

King Saud University
College of Computer and Information Sciences
Computer Science Department
CSC 215

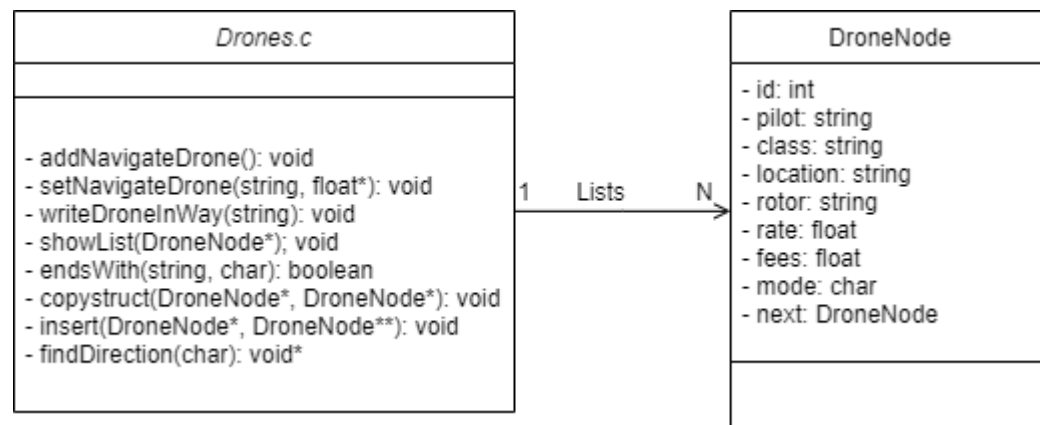
Drone navigation system

Student Name	Fatimah Alhumaidhi	Email	441200921@student.ksu.edu.sa
--------------	--------------------	-------	------------------------------

1- Introduction:

The main objective to the drone navigation system is to save data about the drones from a file to a dynamic linked list, applying all skills we learned through the the procedural programming with c course, from dealing with pointers and memory allocations to using structures and c libraries.

2- Problem Definition:



3 functions were added:

`endsWith(string, char)`: compares the last character in a string with character d for direction, and makes sure that str is not null

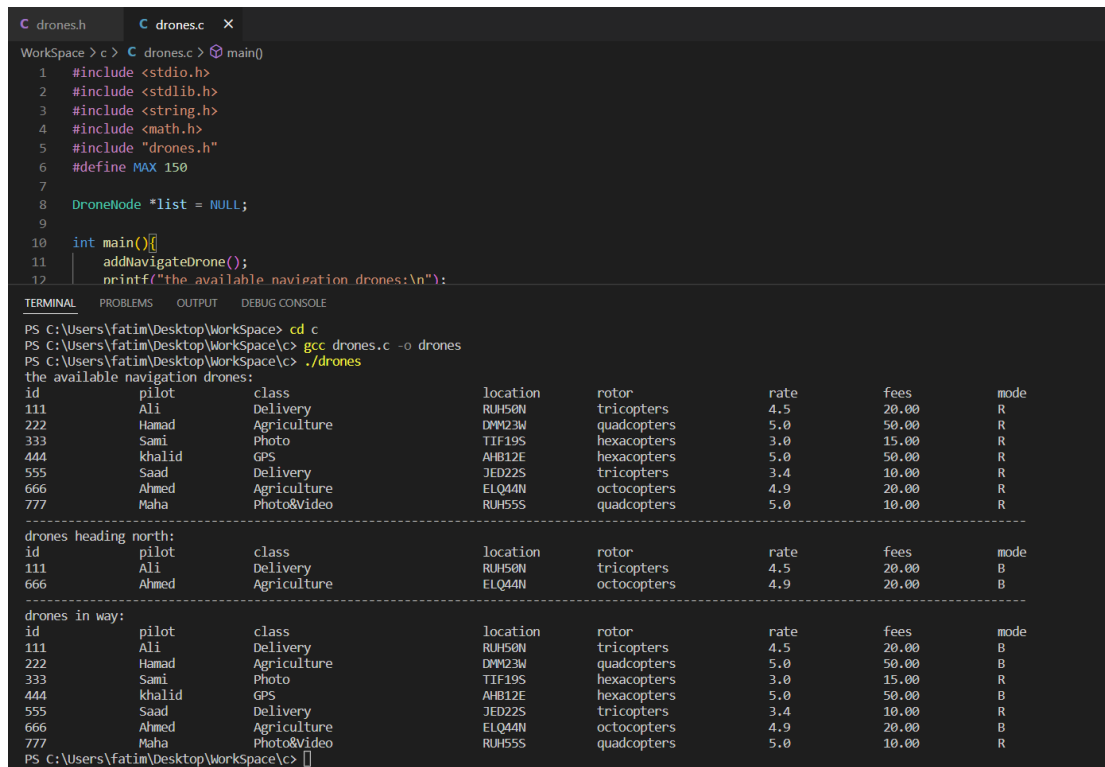
`copystruct(DroneNode*, DroneNode*)`: copies drone information from a source node to a destination node.

`insert(DroneNode*, DroneNode**)`: inserts a node into sorted list in it's appropriate place depending on drone id.

All three functions were made to reduce the `findDirection` function, and make the code more readable.

3- Test and Run:

I tested my code by printing each function's output on the console, first list tested addNavigateDrone and showList functions, the second tested setNavigateDrone and findDrection, all outputs were as expected, here are screenshots of the results:



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <math.h>
5 #include "drones.h"
6 #define MAX 150
7
8 DroneNode *list = NULL;
9
10 int main()
11 {
12     addNavigateDrone();
13     printf("the available navigation drones:\n");
14 }
```

PS C:\Users\Fatim\Desktop\workspace> cd c

PS C:\Users\Fatim\Desktop\workspace> gcc drones.c -o drones

PS C:\Users\Fatim\Desktop\workspace> ./drones

the available navigation drones:

id	pilot	class	location	rotor	rate	fees	mode
111	Ali	Delivery	RUH50N	tricopters	4.5	20.00	R
222	Hamad	Agriculture	DMM23W	quadcopters	5.0	50.00	R
333	Sami	Photo	TIF19S	hexacopters	3.0	15.00	R
444	khalid	GPS	AHB12E	hexacopters	5.0	50.00	R
555	Saad	Delivery	JED22S	tricopters	3.4	10.00	R
666	Ahmed	Agriculture	ELQ44N	octocopters	4.9	20.00	R
777	Maha	Photo&Video	RUH55S	quadcopters	5.0	10.00	R

drones heading north:

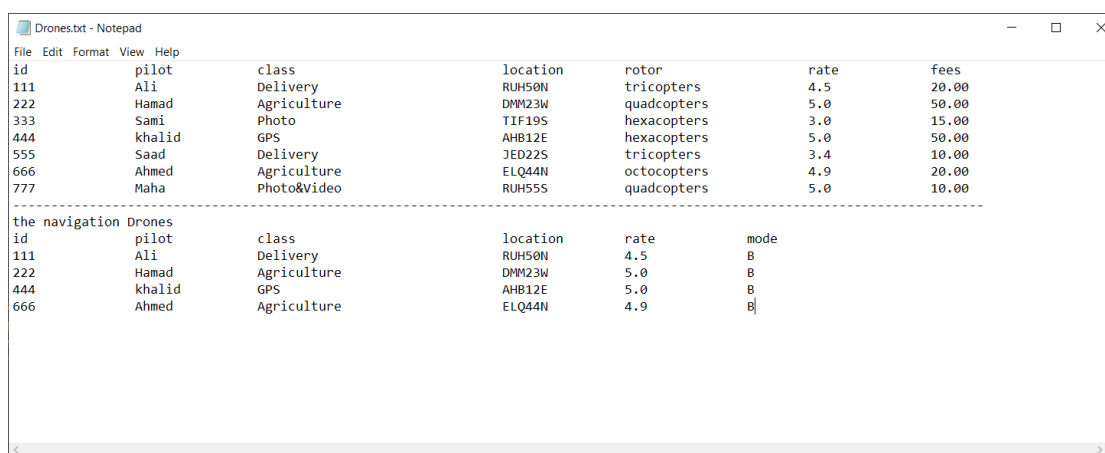
id	pilot	class	location	rotor	rate	fees	mode
111	Ali	Delivery	RUH50N	tricopters	4.5	20.00	B
666	Ahmed	Agriculture	ELQ44N	octocopters	4.9	20.00	B

drones in way:

id	pilot	class	location	rotor	rate	fees	mode
111	Ali	Delivery	RUH50N	tricopters	4.5	20.00	B
222	Hamad	Agriculture	DMM23W	quadcopters	5.0	50.00	B
333	Sami	Photo	TIF19S	hexacopters	3.0	15.00	R
444	khalid	GPS	AHB12E	hexacopters	5.0	50.00	B
555	Saad	Delivery	JED22S	tricopters	3.4	10.00	R
666	Ahmed	Agriculture	ELQ44N	octocopters	4.9	20.00	B
777	Maha	Photo&Video	RUH55S	quadcopters	5.0	10.00	R

PS C:\Users\Fatim\Desktop\workspace>

And by checking the output in the text file I tested writeDroneInWay, which prints the drones with mode B:



```
File Edit Format View Help
id pilot class location rotor rate fees
111 Ali Delivery RUH50N tricopters 4.5 20.00
222 Hamad Agriculture DMM23W quadcopters 5.0 50.00
333 Sami Photo TIF19S hexacopters 3.0 15.00
444 khalid GPS AHB12E hexacopters 5.0 50.00
555 Saad Delivery JED22S tricopters 3.4 10.00
666 Ahmed Agriculture ELQ44N octocopters 4.9 20.00
777 Maha Photo&Video RUH55S quadcopters 5.0 10.00

the navigation Drones
id pilot class location rate mode
111 Ali Delivery RUH50N 4.5 B
222 Hamad Agriculture DMM23W 5.0 B
444 khalid GPS AHB12E 5.0 B
666 Ahmed Agriculture ELQ44N 4.9 B
```

One problem I had was comparing equality of two floating point numbers, taking into account rounding error -showed in the picture-, which I solved easily by using the fabs() function found in the math.h library, which returns the absolute value of a number:

```
rate){
    (double)(0.01000000000000000021)
    && fabs(*rate - current->rate)<=0.01){
```

To make sure that the sorting and findDirection function work correctly, I reordered the information in Drones.txt file for drones heading south and tested the output, which also printed as expected:

Drones.txt - Notepad							
id	pilot	class	location	rotor	rate	fees	
111	Ali	Delivery	RUH50N	tricopters	4.5	20.00	
222	Hamad	Agriculture	DMM23W	quadcopters	5.0	50.00	
777	Maha	Photo&Video	RUH55S	quadcopters	5.0	10.00	
444	khalid	GPS	AHB12E	hexacopters	5.0	50.00	
555	Saad	Delivery	JED22S	tricopters	3.4	10.00	
666	Ahmed	Agriculture	ELQ44N	octocopters	4.9	20.00	
333	Sami	Photo	TIF19S	hexacopters	3.0	15.00	

```

C:\drones> g++ drones.c
C:\drones> ./drones
the available navigation drones:
id      pilot      class      location    rotor      rate      fees      mode
111     Ali        Delivery   RUH50N      triicopters 4.5       20.00     R
222     Hamad      Agriculture DMM23W      quadcopters 5.0       50.00     R
777     Maha       Photo&Video RUH55S      quadcopters 5.0       10.00     R
444     khalid     GPS        AHB12E      hexacopters 5.0       50.00     R
555     Saad       Delivery   JED22S      triicopters 3.4       10.00     R
666     Ahmed     Agriculture ELQ44N      octocopters 4.9       20.00     R
333     Sami       Photo      TIF19S      hexacopters 3.0       15.00     R

drones heading south:
id      pilot      class      location    rotor      rate      fees      mode
333     Sami       Photo      TIF19S      hexacopters 3.0       15.00     R
555     Saad       Delivery   JED22S      triicopters 3.4       10.00     R
777     Maha       Photo&Video RUH55S      quadcopters 5.0       10.00     R

drones in way:
id      pilot      class      location    rotor      rate      fees      mode
111     Ali        Delivery   RUH50N      triicopters 4.5       20.00     B
222     Hamad      Agriculture DMM23W      quadcopters 5.0       50.00     B
777     Maha       Photo&Video RUH55S      quadcopters 5.0       10.00     R
444     khalid     GPS        AHB12E      hexacopters 5.0       50.00     B
555     Saad       Delivery   JED22S      triicopters 3.4       10.00     R
666     Ahmed     Agriculture ELQ44N      octocopters 4.9       20.00     B
333     Sami       Photo      TIF19S      hexacopters 3.0       15.00     R
```

4- Implementation:

Header file:

```
typedef struct Drone{
    int id;
    char pilot[20];
    char class[20];
    char location[8];
    char rotor[20];
    float rate;
    float fees;
    char mode;
    struct Drone* next;
}DroneNode; //project description specified not creating structures
other than struct Drone

void addNavigateDrone();
void setNavigateDrone(char*, float*);
void writeDroneInWay(char*);
void showList(DroneNode*);
void* findDirection(char);
```

c file:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "drones.h"
#define MAX 150

DroneNode *list = NULL;

int main(){
    addNavigateDrone();
    printf("the available navigation drones:\n");
    showList(list);

    float f = 4.5;
    setNavigateDrone("Delivery", &f);
    f = 5.0;
    setNavigateDrone("Agriculture", &f);
```

```

    setNavigateDrone("GPS", &f);
    f = 4.9;
    setNavigateDrone("Agriculture", &f);

    writeDroneInWay("Drones.txt");
    void* headingNorth = findDirection('N');
    puts("-----
-----");
    printf("drones heading north:\n");
    showList(headingNorth);

    puts("-----
-----");
    printf("drones in way:\n");
    showList(list);
    return 0;
}

void addNavigateDrone(){
    DroneNode *current, *tail;
    FILE* file;
    if((file = fopen("Drones.txt", "r")) == NULL){
        exit(1);
    }
    char firstline[MAX];
    fgets(firstline, MAX, file);

    while(!feof(file)){
        if((current = (DroneNode*)malloc(sizeof(DroneNode))) == NULL){
            exit(1);
        }
        fscanf(file, "%d%s%s%s%f%f", &current->id, current->pilot,
current->class, current->location, current->rotor, &current->rate,
&current->fees);
        current->mode = 'R';
        if(!list){
            tail = list = current;
        }
        else{
            tail->next = current;
            tail = tail->next;
        }
    }
    tail->next = NULL;
    fclose(file);
}

```

```

void setNavigateDrone(char* class, float* rate){
    DroneNode* current = list;
    while(current != NULL){
        if(!strcmp(current->class, class) && fabs(*rate - current-
>rate)<=0.01){
            current->mode = 'B';
        }
        current = current->next;
    }
}

void writeDroneInWay(char* fileName){
    FILE* file;
    if((file = fopen(fileName, "a")) == NULL){
        exit(1);
    }

    fputs("\n-----",
file);
    fputs("\nthe navigation Drones\n", file);
    fprintf(file, "%-16s%-16s%-32s%-16s%-16s\n", "id", "pilot",
"class", "location", "rate", "mode");
    DroneNode* current = list;
    while(current){
        if(current->mode == 'B'){
            fprintf(file, "%-16d%-16s%-32s%-16s%-16.1f%c\n", current-
>id, current->pilot, current->class, current->location, current->rate,
current->mode);
        }
        current = current->next;
    }
    fclose(file);
}

void showList(DroneNode* list){
    if(list == NULL){
        printf("empty list.");
        exit(1);
    }
    printf("%-16s%-16s%-32s%-16s%-24s%-16s%-16s\n", "id", "pilot",
"class", "location", "rotor", "rate", "fees", "mode");
    DroneNode* current = list;
    while(current != NULL){

```

```

        printf("%-16d%-16s%-32s%-16s%-24s%-16.1f%-16.2f%c\n", current-
>id, current->pilot, current->class, current->location, current->rotor,
current->rate, current->fees, current->mode);
        current = current->next;
    }
}

int endsWith(const char* str, char d){
    return (str && str[strlen(str) - 1] == d);
} //compares the last character in a string with character d for
direction, and makes sure that str is not null

void copystruct(DroneNode* source, DroneNode* distenation){
    distenation->id = source->id;
    strcpy(distenation->pilot, source->pilot);
    strcpy(distenation->class, source->class);
    strcpy(distenation->location, source->location);
    strcpy(distenation->rotor, source->rotor);
    distenation->fees = source->fees;
    distenation->mode = source->mode;
    distenation->rate = source->rate;
    distenation->next = NULL;
} //copies drone information from node source to node distenation

void insert(DroneNode* newnode, DroneNode** sortedList) {

    if (*sortedList == NULL || (*sortedList)->id > newnode->id) {
        newnode->next = *sortedList;
        *sortedList = newnode;
    }
    else {
        DroneNode* current = *sortedList;

        while (current->next != NULL && current->next->id <= newnode-
>id) {
            current = current->next;
        }
        newnode->next = current->next;
        current->next = newnode;
    }
} //inserts a node into sorted list in it's appropriate place depending
on drone id

void* findDirection(char d){

    DroneNode *current = list, *newNode = NULL, *sortedList = NULL;
    while (current) {
        if(endsWith(current->location, d)){

```



```
        if((newNode = (DroneNode*)malloc(sizeof(DroneNode))) ==  
NULL) {  
            exit(1);  
        }  
        copystruct(current, newNode);  
        insert(newNode, &sortedList);  
    }  
    current = current->next;  
}  
  
return sortedList;  
}
```