ASSIGNMENT 2 - 62070501064 ONWIPA KUJAROENPAISAN –

Source Code

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

typedef struct DATAstruct{ //declare array to keep data in 3 fields
    long col;
    long long f1;
    char f2[100], f3[100];
} DATAstruct; //declare in name DATAstruct

void readfile(DATAstruct data[], int *count){ //declare function to read file .csv
    FILE *fp;
    DATAstruct x;
    if((fp = fopen("test.csv","r")) != NULL){ //if file was found
        while(fscanf(fp, "%lu, %llu, %[^,], %s", &x.col, &x.f1, &x.f2, &x.f3) == 4){ //then recieve in array each field
            data[*count] = x; //and count number of data
            *count = *count +1;
        }
        fclose(fp);
    }else {
        printf("\nCannot read file.\n"); //if file was not found then show warning
    }
}

void swap(DATAstruct *a, DATAstruct *b){ //function to swap data
    DATAstruct c;
    c = *a;
    *a = *b;
    *b = c;
}

//function of selection sort
void sltsort(DATAstruct data[], int start, int stop){
    int i, j, min;
    for(i = start; i < stop; i++){
        min = i;     //give first location is minimum
        for(j = i + 1; j <= stop; j++){
            if(data[j].f1 < data[min].f1) { //if next location is less than minimum
                min = j;
            }
        }
        swap(&data[min] , &data[i]); //then swap data in location
    }
}

//function to count time in selection sort
double time_slt(DATAstruct data[], DATAstruct test[], int count){
    clock_t start, end;
    int i, j;
    double time;
    for(i = 0; i < count; i++){ //copy data in array data to array test
                                //for data in array data will not change
        test[i] = data[i];
    }
        start = clock();
        sltsort(test, 0, count);
        end = clock();
        time = (double)(end - start) / CLOCKS_PER_SEC;
        return time;
}

//function for insertion sort
void istsort(DATAstruct data[],int start, int stop){
    int i, j, x;
    for(i = start + 1 ; i <= stop; i++){
        x = data[i].f1;
        for(j = i; ((j >start)&&(x < data[j-1].f1)); j--){ //loop to find data that less than x then swap
            data[j].f1 = data[j-1].f1; //exchange data in every location by location-1
            data[j].f1 = x;
        }
    }
}

//function to count time in insertion sort
double time_ist(DATAstruct data[], DATAstruct test[], int count){
    clock_t start, end;
    int i, j;
    double time;
    for(i = 0; i < count; i++){
```

```
        test[i] = data[i];
    }   //copy data in array data to array test
        //for data in array data will not change

        start = clock();
        istsort(test, 0, count-1);
        end = clock();
        time = (double)(end - start) / CLOCKS_PER_SEC;
        return time;
}

//function for bubble sort
void bbsort(DATAstruct data[], int start, int stop){
    int i,j;
    for(i = start ; i <= (stop-1); i++){
        for(j = stop; j > i; j--){
            if(data[j].f1 < data[j-1].f1){ //compare data
                swap(&data[j], &data[j-1]); //and swap
            }
        }
    }
}

//function to count time in bubble sort
double time_bb(DATAstruct data[], DATAstruct test[], int count){
    clock_t start, end;
    int i, j;
    double time;
    for(i = 0; i < count; i++){
        //copy data in array data to array test
        //for data in array data will not change
        test[i] = data[i];
    }
        start = clock();
        bbsort(test, 0, count-1);
        end = clock();
        time = (double)(end - start) / CLOCKS_PER_SEC;
        return time;
}

```

```
//function for shell sort
void shsort(DATAstruct data[], int start, int stop){
    int gap, changed, i;
    gap = stop - start + 1;
    do{
        gap = gap / 2; //fix the gap between data
        do{
            changed = 0; //changed is for checking if data is swap
            for(i = start; i < stop - gap + 1; i++){
                if(data[i].f1 > data[i+gap].f1 ){
                    swap(&data[i], &data[i+gap]);
                    changed = 1; //if there's changed it's mean sorting is not complete
                                 // then sort until it have no changed
                }
            }
        } while(changed == 1);
    }while(gap > 1); //loop until there will be one group
}

//function to count time in shell sort
double time_sh(DATAstruct data[], DATAstruct test[], int count){
    clock_t start, end;
    int i, j;
    double time;
    for(i = 0; i < count; i++){
        //copy data in array data to array test
        //for data in array data will not change
        test[i] = data[i];
    }
        start = clock();
        shsort(test, 0, count-1);
        end = clock();
        time = (double)(end - start) / CLOCKS_PER_SEC;
        return time;
}

//function to descending sort by selection sort
void dessltsort(DATAstruct data[], int start, int stop){
    int i, j, min;
    for(i = start; i < stop; i++){
```

```
            min = i;
            for(j = i + 1; j <= stop; j++){
                if(data[j].f1 > data[min].f1) { //sort from max to min
                    min = j;
                }
            }
            swap(&data[min] , &data[i]);
        }
    }

    //function to count time in descending sort of selection sort
    double destime_slt(DATAstruct data[], DATAstruct test[], int count){
        clock_t start, end;
        int i, j;
        double time;
        for(i = 0; i < count; i++){
            test[i] = data[i];
            //copy data in array data to array test
            //for data in array data will not change
        }
            start = clock();
            dessltsort(test, 0, count);
            end = clock();
            time = (double)(end - start) / CLOCKS_PER_SEC;
            return time;
    }

    //function for descending sort by insertion sort
    void desistsort(DATAstruct data[],int start, int stop){
        int i, j, x;
        for(i = start+1; i <= stop; i++){
            x = data[i].f1;
            for(j = i; ((j > start)&&(x > data[j-1].f1)); j--){ //sorting from max to min
                data[j].f1 = data[j-1].f1; //exchange data in each location
                data[j].f1 = x;
            }
        }
    }

    //function to count time in descending sort of insertion sort
    double destime_ist(DATAstruct data[], DATAstruct test[], int count){
        clock_t start, end;
        int i, j;
        double time;
        for(i = 0; i < count; i++){
        //copy data in array data to array test
        //for data in array data will not change
            test[i] = data[i];
        }
            start = clock();
            desistsort(test, 0, count-1);
            end = clock();
            time = (double)(end - start) / CLOCKS_PER_SEC;
            return time;
    }

    //function for descending sort by bubble sort
    void desbbsort(DATAstruct data[], int start, int stop){
        int i,j;
        for(i = start ; i <= (stop-1); i++){
            for(j = stop; j > i; j--){
                if(data[j].f1 > data[j-1].f1){ //if data in next location is more than before
                    swap(&data[j], &data[j-1]); //then swap data
                }
            }
        }
    }

    //function to count time in descending sort of bubble sort
    double destime_bb(DATAstruct data[], DATAstruct test[], int count){
        clock_t start, end;
        int i, j;
        double time;
        for(i = 0; i < count; i++){
        //copy data in array data to array test
        //for data in array data will not change
            test[i] = data[i];
        }
            start = clock();
            desbbsort(test, 0, count-1);
```

```
        end = clock();
        time = (double)(end - start) / CLOCKS_PER_SEC;
        return time;
}

//function for descending sort by shell sort
void desshsort(DATAstruct data[], int start, int stop){
    int gap, changed, i;
    gap = stop - start + 1;
    do{
        gap = gap / 2; //fixed tha gap between data
        do{
            changed = 0; //changed is for checking if data is swap
            for(i = start; i < stop - gap + 1; i++){
                if(data[i].f1 < data[i+gap].f1 ){
                    swap(&data[i], &data[i+gap]);
                    changed = 1;
                    //if there's changed it's mean sorting is not complete
                    // then sort until it have no changed
                }
            }
        } while(changed == 1);
    }while(gap > 1);
}

//function to count time in descending sort of shell sort
double destime_sh(DATAstruct data[], DATAstruct test[], int count){
    clock_t start, end;
    int i, j;
    double time;
    for(i = 0; i < count; i++){
    //copy data in array data to array test
    //for data in array data will not change
        test[i] = data[i];
    }
        start = clock();
        desshsort(test, 0, count-1);
        end = clock();
        time = (double)(end - start) / CLOCKS_PER_SEC;
        return time;
}

int main(){
    int count = 0;
    DATAstruct *data, *test;
    double time1, time2, time3,time4, time5, time6, time7, time8;
    double time51, time61, time71, time81;

    data = (DATAstruct *)malloc(100000*sizeof(DATAstruct));
    test = (DATAstruct *)malloc(100000*sizeof(DATAstruct));
    //declare 2 arrays because array data will not change data inside array

    readfile(data, &count); //order program to read file and count number of data
    printf("Loading..\n -> %d records\n",count); //show number of data

    //first round of sorting is normal sorting -- from min to max
    printf("\nFirst Round of Sorting Progess..\n");
    time1 = time_slt(data, test, 10000);
    time2 = time_ist(data, test, 10000);
    time3 = time_bb(data, test, 10000);
    time4 = time_sh(data, test, 10000);

    //I order to print data to check if it's sorting correctly
    printf("\n%llu %s %s\n",test[0].f1, test[0].f2, test[0].f3);
    printf("\n%llu %s %s\n",test[4999].f1, test[4999].f2, test[4999].f3);
    printf("\n%llu %s %s\n",test[9999].f1, test[9999].f2, test[9999].f3);

    //second round is sorting by add 1 data from data to sorting together
    printf("\n\nAdd 1 data");
    printf("\nSecond Round of Sorting Progess..\n\n");
    time5 = time_slt(data, test, 10001);
    time6 = time_ist(data, test, 10001);
    time7 = time_bb(data, test, 10001);
    time8 = time_sh(data, test, 10001);

    //I order to print data to check if it's sorting correctly as before
    printf("\n%llu %s %s\n",test[0].f1, test[0].f2, test[0].f3);
    printf("\n%llu %s %s\n",test[4999].f1, test[4999].f2, test[4999].f3);
    printf("\n%llu %s %s\n",test[9999].f1, test[9999].f2, test[9999].f3);

```

```c
    //third round is descending sort -- sort from max to min
    printf("\n\nDescending");
    printf("\nThird Round of Sorting Progess..\n\n");
    time51 = time_slt(data, test, 10001);
    time61 = time_ist(data, test, 10001);
    time71 = time_bb(data, test, 10001);
    time81 = time_sh(data, test, 10001);

    //I order to print data to check if it's sorting correctly as i did before
    printf("\n%llu %s %s\n",test[0].f1, test[0].f2, test[0].f3);
    printf("\n%llu %s %s\n",test[4999].f1, test[4999].f2, test[4999].f3);
    printf("\n%llu %s %s\n",test[9999].f1, test[9999].f2, test[9999].f3);

    //then show table of time that program use to sort in each solution and each round
    printf("\n\n-     Sort     -   Random data   -   Insert data   -   Descending   -\n");
    printf("-   Selection  -      %.3lf s    -     %.3lf s     -     %.3lf s    -\n",time1, time5, time51);
    printf("-   Insertion  -      %.3lf s    -     %.3lf s     -     %.3lf s    -\n",time2, time6, time61);
    printf("-   Bubble     -      %.3lf s    -     %.3lf s     -     %.3lf s    -\n",time3, time7, time71);
    printf("-   Shell      -      %.3lf s    -     %.3lf s     -     %.3lf s    -\n",time4, time8, time81);

    printf("\nEnd Program\n");
    printf("\nProgram writen by Onwipa Kujaroenpaisan 62070501064"); //show my name at the last line :)
    return 0;
}
```

อธิบายโค้ดคร่าวๆ คือการเปิดและอ่านไฟล์ .csv แล้วแบ่งข้อมูลที่ได้รับออกเป็น 3 ฟิลด์ และนำฟิลด์ที่1 (ซึ่งเป็นตัวเลข) จำนวน 10,000 ตัว มาเรียงด้วยจากน้อยไปมากวิธีการทั้ง 4 แบบ ได้แก่ Selection sort, Insertion sort, Bubble sort และ Shell sort โดยจับเวลาที่ใช้ในการเรียงแต่ละครั้งไว้ด้วย จากนั้นใช้ฟิลด์ตัวเลขต้นฉบับ (ที่ยังไม่ได้เรียง) จำนวน 10,001 ตัว (เพิ่มจากเดิม 1 ตัว) มาเรียงซ้ำอีกครั้ง หลังจากเรียงอีกครั้งแล้ว ก็นำ 10,001 ตัวที่เป็นต้นฉบับ มาเรียงอีกครั้ง แต่ครั้งสุดท้ายจะเรียงจากมากไปน้อย โดยจับเวลาทุกครั้ง และสร้างตารางแสดงและเปรียบเทียบเวลาที่ใช้ในการเรียงลำดับแต่ละครั้ง โดยทั้งหมดนี้จะจบในการรันครั้งเดียว

Test Case

```
-     Sort      -   Random data   -   Insert data   -   Descending   -
-   Selection   -      0.173 s    -     0.161 s     -     0.177 s    -
-   Insertion   -      0.001 s    -     0.000 s     -     0.001 s    -
-   Bubble      -      1.304 s    -     1.270 s     -     1.273 s    -
-   Shell       -      0.013 s    -     0.012 s     -     0.012 s    -

End Program

Program writen by Onwipa Kujaroenpaisan 62070501064
```

ตารางด้านบนนี้คือ เวลาที่ใช้ในการเรียงแต่ละครั้ง โดยจะเห็นได้ว่า Insertion sort ใช้เวลาน้อยที่สุด และ Bubble sort ใช้เวลามากที่สุด จึงสรุปได้ว่า Insertion sort เป็นวิธีที่ดีที่สุด

ปัญหาที่พบในการทำ ASSIGNMENT

- เวลาที่ใช้เรียงของวิธี insertion sort น้อยเกินไป //ไม่ทราบสาเหตุ

Self-Assessment : 4 ทำงานด้วยตนเอง แก้ไขปัญหาด้วยตนเองได้ แต่บางครั้งยังต้องขอคำแนะนำจากผู้อื่น