스를 상속하여서 자신의 데이터로 사용을 할 것이다.

이와 더불어 5가지 알고리즘의 성능 비교를 위해 시간을 측정하는 클래스를 만들어 성능을 비교할 것이다.

Implementation

1. DataStore 클래스 구현
- 소팅에 사용될 데이터를 저장
- 어레이를 argument로 받아서 자신의 멤버 변수 어레이에 저장.
- 파일에 소팅결과를 출력할 수 있는 함수 구현

2. ClockChecker 구현
- 소팅 알고리즘의 성능 측정을 위해 사용.
- time header파일(#include <ctime>)의 clock() 함수를 이용 (clock() : 프로그램이 시작 후 몇 클락 틱(tick)이 지났는지 리턴)

3. 5가지 소팅 알고리즘 구현
- Insertion, Merge, Quick, Stooge, Modified-Merge sort구현
- DataStore 클래스를 상속

Input

데이터 입력은 파일을 통해 이루어진다. 입력파일의 이름은 콘솔창의 parameter입력을 통해 정한다. 이다. 입력파일은 첫 줄에 데이터의 개수와 두 번째 줄에 정수형 숫자 데이터들의 나열로 이루어져 있다.

Output

4가지 소팅 결과를 모두 파일로 출력할 것이다. 파일 이름은 각각 "out_insertion.txt" "out_merge.txt", "out_quick.txt", "out_modifiedmerge.txt", "out_stooge.txt"가 된다. 출력파일에는 소트된 데이터와 각각의 데이터를 구분해줄 스페이스(' ') 이외에는 출력하지 않는다.

Algorithm

Insertion sort, Merge sort, Quick sort : 강의노트 참고

Modified-Merge sort : 기존 데이터 셋를 2등분해서 merge 하던 것을 3등분 하여 Merge하는 것으로 수정

Stooge sort : 본 문서 뒷부분 참고

Skeleton Code

- 스켈레톤 코드는 구현 사항에 따라 수정 가능
- 아래의 코드는 단지 예시임..

```
class DataStore
{
        protected :
                int *data;
                int numData;

        public :
                void GetInput(int *arr);                            // 입력
                void OutputDataToFile(ofstream &ofs) const;         // 파일로 출력

                DataStore(int n) { }
                ~DataStore() { }
};

class ClockChecker
{
private :
        clock_t startClock;
        clock_t endClock;

public:
        void SetStartClock();
        void SetEndClock();
        clock_t ElapsedClock() const;                       // endClock - startClock
};

class InsertionSorter : public DataStore
{

          public :
                 void Sort();
                 void InsertionSort();
```

```
                InsertionSorter(int n) : DataStore(n) { };

};

class MergeSorter : public DataStore
{

        public :
                void Sort();
                void MergeSort(int *array, int start, int end);
                void Merge(int *array, int start, int q, int end);
                MergeSorter(int n) : DataStore(n) { };

};

class QuickSorter : public DataStore
{
        public :
                        void Sort();
                        void QuickSort(int *array, int l, int r);
                        int Partition(int *array, int l, int r);
                        QuickSorter(int n) : DataStore(n) { }

};

class ModifiedMergeSorter : public DataStore
{
       // MergeSort을 참고하여 작성
};

class StoogeSorter : public DataStore{
        public:
                         void Sort();
                         void StoogeSort(int *array, int start, int end);

                         StoogeSorter(int n) : DataStore(n) { }
```

```
};

int main(int argc, char *argv[])
{
        int num;

        ifstream ifs("data.txt");

        ofstream ofs_i("out_insertion.txt");
        ofstream ofs_m("out_merge.txt");
        ofstream ofs_q("out_quick.txt");
        ofstream ofs_mm("out_modifiedmerge.txt");
        ofstream ofs_s("out_stooge.txt");

        ClockChecker cck;

        ifs >> num;

        int *data = new int[num];
        for(int i=0;i<num;i++)
        {
                ifs >> data[i];
        }
        cout << "number of data: " << num << endl << endl;
        ifs.close();

        InsertionSorter iSorter(num);
        iSorter.GetInput(data);
        cout << "Insertion Sorting..... ";
        cck.SetStartClock();
        iSorter.Sort();
        cck.SetEndClock();
        cout << "Done." << endl;
        iSorter.OutputDataToFile(ofs_m);
        cout << "elapsed Clock of Insertion Sort: "
           << cck.ElapsedClock() << " clock ticks" << endl << endl;
```

```
        ofs_i.close();

        MergeSorter mSorter(num);
        mSorter.GetInput(data);
        cout << "Merge Sorting..... ";
        cck.SetStartClock();
        mSorter.Sort();
        cck.SetEndClock();
        cout << "Done." << endl;
        mSorter.OutputDataToFile(ofs_m);
        cout << "elapsed Clock of Merge Sort: "
           << cck.ElapsedClock() << " clock ticks" << endl << endl;
        ofs_m.close();

        QuickSorter qSorter(num);
        qSorter.GetInput(data);
        cout << "Quick Sorting..... ";
        cck.SetStartClock();
        qSorter.Sort();
        cck.SetEndClock();
        cout << "Done." << endl;
        qSorter.OutputDataToFile(ofs_q);
        cout << "elapsed Clock of Quick Sort: "
           << cck.ElapsedClock() << " clock ticks" << endl << endl;
        ofs_q.close();

        ModifiedMergeSorter mmSorter(num);
        mmSorter.GetInput(data);
        cout << "ModifiedMerge Sorting..... ";
        cck.SetStartClock();
        mmSorter.Sort();
        cck.SetEndClock();
        cout << "Done." << endl;
        mmSorter.OutputDataToFile(ofs_mm);
        cout << "elapsed Clock of ModifiedMerge Sort: "
           << cck.ElapsedClock() << " clock ticks" << endl << endl;
```

```
        ofs_mm.close();

        StoogeSorter sSorter(num);
        sSorter.GetInput(data);
        cout << "Stooge Sorting..... ";
        sSorter.Sort();
        cout << "Done." << endl;
        sSorter.OutputDataToFile(ofs_s);
        cout << "elapsed Clock of Stooge Sort: "
           << sSorter.ElapsedClock() << " clock ticks" << endl << endl;
        ofs_s.close();

        system("PAUSE");
        return EXIT_SUCCESS;
}
```

프로그램 실행 예시

- 콘솔

> project2.exe input.txt

number of data: 100000

Insertion Sorting..... Done.

Elapsed Clock of Insertion Sort: 8492 clocks

Merge Sorting..... Done.

Elapsed Clock of Insertion Sort: 253 clocks

Quick Sorting..... Done.

Elapsed Clock of Quick Sort: 46 clocks

ModifedMerge Sorting..... Done.

Elapsed Clock of ModifiedMerge Sort: 141 clocks

Stooge Sorting..... Done.

Elapsed Clock of Stooge Sort: 8735198 clocks

- 입력파일

100000

42 8468 6335 6501 9170 5725 1479 9359 6963 4465 5706 8146 3282 6828 9962 492 2996 1943 4828 5437 2392 4605 3903 154 293 2383 7422 8717 9719 9896 5448 1727 4772 1539 1870 9913 5668 6300 7036 9895 8704 3812 1323 334 7674 4665 5142 7712 8254 6869 5548 7645 2663 2758 38 2860 8724 9742 7530 779 2317....

- 출력파일

1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 9 ....

**** 출력파일은 꼭 아래와 같은 형식으로 작성...**

숫자(스페이스)숫자(스페이스)숫자(스페이스).....숫자(스페이스)숫자(스페이스)(끝)

**** 안 지킬 경우 감점!!**

Stooge sort 보충설명 (From http://en.wikipedia.org/wiki/Stooge_sort)

Stooge sort is a recursive sorting algorithm with a time complexity of about $O(n^{2.7})$. The exponent's exact value is log(3) / log(1.5) = 2.709...

The algorithm is defined as follows:

If the value at the end is smaller than the value at the start, swap them.
If there are 3 or more elements in the current list subset,
then:
Stooge sort the initial 2/3 of the list
Stooge sort the final 2/3 of the list
Stooge sort the initial 2/3 of the list again
else: exit the procedure

제출사항

1. 제출 방법 및 내용

1) 프로젝트 구현 파일 제출 (DEV C++을 사용하세요!!!)

- 파일은 ee 홈페이지에 과제 제출을 이용하여 제출한다.
- 소스 파일과 컴파일을 할 수 있는 파일과 실행파일을 압축하여 제출한다.
- 각 제출 파일 이름은 다음의 형식을 따른다.

pro_2_학번.zip

예) pro_2_2007-1234?.zip

- 각 소스 파일 상단에 **E-mail주소와 작성자 학번 이름을** 적는다.

2) 보고서 제출

간단한 설명과 discussion을 하드카피로 수업시간 전에 제출한다.

2. 보고서 내용

- 상속에 대한 설명(어떻게 구현 했으며, 어떤 것들을 상속 하는지..)
- 각각의 소팅 알고리즘에 대한 간략한 설명 및 자신의 구현 방법 설명
- 몇 개의 큰 사이즈의 데이터에 대한 시간 측정 결과 기록
- 5가지 알고리즘의 수행시간 차이 분석
- 등등….

2. 제출 일시

파일 제출 마감일: **2008년 5월 5일과 5일 6일사이 새벽 0:00**

구현 파일 제출 시각은 홈페이지 과제 제출 시각에 준한다.

보고서는 2008년 5월 6일 수업시간에 제출

채점 규칙 및 주의사항

- 임의의 입력 파일에 대하여 소팅 결과를 체크
- 5가지의 알고리즘에 대해 적절한 수행시간이 나와야 함
- 스켈레톤 코드는 마음대로 수정할 수 있음. 메인함수도 마찬가지. 단 5가지 소팅 클래스와 데이터, 클락체크 클래스는 반드시 작성할 것. 상속은 필수!
- 출력 파일에 대한 스펙은 꼭 지킬 것

딜레이는 하루에 10%씩 감점

Copy 발견 시 모든 숙제 0점 처리