TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristjan Henri Roots
213450IACB

Kodune ülesanne 2

Programmeerimine I
(IAX0583)

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

12.12.21

Sisukord

Jooniste loetelu

1 Ülesande püstitus

Käesoleva kodutöö ülesandeks oli koostada programm C keeles, mis lahendaks etteantud variant 5 ülesande. Ülesandes oli tegemist kolme spordivõistlustega. Lähteandmeteks on kuus massiivi, kus leiduvad sportlaste nimed ning nende saavutatud punktisumma võistlusel. Ebaõnnestumise korral ei leidu sportlase nime massiivis. Ülesande eesmärgiks oli koostada programm, mis väljastaks kõikidel võistlustel osalenud sportlaste nimed, esikolmiku nende seast ja ka kõik need, kellel osad võistlused ebaõnnestusid.

Lisaparameetritena oli ülesande lahenduses vaja kasutada järgnevaid alamprogramme:

- Massiivide lugemiseks.
- Võistlejate nimede väljastamiseks.
- Esikolmiku väljastamiseks.
- Ebaõnnestujate väljastamiseks.

2 Lahenduse kirjeldus

Loodud programm alustab ülesande lahendamist lähteandmete lugemisest. Selle jaoks kasutasin varasemalt ka tunnist läbi käinud alamfunktsioone, millest esimene loeb sisse numbri massiivi suurusega. Teine neist loeb eelnevalt loetud muutuja suurusega kaks massiivi, mis hoiavad osalenud võistlejate nimesid ning nende saavutatud punktisummat; kuna võistlusi on kokku kolm, siis toimub ka sama tegevus kokku kolm korda.

Olles lugenud edukalt lähteinformatsiooni leiab programm massiivi suuruste summa juhuks, kui kõik sisestatud võistlejate nimed on unikaalsed. Antud muutuja järgi loon eraldi massiivid, mis hakkavad hoidma tervet osalejate nimekirja ning nende punktisummat. Punktisumma jaoks loodud massiivi tegin mitmemõõtmelise, et mugavalt järge pidada mitmel võistlusel iga sportlane lõppkokkuvõttes ikka osales. Loodud massiivid annan ette väiksele alamprogrammile, mis need omakorda algväärtustab. Iga element nimede massiivis saab algväärtusena väärtuse “1”, tavapunktiväärtuseks saab “0” ning osaluseks “1”. Algväärtustamise põhjuseks oli peamiselt lihtsam vigade tuvastamine; vale asja printimisel ei jooksnud programm tervenisti kokku ega ei andud ka meeletuid arve vastuseks.

Järgmiseks loob programm muutuja, mis peab järge mitu unikaalset nime lähteandmetes tegelikult oli, enamus programmi järgnevast tööst põhineb antud muutujal. Järgneb alamprogramm "*allList*", mis lühidalt öeldes lahendab enamuse ülesandest (Joonis 1, lk 7). Alamprogrammi kutsun esile kolmel korral iga võistluse jaoks, selle võlu seisneb ka selles, et esilekutsumise arvul ei ole vahet ning töötaks teoorias ka palju rohkemate võistlustega.

```
Unique names: 10
  1.     Juku - 140.1 - times seen:   3
  2.   Meelis - 141.6 - times seen:   3
  3.   Peeter - 139.6 - times seen:   3
  4.     Test - 206.6 - times seen:   2
  5.      Ats - 135.8 - times seen:   3
  6.     Paul - 130.1 - times seen:   3
  7.     Anti - 138.0 - times seen:   3
  8.     Peep -  86.6 - times seen:   2
  9.     Teet -  89.8 - times seen:   2
 10.     Mart -  57.1 - times seen:   1
```

Joonis 1. Alamfunktsiooni "*allList*" tulemus.

Alamprogrammi põhiosas on kaks *for* tsüklit, mis iga uue nimega lähteandmetest käib läbi ka loodud massiivi, mis hoiab kõiki nimesid. Kui nime ei leita, lisatakse see massiivi ning samale indeksile punktidemassiivis liidetakse punktid ja suurendatakse võistluste arvu ning unikaalsete nimede arvu muutujat. Juhul kui nimi on juba olemas kõikide nimede massiivis liidetakse vaid samale kohale punktid ning samuti suurendatakse võistlustes osalemise arvu (Joonis 2, lk 8). Alamfunktsiooni kolmel väljakutsumisel olen saanud teada mitu võistlejat on kokku, mis on nende punktisumma kokku kõikide võistluste peale ning mitmel võistlusel iga sportlane osales.

```
int i, j;
int breakcount; // Saan aru kas nimi on juba olemas

for(i = 0; i < n; i++){
    breakcount = 0;
    for(j = 0; j < uniqueNames; j++){
        if(strcmp(names[i], allNames[j]) == 0){ // nimi on juba olnud, punktid tuleb liita
            allScore[0][j] += score[i];
            allScore[1][j]++; // osales veel ühel võistlusel
            breakcount++;
            break;
        }
    }
    if(breakcount == 0){ // Nime ei leitud, lisan massiivi
        strcpy(allNames[uniqueNames], names[i]);
        allScore[0][uniqueNames] += score[i];
        uniqueNames++; // uus nimi lisatud
    }
}
```

Joonis 2. Alamfunktsioon "*allList*".

Peale punktide leidmist leiab programm nende alusel kahaneva järjestuse, et mitte rikkuda algandmeid kasutan selle teostamiseks indeksmassiivi täpselt samasuguselt nagu maatriksi järjestamise koduülesandel, kus ülesandeks oli järjestada maatriks kahanevate vahede järjekorras. Alamprogramm algväärtustab indeksmassiivi ning lisab sinna kahanevas järjekorras punktide massiivi indeksid.

Ülesande kirjelduses on küll öeldud, et kõikides võistlustes osalejate ja ebaõnnestujate väljastamiseks on vaja eraldi alamprogramme, kuid informatsioon osaluse kohta on juba kõik ühes kohas olemas ning alamprogrammid erineksid seetõttu üksteisest vaid kahe rea pealt (sõnum kasutajale enne nimesid ja *if* funktsiooni parameeter). Selle tõttu lõin ühe alamprogrammi "*print*" ning kontrollin kõikides osalejate ja ebaõnnestujate väljastamist sisendis etteantud lisamuutujaga. Kui on vaja väljastada kõikides osalejad peab osalus olema vähemalt kolm, ning kui ebaõnnestujad, siis alla kolme.

Viimaseks sammuks jääb esikolmiku väljastamine, lähtandmeteks saab alamprogramm massiivid kogu informatsiooniga ning indeksmassiivi. Alamprogrammis olev *for* tsükkel hakkab läbi käima punktide massiivi indeksmassiivi alusel, kontrollides samal ajal ka punktisummale vastava võistleja osalust, et mitte printida mittetäielike osalusi. Kui sobiv sportlane on leitud käiakse läbi ka veel järgnevad punktisummad. Juhul kui leitakse veel üks sportlane sama suure punktisummaga prinditakse vastaval kohal välja mõlemate sportlaste nimed. Siin tekkis ka väike küsimus, kas sellisel juhul peaksid nad jagama näiteks esimest ja teist kohta ning järgnev sportlane saab kolmanda koha? Otsustasin sama punktisumma omanikud panna jagama sama kohta. Tsükli viimaseks kontrolliks teeb ta selgeks olukorra kui tsükkel on lõppemas, kuid mingile positsioonile ei ole nime leitud. Sel juhul prinditakse koha number koos puuduva sportlasega ning väljutakse programmist (Joonis 3, lk 9).

```
for(i = 0; i < nx; i++){ // nx on uniqueNames, kõik vaatab läbi
    if(allScore[1][idx[i]] >= 3 && place < 4 && allScore[0][idx[i]] != previous){
        printf("%2d. koht: ", place);
        printf("%6s", allNames[idx[i]]);
        for(j = i + 1; j < nx; j++){ // vaatan kas kellelgi oli veel sama palju punkte
            if(allScore[0][idx[i]] == allScore[0][idx[j]] && allScore[1][idx[j]] >= 3){
                printf(", %s", allNames[idx[j]]);
            }
        }
        printf(" tulemusega %5.1f\n", allScore[0][idx[i]]); // prindi punktid
        place++;
        previous = allScore[0][idx[i]]; // kontroll järgmiseks korraks
    }
    if(i + 1 == nx && place <= 3){ // kui massiiv sai otsa ja kõikidel kohtadel pole veel kedagi
        for(j = place; j <= max; j++){
            printf("%2d. koht: -\n", j); // prindib nii mitu korda kui mitu puudu on
        }
    }
}
printf("\n");
```

Joonis 3. Esikolmiku väljastamine.

3 Eriolukorrad

Programmi loomisel üritasin tähelepanu pöörata võimalikele eriolukordadele, mis tekkida võivad. Esimene neist oli lahendada olukord, kui kõik võistlejad on unikaalsed ehk keegi ei lõpeta edukalt. Lahenduses kasutasin muutujat, mis jälgib unikaalseid nimesid ning lisasin juurde lihtsa kontrolli "*allList*" alamfunktsioonile, mis väljastab kasutajale olukorrale vastava info ning väljub vajadusel programmist (Joonis 5, lk 10). Väljumiseks kasutatud käskluse pärast kasutasin ka programmis teeki *stdlib.h*. Programmist väljutakse ka vales formaadis sisendandmete korral.

```
if(uniqueNames == N){
    printf("Mitte ükski võistleja ei lõpetanud kõiki võistlusi\n");
    exit(0);
}
```

Joonis 4. Unikaalsete nimede kontroll.

```
Unique names: 24
  1.     Juku -  49.8 - times seen:   1
  2.   Meelis -  51.0 - times seen:   1
  3.   Peeter -  47.5 - times seen:   1
  4.     Test -   6.6 - times seen:   1
  5.      Ats -  47.7 - times seen:   1
  6.     Paul -  49.2 - times seen:   1
  7.     Anti -  50.5 - times seen:   1
  8.     Peep -  49.5 - times seen:   1
  9.  Meelis1 -  35.2 - times seen:   1
 10.    Juku1 -  33.7 - times seen:   1
 11.  Peeter1 -  32.9 - times seen:   1
 12.    Paul1 -  31.0 - times seen:   1
 13.     Ats1 -  37.1 - times seen:   1
 14.    Teet1 -  36.5 - times seen:   1
 15.    Peep1 -  37.1 - times seen:   1
 16.    Anti1 -  33.3 - times seen:   1
 17.    Juku2 -  56.6 - times seen:   1
 18.  Peeter2 -  59.2 - times seen:   1
 19.    Teet2 -  53.3 - times seen:   1
 20.     Ats2 -  51.0 - times seen:   1
 21.    Paul2 -  49.9 - times seen:   1
 22.    Mart2 -  57.1 - times seen:   1
 23.  Meelis2 -  55.4 - times seen:   1
 24.    Anti2 -  54.2 - times seen:   1


Mitte ükski võistleja ei lõpetanud kõiki võistlusi

krist@DESKTOP-OI1AIHQ ~
$
```

Joonis 5. Eriolukord: Keegi ei lõpetanud edukalt.

Vastupidiselt töötab programm ka ainult ühe sportlase korral või ainult ühe eduka osaluse korral (Joonis 6, lk 11). Andmetes on ka võimalik sama sportlast mitu korda nimetada sama võistluse all ning nende punktid liidetakse, mis oleks teoorias lisasammuna võinud kõrvaldada.

Hetkeseisuga võimaldab programm ka negatiivseid tulemusi, kuid ülesande kirjelduses ei olnud nende olemasolu keelatud ning näiteks penaltipunktide jaoks on negatiivsed arvud sisse jäetud.

Esikolmiku printimisel sama punktitulemuse korral ning mõnede kohtade puudumisel "-" väljastamise lahendasin üleliigselt keeruka printimisfunktsiooniga, mis õnneks antud eriolukordi väldib (Joonis 6, lk 11).

```
Unique names: 1
  1.     Juku -  90.0 - times seen:  18


Kõikidel võistlustel osalesid: Juku

 1. koht:   Juku tulemusega  90.0
 2. koht: -
 3. koht: -


Võistlustel ebaõnnestusid: -
```

Joonis 6. Eriolukord: Üks unikaalne sportlane.

4 Kokkuvõte

Töö käigus loodi programm C keeles, mis kompileerub vigadeta, vastab seatud nõuetele ning lahendab ülesande. Ülesande esimesel lahendamisel proovisin järgida nõudeid täpselt ning teha kõik etteseatud punktid oma alamfunktsioonides, kuid lahendus venis meeletult pikaks uuesti nullist alustades iga punkti lahendamiseks. Ülesannet uuesti alustades otsustasin minna suurte massiividega, mis hoiaksid endas kogu lahenduseks vajaminevat informatsiooni ning ülesande lahendus muutus kordades lihtsamaks ning lühemaks. Kindasti ei väida ma, et tegu on parima lahendusega, kuid seatud eesmärgid on täidetud ning kõik pähetulnud eriolukorrad on suudetud lahendada.

5 Lisad

```
krist@DESKTOP-0I1AIHQ ~
$ ./sp
Sportlaste tulemused
7
Juku    49.8
Meelis  51.0
Peeter  47.5
Ats     47.7
Paul    49.2
Anti    50.5
Peep    49.5
8
Meelis  35.2
Juku    33.7
Peeter  32.9
Paul    31.0
Ats     37.1
Teet    36.5
Peep    37.1
Anti    33.3
8
Juku    56.6
Peeter  59.2
Teet    53.3
Ats     51.0
Paul    49.9
Mart    57.1
Meelis  55.4
Anti    54.2


Kõikidel võistlustel osalesid: Juku, Meelis, Peeter, Ats, Paul, Anti

 1. koht: Meelis tulemusega 141.6
 2. koht:   Juku tulemusega 140.1
 3. koht: Peeter tulemusega 139.6


Võistlustel ebaõnnestusid: Peep, Teet, Mart


krist@DESKTOP-0I1AIHQ ~
$
```

Joonis 7. Normaalvaste.

```
Kõikidel võistlustel osalesid: Juku, Meelis, Peeter, Ats, Paul, Anti

 1. koht:   Juku, Meelis tulemusega 141.6
 2. koht: Peeter tulemusega 139.6
 3. koht:   Anti tulemusega 138.0


Võistlustel ebaõnnestusid: Peep, Teet, Mart
```

Joonis 8. Sama punktisumma.