L09 Tehtävät

- Poikkeustenkäsittely tiedostonkäsittelyn ja käyttäjäsyötteiden kanssa
- Algoritmi datan luokitteluun ryhmiin
- Laajeneva viikkotehtävä

Lue oppaan tämän viikon asioita käsittelevä luku 9. Lisäksi tehtävien suorittamiseen tarvitset aiempien lukujen tietoja. Ohjelmointitehtävissä on oltava otsikkotiedot.

L09T1: Poikkeustenkäsittely tiedostojenkäsittelyn yhteydessä

Tee ohjelma, joka lukee tekstitiedoston listaan yhdessä aliohjelmassa ja kirjoittaa listan tiedostoon toisessa aliohjelmassa. Pääohjelmassa tulee olla listan määrittely, tiedostonimien kysyminen ja luku- sekä kirjoitusaliohjelmakutsut. Molemmissa aliohjelmissa tulee olla tiedostonkäsittelyyn liittyvät poikkeustenkäsittelyt eli tiedoston avaaminen, lukeminen/kirjoittaminen ja sulkeminen tulee tehdä poikkeustenkäsittelijän sisällä, jotta mahdolliset virhetilanteet saadaan käsiteltyä hallitusti. Luettavan ja kirjoitettavan tiedoston rakenne on sama: yksittäisiä kokonaislukuja aina yksi luku rivillä. Katso esimerkkiajosta ohjelman antamat tiedotteet käyttäjälle ohjelman kulun etenemisestä. Tulostettava rivimäärä on tiedostossa olevien tietoa sisältävien rivien määrä.

Ohjelman ydin on tuttu ja varsin lyhyt, mutta poikkeustenkäsittely laajentaa ohjelmaa. Mikäli tiedoston käsittely ei onnistu, ei siitä yritetä toipua vaan käyttäjälle kerrotaan alla olevalla virheilmoituksella tilanteesta ja lopetetaan ohjelman suoritus sys.exit(0) –käskyllä. Heittomerkkien sisällä tulee olla ongelman aiheuttaneen tiedoston nimi:

- `Tiedoston 'x.txt' käsittelyssä virhe, lopetetaan.`

Huomaa, että hakemistorakenteet näyttävät erilaisilta Linuxissa ja Windowsissa, mutta Macin hakemistorakenteet vastaavat Linuxia. Tee ja testaa ohjelmasi IDLEssä esimerkkiajon mukaisesti ja kiinnitä huomiota tiedostonimien esitysasuun tarpeen mukaan CodeGradessa. Windows-koneessa tiedoston kirjoittaminen epäonnistuu tyypillisesti viitattaessa levyyn, jota ei ole käytössä. Esim. tiedostonimi ’p:\eivoikirjoittaa.txt’ johtaa tyypillisesti Windows-koneessa virhe-ilmoitukseen ja jos ei, niin p:n tilalla voi kokeilla jotain muita kirjaimia (tai levyn nimiä), joita ei ole määritelty käytettävässä tietokoneessa. Linuxissa, ja CodeGradessä, tiedoston kirjoitusvirhe saadaan tyypillisesti aikaan /var/eivoikirjoittaa.txt -tyylisellä tiedostonimellä.

Ohjelman esimerkkiajo 1, normaali suoritus:

```
Anna luettavan tiedoston nimi: L09T1D1.txt
Tiedoston 'L09T1D1.txt' lukeminen onnistui, 7 riviä.
Anna kirjoitettavan tiedoston nimi: L09T1T.txt
Tiedoston 'L09T1T.txt' kirjoittaminen onnistui.
Kiitos ohjelman käytöstä.
```

Ohjelman esimerkkiajo 2, luettavan tiedoston kanssa ongelmia:

```
Anna luettavan tiedoston nimi: L09T1D1.txt
Tiedoston 'L09T1D1.txt' käsittelyssä virhe, lopetetaan.
```

Ohjelman esimerkkiajo 3, kirjoitettavan ohjelman kanssa ongelmia:

```
Anna luettavan tiedoston nimi: L09T1D1.txt
Tiedoston 'L09T1D1.txt' lukeminen onnistui, 7 riviä.
Anna kirjoitettavan tiedoston nimi: /var/eivoikirjoittaa.txt
Tiedoston '/var/eivoikirjoittaa.txt' käsittelyssä virhe, lopetetaan.
```

L09T2: Poikkeustenkäsittely käyttäjäsyötteiden yhteydessä

Tee ohjelma, jolla voit testata erilaisia poikkeuksia käyttäjän antamien syötteiden yhteydessä, erityisesti ValueError, IndexError, ZeroDivisionError ja TypeError. Toteuta ohjelmasi normaalin pääohjelman ja valikko-ohjelman avulla, joiden lisäksi kannattaa tehdä oma aliohjelma jokaisen virheen testaamiseen ValueErroria lukuun ottamatta. ValueError testaus kannattaa tehdä valikko-ohjelmassa kun käyttäjän antama syöte muutetaan kokonaisluvuksi – jos se ei onnistu, informoi käyttäjää tilanteesta ja pyydä uutta valintaa.

Kolmessa muussa virheentestausaliohjelmassa kannattaa käyttää rakennetta, jossa ensin pyritään tekemään haluttu operaatio tyypillisesti 2 koodirivillä ja jos se ei onnistu, mennään poikkeuksenkäsittelijään. Poikkeustenkäsittelijän tulee huomioida vain aliohjelman testaama poikkeus. IndexError tulee tyypillisesti, kun viitataan listaan indeksillä, jota ei ole. Näin ollen tee ko. aliohjelmaan lista, jossa on alkioina 11, 22, 33, 44 ja 55. Jakolaskun yhteydessä desimaaliluvun formaatissa on 2 desimaalia. Ja tyyppivirhe tulee esimerkiksi silloin, kun yrittää kertoa kahta merkkijonoa keskenään eli unohtaa muuttaa käyttäjän syötteen merkkijonosta numeroksi.

Huomaa, että CodeGradessa on toinen laajempi testiajo, jossa kaikki kohdat suoritetaan sekä onnistuneesti että poikkeuksen kanssa (paitsi TypeError vain poikkeuksella).

Ohjelman esimerkkiajo:

```
Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
3) Testaa ZeroDivisionError
4) Testaa TypeError
0) Lopeta
Valintasi: 1
Valikko-ohjelma testaa ValueError'n.
Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
3) Testaa ZeroDivisionError
4) Testaa TypeError
0) Lopeta
Valintasi: moi
Anna valinta kokonaislukuna.
Valintasi: 2
Anna indeksi 0-4: 7
Tuli IndexError, indeksi 7.
Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
```

```
3) Testaa ZeroDivisionError
4) Testaa TypeError
0) Lopeta
Valintasi: 3
Anna jakaja: 0
Tuli ZeroDivisionError, jakaja 0.
Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
3) Testaa ZeroDivisionError
4) Testaa TypeError
0) Lopeta
Valintasi: 4
Anna numero: auto
Tuli TypeError, auto*auto merkkijonoilla ei onnistunut.
Mitä haluat tehdä:
1) Testaa ValueError
2) Testaa IndexError
3) Testaa ZeroDivisionError
4) Testaa TypeError
0) Lopeta
Valintasi: 0
Kiitos ohjelman käytöstä.
```

L09T3: Algoritmi merkkijonojen luokitteluun ryhmiin

Tee ohjelma, joka lukee tekstitiedoston sisällön listaan, selvittää tiedostossa olevat eri automerkit, tulostaa niiden lukumäärän ja kaikki eri merkit näytölle sekä tallentaa ne tiedostoon. Määrittele pääohjelmassa listat luettaville tiedoille ja automerkeille sekä nimet luettavalle ja kirjoitettavalle tiedostolle. Lue tiedosto aliohjelmassa, joka saa parametreina tiedoston nimen ja listan. Tämän jälkeen analysoi tiedot yhdessä aliohjelmassa ja tulosta sekä tallenna automerkit kolmannessa aliohjelmassa. Mikäli luetussa tiedostossa ei ollut tietoja, kerro asiasta käyttäjälle äläkä kutsu analysointi- ja kirjoitusaliohjelmia.

Käyttäjälle näytettävät virheviestit ovat seuraavat:

- `"Tiedoston 'x' käsittelyssä virhe, lopetetaan."`, jossa x:n tilalla on virheen aiheuttaneen tiedoston nimi
- `"Tiedosto oli tyhjä, yhtään automerkkiä ei tunnistettu."`, mikäli luettu tiedosto oli tyhjä.

Tiedostossa olevat automerkit on järjestetty siten, että yhdellä rivillä on aina yksi automerkki ja kaikki samanmerkkiset autot ovat peräkkäisillä riveillä. Näin ollen voit selvittää eri merkit käymällä listan läpi ja aina kun automerkki vaihtuu, lisäät edellisen merkin listaan. Ole tarkkana, että ensimmäinen ja viimeinen rivi sekä automerkki tulevat käsiteltyä oikein.

Voit testata ohjelmaa tiedostoilla L09T3D1.txt, L09T3D2.txt ja L09T3D3.txt. Huomaa, että tiedosto L09T3D3.txt on tyhjä, ts. siellä ei ole mitään ja näin ollen sitä ei ole saatavilla Moodlessa, koska Moodle ei hyväksy tyhjiä tiedostoja.

Ohjelman esimerkkiajo:

```
Anna luettavan tiedoston nimi: L09T3D1.txt
Anna kirjoitettavan tiedoston nimi: L09T3T1.txt
Tiedostossa oli 8 eri automerkkiä.
Kia
Mazda
Mercedes-Benz
Opel
Renault
Seat
Toyota
Volkswagen
Kiitos ohjelman käytöstä.
```

L09T4: Laajeneva valikkopohjainen ohjelma virheenkäsittelyllä

Tällä viikolla valikkopohjaiseen ohjelmaan lisätään virheenkäsittely. Tälläkään kertaa ohjelmaan ei siis lisätä uusia ominaisuuksia vaan nyt tavoitteena on estää erilaisista virheistä johtuvat ongelmat ja ohjelman kaatuminen. Keskitytään tällä kurssilla tehtäviin virheentarkistuksiin eli tiedostonkäsittelyyn ja sopimattomien valintojen aiheuttamien ongelmien estämiseen. Tämän tehtävän lähtökohtana kannattaa käyttää itse tekemääsi ohjelmaa L08T4.

Muokkaa L08T4 ohjelmaa lisäämällä aina tiedostonkäsittelyn yhteyteen poikkeustenkäsittely, joka tulostaa tiedostovirheen sattuessa alla olevan ilmoituksen ja lopettaa ohjelman suorituksen. Laita lisäksi pääohjelmaan analysointi- ja kirjoitusvalintojen yhteyteen tarkistus, että ko. toimintoja kutsutaan vain, jos niissä tarvittavat tiedot ovat saatavilla. Muussa tapauksessa tulosta alla olevat virheilmoitukset. Analyysi edellyttää tiedoston lukemista, jonka voi varmistaa luettujen tietojen listasta. Tiedoston kirjoittaminen taas edellyttää analyysin tuloksia, joka näkyy tulos-olion jäsenmuuttujien arvoista.

Ohjelmakoodiin tarvittavat muutokset näkyvät oppimateriaalien esimerkeissä, joten niihin kannattaa tutustua, mikäli nämä asiat eivät ole jo tuttuja. Muutokset eivät näy mitenkään ohjelman normaalissa suorituksessa, mutta poikkeustilanteissa ne estävät ohjelman kaatumisen ja lopettavat ohjelman hallitusti.

Testaa ohjelma samoilla tiedostoilla kuin L07T4 ja L08T4, mutta tallennettavan tiedoston nimi on tehtävän mukaisesti esim. L09T4T1.txt. Moodlessa on testausta varten tiedostot L07T4D1.txt ja L07T4D2.txt. Alla on ohjelman virheilmoitukset kootusti:

- `"Tiedoston 'x' käsittelyssä virhe, lopetetaan."` siten, että x:n tilalla on virheen aiheuttaneen tiedoston nimi
- `"Ei analysoitavia tietoja, ei analysoitu."`
- `"Ei tallennettavia tietoja, tiedostoa ei tallennettu."`

Ohjelman esimerkkiajo:

```
Tämä on valikkopohjainen ohjelma, jossa voit valita haluamasi toiminnon.
Valitse haluamasi toiminto:
1) Lue tiedosto
2) Analysoi
3) Kirjoita tiedosto
0) Lopeta
Anna valintasi: 1
Luettavan tiedoston nimi on ''.
Anna uusi nimi, enter säilyttää nykyisen: L07T4D1.txt
Tiedosto 'L07T4D1.txt' luettu.

Valitse haluamasi toiminto:
1) Lue tiedosto
2) Analysoi
3) Kirjoita tiedosto
0) Lopeta
Anna valintasi: 2
Analyysi suoritettu.

Valitse haluamasi toiminto:
1) Lue tiedosto
2) Analysoi
3) Kirjoita tiedosto
0) Lopeta
Anna valintasi: 3
Kirjoitettavan tiedoston nimi on ''.
Anna uusi nimi, enter säilyttää nykyisen: L09T4T1.txt
Tulokset kirjoitettu tiedostoon.

Valitse haluamasi toiminto:
1) Lue tiedosto
2) Analysoi
3) Kirjoita tiedosto
0) Lopeta
Anna valintasi: 0
Lopetetaan

Kiitos ohjelman käytöstä.
```