Bilregister

Fabian Odén

20 mars 2023

DVGA01 Programeringsteknik
Datavetenskap
Fakulteten för hälsa-, teknik- och naturvetenskap

Innehåll

1 Inledning

I laborationen skulle ett program skapas. Programmet skulle spara, ändra och visa information om fordon. Användare skulle med hjälp av programmet spara informationen i en fil.

2 Översikt

Programmet ska ge användaren 5 alternativ. Lägga till ett fordon, ta bort ett fordon, visa ett fordon, visa alla fordon eller sortera fordonen. Detta ska göras utan att programmet krashar. Programmet förväntas köras på ett linux system.

3 Detaljerad beskrivning

- Entry funktionen(main funktionen) deklarer först två variablar buffer(char array av längden MENU_OPTION_LENGTH) och option(int). En while loop startas sedan som körs tills den breakas eller funktionen returnas. Den är till för att hålla programmet vid liv. Användaren blir sendan promptad genom en printf call att skriva valet av option. Vad andvändaren skriver in i terminalen tas sedan in till buffer med fgets där malängden är MENU_OPTION _LENGTH-1(för end of line karaktären). atoi funktionen konverterar sedan värdet i buffer till en integer. Denna integern jämförs i en serie av if statements(i retrospect kunde switch statement använts). Varje if statement jämför mot MENU_OPTION_X för att se vad användaren valde. Option 1-5 kallar på funktionerna nedan, option 6 exitar programmet och i alla andra fall promptas användaren om att optionen inte fanns och while loopen börjar om.

- Option 1 kallar på funktionen add_vehicle. Den definerar pointern vehicles och allokerar minne till storleken av MAX_CARS stycken Vehicle structs. Med memset sätter alla minnes addresser allokerade till vehicle till -1. Den använder funktionen read_file för att får antalet vehicles sedan jämförs värdet med MAX_CARS. Om MAX_CARS är uppnåt skriver programmet ut det till användaren och returnar från funktionen. Om MAX_CARS inte är uppnåt kommer new_vehicle defineras och allokeras minne till storleken av ett Vehicle struct. Användaren blir sedan promptad till att skriva in properties på vehiclet. Den allokerar en char array med längden NAME_LENGTH, promptar användaren vilken attribut, tar värdet användaren skriver in med fgets, tar bort newline charactären med strtok och kopierar värdet till new_vehicle variablen. Detta upprepas 3 gånger för namn, brand

och registration number. Processen upprepas igen men funktionen int_input kallas på värdet.

Variabel person deklareras med typen Person och metoden ovan använda för att ta input av namn och ålder. Inputten borde has gjorts i sin egen funktion för att undivika onödig mycket code duplication.

I slutet av funktionen visar programmet datan hen skrev in. write_one _person kallas sedan med person som argument samt write_one_vehicle kallas med new_vehicle som argument.

- Option 2 kallar på remove_vehicles. Den definerar pointern vehicles och allokerar minne till storleken av MAX_CARS stycken Vehicle structs. Med memset sätter alla minnes addresser allokerade till vehicle till -1. Antalet fordon defineras och säts till retur värdet av read_file, funktionen kallas med argumentet vehicles. Användaren promptas till att ta bort 1- antalet fordon. Med hjälp av int_input och atoi hanteras buffer och vehicle_num är definerat med användarens val av fordon att ta bort. Programmet jämför ifall användarens val är inom 0 till antalet fordon, ifall den inte är det promptas användaren om att fordonen inte finns. Annars kallas clear_file och en for loop startas för mängden av fordon-1. Ifall indexet har gått större eller likamed användarens val kommer den skriva vehicle i+1 med write_one_vehicle alltså den hoppas över användarens val. Annars kommer den skriva vehicle på index i med wriet _one_vehicle alltså tidigare array. Sen freeas minnet som allokerades till vid pointern vehicles

- Option 3 kallas sort_vehicles. Den definerar pointern vehicles och allokerar minne till storleken av MAX_CARS stycken Vehicle structs. Med memset sätter alla minnes addresser allokerade till vehicle till -1. Antalet fordon defineras och säts till retur värdet av read_file, funktionen kallas med argumentet vehicles. clear_file kallas. Variablen best_choice deklareras med typen int och variabeln taken defineras som en array av ints med längen av antal fordon. En while loop startas som körs tills den breakar eller sort_vehicles funktionen returnar. En for loop men längden av antal fordon körs. Med hjälp av strcmp jämför den vechile på plats index mot vehicle på plats best_choice samt ifall den är tagen genom att jämföra integern i taken vid index index. Ifall fordonet vid den nuvarnde indexen är före i alfabetisk ordning och den är inte tagen blir den den nya best_choice. Fordonet vid index best_choice skrivs sedan med write_one_vehicle och best_choice sätts till 1. En ny for loop går igenom taken arrayen för att jämföra ifall det finns en int som inte är 1. Isåfall säts best_choice till indexet. Efter for loopen jämförs best_choice med -1 för att se ifall det finns en vehicle som inte är

tagen och isåfall breakar från while loopen. Minnet som vehicles pekar på blir freeat.

- Option 4 kallas info_vehicle. Den definerar pointern vehicles och allokerar minne till storleken av MAX_CARS stycken Vehicle structs. Med memset sätter alla minnes addresser allokerade till vehicle till -1. Antalet fordon defineras och säts till retur värdet av read_file, funktionen kallas med argumentet vehicles. Användaren promptas till att välja ett fordon genom int_input och atoi. Programmet jämför ifall användarens val är inom 0 till antalet fordon, ifall den inte är det promptas användaren om att fordonen inte finns. Variablen person defineras till typen Person. Variablen age av typen integer sätt till retur värdet av get_person_age som kallas med argumentet owner_name. Informationen skrivs ut till användaren. Minnet som vehicle pekar på blir freeat.

- Option 4 kallas show_vehicle. Den definerar pointern vehicles och allokerar minne till storleken av MAX_CARS stycken Vehicle structs. Med memset sätter alla minnes addresser allokerade till vehicle till -1. En rad med columnnamn printas formaterat. En for loop går igenom vehicles och kontrollerar att vehicle vid indexet existerar och isåfall skriver ut till stdout(ternimalen). Minnet som vehicle pekar på blir freeat.

- file_exists försöker öppna filen med filvägen file_name och mode 'r'. Ifall filen inte finns skapar den en fil genom att öppna den med fopen och mode 'w' vilket skapar en fil ifall den inte redan finns. Filen stängs sedan med fclose.

- read_file kallas file_exists och filen öppnas i reading mode. Den kontrollerar att den lyckades öppna annars exitar den(detta fallet borde aldrig hända ifall file_exists fungerar som förväntat). Den startar en while look med jämförelsen fscanf och EOF. fscanf kallas med minnes adresserna till ett vehicle i en lista med vehicles. I while loopen ökar den i för att ta nästa vehicle. Den stängder sedan filen och returar antalet vehicles lästa.

- int_input läser från stdin med fgets och tar bort new line med strtok. En while loop startas med jämförelse mellan 1 och is_number, ifall is_numer retunerar 1 promptas användaren att skriva in ett nytt tal.

- is_number kör en for loop för varje index i input char arrayn. Varje char vid indexen är argument till idigit. Ifall isdigit returnerar 0 returnerar funktionen 0.

- write_one_person kallar file_exists och öppnar sedan filen 'persons.dat' med mode append. Den kontrollerar att den lyckades öppna annars exitar den(detta fallet borde aldrig hända ifall file_exists fungerar som förväntat). Den skriver till filen med fprintf. Filen stängs sedan med fclose.

- write_one_vehicle kallar file_exists och öppnar sedan filen 'vehicles.txt' med mode append. Den kontrollerar att den lyckades öppna annars exitar den(detta fallet borde aldrig hända ifall file_exists fungerar som förväntat). Den skriver till filen med fwrite. Filen stängs sedan med fclose.

- clear_file öppnar 'vehicles.txt' filen i write mode vilket rensar filen. Filen stängs sedan med fclose.

- get_person_age kallar file_exists och öppnar sedan filen 'persons.dat' med mode append. Den kontrollerar att den lyckades öppna annars exitar den(detta fallet borde aldrig hända ifall file_exists fungerar som förväntat). Deklarerar sedn variablerna age(int) och temp_person (Person). En while loop med fread som argument. Den fortsätter läsa rader tills fread kommer till slutet. I loopen jämförs namnet på temp_person och namnet som var parametern vid funktionens början. Filen stängs sedan med fclose och age returneras

4 Problem

Ett problem som uppkom var att den kunde spara och läsa filen endast första gången koden kördes. Detta löstes genom att deklarera storleken på arraysen i structet. Detta gjorde att programmet visste exakt hur stor plats att allokera till structen.

Programmet verkade tycka att det fanns saker i minne efter allokerat med malloc. Detta löstes genom att byta minnet med hjälp av memset. Jag kunde också använt calloc.

5 Sammanfattning

Det jag har lärt mig mest om är minnes allokering. Jag spenderade 10 timmar ungefär.

A Programkod

Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#include "fil.h"
#include "main.h"

int is_number(char *string)
{
    for (int i = 0; i < strlen(string) - 1; i++)
    {
        if (isdigit(string[i]) == 0)
        {
            return 0;
        }
    }
    return 1;
}



void int_input(char *buffer)
{
    fgets(buffer, NAME_LENGTH, stdin);
    strtok(buffer, "\n");
    while (is_number(buffer) != 1) // not working for
       letters
    {
        printf("Error: not a number. Enter a number:");
        fgets(buffer, NAME_LENGTH, stdin);
        strtok(buffer, "\n");
    }
}

void add_vehicle()
{

    struct Vehicle *vehicles = malloc(sizeof(struct Vehicle
       ) * MAX_CARS);
```

```
memset(vehicles, -1, sizeof(struct Vehicle) * MAX_CARS)
   ;
if (read_file(vehicles) == MAX_CARS)
{
    printf("\nError: Maximum allowed cars(10) already
       added.");
    return;
}

struct Vehicle new_vehicle;
printf("\nAdding a vehicle.");

char typeb[NAME_LENGTH];
printf("\nType: ");
fgets(typeb, NAME_LENGTH, stdin);
strtok(typeb, "\n");
strcpy(new_vehicle.type, typeb);

char brandb[NAME_LENGTH];
printf("Brand: ");
fgets(brandb, NAME_LENGTH, stdin);
strtok(brandb, "\n");
strcpy(new_vehicle.brand,brandb);

char regb[NAME_LENGTH];
printf("Registration number: ");
fgets(regb, NAME_LENGTH, stdin);
strtok(regb, "\n");
strcpy(new_vehicle.reg_num,regb);

char ab[NAME_LENGTH];
printf("Age: ");
int_input(ab);
new_vehicle.age = atoi(ab);

struct Person person;
char owner_nameb[NAME_LENGTH];
printf("Owner name: ");
fgets(owner_nameb, NAME_LENGTH, stdin);
strtok(owner_nameb, "\n");
strcpy(person.name, owner_nameb);
strcpy(new_vehicle.owner_name,owner_nameb);
```

```
    char owner_ageb[NAME_LENGTH];
    printf("Owner age: ");
    int_input(owner_ageb);
    person.age = atoi(owner_ageb);

    printf("\nNew vehicle   %s, %s, %s, %d %s", new_vehicle
       .type, new_vehicle.brand, new_vehicle.reg_num,
       new_vehicle.age, new_vehicle.owner_name);

    write_one_person(person);
    write_one_vehicle(new_vehicle);
}

void remove_vehicle()
{
    struct Vehicle *vehicles = malloc(sizeof(struct Vehicle
       ) * MAX_CARS);
    memset(vehicles, 0, sizeof(struct Vehicle) * MAX_CARS);
    int amount= read_file(vehicles);
    printf("\nRemove vehicle(1-%d): ", amount);
    char buffer[NAME_LENGTH];
    int_input(buffer);
    int vehicle_num = atoi(buffer);
    if (vehicle_num < 0 || vehicle_num > amount)
    {
        printf("Error: vehicle %d doesnt exist",
           vehicle_num);
    }
    else
    {

        int new_amount = amount - 1;
        clear_file();
        for (int i = 0; i < new_amount; i++)
        {
            if (vehicle_num - 1 <= i)
            {
                write_one_vehicle(vehicles[i + 1]);
            }
            else
            {
                write_one_vehicle(vehicles[i]);
            }
```

```
        }
    }
free(vehicles);
}

void sort_vehicles()
{
    struct Vehicle *vehicles = malloc(sizeof(struct Vehicle
       ) * MAX_CARS);
    memset(vehicles, 0, sizeof(struct Vehicle) * MAX_CARS);
    int amount =read_file(vehicles);
    clear_file();
    int best_choice = 0;
    int taken[amount];
    while (1)
    {
        for (int i = 0; i < amount; i++)
        {
            if (strcmp(vehicles[i].brand, vehicles[
               best_choice].brand) < 0 && taken[i] != 1)
            {
                best_choice = i;
            }
        }
        write_one_vehicle(vehicles[best_choice]);
        taken[best_choice] = 1;
        // vehicles[best_choice].brand = "NULL";
        best_choice = -1;
        for (int o = 0; o < amount; o++)
        {
            if (taken[o] != 1)
            {
                best_choice = o;
            }
        }
        if (best_choice == -1)
        {
            break;
        }
    }
    free(vehicles);
}
```

```
void info_vehicle()
{
    struct Vehicle *vehicles = malloc(sizeof(struct Vehicle
       ) * MAX_CARS);
    memset(vehicles, 0, sizeof(struct Vehicle) * MAX_CARS);
    int amount = read_file(vehicles);
    printf("\nInfo about vehicle(1-%d): ", amount);
    char buffer[NAME_LENGTH];
    int_input(buffer);
    int vehicle_num = atoi(buffer);
    if (vehicle_num < 0 || vehicle_num > amount)
    {
        printf("\nError: vehicle %d doesnt exist",
           vehicle_num);
    }
    else
    {
        struct Person person;
        int age = get_person_age(vehicles[vehicle_num - 1].
           owner_name);
        printf("Vehicle %d\nType:%s \nBrand:%s \nReg num:%s
            \nAge:%d \nOwner:%s \nOwner age:%d",
               vehicle_num, vehicles[vehicle_num - 1].type,
                   vehicles[vehicle_num - 1].brand,
                  vehicles[vehicle_num - 1].reg_num,
                  vehicles[vehicle_num - 1].age, vehicles[
                  vehicle_num - 1].owner_name, age);
    }
    free(vehicles);
}

void show_vehicles()
{
    struct Vehicle *vehicles = malloc(sizeof(struct Vehicle
       ) * MAX_CARS);
    memset(vehicles, 0, sizeof(struct Vehicle) * MAX_CARS);
    int amount = read_file(vehicles);
    printf(" # %15s %15s %15s %5s", "type", "brand", "reg",
        "age");
    for (int i = 0; i < amount; i++)
    {
        if (vehicles[i].type == NULL)
        {
```

```
            break;
        }
        printf("\n%2d %15s %15s %15s %3d", i + 1, vehicles[
           i].type, vehicles[i].brand, vehicles[i].reg_num,
            vehicles[i].age);
    }
    free(vehicles);
}

int main()
{
    char buffer[MENU_OPTION_LENGTH];
    int option;

    while (1)
    {
        // menu
        printf("\n\n1. Add vehicle\n2. Remove vehicle\n3.
           Show vehicles sorted by car brand\n4.
           Information about a vehicle\n5. Show all
           vehicles\n0. Exit\n# ");
        fgets(buffer, MENU_OPTION_LENGTH - 1, stdin);
        option = atoi(buffer);
        if (option == MENU_OPTION_1)
        {
            add_vehicle();
        }
        else if (option == MENU_OPTION_2)
        {
            remove_vehicle();
        }
        else if (option == MENU_OPTION_3)
        {
            sort_vehicles();
        }
        else if (option == MENU_OPTION_4)
        {
            info_vehicle();
        }
        else if (option == MENU_OPTION_5)
        {
            show_vehicles();
        }
```

```
        else if (option == MENU_OPTION_6)
        {
            exit(0);
        }
        else
        {
            printf("\nError: option doesnt exist");
        }
    }
}
```