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 otettua haltuun. 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 lopettaa 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, kun taas 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 monissa tietokoneissa virheilmoitukseen ja jos ei, niin p:n tilalla voi kokeilla jotain muita kirjaimia (tai levyn nimiä), joita ei ole määritelty käytettävässä tietokoneessa.

Ohjelman esimerkkiajo 1, normaali suoritus:

```
Anna luettavan tiedoston nimi: L09T1D1.txt
Tiedoston 'L09T1D1.txt' lukeminen onnistui, 7 riviä.
Anna kirjoitettavan tiedoston nimi: L09T1T1.txt
Tiedoston 'L09T1T1.txt' kirjoittaminen onnistui.
Kiitos ohjelman käytöstä.
```

Ohjelman esimerkkiajo 2, luettavan tiedoston kanssa ongelmia:

```
Anna luettavan tiedoston nimi: L09T1D10.txt
Tiedoston 'L09T1D10.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/L09T1D1.txt
Tiedoston '/var/L09T1D1.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 on luonnollinen osa valikko-ohjelmaa, 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 perusformaatti on 5 merkin kenttä ja 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 eri merkkien määrän sekä kaikki eri merkit näytölle ja tallentaa eri automerkit tiedostoon. Määrittele pääohjelmassa lista luettaville tiedoille sekä eri automerkeille ja 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 mitään tietoja, kerro asiasta käyttäjälle äläkä kutsu turhaan 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 automerkki ja rivi 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: Valikkopohjainen ohjelma / laskin virheenkäsittelyllä, jatkoa

Tämä tehtävä laajentaa aiemmin tehtyä laskinta, jossa valikkopohjaisen laskimen valintarakenne tehtiin tehtävässä L03T3, toistorakenne L04T5, aliohjelmarakenne L05T5, tiedostonkäsittely L06T5, listaratkaisu L07T5 ja useaan tiedostoon jako L08T4. Muokkaa L08T4 ohjelmaa lisäämällä tiedostonkäsittelyn yhteyteen poikkeustenkäsittely, joka tulostaa tiedostovirheen sattuessa tekstin `"Tiedoston 'x' käsittelyssä virhe, lopetetaan."` siten, että x:n tilalla on virheen aiheuttaneen tiedoston nimi. Laita lisäksi pääohjelmaan ennen kirjoitusaliohjelman kutsua tarkistus, jolla varmistetaan, että tallennettavassa listassa on tietoja ja jos näin ei ole, kerro tilanne käyttäjälle viestillä `"Ei tallennettavia tietoja, tiedostoa ei tallennettu."`. Tämän tehtävän lähtökohtana kannattaa käyttää itse tekemääsi ohjelmaa L08T4.

Poikkeustenkäsittely ja turhan aliohjelmakutsun välttäminen tehdään laittamalla sopiviin paikkoihin muutama sopiva koodirivi. Ohjelmakoodiin tarvittavat muutokset näkyvät oppimateriaalien esimerkeissä, joten niihin kannattaa tutustua, jos nämä asiat eivät ole entuudestaan tuttuja. Nämä muutokset eivät näy mitenkään ohjelman normaalissa suorituksessa, mutta poikkeustilanteissa ne estävät ohjelman kaatumisen ja lopettavat ohjelman hallitusti.

Tämä ohjelma lukee ja kirjoittaa samoja tiedostoja kuin ohjelma L06T5, vaikka tallennettavan tiedoston nimi onkin L09T4T.txt. Moodlessa on testausta varten tiedostot L06T5D1.txt ja L06T5D2.txt.

Ohjelman esimerkkiajo 1:

```
Anna luettavan tiedoston nimi: L06T5D2.txt
Anna kirjoitettavan tiedoston nimi: L09T4T1.txt
Tämä laskin osaa seuraavat toiminnot:
1) Anna luvut
2) Summa
3) Osamäärä
0) Lopeta
Valitse toiminto (0-3): 1
Luettu tiedosto 'L06T5D2.txt'.
Luettiin luvut 1 ja 2
Tämä laskin osaa seuraavat toiminnot:
1) Anna luvut
2) Summa
3) Osamäärä
0) Lopeta
Valitse toiminto (0-3): 2
Tulos lisätty listaan.
Tämä laskin osaa seuraavat toiminnot:
1) Anna luvut
2) Summa
3) Osamäärä
0) Lopeta
Valitse toiminto (0-3): 1
Luvut loppuivat, lopeta ohjelma.
Tämä laskin osaa seuraavat toiminnot:
1) Anna luvut
2) Summa
3) Osamäärä
0) Lopeta
Valitse toiminto (0-3): 0
Tallennettu tiedosto 'L09T4T1.txt'.
```

```
Kiitos ohjelman käytöstä.
```