TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristjan Henri Roots
213450IACB

Kodune ülesanne 1

Programmeerimine II
(IAX0584)

Juhendaja: Lebit Jürimägi
Magister

Tallinn 2021

Autorideklaratsioon

Kinnitan, et olen koostanud antud kodutöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kristjan Henri Roots

20.03.22

Sisukord

Jooniste loetelu

1 Ülesande püstitus

Käesoleva kodutöö ülesandeks oli koostada programm C keeles, mis lahendaks etteantud variant 8 ülesande. Ülesandes oli tegemist kasutaja poolt ette antud filtrile vastavalt koostada statistika erinevatel Eesti jahialadel kütitud liikide ja nende arvukuse kohta. Lähteandmeteks on kaks faili, jahiandmed kütitud liikide kohta ning fail andmetega jahialade kohta. Programmi eesmärgiks oli võtta kasutaja sisestatud filtrid aasta ning liigi nime kohta, koostada sellele vastav statistika igas maakonnas ning väljastada saadud andmed tabelis maakondade tähestikulises järjekorras (Joonis 1, lk 5).

Lisaparameetrina oli ülesande lahenduses vajalik koostada andmestruktuurid:

- Jahiandmete jaoks.
- Jahialade jaoks.
- Statistika jaoks.

Aasta = 2010
Liik = Capreolus_capreolus

Maakond	Täiskasvanud	Keskmine	Noor	Teadmata
Harju	146	0	34	163
Lääne-Viru	2	0	2	16

Joonis 1. Programmi väljundi näide testandmetega.

2 Andmemudeli ja lähtefailide kirjeldus

Ülesande lahenduse parameetritele vastavalt koostasin eraldi andmestruktuurid jahiandmetele, jahialadele ja statistikale, ning kõigile veel omakorda andmestruktuuri, et koostada dünaamilised andmestruktuuride massiivid, kus hoida ka nende elementide arvu (Joonis 2, lk 7). Lähtefailidena otsustasin kasutada CSV faile, kuna olin ülesandega alustanud enne vajalike TXT failide olemasolu. Teoreetiliselt oleks loodud lahendusega väga lihtne ka tekstifailide töötlemine ja nõuaks ainult mõnda muutust tänu strtok funktsiooni kasutusele, kuid ei pidanud seda hädavajalikuks.

Esimesena loodud jahiandmete andmestruktuuris ei ole ülesande lahenduseks vajalik hoida kõiki failis leiduvaid andmeid. Andmestruktuuride massiivis on vajalik vaid hoida jahiala koodi, looma vanuse koodi ning loomade arvukust. Ülejäänud faili andmeveergude: aasta, liigi ja põhjuse eest hoolitseb andmete lugemisel filter ning looma sugu on antud juhul ebaoluline. Filtrile vastavalt ei loeta mällu kogu andmestikku, vaid ainult kasutaja soovile vastavad read (Joonis 3, lk 7).

Jahialade fail on vajalik mällu lugeda terves pikkuses, kuid andmetes leiduvad ka ebavajalikud veerud jahiala nime ning pindalaga mida ignoreeritakse. Selle tõttu olen ka mälu dünaamilist kasutust realiseerinud erinevalt, kuna eeldatavalt ei muutu Eestis olevate jahialade arv, siis eraldatakse andmestruktuuride massiivile kindel ridade arv, kuhu kogu fail ära mahub. Jahiandmete lugemisel suurendatakse allutatud mälu suurust dünaamiliselt, kuna sõltuvalt filtrile erineb sisseloetavate ridade arv.

Viimane andmestruktuur on loodud kasutaja sisestatud filtrite, statistika ja ka failitüübile omase eraldaja hoidmiseks. Teoreetiliselt ei ole rakendusel piiranguid andmemahu piires, kuna mälu kasutatakse dünaamiliselt, kuid see on eeldusel, et Eestis olevate jahialade arvukus ei kasva.

```c
typedef struct jahiandmed{ // jahiandmete hoidmine
    char code[CODELEN]; // jahiala kood
    int age; // vanus ja kogus
    int count;
}jdata_t;

typedef struct jahiandmed_arr{ // massiiv structidest, koos pikkusega
    unsigned int len;
    jdata_t *arr;
}animals_t;

```

Joonis 2. Jahiandmete andmestruktuuri näidis.

```c
    if(stats->year == atoi(format[0]) && strcmp(stats->species, format[1]) == 0){
        if(stats->reason == 1){ // lugemist on kahel korral eraldi juhuks kui kasutaja soovib kõiki vanuseid
            if(match == size){ // struct array on vaja suuremaks teha
                size = size * 2;
                jdata_t *p = realloc(animals->arr, size * sizeof(jdata_t));
                if(p == NULL){
                    printf("Error reallocating memory\n");
                    exit(0);
                }
                animals->arr = p;
            }
            readHunt(animals, format, match);
            match++;
```

Joonis 3. Jahiandmete lugemine.

3 Lahenduse kirjeldus

Programm alustab tööd andmestruktuuri algväärtustamisega ning failis oleva eraldaja määramisega, valitud CSV faili puhul semikoolon. Järgmiseks küsitakse kasutaja käest filtrid: aasta, liigi nimi ning hukkumise põhjus. Põhjuse valik ei olnud küll programmi üks nõuetest, kuid kuna if lause lisatingimus erilist probleemi ei tekitanud otsustasin ka selle lisada. Antud lahenduse puhul peavad filtrid olema teada enne failide avamist, kuna mällu loetakse ainult filtrile vastavad andmed.

Failide avamine on lahendatud liigse keerukusega, kuid pakub kasutajale mitu võimalust. Nimelt proovib programm kõigepealt ise avada sama nimega failid, nagu nad olid Teams keskkonnas saadaval (“jahiandmed.csv”, “jahiala.csv”), ebaõnnestumise korral vaadatakse ega kasutaja ei ole käsurealt midagi ette andnud ning viimasena pöördutakse kasutaja poole. Antud meetod tekitas eriolukorra, sest kasutaja ei tea täpselt, mis järjekorras failid sisestada ega tahagi meelde jätta. Alamprogramm fileID, mis tagastab sisestatud faili tüübi, sai loodud olukorra lahendamiseks. Rakenduse jaoks ei ole enam failide sisestamise järjekord oluline ning kahe samasuguse faili sisestamine ei lähe samuti läbi. Failide kontroll töötab väga lihtsalt, kontrollitakse tabeli päist. Tehniliselt on võimalik sisestada vale päisega fail, kuid kasutaja on siis juba tõenäoliselt teadlikult pahatahtlik.

Andmete lugemine toimub samas alamprogrammis fileopen peale faili tüübi teada saamist, kui tegemist on jahialadega loetakse fail mällu täies pikkuses paari veerulise erandiga, kuid jahiandmetest loetakse mällu vaid filtrile vastavad andmed. Tegemist ei ole elegantse lahendusega, kuna kõikide põhjuste soovimise puhuks on sisselugemise funktisoon kopeeritud uuesti. Andmete salvestamine stuktuuridesse on viidud eraldi alamfunktsioonidesse readHunt ja readArea lihtsamaks modifitseerimiseks, kuna testimise ajal oli pikast lugemise funktsioonist õiget kohta keeruline leida.

Peale failide korrektset lugemist asub programm statistikat koostama alamfunktsiooniga killCount. Kõigepealt eraldatakse mälu statistika massiivile kõikide Eesti maakondade jaoks ning algväärtustatakse. Statistika liitmine toimib jahiandmete alusel, käiakse läbi terve massiiv ning otsitakse jahialade andmete seast jahiala koodile vastav maakond ning loomade arvukus liidetakse statistikas maakonnale vastavale kohale. Lisafunktsioon findIndex kontrollib, kas ja kus maakond statistikas on, lisades selle vajadusel statistikasse juurde. Eri vanuste arvukuse hoidmiseks on programmis üks neljakohaline massiiv, kus igal vanusel on oma element kuhu arvukust pidevalt liidetakse.

Programmi viimaseks sammuks on statistika väljastamine tabelina, kuid eelnevalt on vaja statistika sorteerida tähestikulises järjekorras maakonna järgi, mis toimub kasutades qsort funktsiooni namesort võrdlejaga. Enne väljastamist vabastatakse ka jahialade ning jahiandmete andmestruktuuride massiivide mälu, tabeli väljastamiseks on vajalik vaid loodud statistika. Statistika väljastatakse alamfunktsioonis countyPrint, maakonnad on paigutatud tabeli lõppu ühtluse põhjustel, kuna täpitähtedega maakondi ei saa ette antud printimispikkusega korrektselt väljastada. Peale tabeli väljastamist vabastatakse statistikale allutatud mälu ning väljutakse programmist.

4 Eriolukorrad

Programmi loomisel üritasin tähelepanu pöörata võimalikele eriolukordadele, mis tekkida võivad. Esimene neist oli lahendada juba mainitud olukord failide lugemisel. Loodud kontrollfunktsioon fileID leiab päise järgi failitüübi ning järgmisel väljakutsumisel mäletab eelmise faili tüüpi staatiliste muutujate abil (Joonis 4, lk 10). Lubatud ei ole sisestada ka faili, millel puudub päis või nad on samad.

Oluline kontroll toimus ka jahiandmete lugemisel, kuna otsustasin mällu lugeda ainult filtrile vastavad andmed tuleb kontrollida, kas neid üldse failis leidub. Funktsionaalsuse teostasin lihtsa abimuutujaga, mis loendab edukalt loetud ridu ning teavitab kasutajat juhul kui otsitavad andmed puuduvad.

Viimane suurem eriolukord, millele tähelepanu pöörasin on juhul kui statistikas ei kajastu kõik maakonnad või on jahialade andmed puudulikud. Mälu allutatakse küll kõikide maakondade jaoks, kuid statistika koostamisel loodud lisafunktsioon findIndex lisab uusi maakondi statistikasse jooksvalt, mille tulemusena ilmub tabel ka antud juhul edukalt. Jahialade andmestiku kontrollimiseks peab programm järge, kas jahiala kood andmestikust ikka leiti.

```
int fileID(FILE *f){ //  saan aru kumb fail on juhul kui programmi sisenditeks pannakse suvalises järjekorras
    static short next, read = 0;
    char template[2][NAMELEN] = {"aasta", "kood"};
    char temp[MAXROW];
    fscanf(f, "%s", temp);
    if(read == 0){
        if(strncmp(temp, template[0], 5) == 0){ // jahiandmed
            read++;
            next = 1;
            return 1;
        }
        else if(strncmp(temp, template[1], 4) == 0){ // jahialad
            read++;
            next = 0;
            return 2;
        }
        else
            return 0;
    }
    else{
        if(strncmp(temp, template[next], 5 - next) == 0){ // teise faili lugemise korral peab olema esimesest failist erinev
            return next + 1;
        }
        else
            return 0;
        }
    return 0;
}
```

Joonis 4. Alamfunktsioon *fileID*.

5 Kokkuvõte

Töö käigus loodi programm C keeles, mis kompileerub vigadeta, vastab seatud nõuetele ning lahendab ülesande. Kindasti ei ole tegemist kõige optimaalsema lahendusega, kuid seatud eesmärgid on täidetud ning kõik pähetulnud eriolukorrad on suudetud lahendada. Programmi loomisel erilisi murekohti ega probleeme ei tekkinud ning otsustasin lisada ka paar lisafunktsiooni muutes kasutajale rakenduse kasutamise mugavamaks ja andes rohkem valikuid andmete filtreerimisel.

6 Lisad

```
krist@DESKTOP-OI1AIHQ ~
$ ./kt
Enter year: 2008
Enter species: Alces alces
Sisestage põhjus
100 - Muu
101 - Küttimine
102 - Liikluses hukkunud
103 - Loendamine
  1 - Kõik
1
Filtrile vastavad 1234 rida
 Täiskasvanud | Keskmine |  Noor | Teadmata | Maakond
         2292 |         0 |  1065 |      235 |   Harju
           12 |         0 |     9 |        4 |   Hiiu
            1 |         0 |     2 |        2 |   Ida-Viru
           48 |         0 |    24 |        6 |   Järva
           76 |         0 |    19 |        2 |   Jõgeva
          133 |         0 |    59 |       25 |   Lääne
            3 |         0 |     0 |        0 |   Lääne-Viru
          135 |         0 |    57 |        5 |   Pärnu
           17 |         0 |    15 |        1 |   Põlva
           35 |         0 |    25 |        5 |   Rapla
            6 |         0 |     6 |        1 |   Saare
            2 |         0 |     3 |        2 |   Tartu
           14 |         0 |     8 |        2 |   Valga
           16 |         0 |     9 |        0 |   Viljandi
           29 |         0 |    13 |        4 |   Võru
```

Joonis 5. Rakenduse töö näidis.

```
krist@DESKTOP-OI1AIHQ ~
$ ./kt
Enter year: 2022
Enter species: Capreolus capreolus
Sisestage põhjus
100 - Muu
101 - Küttimine
102 - Liikluses hukkunud
103 - Loendamine
  1 - Kõik
1
Filtrile vastavad 0 rida
Failis ei ole vastavaid andmeid!
```

```
krist@DESKTOP-OI1AIHQ ~
$ ./kt
Enter year: 2008
Enter species: Alces alces
Sisestage põhjus
100 - Muu
101 - Küttimine
102 - Liikluses hukkunud
103 - Loendamine
  1 - Kõik
1
Error with file
```

Joonis 6. Eriolukordade näidised.