ASSIGNMENT 3 - 62070501064 ONWIPA KUJAROENPAISAN –

Source Code

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

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

void readfile(Struct data[], int *count){ //declare function to read file .csv
    FILE *fp;
    Struct x;
    if((fp = fopen("test.csv","r")) != NULL){ //if file was found
        while(fscanf(fp, "%llu, %[^,], %s", &x.f1, x.f2, x.f3) == 3){ //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(Struct *a, Struct *b){ //function to swap data
    Struct c;
    c = *a;
    *a = *b;
    *b = c;
}

int cmp_int(const void *x, const void *y){ //function to compare integer
    double a, b;
    a = (*(Struct *)x).f1;
    b = ((Struct *)y)->f1;

    //check different between 2 int
    if((a - b) > 0){
        return 1;
    }else if((a - b) < 0){
        return -1;
    }else{
        return 0;
    }
}

int cmp_str(const void *x, const void *y){ //function to compare string
    char a[100], b[100];
    int ans;
    strcpy(a,((Struct *)x)->f3);
    strcpy(b,((Struct *)y)->f3);
    ans = strcmp(a,b);
    return ans;
}

//function to use qsort to compare integer in field 1
//and count the time
double qsortInt(Struct data[], Struct test[], int count){
    clock_t start, end;
    int i, j;
    double min_t = 10000, time; //give min_t to be the minimum time
    printf("\n");
    for(i = 0; i < 3; i++){ //loop to compare for 3 times
        for(j = 0; j < count; j++){
            //copy data in array data to array test
            //for data in array data will not change
            test[j] = data[j];
        }
        printf("qsort for integer round %d", i+1); //show round of sorting
        start = clock();
        qsort(test, count, sizeof(Struct), cmp_int); //send data to qsort
        end = clock();
```

```
            time = ((double)(end - start)) / CLOCKS_PER_SEC;
            printf("   // Time --> %lf s\n", time); //show the time that used in that round
            if(time < min_t){ //if time that used is less than minimum time
                min_t = time; //then give minimum time be that time
            }
        }
        printf("\nBEST TIME is %lf s\n", min_t); //show the smallest time be the best
        return min_t; //return the smallest time to main
    }

    //function to use qsort to compare string in field 3
    //and count the time
    double qsortStr(Struct data[], Struct test[], int count){
        clock_t start, end;
        int i, j;
        double min_t = 10000, time; //give min_t to be the minimum time
        printf("\n");
        for(i = 0; i < 3; i++){ //loop to compare for 3 times
            for(j = 0; j < count; j++){
                //copy data in array data to array test
                //for data in array data will not change
                test[j] = data[j];
            }
            printf("qsort for strings round %d", i+1); //show round of sorting
            start = clock();
            qsort(test, count, sizeof(Struct), cmp_str); //send data to qsort
            end = clock();
            time = ((double)(end - start)) / CLOCKS_PER_SEC;
            printf("   // Time --> %lf s\n", time); //show the time that used in that round
            if(time < min_t){ //if time that used is less than minimum time
                min_t = time; //then give minimum time be that time
            }
        }
        printf("\nBEST TIME is %lf s\n", min_t); //show the smallest time be the best
        return min_t; //return the smallest time to main
    }
```

```
    //function to use quick sort by manual to compare integer in field 1
    //and count the time
    void quickSort_int(Struct data[], int first, int last){
        int i = first, j = last;
        if(first < last){
            do{
                //find the next data that smaller then swap
                while(data[i].f1 <= data[j].f1 && (i < j)){
                    j--;
                }
                if(data[i].f1 > data[j].f1){
                    swap(&data[i], &data[j]);
                    i++;
                }
                //find the data before that bigger then swap
                while(data[i].f1 <= data[j].f1 && (i < j)){
                    i++;
                }
                if(data[i].f1 > data[j].f1){
                    swap(&data[i], &data[j]);
                    j--;
                }
            }while(i < j); //loop until i = j

            //check the correct of sorting
            if(first < (j-1)){
                quickSort_int(data, first, (j-1));
            }
            if((i + 1)< last){
                quickSort_int(data, (i + 1), last);
            }
        }
    }
```

```c
//function to use quick sort by manual to compare string in field 3
//and count the time
void quickSort_str(Struct data[], int first, int last){
    int i = first, j = last;
    if(first < last){
        do{
            //find the next data that smaller then swap
            while(strcmp(data[i].f3, data[j].f3) <= 0 && (i < j)){
                j--;
            }
            if(strcmp(data[i].f3, data[j].f3) > 0){
                swap(&data[i], & data[j]);
                i++;
            }
            //find the data before that bigger then swap
            while(strcmp(data[i].f3, data[j].f3) <= 0 && (i < j)){
                i++;
            }
            if(strcmp(data[i].f3, data[j].f3) > 0){
                swap(&data[i], & data[j]);
                j--;
            }
        }while( i < j); //loop until i = j

        //check the correct of sorting
        if(first < (j-1)){
            quickSort_str(data, first, (j-1));
        }
        if((i+1) < last){
            quickSort_str(data, (i+1), last);
        }
    }
}
```

```c
//function to count the time of quick sort by manual that compare integer in filed 1
double time_quickint(Struct data[], Struct test[], int count){
    clock_t start, end;
    int i, j;
    double min_t = 10000, time; //give min_t to be the minimum time
    for(i = 0; i < 3; i++){ //loop to compare for 3 times
        for(j = 0; j < count; j++){
            //copy data in array data to array test
            //for data in array data will not change
            test[j] = data[j];
        }
        printf("quick sort for integer round %d", i+1); //show round of sorting
        start = clock();
        quickSort_int(test, 0, count-1);//send data to quick sort
        end = clock();
        time = ((double)(end - start)) / CLOCKS_PER_SEC;
        printf("   // Time --> %lf s\n", time); //show the time that used in that round
        if(time < min_t){ //if time that used is less than minimum time
            min_t = time; //then give minimum time be that time
        }
    }
    printf("\nBEST TIME is %lf s\n", min_t); //show the smallest time be the best
    return min_t; //return the smallest time to main
}
```

```c
//function to count the time of quick sort by manual that compare string in filed 3
double time_quickstr(Struct data[], Struct test[], int count){
    clock_t start, end;
    int i, j;
    double min_t = 10000, time; //give min_t to be the minimum time
    for(i = 0; i < 3; i++){ //loop to compare for 3 times
        for(j = 0; j < count; j++){
            //copy data in array data to array test
            //for data in array data will not change
            test[j] = data[j];
        }
        printf("quick sort for string round %d", i+1); //show round of sorting
        start = clock();
        quickSort_str(test, 0, count-1); //send data to quick sort
        end = clock();
        time = ((double)(end - start)) / CLOCKS_PER_SEC;
        printf("   // Time --> %lf s\n", time);  //show the time that used in that round
        if(time < min_t){ //if time that used is less than minimum time
            min_t = time; //then give minimum time be that time
        }
    }
    printf("\nBEST TIME is %lf s\n", min_t); //show the smallest time be the best
    return min_t; //return the smallest time to main
}
```

```c
int main(){
    int count = 0;
    Struct *data, *test;
    double time1, time2, time3, time4;

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

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

    //show data before sorting
    printf("\n=====Show data before sorting=====\n");
    printf("\ndata[0] :");
    printf("\n%llu - %s - %s\n",data[0].f1, data[0].f2, data[0].f3);
    printf("\ndata[49999] :");
    printf("\n%llu - %s - %s\n",data[49999].f1, data[49999].f2, data[49999].f3);
    printf("\ndata[99999] :");
    printf("\n%llu - %s - %s\n",data[99999].f1, data[99999].f2, data[99999].f3);
    printf("\n================================");

    //sorting by qsort and quick sort by manual and get the time
    printf("\n\nSorting..\n\n");
    time4 = time_quickstr(data, test, count);
    time2 = time_quickint(data, test, count);
    time3 = qsortStr(data, test, count);
    time1 = qsortInt(data, test, count);
```

```c
    //show data after sorting to check the correct of sorting
    printf("\n=====Show data after sorting=====\n");
    printf("\ndata[0] :");
    printf("\n%llu - %s - %s\n",test[0].f1, test[0].f2, test[0].f3);
    printf("\ndata[49999] :");
    printf("\n%llu - %s - %s\n",test[49999].f1, test[49999].f2, test[49999].f3);
    printf("\ndata[99999] :");
    printf("\n%llu - %s - %s\n",test[99999].f1, test[99999].f2, test[99999].f3);
    printf("\n================================");

    printf("\n\n *****SHOWTIME***** \n\n");
    printf(" Int Sorting");
    //compare the time that each solution used
    if(time1 < time2){
        printf("\n -qsort function- is better than -quick sort by manual-\n");
    }else{
        printf("\n -quick sort by manual- is better than -qsort function-\n");
    }

    printf("\n String Sorting");
    //compare the time that each solution used
    if(time3 < time4){
        printf("\n -qsort function- is better than -quick sort by manual-\n");
    }else{
        printf("\n -quick sort by manual- is better than -qsort function-\n");
    }

    printf("\n\nEND PROGRAM");
    printf("\nProgram writen by ONWIPA KUJAROENPAISAN 62070501064");
}
```

คำอธิบายโปรแกรม

เมื่อเริ่มต้นโปรแกรมจะทำการอ่านไฟล์ test.csv โดยเก็บข้อมูลที่อ่านได้ในอาร์เรย์ data โดยจะแบ่งเป็น 3 ชุดข้อมูล ชุดแรกคือตัวเลข ชุดที่สองและสามเป็นสตริง หลังจากรับข้อมูลแล้วก็ทำการนับจำนวนของข้อมูลในไฟล์ด้วย (โดยที่ 1 บรรทัด คือข้อมูล 1 ตัว) เมื่ออ่านไฟล์เสร็จแล้วก็แสดงข้อมูลตัวแรก(0) ตัวที่ 49,999 และตัวสุดท้าย(99999) ของข้อมูล จากนั้นเมื่อทำการเรียง ในการเรียงทุกครั้ง จะมีการ copy ค่าจาก data ไปใส่ในอีกอาร์เรย์ที่ชื่อว่า test เพื่อที่ลำดับของข้อมูลใน data จะไม่เปลี่ยนแปลง

การเรียงข้อมูลจะแบ่งเป็น 4 แบบ

การสร้างฟังก์ชัน quick sort เพื่อเรียงลำดับของข้อมูลตัวเลขจำนวนเต็ม(Int)

การสร้างฟังก์ชัน quick sort เพื่อเรียงลำดับของข้อมูลสตริงชุดสุดท้าย

การสร้าง compare function เพื่อใช้คำสั่ง qsort เพื่อเรียงลำดับของข้อมูลตัวเลขจำนวนเต็ม

การสร้าง compare function เพื่อใช้คำสั่ง qsort เพื่อเรียงลำดับของข้อมูลสตริงชุดสุดท้าย

โดยที่ในการเรียงทุกแบบจะมีการจับเวลาที่ใช้ในการเรียง แสดงเวลาที่ใช้ และจะเรียงข้อมูลทั้งหมด แบบละ 3 ครั้ง เพื่อการเปรียบเทียบเวลาและหาเวลาที่น้อยที่สุดของแบบนั้นๆ

เมื่อเรียงเรียบร้อย และแสดงเวลาที่ใช้ในแต่ละรอบแล้ว โปรแกรมจะแสดงผลข้อมูลทั้ง 3 ตำแหน่งอีกครั้ง เพื่อเช็คความถูกต้องในการเรียง และมีการเปรียบเทียบเวลาว่า การเรียงสตริง qsort หรือ quick sort ใช้เวลาน้อยกว่า และการเรียงตัวเลขจำนวนเต็ม quick sort หรือ qsort ใช้เวลาน้อยกว่า จากนั้นจึงจบโปรแกรม

Test Case

```
Loading..
 -> 100000 records

=====Show data before sorting=====

data[0] :
111111111111 - AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA - ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

data[49999] :
725622549999 - EAHCDMRGCOHEVVCTJGFSDXZMZUSQTZFZIEXOQCCWWUVMYMKKKKKK - EQEVXBWZOYLTJAWDEILNMTRPCHXHGOLWCEEFRBQJAXKSAIKKKKKK

data[99999] :
999999900000 - ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ - AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

==================================

Sorting..

quick sort for string round 1   // Time --> 0.073000 s
quick sort for string round 2   // Time --> 0.064000 s
quick sort for string round 3   // Time --> 0.062000 s

BEST TIME is 0.062000 s
quick sort for integer round 1   // Time --> 0.048000 s
quick sort for integer round 2   // Time --> 0.072000 s
quick sort for integer round 3   // Time --> 0.055000 s

BEST TIME is 0.048000 s

qsort for strings round 1   // Time --> 0.133000 s
qsort for strings round 2   // Time --> 0.139000 s
qsort for strings round 3   // Time --> 0.142000 s

BEST TIME is 0.133000 s

qsort for integer round 1   // Time --> 0.080000 s
qsort for integer round 2   // Time --> 0.085000 s
qsort for integer round 3   // Time --> 0.085000 s

BEST TIME is 0.080000 s
```

```
=====Show data after sorting=====

data[0] :
100000000000 - ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ - ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ

data[49999] :
556526160000 - RINZGHRJWITFMUHTNNNXIFMVNGGOYIDLBNIUIJHEVQEBNZAXAAAA - YCDKDIGPVNIDHDKORMPQEGKAIRTVANUZNPILJDHZMIDGMPDFZZZZ

data[99999] :
999999999999 - ZYXWVUTSRQPONMLKJIHGFEDCBAZYXWVUTSRQPONMLKJIHGFEDCBA - ZYXWVUTSRQPONMLKJIHGFEDCBAZYXWVUTSRQPONMLKJIHGFEDCBA

==================================

 *****SHOWTIME*****

 Int Sorting
 -quick sort by manual- is better than -qsort function-

 String Sorting
 -quick sort by manual- is better than -qsort function-


END PROGRAM
Program writen by ONWIPA KUJAROENPAISAN 62070501064
```

จาก test case จะเห็นได้ว่า quick sort ใช้เวลาน้อยกว่า qsort ทั้งการเรียงตัวเลขจำนวนเต็มและสตริง

ปัญหาที่พบในการทำ ASSIGNMENT

เขียนฟังก์ชัน compare ผิด

ไปเรียงข้อมูลชุดที่สอง จากที่ต้องเรียงชุดที่สาม

สับสนตัวแปรที่ใช้ชื่อใกล้กัน หรือต่างกันแค่ตัวเลข

Self-Assessment : 3 เข้าใจแต่มีปัญหาบางช่วงยังต้องขอความช่วยเหลือ