**Juho Jansson** [**juho.jansson@tuni.fi**](mailto:juho.jansson@tuni.fi) **045-3299710**

**TIRA Postfix laskin, arvosana 5.**

**Ajankäyttö:**

5. – 6.3.2022 yht 12h symbolien tasapainotusohjelman kanssa pään seinään hakkaamista

6.3.2022 n. 4h postfix laskimen tekeminen ja sen testaileminen, josta ehkä noin 1h syötteen oikein saamisen tutkimiseen.

9.3.2022 n. ½h tämän kirjoittamiseen ja muutaman huomatun kirjoitusvirheen muokkaamiseen.

**Käyttöohjeet**:

Kokoomalla normaalisti g++ \*.cpp -o a , saa rakennettua laskimen kansioon nimellä a.

Ajo tapahtuu komennolla ./a , joka käynnistää tervetulotekstin sekä lyhyen valikon, jossa voi valita haluaako tarkastella 1 pisteen tehtävää (valmiskoodista koottu) vai 2-4 pisteen tehtävää. Valinta on tehty siksi, että annettujen syötteiden ja valmiskoodin välillä on merkittävä ero johtuen numeroiden syöttämisestä ?-merkki parametrin avulla. Laskimen voi myös ajaa -p flagilla päästäkseen vastaavaan kohtaan.

Valinnan suorittamisen jälkeen tulostuu manuaali, mitä kaikkia parametrejä on käytettävissä.

1 pst. tehtävässä ei sinänsä ole mitään erikoista, pois lukien se, että kyseiseen koodin on toki lisätty myös myöhemmissä kohdissa käytettävät funktiot.

2-4 pst. tehtävässä syötteen voi antaa joko putkeen, jokaisen operaattorin ja numerot erikseen tai sekä että useammalle riville. Ajoesimerkkejä alla:

Text

Description automatically generated

5 pst. Ohjelman voi myös ajaa asettamalla koko laskusuoritteen parametreinä laskimen ajon yhteydessä, jolloin tämä tulostaa vain tuloksen (huom ’=’ -merkkiä ei tarvita):

Background pattern

Description automatically generated

**Rakaisut:**

Huomattavasti hankalin osa-alue oli saada syöte luettua oikein, mutta loput koodin osa-alueet eivät olleen liiemmin hankaluuksia tuottavia stackin avulla ja voinee sanoa, että jopa itsestäänselviä.

Kaikki komennot luetaan käyttäjäsyötteestä ’\n’ -merkkiin (ENTER) asti INPUT -stringiin, joka syötetään sellaisenaan laskin-funktioon stored\_numbers -stackin kanssa. Vastaavasti argumenttien ollessa laskutoimituksen kohde, nämä yhdistetään pitkäksi stringiksi, jonka perään asetetaan = merkki ja valmistunut tuotos heitetään laskimeen.

Calculator -funktio käsittelee parametrinä tulleen merkkijonon (string) merkki kerrallaan for-loopissa sen pituutta vastaan ja numeroiden sekä ’.’:n ollessa kyseessä syöttää ne uuteen temp -stringiin. Lisäksi ’-’ -merkki syötetään vastaavasti temppiin, mikäli seuraava merkki on numero. Jos käsitelty merkki on välilyönti, yritetään istringstreamaa temp-merkkijono double-merkkiseen muuttujaan, tämän onnistuessa kyseinen luku laitetaan stackiin ja temp ”tyhjennetään” (asetetaan $-merkki). Jos seuraava merkki syötteestä tulleessa merkkijonossa, tarkastetaan onko kyseessä operaattori isOperator() -funktiolla ja mikäli tämä palauttaa arvon ”true”, se syötetään do\_command -funktioon toisena parametrinä stackin kanssa. Lopuksi vielä tarkistetaan, ettei tempiin ole jäänyt lukuja, jos käyttäjä syöttää vain yhden reaaliluvun (jolloin välilyöntiä ei luonnollisesti ole tarkasteltavassa merkkijonossa). Mikäli käyttäjä päättää lopettaa ohjelman [q]-merkillä, palauttaa laskin arvon bool cntn -vakioon pääohjelmassa, josta while-loopin pyöriminen on riippuvainen.

isOperator() -funktio tarkastaa for loopissa globaalia vakiomuuttujaa vasten globaalista vakio char-arraystä parametrinä tulleen merkin ja mikäli se löytyy sieltä, palauttaa boolean arvon ”true”, muuten palautetaan boolean arvo ”false”.

instruction() -funktio ottaa parametrinä boolean arvon true/false riippuen ollaanko 1pst. tehtävässä vai +2 pst. tehtävässä, jotta saadaan tulostettua oikeat ohjeet.