1. Johdanto

1.1. Tehtävä

Lausekielinen ohjelmointi II -kurssin toisena harjoitustyönä toteutetaan Java-ohjelma ASCII-grafiikkana [1] esitettyjen kuvien katseluun ja käsittelyyn. Ohjelma lataa käynnistyessään kuvan lukuesityksen tekstitiedostosta keskimuistiin. Kuvaa voidaan katsella kahdessa eri muodossa ja kuvan tiedot voidaan tulostaa näytölle. Tiedot sisältävät kuvan koon ja kuvan eri merkkien lukumäärät. Kuvan muokkaus tapahtuu keskiarvosuotimella. Muokkaus perutaan lataamalla kuva uudelleen tiedostosta. Ohjelma on voitava pysäyttää käyttäjän toimesta.

1.2. Kuva ASCII-muodossa

Harjoitustyössä käsitellään kuvia, joissa ajatellaan olevan 16 harmaan värisävyä. Jokaiselle värille on määritelty merkki (taulukko 1). Mustaa väriä vastaa ristikkomerkki ja valkoista väriä välilyönti. Näiden värien välissä on 14 harmaan sävyä siten, että harmaa tummenee kohti mustaa ja vaalenee kohti valkoista.

Merkit on pyritty valitsemaan siten, että merkin kuvapisteiden määrä vastaa karkeasti harmaan sävyä. Valkoinen esitetään tästä syystä "näkymättömällä" merkillä, jossa ei ole ollenkaan kuvapisteitä. Merkeillä on myös lukukoodit, joita tarvitaan, kun kuvaa käsitellään keskiarvosuotimella (luku 1.3). Kuvassa 1 on esitetty 16×16 -kokoinen *tree*-kuva merkki- ja lukumuodossa.

Kuva esitetään ohjelmassa vähintään yhtenä **kaksiulotteisena taulukkona**, jonka n rivin ja m sarakkeen kokoisena taulukkona ($n > 0$, $m > 0$).

Taulukoita voi olla myös kaksi kappaletta, jolloin ensimmäisessä taulukossa on kuvan merkit (**char**) ja toisessa taulukossa merkkejä vastaavat lukukoodit (**int**). Tässä tapauksessa kuvassa 1 merkkeinä esitetty puu (vasemmalla) olisi ensimmäisessä taulukossa ja merkkejä vastaavat luvut (oikealla) olisivat toisessa taulukossa. Huomaa, että Javan API-luokkien palvelujen käyttöä on rajoitettu (luku 3).

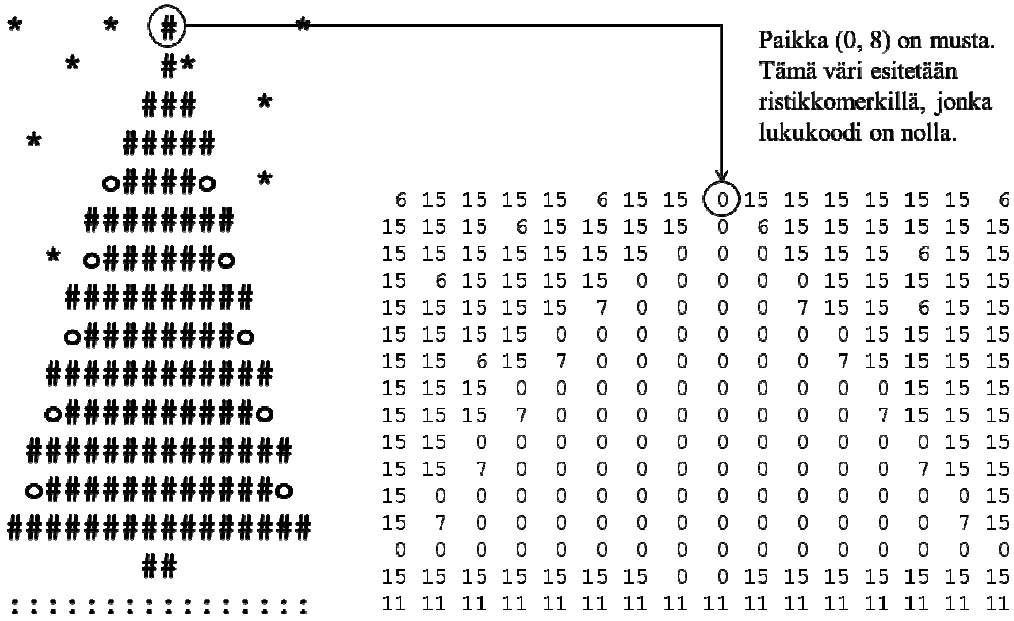
Kuvan paikkoihin viitataan jatkossa taulukon alkioiden tapaan kahdella nollasta alkavalla indeksiarvolla. Kuvan ensimmäinen paikka (0, 0) on vasemmassa yläkulmassa. Viimeinen paikka (n - 1, m - 1) on puolestaan oikeassa alakulmassa.

Kuva on aluksi tekstitiedostossa, josta se ladataan tietokoneen keskusmuistissa olevaan taulukkoon. Tekstitiedostossa oleva kuva koostuu merkeistä. Tiedostossa ei ole muita tietoja. Näin esimerkiksi kuvan koko täytyy päätellä latauksen yhteydessä. Kuvan sisältävän tiedoston voi olettaa olevan aina kunnossa. Kuvassa ei esimerkiksi ole taulukoon 1 kuulumattomia merkkejä.

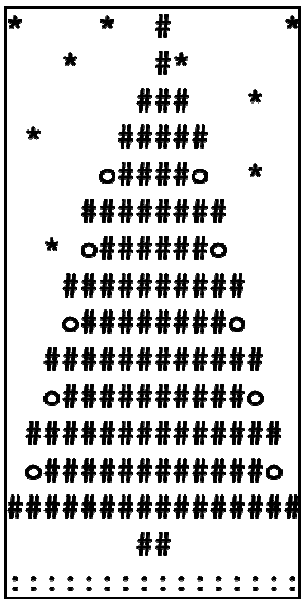
Kuvassa 2 on esitetty tiedoston *file_tree.txt* sisältö. Testeissä käytettävät kuvat julkaistaan kurssin kotisivuilla. Kuvia voi "piirtää" myös itse tekstieditorissa, josta kuvan voi tallentaa tekstitiedostoon ohjelmaan ladattavaksi.

Taulukko 1: Värejä vastaavat merkit ja merkkien lukukoodit.

#	@	&	$	%	x	*	o	|	!	;	:	'	,	.	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Kuva 1: *Tree*-kuva merkkeinä ja lukuina.



Kuva 2: Tiedoston *file_tree.txt* merkit. Jokaisella rivillä on 16 merkkiä.

1.3. Keskiarvosuodin

Keskiarvosuodinta [2] käytetään signaalin- ja kuvankäsittelyssä kohinan eli melun poistoon. Suodin poistaa kuvasta myös hyödyllisiä piirteitä kuten yksityiskohtia ja reunoja, jolloin kuva muuttuu sumeammaksi. Kuvankäsittelyssä keskiarvosuodin on tyypillisesti neliömäinen ikkuna, jonka sivun pituus on pariton ja vähintään kolme, kuten tässä työssä oletetaan.

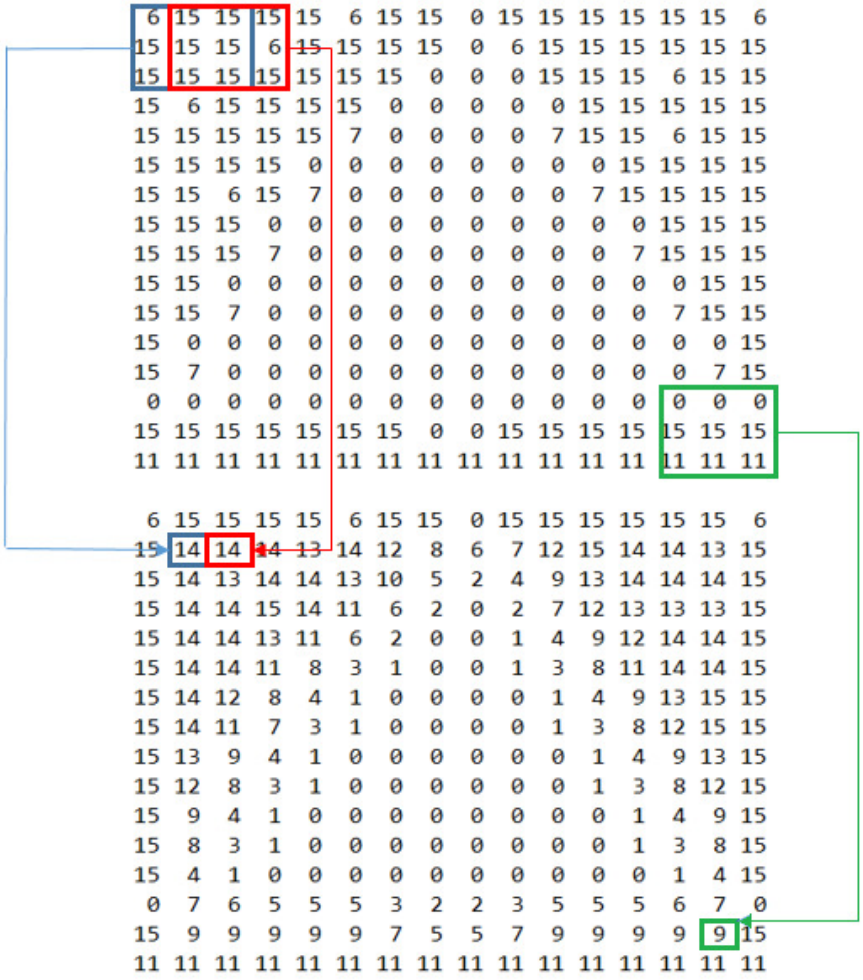
Suodinikkuna asetetaan lähtökuvan päälle kuvan reunojen suuntaisesti siten, että ikkunan keskikohta on lähtökuvan käsiteltävän paikan (*i*, *j*) päällä. Suodin laskee alueelleen kuuluvissa paikoissa olevien lukujen keskiarvon, joka on suotimen tulos. Keskiarvo sijoitetaan tuloskuvassa paikkaan (*i*, *j*). Harjoitustyössä pyöristäminen tehdään aina *Math*-luokan *round*-metodilla, jotta pyöristyksessä käytetään varmasti samoja sääntöjä. *Round*-operaation palauttaman **long**-tyyppisen arvon voi muuntaa tässä työssä **int**-tyyppiseksi arvoksi ilman pelkoa ylivuodosta.

Suodinta liu'utetaan kuvan yli siten, että suodatuksen aikana käsitellään jokainen lähtökuvan kohta, jonka päälle asetettu suodin mahtuu kuvaan kokonaisena. Tällöin kuvasta jää suodattamatta reuna, jonka leveys riippuu suotimen sivun pituudesta. Esimerkiksi 3 × 3 -kokoista suodinta käytettäessä suodinikkuna ei mahdu kokonaisena kuvan reunimmaisiin paikkoihin ja kuvaan jää yhden merkin levyinen käsittelemätön reuna. Jatkossa oletetaan, että kuvan käsitellään rivi kerrallaan edeten. Lähtökuva ja tuloskuva ovat suodatuksen aikana täysin erilliset. Jokainen keskiarvo lasketaan lähtökuvasta ja sijoitetaan tuloskuvaan. Keskiarvoa laskettaessa lähtökuvaa ei muuteta.

Kuvassa 3 esitetään, kuinka *tree*-kuvan kaksi ensimmäistä suodatukseen soveltuvaa paikkaa (1, 1) ja (1, 2) käsitellään samoin kuin miten viimeinen suodatukseen soveltuva paikka (14, 14) käsitellään. Suodatus aloitetaan kuvan toiselta riviltä, koska ensimmäiselle riville asetettu suodinikkunan ei mahdu kuvaan kokonaan. Toisen rivin ensimmäinen paikka (1, 0) ei käy, koska ensimmäiseen sarakkeeseen asetettu suodin peittää kuvan vain osittain. Suodatus aloitetaan paikasta (1, 1). Tuloskuvan paikkaan (1, 1) tulee (6 + 15 + 15 + 15 + 15 +15 + 15 + 15 +15) / 9 = 14. Tuloskuvan paikkaan (1, 2) tulee myös luku 14. Paikan (14, 14) suodatuksen tulos on (0 + 0 + 0 + 15 + 15 + 15 + 11 + 11 + 11) / 9 ≈ 9. Reuna-arvot on siirretty suoraan sellaisinaan tuloskuvaan, koska reunaa ei suodateta.

Huomaa, että ohjelmassa kuvan suodatus ei käsitä vain esimerkkinä annetun kolmea paikkaa, vaan kahden ensimmäisen paikan jälkeen ensimmäinen rivi käytäisiin loppuun ja suodatus jatkuisi aina seuraavalta riviltä, kunnes olisi käsitelty paikka (14, 14), jota seuraaviin paikkoihin suodin ei enää mahdu.

Keskiarvosuotimen toimintaan voi tutustua tarkemmin esimerkiksi katsomalla YouTubesta videon [3].



Kuva 3: Keskiarvosuodatusta *tree*-kuvalla. Paikkoihin (1, 1), (1, 2) ja (14, 14) asetettujen suotimien alueet on merkitty lähtökuvaan vastaavasti sinisellä, punaisella ja vihreällä neliöllä. Kunkin suodatuksen tulos on tuloskuvassa pienemmän neliön ympäröimä luku

1.4. Valmiin koodin käytöstä

Harjoitustyön teossa saa käyttää kurssin verkkosivuilla julkaistuja mallivastauksia joko sellaisenaan tai muokattuna. Osa viikkoharjoituksen tehtävistä ja niiden mallivastauksista on tarkoitettu avuksi harjoitustyön teossa. Tehtävät 4.2 (muunnos taulukon avulla), 4.7 (taulukon tulostus), 5.5 (frekvenssien laskeminen) ja 5.6 (tiedoston rivien laskeminen) ovat työssä erityisen hyödyllisiä.

Javan API:n tietorakenneluokkia ei saa käyttää; ainoa sallittu tietorakenne on taulukko. Javan API:n taulukkojen käsittelyyn liittyvät operaatiot on myös kielletty. Esimerkiksi *Arrays*-luokan palvelut ovat kieltolistalla. Tiedustele harjoitustyön ohjaajaan mielipidettä, jos Javan API:sta sattuu löytymään palvelu, jolla harjoitustyö ratkeaa kovin helposti. Harjoitustyön ohjaajaan kannattaa ottaa yhteyttä muutenkin, mikäli on epävarma siitä mitä saa tehdä ja mitä ei.

Harjoitustyö tehdään itse ja pääosin omalla ajalla. Kaverien kanssa saa keskustella, mutta suora kopiointi eli plagiointi on kiellettyä. Verkosta löytyviä ja harjoitustyötä muistuttavia ohjelmia saa ajaa, mutta niiden koodia ei saa kopioida. Muualta kuin kurssisivuilta poimitun koodin käyttö katsotaan plagioinniksi.

1.5. Pakollisuus ja korvaavuudet

Harjoitustyö on **pakollinen**. Ainoa poikkeus tähän sääntöön ovat harjoitustyön korvanneet opiskelijat. Harjoitustyön voi korvata:

A) Muiden oppilaitosten opinnoilla.

B) Syksyllä 2017 luennoidun Lausekielinen ohjelmointi II -kurssin hyväksytyllä toisella harjoitustyöllä, jos kurssi jäänyt kesken esimerkiksi ase- tai siviilipalveluksen tapaisesta pakottavasta syystä. Kurssin kotisivuilla on annettu tarkempia tietoja osasuorituksista.

C) Edellisten kohtien tapaisella painavalla syyllä.

Kohdan A perusteella on annettu kaikki opintokoordinaattorin (Heli Rikala) kurssin vastuuopettajalle (Jorma Laurikkala) esittämät korvaavuudet. B- ja C-kohtien osalta on tärkeintä muistaa, että harjoitustyö korvautuu vain, jos ottaa yhteyttä kurssin vastuuopettajaan. Tähän mennessä tulleet yhteydenotot ja sopimukset on kirjattu ylös eikä uusia yhteydenottoja näiltä osin tarvita.

2. Ohjelman toiminnot

Seuraavassa esitellään ohjelman toiminnallisuutta pienten esimerkkien avulla. Laajempia esimerkkiajoja julkaistaan kurssin kotisivujen *Opetus* | *Harjoitustyöt* | *Harjoitustyö 2* -kohdassa.

Ohjelmassa on komennot kuvan tulostamiseen näytölle merkki- ja lukumuodossa (luvut 2.4 ja 2.5), kuvan tietojen tulostamiseen (luku 2.6), kuvan suodattamiseen (luku 2.7), kuvan lataamiseen (luku 2.8) ja ohjelman lopettamiseen (luku 2.9).

2.1. Ohjelman käynnistys

Ohjelma lataa käynnistyessään tekstitiedostossa olevan ASCII-grafiikkakuvan lukuesityksen taulukkoon. Tiedoston nimi välitetään ohjelmalle komentoriviparametrina. Ohjelma käynnistetään seuraavalla komennolla:

```
java LakiHT2 file_tree.txt
```

kun kuvan merkit halutaan lukea tiedostosta *file_tree.txt*.

Uudelleenohjausta käytettäessä komento voi olla esimerkiksi:

```
java LakiHT2 file_tree.txt < input_tree.txt > out.txt
```

Yllä standardisyötevirta liitetään ohjelman suorituksen ajaksi näppäimistön asemasta *input_tree.txt*-tiedostoon, josta syötteet luetaan ohjelmaan rivi kerrallaan. Standarditulosvirta puolestaan liitetään näytön asemasta *out.txt*-tiedostoon, jolloin ohjelman tulosteet saadaan tallennettua tiedostoon.

Ohjelman tulostaa tervehdyksen (luku 2.2) jälkeen virheilmoituksen "Invalid command-line argument!" ja jäähyväiset (luku 2.9), jos komentoriviparametreja ei ole annettu, komentoriviparametreja on enemmän kuin yksi tai jos tiedostoa ei löydetä. Jos ohjelmaa pyydetään lataamaan jossain muussa kuin työhakemistossa oleva *file-x.txt*-tiedosto:

```
java LakiHT2 file-x.txt
```

niin tulostetaan:

```
------------------
| A S C I I A r t |
------------------
Invalid command-line argument!
Bye, see you soon.
```

2.2. Tervehdystekstin tulostaminen

Ohjelma tulostaa käynnistyessään näytölle miinusmerkeillä ('-') ja kahdella putkimerkillä ('|') kehystetyn viestin:

```
------------------
| A S C I I A r t |
------------------
```

Tämä teksti tulostetaan vain kerran.

2.3. Komennon lukeminen

Heti tervehdysrivien jälkeen tulostetaan omalle rivilleen ohjerivi "printa/printi/info/filter [n]/reset/quit?" ja jäädään odottamaan käyttäjän komentoa seuraavalla rivillä. Ohjerivi tulostetaan uudelleen jokaisen komennon jälkeen.

```
------------------
| A S C I I A r t |
------------------
printa/printi/info/filter [n]/reset/quit?
```

Harjoitustyössä oletetaan, että ohjelmalle annettu komento on aina oikeellinen.

2.4. Kuvan tulostaminen merkkimuodossa

Kuvan tulostetaan näytölle *printa*-komennolla. Rivit tulostetaan **täysimittaisina**, vaikka lopussa olisi vain välilyöntejä (valkoista väriä). Alla tulostetaan tiedostosta *file_tree.txt* ladattu kuva näytölle siten, että jokaisella rivillä on 16 merkkiä

```
printa
*    *  #      *
   *    #*
       ###   *
 *    #####
     o####o  *
    ########
  * o######o
   ##########
   o########o
  ############
  o##########o
 ##############
 o############o
################
       ##
:::::::::::::::
printa/printi/info/filter [n]/reset/quit?
```

2.5. Kuvan tulostaminen merkkimuodossa

Komento *printi* tulostaa kuvan näytölle lukumuodossa. Luvut erotetaan toisistaan välilyönnillä. Yksinumeroisen luvun eteen tulostetaan ylimääräinen välilyönti. Rivin alussa on yksi välilyönti, jos rivi alkaa yksinumeroisella luvulla. Rivien lopussa ei ole välilyöntejä.

Tulostetaan tiedostosta *file_tree.txt* ladattu kuva näytölle:

```
printi
 6 15 15 15 15  6 15 15  0 15 15 15 15 15 15  6
15 15 15  6 15 15 15 15  0  6 15 15 15 15 15 15
15 15 15 15 15 15 15  0  0  0 15 15 15  6 15 15
15  6 15 15 15 15  0  0  0  0  0 15 15 15 15 15
15 15 15 15 15  7  0  0  0  0  7 15 15  6 15 15
15 15 15 15  0  0  0  0  0  0  0  0 15 15 15 15
15 15  6 15  7  0  0  0  0  0  0  7 15 15 15 15
15 15 15  0  0  0  0  0  0  0  0  0  0 15 15 15
15 15 15  7  0  0  0  0  0  0  0  0  7 15 15 15
15 15  0  0  0  0  0  0  0  0  0  0  0  0 15 15
15 15  7  0  0  0  0  0  0  0  0  0  0  7 15 15
15  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15
15  7  0  0  0  0  0  0  0  0  0  0  0  0  7 15
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
15 15 15 15 15 15 15  0  0 15 15 15 15 15 15 15
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
printa/printi/info/filter [n]/reset/quit?
```

2.6. Kuvan tietojen tulostaminen

Komento *info* tulostaa näytölle kuvan koon ja värejä symboloivien merkkien lukumäärät. Ensin tulostetaan kuvan koko muodossa "n x m", missä *n* on kuvan rivien lukumäärä (korkeus) ja *m* on sarakkeiden lukumäärä (leveys). Sitten tulostetaan kukin merkeistä lukumääränsä kanssa omalle rivilleen taulukon 1 järjestyksessä (musta alussa, valkea lopussa). Merkki ja lukumäärä erotetaan yhdellä välilyönnillä.

Tree-kuvan nähdään olevan neliömäinen: sekä rivejä että sarakkeita on 16. Kuvassa on mustan ja valkoisen värin lisäksi kolme harmaasävyä. Mustaa vastaavia ristikkomerkkejä on 112 kappaletta, kun taas valkoista väriä vastaavia välilyöntejä on 109 kappaletta. Harmaan eri sävyjä on selvästi vähemmän.

```
info
16 x 16
# 112
@ 0
& 0
$ 0
% 0
x 0
* 9
o 10
| 0
! 0
; 0
```

```
: 16
' 0
, 0
. 0
  109
printa/printi/info/filter [n]/reset/quit?
```

2.7. Kuvan suodattaminen

Kuva käsitellään neliömäisellä keskiarvosuotimella, kun ohjelmalle annetaan komento *filter n*.

Komennon parametri *n* määrittelee suodinikkunan sivun pituuden, joka mitataan merkkinä. Komento erotetaan parametristaan yhdellä välilyönnillä. Esimerkiksi komennolla *filter 5* käytetäisiin 5×5 -kokoista keskiarvosuodinta. Sivun pituus on kolme merkkiä, jos komento annetaan ilman parametria.

Tässä työssä suodinikkunan sivun pituuden oletetaan olevan pariton ja vähintään kolme merkkiä. Suotimen oletetaan myös aina mahtuvan kuvaan.

Alla suodatetaan *tree*-kuva 3×3 -kokoisella suotimella ja tarkastellaan tuloskuvaa lukumuodossa.

```
filter
printa/printi/info/filter [n]/reset/quit?
printi
 6 15 15 15 15  6 15 15  0 15 15 15 15 15 15  6
15 14 14 14 13 14 12  8  6  7 12 15 14 14 13 15
15 14 13 14 14 13 10  5  2  4  9 13 14 14 14 15
15 14 14 15 14 11  6  2  0  2  7 12 13 13 13 15
15 14 14 13 11  6  2  0  0  1  4  9 12 14 14 15
15 14 14 11  8  3  1  0  0  1  3  8 11 14 14 15
15 14 12  8  4  1  0  0  0  0  1  4  9 13 15 15
15 14 11  7  3  1  0  0  0  0  1  3  8 12 15 15
15 13  9  4  1  0  0  0  0  0  0  1  4  9 13 15
15 12  8  3  1  0  0  0  0  0  0  1  3  8 12 15
15  9  4  1  0  0  0  0  0  0  0  0  1  4  9 15
15  8  3  1  0  0  0  0  0  0  0  0  1  3  8 15
15  4  1  0  0  0  0  0  0  0  0  0  0  1  4 15
 0  7  6  5  5  5  3  2  2  3  5  5  5  6  7  0
15  9  9  9  9  9  7  5  5  7  9  9  9  9  9 15
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11

printa/printi/info/filter [n]/reset/quit?
```

2.8. Kuvan lataaminen

Alkuperäinen kuva saadaan käyttöön *reset*-komennolla, joka lataa kuvan uudelleen tiedostosta keskusmuistiin.

Ladataan kuva uudestaan ja katsotaan sitä *printa*-komennolla:

```
reset
printa/printi/info/filter [n]/reset/quit?
printa
*    *  #      *
   *    #*
       ###   *
 *    #####
     o####o  *
    ########
  * o######o
   ##########
   o########o
  ############
  o##########o
 ##############
 o############o
################
       ##
:::::::::::::::
printa/printi/info/filter [n]/reset/quit?
```

2.9. Ohjelman lopettaminen

Ohjelman lopetetaan *quit*-komennolla. Ohjelma tulostaa ennen pysähtymistään lyhyet jäähyväiset "Bye, see you soon."

```
quit
Bye, see you soon.
```

3. Koodista

Ohjelma kirjoitetaan tuttuun tapaan *main*-operaation sisältävään luokkaan. Ensimmäisestä harjoitustyöstä poiketen ohjelma on kuitenkin jaettava **operaatioihin**. Operaatioiden lukumäärä riippuu paljon ohjelmoijasta. Operaatiot eivät kuitenkaan saa olla liian pitkiä. Erityisesti *main*-operaatio on usein vaarassa venyä liian pitkäksi.

Ohjelmassa on vältettävä saman koodin kopioimista eri paikkoihin. Toisteinen koodi pitää eristää omaksi operaatiokseen, jota kutsutaan ohjelmassa kohdista, joissa oli operaatioon siirrettyä koodia.

Muista noudattaa hyvää ohjelmointitapaa. Sisennä koodia johdonmukaisesti luettavuuden parantamiseksi, kommentoi riittävästi ja oikeissa paikoissa, liitä jokaiseen operaatioon yleisluonteinen kommentti, nimeä vakiot, muuttujat ja operaatiot järkevästi, käytä vakioita, pidä rivit riittävän lyhyinä, käytä välejä lauseiden sisällä ja erota loogiset kokonaisuudet toisistaan väliriveillä ja pidä operaatiot järkevän mittaisia – operaation tulisi mahtua yhdelle A4-kokoiselle sivulle.

Voit kerrata hyvän ohjelmointitavan lukemalla Lausekielinen ohjelmointi I -kurssin luentorungon 14. luvun [4] ja Lausekielinen ohjelmointi II -kurssin luentomateriaalin kuudennen luvun [5].

Harjoitustyössä ei saa käyttää normaaleja, muuttuvia attribuutteja. Harjoitustyössä on tavoitteena jakaa ohjelma helposti hallittaviksi operaatioiksi, jotka kommunikoivat ympäristönsä kanssa parametrilistan ja paluuarvon kautta. Attribuutit näkyvät kaikkiin operaatioihin, jolloin ohjelman tietoja voidaan muuttaa missä tahansa. Paljon attribuutteja sisältävät ohjelmat ovat huonoja muun muassa siksi, että tällaisen ohjelman toimintaa on hyvin vaikea hahmottaa. Vakioituja attribuutteja saa käyttää vapaasti, koska vakion arvoa ei voi muuttaa operaatioissa.

Älä käytä muita tietorakenteita kuin taulukkoja. Esimerkiksi listaa ei saa käyttää oli kyseessä Javan API:n valmis luokka tai itse toteutettu koodi.

Älä käytä *Arrays*-luokkaa tai vastaavia luokkia, joilla taulukoiden käsittelyn salat on helppo ulkoistaa Javan omille palveluille.

Sisennä koodi välilyönnein. WETO ei hyväksy muilla tavoilla sisennettyä lähdekoodia, jotta koodi näkyisi opettajan editorissa täsmälleen opiskelijan aikomalla tavalla.

Lue tiedot näppäimistöltä *In*-luokalla. Harjoitustyössä käyttäjän syötteiden lukuun ei saa käyttää *Scanner*-luokan tapaisia vaihtoehtoisia menetelmiä, jotta töiden automaattinen tarkistus (luku 6) onnistuisi paremmin. *In*-luokka löytyy kurssin sivuilta.

4. Dokumentointi

Harjoitustyöstä kirjoitetaan dokumentti, jonka tulee sisältää seuraavat asiat:

- Kansilehdellä ovat tekijän nimi, opiskelijanumero, sähköpostiosoite, tiedekunta ja tutkinto-ohjelma. Avoimen yliopiston opiskelijat voivat merkitä tiedekunnan ja tutkinto-ohjelman sijasta organisaatiokseen pelkästään avoimen yliopiston. Sivun keskellä tulisi olla suuremmalla fontilla dokumentin nimi. Kurssin kotisivuilla julkaistaan esimerkinomainen kansilehti.
- Ohjelman rakenteen ja toiminnan kuvaus korkeintaan puolen sivun mittaisena vapaamuotoisena tekstinä.
- Omia ajatuksia. Esimerkiksi: Oliko työ helppo, sopiva vai vaikea? Jos helppo tai vaikea, niin miksi? Mitä uutta opittiin? Oliko työstä mitään hyötyä tekijälleen? Montako tuntia työn tekemiseen meni?

Dokumentin leipäteksti kirjoitetaan 12 pisteen fontilla ja yhdellä rivinvälillä. Valmiin tekstin lukeminen pariin otteeseen on suotavaa. Tekstinkäsittelyohjelma on oikea työväline dokumentin tuottamiseen, koska esimerkiksi kielentarkistus oikolukutoiminnolla on tällaisessa ohjelmassa vaivatonta.

Dokumentti on palautettava PDF-muodossa. Muut tiedostomuodot eivät kelpaa. Luvussa 6 ja kurssin verkkosivuilla kerrotaan tarkemmin dokumentin ja koodin palauttamisesta.

5. Harjoitustyön ohjaus

Harjoitustöitä ohjataan mikroluokissa. Ohjaukseen voi tulla harjoitusryhmästään riippumatta. Ohjausajat eivät sulje toisiaan pois. Halutessaan voi osallistua vaikka kaikkiin harjoitustyön ohjauksiin. Huomaa, että ohjauksiin voi tulla ilman, että ohjelmassa on valmiiksi jokin ongelma, koska luokassa voi vain kirjoittaa ohjelmaa ja kysyä tarvittaessa neuvoa ilmenneisiin ongelmiin. Apua saa myös harjoitusten yhteydessä.

Sähköpostilla lähetetyt kysymykset on hyvä laittaa oman ryhmän vetäjälle, jotta kysymykset jakaantuisivat tasaisesti kurssin opettajille. Vastuuopettajan ryhmä on poikkeus: Jorman ryhmäläisten tulisi lähettää kysymykset ensisijaisesti harjoitustyön laatijalle (Jyrki Rasku, Jyrki.Rasku@uta.fi).

Tuntiopettajat vastaavat ryhmäläistensä töiden tarkistuksesta. Vastuuopettajan ryhmäläisten harjoitustyöt jaetaan tuntiopettajille tarkistettaviksi.

6. Harjoitustyön palautus

Ohjelma ja dokumentti täytyy palauttaa viimeistään **keskiviikkona 19.12.2018 klo 12.00** WETO-järjestelmään. Tarkemmat palautusohjeet julkaistaan myöhemmin kurssin verkkosivuilla.

Harjoitustöiden toiminnallisuuden tarkistamiseen käytetään WETO-järjestelmää, joka vertailee automaattisesti mallivastauksen ja opiskelijoiden ratkaisujen tulosteita. Tästä syystä **edellä annettuja tulostemäärittelyjä on seurattava merkilleen**. Automaattinen vertailu vähentää testaustyötä, jolloin opettajille jää enemmän aikaa mielekkäämpään työhön eli ohjelman rakenteen ja tyylin tutkimiseen. Opiskelijat hyötyvät tästä perusteellisempien kommenttien muodossa.

Lisäaikaa työn tekoon voi saada muutaman päivän hyvästä syystä. Lisäajasta on sovittava harjoitusryhmän vetäjän kanssa ajoissa eli viimeistään päivää tai paria ennen palautuksen takarajaa.

Ennen palautusta on syytä varmistaa, että dokumentissa on mukana kaikki edellä mainitut kohdat. Lisäksi kannattaa tarkistaa, että ohjelma toimii varmasti oikein viimeisten muutosten jälkeen.

7. Harjoitustyön arvostelu

Harjoitustyö arvostellaan asteikolla hyväksytty/hylätty. Hylkäyksen perusteena voi olla esimerkiksi ohjelman virheellinen toiminta, hyvän ohjelmointitavan noudattamatta jättäminen, ohjelman rakenne (erityisesti koodin monistaminen ja liian pitkät operaatiot), huono dokumentti ja/tai plagiointi. Plagiointiin liittyy sanktio, joka koskee molempia opiskelijoita. Toiselta opiskelijalta tämän tietämättä kopioidun koodin käyttö johtaa kopioijan koko kurssisuorituksen hylkäämiseen.

Suurempia korjauksia vaativa hylätty työ on korjattava pääsääntöisesti viikon sisällä hylkäyksestä. Työn voi palauttaa opettajan tarkistettavaksi korkeintaan neljä kertaa ilman pakottavaa syytä lisäpalautuksiin

Lähteet

[1] Wikipedia-yhteisö: ASCII art,
https://en.wikipedia.org/wiki/ASCII_art
(Luettu viimeksi 26.11.2018.)

[2] R. C. Gonzalez, R. E. Woods. 2001. Digital Image Processing (2nd ed.).
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[3] Udacity:
https://www.youtube.com/watch?v=ZoaEDbivmOE
(Katsottu viimeksi 26.11.2018.)

[4] J. Laurikkala: Lausekelinen ohjelmointi I -kurssin luentorunko, luku 14,
http://www.sis.uta.fi/~laki1/luennot/luento08/
(Luettu viimeksi 30.11.2018.)

[5] J. Laurikkala: Lausekelinen ohjelmointi II -kurssin luentorunko, luku 6,
http://www.sis.uta.fi/~laki2/luennot/luento06/
(Luettu viimeksi 30.11.2018.)