Janita Karppinen

TEKSTINKÄSITTELYOHJELMIEN JA EDITORIEN OMINAISUUDET JA EROT

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Syksy 2004

Kajaanin ammattikorkeakoulu

INSINÖÖRITYÖ
TIIVISTELMÄ

Osasto Tekniikka	Koulutusohjelma Tietotekniikan insinööri
Tekijä(t) Karppinen Tiina Janita	
Työn nimi Tekstinkäsittelyohjelmien ja editorien ominaisuudet ja erot	
Vaihtoehtoiset ammattiopinnot Ohjelmistotekniikka	Ohjaaja(t) Jukka Heino
Aika Syksy 2004	Sivumäärä 33+16

Tiivistelmä

Insinöörityön tavoitteena oli tutustua tekstinkäsittelyohjelmien sekä editorien ominaisuuksiin ja niiden välisiin eroihin. Lisäksi työssä tarkasteltiin tekstinkäsittelyohjelmien ja editorien toteuttamista.

Tavoitteena oli myös suunnitella ja toteuttaa Kajaanin ammattikorkeakoululle ASCII-editori. ASCII-editorin tilaajana oli opettaja Jukka Heino. Editorin tarkoitus on toimia Kajaanin ammattikorkeakoulun 8051-prosessorikortin editorina.

Editori tehtiin C-kielellä. Editorin koko rajattiin pariin kilotavuun, mikä otettiin huomioon editorin suunnittelussa ja toteutuksessa. Editori sisältää tekstinkäsittelyn perustoiminnot, kuten tekstin lisäämisen ja poistamisen. Editori toimii sarjaportin kautta käyttäen hyväksi jo olemassa olevia I/O-rutiineja.

Luottamuksellinen Kyllä Ei X	
Hakusanat Editori	
Säilytyspaikka Ammattikorkeakoulu	

Kajaanin ammattikorkeakoulu

Kajaani Polytechnic

ABSTRACT
THESIS

Faculty Faculty of Engineering	Degree programme Information Technology
Author(s) Karppinen Tiina Janita	
Title Features and Differences between Word Processing Programs and Editors	
Optional professional studies Software technology	Instructor(s) / Supervisor(s) Jukka Heino
Date Fall 2004	Total number of pages 33+16

Abstract

The Bachelor's thesis was made for Kajaani Polytechnic. Its goal was to study word processing programs and editors, and to compare differences between them. Furthermore, the purpose was to study the making of word processing programs and editors in general.

The purpose of the thesis was also to design and make an editor for the Kajaani Polytechnic 8051 processor card. The thesis was commissioned by Mr Jukka Heino.

The editor includes basic functions of word processing such as adding and removing text. The size of the editor was limited to a few kilobytes, which was taken into consideration when designing and making the editor. The editor works through a serial port with already existing I/O routines. The editor was made with the C programming language.

Confidential Yes No X	
Keywords Editor	
Deposited at Kajaani Polytechnic	

SISÄLLYSLUETTELO

1 JOHDANTO

Tekstinkäsittelyllä tarkoitetaan kirjoitettavan tekstin tuottamista ja muokkaamista. Tekstinkäsittelyä varten on olemassa erilaisia ohjelmia, joilla tekstiä voi tuottaa. Nykyaikana on erittäin yleistä, että ihmiset hankkivat tietokoneen pelkästään tekstin käsittelemistä varten. Tekstinkäsittelyyn on olemassa kahta erityyppistä ohjelmaa, tekstinkäsittelyohjelmat sekä editorit.

Tekstinkäsittelyohjelmien avulla tekstiä voidaan tuottaa, muokata sekä tulostaa haluttuun muotoon. Tekstinkäsittelyohjelmat ovatkin lähes korvanneet kirjoituskoneet. Tekstinkäsittelyohjelmien suurimpana etuna kirjoituskoneisiin verrattuna on kerran kirjoitetun tekstin uudelleenkäytettävyys. Tekstiä voidaan muokata helposti, siihen voidaan tehdä korjauksia ja se voidaan siirtää tai kopioida paikasta toiseen. Tekstin osien siirtäminen ja kopioiminen onnistuu myös eri dokumenttien välillä. Tekstinkäsittelyohjelmat ovatkin tietokoneiden yleisimpiä sovelluksia. Tekstinkäsittelyohjelma ei kuitenkaan ole pelkästään kirjoituskoneen korvike, sillä nykyiset ohjelmat sisältävät useita kirjoittamista helpottavia toimintoja, joita ei ole edes mahdollista tehdä kirjoituskoneella.

Editorit puolestaan ovat yksinkertaiseen tekstin muokkaamiseen tarkoitettuja ohjelmia. Niiden avulla voidaan tuottaa tekstiä, mutta tekstin muokkaamiseen tarkoitettuja työkaluja ei ole niin paljon kuin varsinaisissa tekstinkäsittelyohjelmissa.

Tämän insinöörityön tavoitteena on tutustua tekstinkäsittelyohjelmien sekä editorien ominaisuuksiin ja eroavaisuuksiin. Lisäksi tarkasteltiin tekstinkäsittelyohjelmien ja editorien toteuttamista. Tekstinkäsittelyohjelmista tutustutaan lähinnä vain Microsoft Wordin tarjoamiin ominaisuuksiin. Editoreja sen sijaan on niin runsaasti, että editorien kirjo vaatii tutustumaan useampaan editoriin, koska jokainen editori on erilainen ja sisältää erilaisia ominaisuuksia.

Lisäksi tavoitteena on perehtyä editoreihin tarkemmin toteuttamalla Kajaanin ammattikorkeakoulun 8051-prosessorikortille pieni merkkieditori. Editori toimii sarjaportin kautta käyttäen hyväksi jo olemassa olevia I/O-rutiineja. Editorin koko saa olla korkeintaan pari kilotavua ja tämä onkin yksi editorin tärkeimpiä ominaisuuksia. Lisäksi editorilla tulee olla tekstinkäsittelyn perustoiminnot, kuten tekstin lisääminen ja poistaminen.

2 EDITORIT

Tekstieditorit ovat yksinkertaisia tekstintuottamisen perusvälineitä. Tekstieditorin määrittelevä ominaisuus on se, että editorilla käsitellään ASCII-tekstiä, joka ei sisällä mitään muotoiluja, ainoastaan merkkejä, välilyöntejä ja rivinvaihtoja. Tekstieditorit ovat usein kooltaan pieniä ja prosessorivaatimuksiltaan vaatimattomia. Tekstieditorit ovat siten yleensä kevyitä ja nopeita käyttää. Editorien tuottamat tekstitiedostot ovat myös kooltaan paljon tekstinkäsittelyohjelmien tuottamia tiedostoja pienempiä. Tekstitiedostot ovat myös hyvin siirrettäviä koneelta tai järjestelmästä toiseen. Käytännössä kaikista tietokonekokoonpanoista löytyy ohjelmia tekstitiedoston editoimiseen. [1.]

Tekstieditorit soveltuvat kaikenlaiseen tekstinkäsittelyyn, erityisesti silloin kun ensisijainen tarve ei ole tekstin tulostaminen paperille. Editoreja käytetään erityisesti ohjelmoinnissa ja muussa koodauksessa. Tekstieditorit soveltuvat kuitenkin myös tulostettavan tekstin tuottamiseen, koska editorilla tuotettu teksti voidaan siirtää muotoilua varten tekstinkäsittely- tai taitto-ohjelmaan. [1.]

Editoreja on periaatteessa kahdenlaisia: rivi- ja kuvaruutueditoreja (line editors, full-screen editors). Käytännössä rivieditoreja ei nykyään käytetä paljoakaan, koska kuvaruutueditorit ovat huomattavasti havainnollisempia. Rivieditoreja ohjataan antamalla komentoja, vähän samaan tapaan kuin DOSin komentotulkissa. Muokattavan tekstin näkee vain erillisellä komennolla. Kuvaruutueditoreissa käyttäjä näkee koko ajan tekstistä kohdan, johon on kirjoittamassa. [2.]

2.1 Rivieditori

Rivieditori on yksi vanhimmista markkinoilla olevista tekstinkäsittelyyn tarkoitetuista ohjelmista. Tällainen ohjelma mahdollistaa vain yhden tekstirivin kirjoittamisen ja muokkaamisen kerrallaan. Esimerkiksi UNIXin ja vanhan MS-DOSin niin sanotut komentopohjaiset käyttöliittymät perustuvat juuri rivieditoreihin.

Periaatteessa rivieditorilla voi kirjoittaa useastakin rivistä koostuvia kokonaisuuksia ja tallentaa niitä tiedostoihin, mutta tämä on erittäin vaivalloista. Tiedostoa voi kelata riveittäin eteen- ja taaksepäin ja tehdä tarvittavia muutoksia. Nykyisin rivieditorin käyttötapa on kuitenkin tavallisen käyttäjän kannalta toki lähinnä historiallinen kuriositeetti. [3.]

2.2 Kuvaruutueditori

Kuvaruutueditoreilla asiakirjasta voi nähdä samalla kertaa useampia kuin yhden rivin. Näytöllä oleva ikkuna asiakirjasta voi olla vaikka koko näytön kokoinen. Varsinaisista tekstinkäsittelyohjelmista kuvaruutueditorit eroavat lähinnä vain suppeamman toimintovalikoimansa osalta. Nimitys editori suomennetaan joskus tekstintoimittimeksi, koska kyseessähän on suora lainaus englannin sanasta editor, joka taas on johdettu tekstin muokkaamista tarkoittavasta verbistä edit. [3.]

Erilaisia kuvaruutueditoreita tulee usein käyttöjärjestelmien ja sovellusohjelmistojen mukana, ja lisäksi niitä on runsaasti Internetissäkin vapaasti omalle koneelle imuroitavissa. Ne voivat olla yleiskäyttöisiä tai vain johonkin tiettyyn tarkoitukseen tehtyjä. [3.]

MS Windows -käyttöjärjestelmissä on yleensä yksinkertainen Muistio ja sitä aavistuksen verran monipuolisempi Wordpad. UNIX-järjestelmissä on yleensä ainakin editori nimeltä Vi, joka on kylläkin kaikkea muuta kuin helppokäyttöinen.

Tarjolla saattaa lisäksi olla esimerkiksi Micro Emacs tai Pine-sähköpostiohjelman mukana tuleva, suhteellisen käyttäjäystävällinen, editori nimeltä Pico. Picolla on myös avointa lähdekoodia edustava serkku nimeltä Nano. Pinen ohella toiseksi esimerkiksi kuvaruutueditoreita sisältävistä sovellusohjelmista voi ottaa Mozillan, joka mahdollistaa kirjeiden, uutisryhmiin lähetettävien artikkeleiden ja WWW-sivujen kirjoittamisen. [3.]

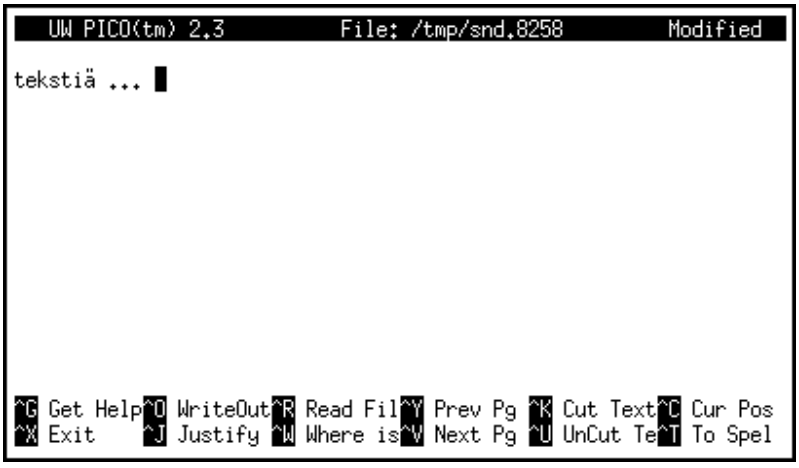
2.3 Editorien ominaisuuksia

Editorien ominaisuudet vaihtelevat eri editorien välillä, ja editoreja on useita erilaisia. Editori voidaan jopa tehdä vain jotain tiettyä käyttötarkoitusta varten. Usein käyttäjä ei edes havaitse kaikkia editoreja, sillä lähes kaikki ohjelmat, joilla käsitellään tekstiä sisältävät editorin. Esimerkiksi sähköpostiohjelmien sisällä asuu pieni editori.

Yleensä editoreja käytetäänkin ohjelmointiin, harvemmat käyttävät editoreja varsinaiseen tekstinkäsittelyyn. Ohjelmointiin tarkoitetut editorit sisältävät usein toimintoja, jotka helpottavat koodin tuottamista. Eräs näistä on toiminto, joka tunnistaa ohjelmointikomennot ja muuntaa niiden värin esimerkiksi tummemmaksi. Jotkut editorit helpottavat ohjelmoijan työtä jopa tekemällä sisennyksetkin automaattisesti. Näin koodia on helpompaa tuottaa ja eri tekijöiden väliset erot pienenevät. Tämä tekee koodista luettavampaa myös muiden kuin tekijänsä silmissä.

2.3.1 Pico

Pico-editorilla on neljä perusominaisuutta, jotka ovat kappaleen tasaus, merkkijonojen haku, oikeinkirjoituksen tarkistus ja tiedostojen selaus. Näillä ominaisuuksilla on kuitenkin tiettyjä rajauksia, kuten merkkijonojen haussa ei eritellä isoja ja pieniä kirjaimia ja oikeinkirjoituksen tarkistus on vain englanninkielisille teksteille. Kuva 1 esittää erään Pico-editorin käyttöliittymän.



Kuva 1. Pico-editori [4]

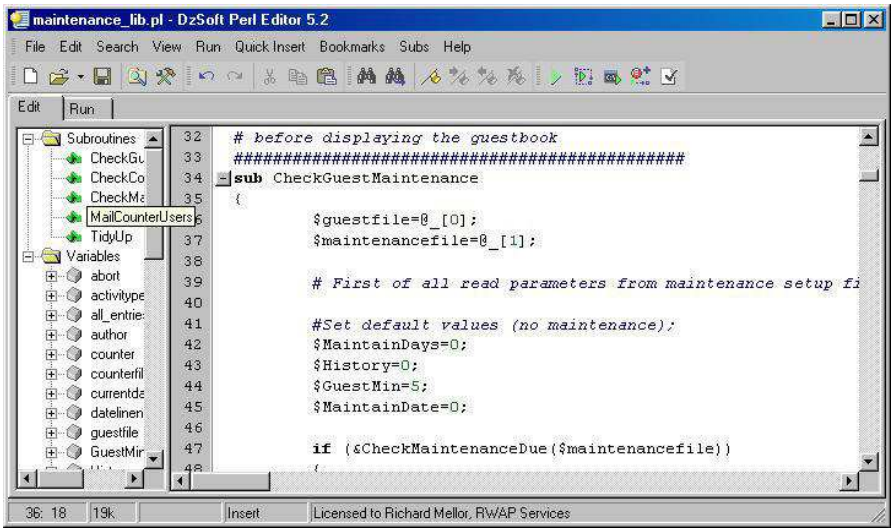
2.3.2 Emacs

Emacs-editori on suhteellisen monipuolinen kuvaruutueditori, jolla voi muokata tekstiä, etsiä haluttua merkkijonoa sekä tehdä automaattisia korjauksia tekstiin. Emacs-editori ei kuitenkaan sisällä tavutusta eikä tekstin tasaamista, joten suurempia tekstinkäsittelyä vaativia dokumentteja ei välttämättä kannata Emacs-editorillakaan tehdä. Kuvassa 2 on esitetty Emacs-editorin käyttöliittymä.

Kuva 2. Emacs-editori [4]

2.3.3 Perl

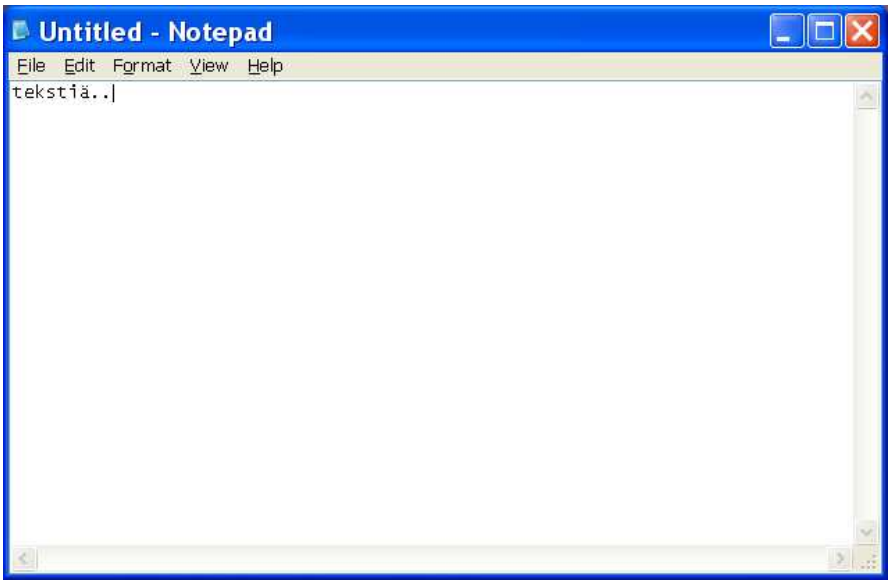
Perl-editori soveltuu puolestaan loistavasti suurempienkin scriptien kirjoittamiseen. Ensimmäisellä kerralla kirjoitettaessa voi kirjoittaminen tuntua vaivalloiselta, mutta jatkossa scriptiä voi muokata yksinkertaisilla komennoilla ja näin voi välttyä koko scriptin kokonaan korvaamiselta. Kuvassa 3 on esitetty erään Perl-editorin käyttöliittymä.



Kuva 3. Perl-editori [5]

2.3.4 Notepad eli Muistio

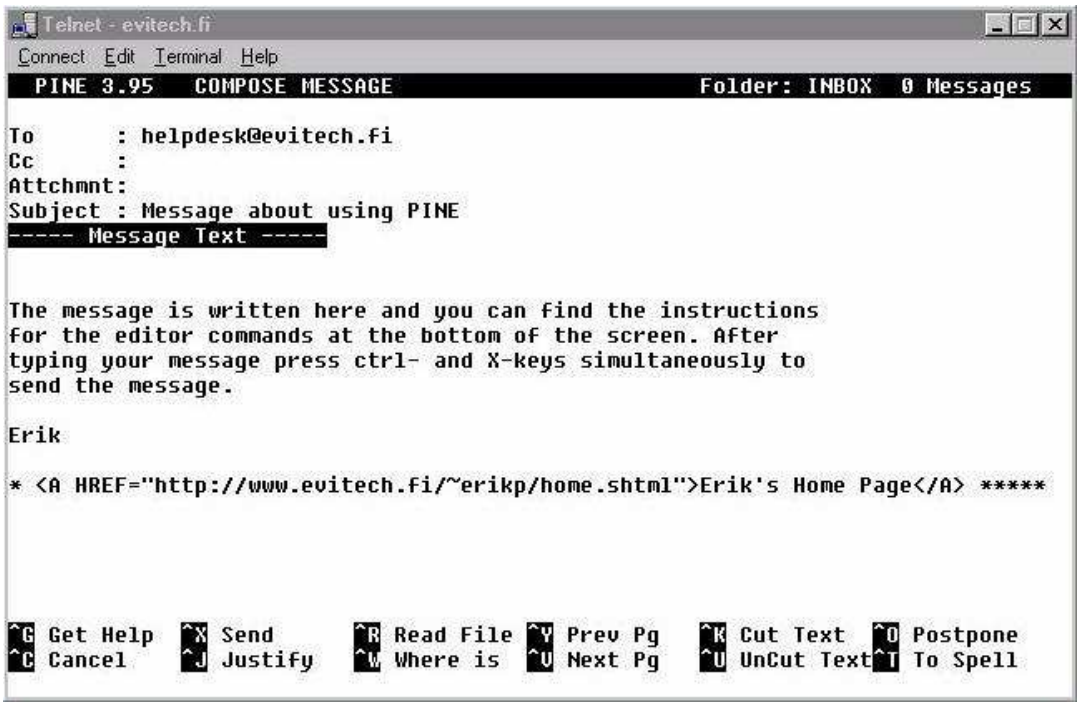
Notepad on Microsoft Windowsin mukana tuleva standardi tekstieditori. Esimerkiksi Notepad-editorilla voi tuottaa html-koodia ja näin tehdä esimerkiksi kotisivuja. Kuvassa 4 on Notepadin eli Muistion käyttöliittymä.



Kuva 4. Muistio-editori.

2.3.5 Pine

Pine on hyvä esimerkki sähköpostiohjelmasta, joka toimii kuten editori. Pine-sähköpostiohjelmalla (Program for Internet News and Email) avulla voidaan lähettää ja vastaanottaa sähköpostiviestejä ja tiedostoja. Pine on monipuolinen sähköpostiohjelma, ja sillä voidaan sähköpostin lähettämisen ja vastaanottamisen lisäksi arkistoida posteja. Kuvassa 5 on esitetty Pinen käyttöliittymä.



Kuva 5. Pine [6]

2.4 Editorien toteuttaminen

Editorit ovat yleensä toiminnoiltaan niin yksinkertaisia, että ne on mahdollista toteuttaa vaikka Assembly-kielellä. Editoreja voidaan siis toteuttaa kaikilla olemassa olevilla ohjelmointikielillä. Esimerkiksi Jedit-editori ja Jext-editori on toteutettu java-ohjelmointikielellä. Jedit-editori on pääosin Slava Pestovin kehittämä, ohjelmoijille tarkoitettu editori. Jext-editori on myös tarkoitettu ohjelmoijille ja se tukee monia ohjelmointikieliä, kuten C, C++, TeX, ja XHTML.

Emacs-editori on toteutettu alun perin Lisp-ohjelmointikielellä. Nykyisissä uudemmissa versiossa on lisäksi käytetty myös C-kieltä. Emacsin lähdekoodi on tietyin ehdoin kaikkien käytettävissä ja muokattavissa.

3 TEKSTINKÄSITTELYOHJELMAT

Tekstinkäsittelyohjelmat ovat nykyajan kirjoituskoneita, joilla tuotetaan ensisijaisesti tulostettavaksi tarkoitettuja tekstejä. Tekstinkäsittelyohjelmilla tekstin ulkoasua voi muokata monipuolisesti ja tekstin sekaan voi myös asetella kuvia ja muuta grafiikkaa. Tämän vuoksi ohjelmilla tehdyt dokumentit ovat tavallisia tekstitiedostoja huomattavasti suurempia, ja tiedosto on luettavissa ja muokattavissa vain samalla ohjelmalla kuin millä se on tuotettukin. Tekstinkäsittelyohjelmilla on usein omat tallennusformaattinsa, joten yhdellä teksturilla tuotettu dokumentti ei välttämättä ole luettavissa toisella. Tosin tekstinkäsittelyohjelmilla voidaan tekstit tallentaa myös ASCII-muodossa ilman, että se sisältää erikoiskomentoja. Tämä täytyy huomioida tekstin tallennusvaiheessa. [1.]

Useimmat tekstinkäsittelyohjelmat, varsinkin MS Windows -ympäristöstä puhuttaessa, ovat nykyään WYSIWYG-tyyppisiä. WYSIWYG tulee sanoista What You See Is What You Get, eli asiakirjaa voi tarkastella näytöllä lähes siinä muodossa kuin se paperillekin tulostuisi. Vanhat MS-DOS-tekstinkäsittelyohjelmat eivät yleensä tätä mahdollisuutta tarjonneet. [3.]

Nykyiset tekstinkäsittelyohjelmat näyttävät tekstin jo ruudulla kirjoitettaessa siinä muodossa, kuin se on tulostettaessa. Tekstinkäsittelyohjelmat asettuvatkin ominaisuuksiltaan tekstieditorien ja taitto-ohjelmien väliin, koska niillä voidaan sekä tuottaa tekstiä että saattaa se painettavaan kuntoon. [1.]

3.1 Tekstinkäsittelyohjelmien ominaisuuksia

Tekstinkäsittelyohjelmien, kuten Microsoft Word, tärkeimpiä ominaisuuksia on tekstin tuottamisen lisäksi tekstin muokattavuus. Tekstinkäsittelyohjelmat sisältävät lukuisia tekstin muokkaamiseen tarkoitettuja ominaisuuksia, jotka helpottavat käyttäjän tekstin tuottamista.

Microsoft Word sisältää kopiointiin, leikkaamiseen ja liittämiseen liittyvät toiminnot, mutta lisäksi ohjelmalla voi muun muassa tavuttaa, tarkistaa ja korjata tekstiä automaattisesti. Lisäksi Microsoft Word mahdollistaa kappalejakojen tekemisen, sisällysluettelon automaattisen tekemisen, sivunumeroinnin sekä ylä- ja alatunnisteiden käyttämisen. Microsoft Word sisältää myös erilaisia asiakirjamalleja, joiden pohjalta käyttäjä voi helposti luoda oman asiakirjansa.

Tekstinkäsittelyohjelmat mahdollistavat myös erilaisten graafisten esitysten liittämisen tekstiin. Myös kuvat ja taulukot ovat liitettävissä tekstin yhteyteen.

3.2 Tekstinkäsittelyohjelmien toteuttaminen

Tekstinkäsittelyohjelmista puhuttaessa tärkeimpiä seikkoja niiden tuottamisessa ovat Lisp ja linkitetyt listat. Muun muassa Microsoft Wordin toteuttamisessa on käytetty hyväksi yhteen suuntaan linkitettyjä listoja.

3.2.1 Lisp-ohjelmointi

Lisp-ohjelmointi on John McCarthyn 1950-luvun loppupuolella kehittämä ohjelmointikieli. Lisp-ohjelmointikielen sanotaan C-kielen ohella olevan pohja nykyisen ohjelmoinnin eri suuntauksille. Nykyajan ohjelmointikielet muistuttavat Lisp-kieltä koko ajan enenevissä määrin.

Lisp-kielen mukana McCarthy toi esiin uusia ohjelmointiominaisuuksia. Yksi niistä oli ehtolauseet, joita tänä päivänä pidetään ohjelmoinnissa itsestäänselvyytenä. Myös pointtereiden käyttäminen sekä muistiroskien siivoaminen olivat Lisp-kielen mukana tulleita uusia ominaisuuksia. Funktiokutsujen avulla rakennetut ohjelmat tulivat myös Lisp-kielen kehityksen mukana. [7.]

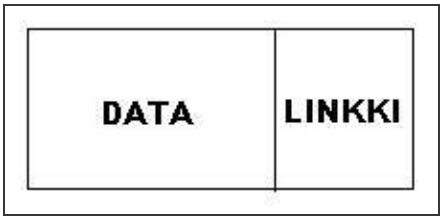
Lisp on niin sanottu funktionaalinen ohjelmointikieli. Se perustuu listarakenteiden algebraan, lambda-kalkyyliin ja rekursiivisten funktioiden teoriaan. Funktionaalisen ohjelmoinnin lisäksi Lispissä voidaan käyttää tavanomaista peräkkäisiin käskyihin perustuvaa ohjelmointia sijoitus-, hyppy- ja erilaisine toistokäskyineen. Makro-ohjelmoinnin avulla voidaan kieleen ohjelmoida uusia rakenteita tai toteuttaa kokonaan uusia kieliä. [8.]

Ohjelman suorituksesta eli funktioiden kutsujen evaluoinnista huolehtii ilmausten redusointiin kykenevä ohjelma eli Lisp-tulkki. Tulkki redusoi käyttäjän määrittelemiä funktioita sisältä ulos ja suorittaa erikoisreduktion niin sanottujen erikoisten funktioiden yhteydessä. [9.]

Lisp-kielen hyviä puolia ovat syntaksin yksinkertaisuus, tyypittömyys, tulkitseva suoritus ja standardin puute. Toisaalta nämä voidaan tulkita myös Lisp-kielen huonoiksi ominaisuuksiksi.

3.2.2 Linkitetyt listat

Linkitetty lista on ohjelmoinnissa käytettävä rakenne, jossa käytetään hyväksi alkioita ja linkkejä. Linkitetty lista on ikään kuin jono alkioita ja linkkejä, jotka osoittavat seuraavaan tai edelliseen alkioon. Kuvassa 6 on kuvattu alkiota, jossa on sekä dataa että linkki.



Kuva 6. Alkio.

Linkitetty lista on monessa suhteessa taulukkoa joustavampi lineaarinen tietorakenne. Listan alkioissa on tallennetun datan lisäksi osoitin listan seuraavaan alkioon. Listan kokoa ei ole rajoitettu ennalta. Listaan voidaan lisätä alkioita mihin paikkaan vain ja siitä voidaan poistaa alkioita mistä paikasta vain. Lisäksi alkioiden paikkaa voidaan vaihtaa. Luonnollisesti myös alkioiden sisältöjä voidaan vaihtaa keskenään. Linkitetyn listan suuri rajoitus on se, että alkioita voidaan käydä läpi vain järjestyksessä. Eli jotta voitaisiin käsitellä tiettyä alkiota, on se haettava käymällä lista läpi alusta siihen saakka. Linkitetty lista voidaan toteuttaa eri tavoilla. Tyypillisesti toteutuksessa käytetään osoitinmuuttujia ja dynaamista muistinvarausta. Toteutus voidaan kuitenkin tehdä taulukoilla simuloimalla osoittimia. [10.]

Linkitetty lista rakenteena voi viedä enemmän muistitilaa, koska se tarvitsee tilaa myös linkeille. Tämä voi olla monesti tekijä, jonka takia valitaan käyttöön jokin toinen rakenne. Kuitenkin usein on niin, että alkio ei tule täyteen tiedosta, jota siihen tallennetaan, joten usein tilaa ei tarvita sen enempää kuin muita rakenteita käyttämälläkään. Linkitettyjä listoja käytettäessä on myös mahdollista yhdistää yhteen alkioon useiden eri alkioiden dataa ja näin ollen tarvitaan vain yksi linkki, jolla voidaan osoittaa useampaan dataan. [11.]

Listarakenteen etuna on se, että listan kokoa voidaan muuttaa dynaamisesti, jolloin tilaa ei tarvitse varata etukäteen. Näin ollen myöskään ohjelman rakenteeseen ei tarvitse ohjelmallisia muutoksia, vaikka alkioiden määrä muuttuisikin. Linkitetyn listan rakenteen käyttömahdollisuus sisältyy useimpiin ohjelmointikieliin.

Linkitettyjä listoja käytettäessä lisääminen ja poistaminen listasta on helppoa, vain kahden linkin osoitetta on muutettava. Myös kahden tai useamman listan yhteen linkittäminen käy helposti.

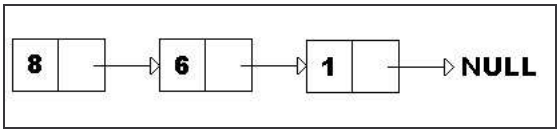
Linkitettyjä listoja käytettäessä on myös omat riskinsä. Linkitettyjä listoja käytettäessä on tärkeätä määritellä kaikki mahdolliset olosuhteet huolellisesti, erityisesti tapaus, että lista on tyhjä. Tämä onkin yksi yleisimmistä ohjelmointivirheistä linkitettyjä listoja käytettäessä, eli tyhjiä listoja ei osata käsitellä oikein. [11.]

Toinen hyvin yleinen virhe on unohtaa vaihtaa linkit, kun listan rakennetta muokataan. Esimerkiksi kun lisätään tai poistetaan listasta alkioita, tulisi muistaa laittaa linkit oikein. Tämän virheen välttämiseksi tulisi piirtää tarkat kaaviot listan rakenteesta ja linkkien osoitteista, jotta rakennetta muokattaessa nähtäisiin, minkä linkkien osoitteita on vaihdettava. Kaaviot tulee sitten piirtää uudestaan, jotta muutokset olisivat taas selkeästi nähtävillä kaavioista ja mahdolliset seuraavat muutokset olisivat mahdollisia. [11.]

Linkitetyillä listoilla on monta erilaista rakennetta: Yhteen suuntaan linkitetty lista, kahteen suuntaan linkitetty lista sekä yksi- ja kaksisuuntainen rengas.

Yhteen suuntaan linkitetty lista

Yhteen suuntaan linkitetyssä listassa on kaksi osaa: Osoitin seuraavaan alkioon sekä itse alkion tieto. Yhteen suuntaan linkitetyn listan rakenne on siis kuvan 7 kaltainen. [12.]



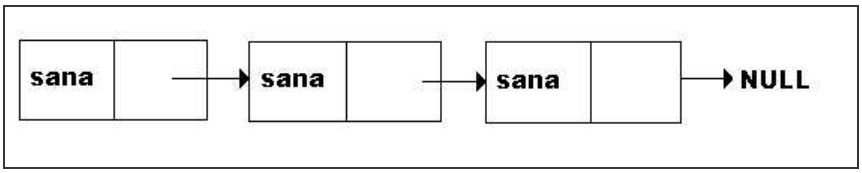
Kuva 7. Yhteen suuntaan linkitetyn listan rakenne [12]

Tekstinkäsittelyohjelmista esimerkiksi Microsoft Word -tekstinkäsittelyohjelma käyttää yksisuuntaista listaa. Käyttäjän lisätessä merkkejä syntyy yksisuuntainen lista, jossa edellisen merkin osoitin osoittaa seuraavaan merkkiin ja näin ollen kehittyy lista, jossa sanat ovat linkitetty toisiinsa.

Merkkien poistaminen yhteen suuntaan linkitetystä listasta on helppoa. Vain linkkien osoittamat osoitteet on vaihdettava.

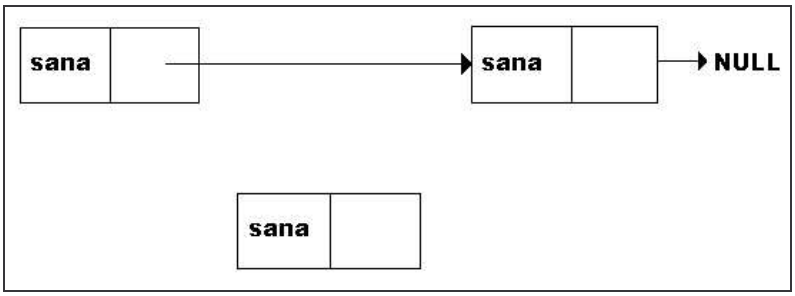
Kun merkkejä poistetaan tekstin keskeltä, syntyy tilanne, jossa osoittimien on osoitettava uuteen paikkaan ja poistettu sana poistuu listasta. Näin poistetut merkit jäävät ikään kuin ilmaan leijumaan, koska ne eivät enää osoita mihinkään, eikä mikään osoitin osoita niihin.

Ensin on siis kirjoitettua tekstiä, jossa jokaisesta merkistä on osoitin seuraavaan merkkiin. Kirjoitetuista merkeistä muodostuu yksisuuntainen linkitetty lista, kuten kuva 8 esittää.



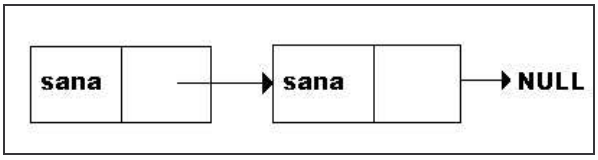
Kuva 8. Tekstin muodostamat linkit.

Tämän jälkeen poistetaan kirjoitetusta tekstistä sana, jolloin osoittimien osoittamat osoitteet muuttuvat ja poistettu sana poistuu listasta kuvan 9 mukaisesti. Poistettu sana ei siis osoita mihinkään, eikä mikään osoita siihen. Sanasta on tullut muistiroska. Kuva 9 kuvastaa tilannetta, jossa sana on muistiroskana.



Kuva 9. Poistetaan sana tekstistä.

Jotta sana ei jäisi niin sanotusti ilmaan leijumaan, on olemassa eräänlaisia roskan keräämisohjelmia, jotka huolehtivat turhien merkkien ja sanojen, eli muistiroskien, poistamisesta. Tämä roskien poistaminen tapahtuu tekstinkäsittelyohjelmissa yleensä viimeistään tallennusvaiheessa. Tällaisia roskankeräämisohjelmia kutsutaan Garbage-collection-ohjelmiksi. Tallennusvaiheessa listat myös järjestellään uudestaan, koska kirjoittaja on voinut tehdä useitakin muutoksia tekstiinsä. Kun käyttäjä tallentaa tekstin, listat järjestellään siistiksi niin, että edellisen merkin osoitin osoittaa seuraavaa merkkiä. Kuva 10 kuvastaa tilannetta, jossa poiston jälkeen linkit on järjestelty uudelleen ja poistettu sana ei enää ole listassa.

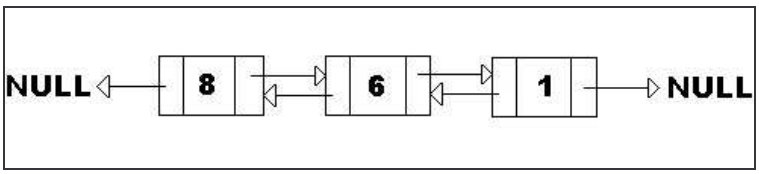


Kuva 10. Linkitetty lista sanan poiston jälkeen.

Kahteen suuntaan linkitetty lista

Kahteen suuntaa linkitetyssä listassa kukin alkio osoittaa sekä edelliseen että seuraavaan listan alkioon. Kahteen suuntaan linkitetyllä listalla on huomattavia etuja muun muassa poistettaessa alkioita. Kaksisuuntaisen listan edut tulevat esille myös silloin, kun alkiot on pystyttävä lukemaan käänteisessä järjestyksessä. Kaksisuuntaista listaa kutsutaan usein myös pakaksi. Kahteen suuntaan linkitetyn listan rakenne on esitetty kuvassa 11.

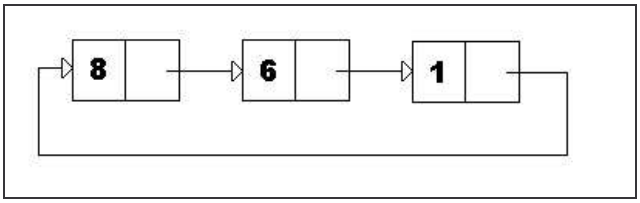
Kahteen suuntaan linkitetyn listan alkioilla on siis kaksi linkkiä, joista toinen osoittaa seuraavaan alkioon ja toinen edelliseen. Usein linkkejä kutsutaankin nimillä LLINK ja RLINK. LLINK nimitys tulee englannin kielestä eli left link, joka siis tarkoittaa, että linkki osoittaa vasemmalle eli edelliseen alkioon. RLINK puolestaan tulee sanoista right link, eli linkki osoittaa oikealle eli seuraavaan alkioon. [11.]



Kuva 11. Kahteen suuntaan linkitetyn listan rakenne.

Yksisuuntainen rengaslista

Yksisuuntaisessa renkaassa viimeisen alkion linkki osoittaa takaisin ensimmäiseen alkioon eikä siis ole tyhjä (NULL). Yksisuuntaisen renkaan rakenne on kuvattu kuvassa 12. Listan käsittelyn voi aloittaa mistä alkiosta tahansa, eikä listaa käyttäessä tarvitse miettiä, että sillä olisi alku ja loppu.



Kuva 12. Yksisuuntaisen renkaan rakenne.

Rengaslistaa voidaan siis käyttää renkaan muotoisten rakenteiden toteuttamiseen, mutta sitä voidaan käyttää myös lineaarisiin rakenteisiin. Rengaslista, jossa yksi linkki osoittaa viimeistä alkiota vastaa lineaarista listaa, jossa kaksi linkkiä osoittaa sekä alkuun että loppuun. [11.]

Koska rengaslistoissa ei ole loppua vaan viimeinen linkki osoittaa takaisin alkuun, voidaan rengaslistoihin laittaa tietyin välein tietynlaisia tunnistettavia alkioita. Nämä alkiot toimivat käytännöllisinä pysähdyspaikkoina. Näitä alkioita kutsutaan List Headiksi, eli listan aluksi. Jokaisella rengaslistalla tulisikin olla ainoastaan yksi alkio, jonka nimi on List Head. Etuna tästä on, että näin lista ei koskaan voi olla tyhjä. [11.]

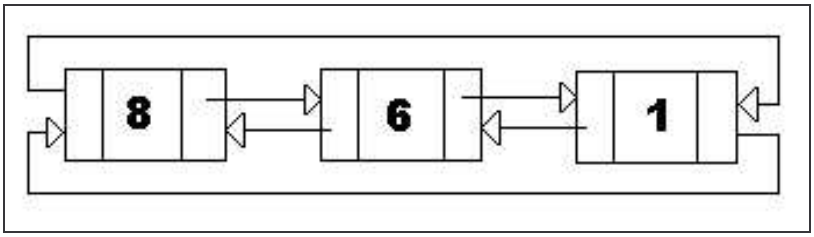
Rengaslistaa voidaan käyttää sekä jonona että pinona. Pino on järjestetty alkiolista, jossa alkioiden lisäykset ja poistot suoritetaan aina pinon päältä. Se on siten lautaspinoa vastaava abstrakti tietorakenne, missä lautasta ei saa ottaa pinon keskeltä tai pohjalta. Viimeksi pinoon lisätty alkio poistetaan siitä ensimmäisenä, joten pino toimii LIFO-periaatteella (Last In First Out). Ohjelmat käyttävät käynnistysvaiheessa varattua systeemipinoa niin sanotun pinokehysten (stack frame) tallentamiseen. Pinokehys tarkoittaa funktiokutsusta systeemipinoon talletettavaa tietorakennetta, joka sisältää osoittimen edelliseen pinokehykseen, paluuosoitteen sekä mahdolliset paikalliset muuttujat. Ohjelmoijan luomat omat pinotietorakenteet eivät liity mitenkään automaattisesti käytettävään systeemipinoon. [13.]

Jono on järjestetty alkiolista, jossa alkioiden lisäykset tehdään vastakkaiseen päähän kuin mistä alkiot poistetaan. Jonossa ensimmäiseksi jonoon lisätty alkio poistetaan siitä ensimmäisenä, joten se toimii FIFO-periaatteella (First In First Out). Renkaana käsiteltävä jono täyttyy vain siinä tilanteessa, että jonoon yritetään lisätä liian monta alkiota. Jonoon mahtuu ALKIOITA - 1 alkiota, koska jonon tyhjyys ja täyttyminen on pystyttävä erottamaan toisistaan. [13.]

Kaksisuuntainen rengaslista

Kaksisuuntainen rengaslista on rakenteeltaan kuvan 13 kaltainen. Kussakin alkiossa on kaksi linkkiä, jotka osoittavat seuraavaan sekä edelliseen alkioon. Lisäksi rengaslistalle tyypillisesti viimeisen alkion linkki ei ole tyhjä, vaan osoittaa listan alkuun eli List Headiin, josta voidaan käyttää myös nimitystä pääalkio. Usein rengaslistaan lisätään myös häntäalkio, johon ei talleteta varsinaista tietoa vaan se toimii hännän osoittajana, kuten pääalkio toimii listan alun osoittajana.

Kaksisuuntainen rengaslista, kuten myös kahteen suuntaan linkitetty lista, vie hieman enemmän muistitilaa kuin yksisuuntainen rengaslista, koska linkkejä on kaksi. Usein alkiossa on kuitenkin tilaa jopa kahdelle linkille, joten monesti tämä ei vaikuta rakenteen valintaan. Kaksisuuntainen rengaslista on kuitenkin ominaisuuksiltaan niin paljon muita rakenteita monipuolisempi, että usein edut voittavat haitat. [11.]



Kuva 13. Kaksisuuntaisen renkaan rakenne.

Kaksisuuntaista rengaslistaa käytettäessä alkion poistaminen on erittäin helppoa. Alkion poistaminen onnistuu pelkästään alkion arvon avulla. Kun alkio on poistettu, laitetaan linkit osoittamaan uusiin osoitteisiinsa ja näin alkio on poistettu. Myös alkion lisääminen listaan on samalla tavalla helppoa. [11.]

4 TEKSTINKÄSITTELYOHJELMIEN JA EDITORIEN VÄLISIÄ EROJA

Tekstinkäsittelyohjelmien ja tekstieditorien välillä on monia hyvinkin suuria eroja. Tekstinkäsittelyohjelmat ovat yleensä kooltaan paljon tekstieditoreja suurempia, koska tekstinkäsittelyohjelmat tarjoavat käyttäjälle enemmän tekstinkäsittelyominaisuuksia. Tekstieditorit ovat yleensä vain muutaman kilotavun kokoisia ja ne tarjoavat käyttäjälle vain ne välttämättömimmät tekstinkäsittelytoiminnot, kuten tekstin lisääminen ja poistaminen. Tekstinkäsittelyohjelmilla sen sijaan käyttäjä pystyy muokkaamaan tekstiä erilaisten tekstinkäsittelyohjelmien tarjoamien toimintojen avulla hyvinkin helposti. Tekstin tasaaminen, tavuttaminen, rivinvälin valinta ja erilaisilla tyyleillä kirjoittaminen ovat vain muutamia esimerkkejä tekstinkäsittelyohjelmien tarjoamista toiminnoista. Joten voisi sanoa, että tekstinkäsittelyohjelmat tarjoavat käyttäjälle paljon enemmän erilaisia toimintoja, mutta vievät myös enemmän tilaa. Joskus käyttäjät voivat jopa turhautua tekstinkäsittelyohjelmien tarjoamiin automaattisiin toimintoihin ja vaihtaa yksinkertaisempaan eli tekstieditoriin, joka ei tarjoa liikaa ominaisuuksia käyttäjälle.

Tekstinkäsittelyohjelmien ja tekstieditorien toteuttamisessakin on runsaasti eroja. Tekstieditorit ovat niin yksinkertaisia rakenteeltaan, että ne voidaan toteuttaa esimerkiksi Assembly-ohjelmointikielellä tai C-kielellä. Tekstinkäsittelyohjelmat sen sijaan vaativat jo sen verran vaativia rakenteita, ettei niitä yleensä näillä kielillä toteuteta. Tekstinkäsittelyohjelmien rakenteen luomiseksi käytetäänkin yleensä linkitettyjä listoja, joka mahdollista tekstin helpon muokkaamisen.

Esimerkiksi voisi mainita tekstin siirtämisessä näkyvät erot. Tekstinkäsittely-ohjelmat käyttävät tekstin siirtämiseen linkitettyä listaa eli kun tekstistä halutaan siirtää yksi sana eri paikkaan, niin ohjelmallisesti tekstinkäsittelyohjelman tarvitsee vain muuttaa sanan osoitin osoittamaan uuteen paikkaan, sekä järjestellä lista uudestaan. Kun taas tekstieditorissa tekstin siirtäminen toteutetaan siten, että koko tekstiä siirretään merkki kerrallaan jos tekstistä poistetaan yksikin merkki.

5 ASCII-EDITORIN SUUNNITTELU JA TOTEUTUS

Tavoitteena oli toteuttaa pieni merkkieditori Kajaanin ammattikorkeakoulun 8051:n kortille. Editorin tuli sisältää tekstinkäsittelyn perustoiminnot, ja sen koon tuli olla korkeintaan pari kilotavua. Tarvittavat sarjaportin kirjoitus- ja lukufunktiot olivat valmiina, joten niitä ei tarvinnut toteuttaa.

5.1 ASCII-editorin suunnittelu

Ensimmäiseksi alettiin suunnitella editorin sisältämiä toimintoja. Kun toimintojen lista oli suunniteltu, alettiin miettiä editorin toteutusta. Ensimmäisenä täytyi valita ohjelmointikieli, jolla editori toteutettaisiin. Tämän jälkeen voitiin alkaa suunnittelemaan toimintojen toteuttamista kyseisellä kielellä.

Editoriin suunnitellut toiminnot on listattu liitteessä A olevassa taulukossa 1.

5.2 ASCII-editorin toteuttaminen

Ensimmäiseksi päätettiin, että editori toteutettaisiin C-kielellä. Tämän jälkeen alettiin suunnittelemaan editoriin tulevien toimintojen toteuttamista. Aluksi suunniteltiin, kuinka merkkijonon siirtäminen, poistaminen sekä korvaaminen toisella merkkijonolla toteutettaisiin. Kun näiden toimintojen toteuttaminen yksinkertaisesti oli suunniteltu, alettiin suunnitella muiden toimintojen toteuttamista. Toimintojen toteuttamisen jälkeen piti vielä huomioida mahdolliset poikkeukset. Poikkeuksia olivat muun muassa rivinloppumerkkien sekä editorin loppumerkin käsittelyyn liittyvät toiminnot, sillä näitä merkkejä ei missään tapauksessa saa poistaa eikä korvata.

Koska kyseessä on yksinkertainen merkkieditori, toiminnot olivat luonteeltaan perustoimintoja, kuten lisääminen, poistaminen, tulostaminen sekä erilaiset kursorin siirtämiseen liittyvät toiminnot. Näiden toteuttamisessa tuli ottaa huomioon editorin pieni koko. Editorin toteutus ja kommentoitu lähdekoodi ovat liitteessä B.

Ohjelman kuvaus

Kun editori käynnistetään, valitaan ensimmäiseksi toiminto. Toimintojen listan saa esiin help toiminnon avulla eli painamalla näppäintä h. Kun valitaan I eli insert, voidaan editorilla kirjoittaa. Näppäintä P painamalla tulostetaan editorin sisällön näytölle. Painamalla ?-näppäintä saadaan esiin merkkipointterin paikka. Nuolinäppäimillä < ja > voidaan liikkua niiden osoittamiin suuntiin. Editorin alkuun päästään t-näppäintä painamalla ja tekstin loppuun päästään painamalla b-näppäintä. Lisäksi yhden rivin ylöspäin pääsee näppäimellä u ja seuraavalle riville puolestaan näppäimellä d. Rivin alkuun pääsee painamalla näppäintä : ja rivin loppuun pääsee painamalla .-näppäintä. Merkkejä voi etsiä painamalla näppäintä f. Merkki poistetaan puolestaan näppäimellä x. Jäljellä olevan RAM-muistin koon saa esiin =-näppäimellä. Editorin käyttö lopetetaan valitsemalla q.

Ohjelman vuokaavio on liitteessä C.

Editoria varten on olemassa vain yksi prosessorikortti, joten testaaminen suoritettiin tällä kortilla. Ensimmäiseksi lisättiin sarjaportin kautta tapahtuvan liikenteen vuoksi kirjoitus- ja lukufunktiot, joilla sarjaportin kautta tapahtuva liikenne tapahtuu merkki kerrallaan. Testausvaiheessa jokainen toteutettu funktio testattiin erikseen. Testausvaiheessa ei havaittu virheitä, eli editori toimi moitteettomasti.

6 YHTEENVETO

Työn tavoitteena oli tutustua tekstinkäsittelyohjelmiin ja editoreihin sekä niiden toteuttamiseen. Lisäksi tavoitteena oli toteuttaa ASCII-editori. Tekstinkäsittelyohjelmien ja editorien ominaisuuksissa oli paljonkin eroja. Tekstinkäsittelyohjelmat muun muassa sisältävät huomattavasti enemmän tekstin muokkaamista helpottavia toimintoja, kun taas editorissa on vain välttämättömimmät tekstin tuottamista helpottavat toiminnot. Lisäksi tekstinkäsittelyohjelmat vievät huomattavasti enemmän tilaa tietokoneelta kuin editorit, jotka ovat yleensä vain muutamien kilotavujen kokoisia.

Koska tekstinkäsittelyohjelmilla on niin paljon enemmän erilaisia ominaisuuksia, on myös sen tekeminen huomattavasti editoria vaikeampaa. Editorit voidaankin toteuttaa lähes millä tahansa ohjelmointikielellä, kun taas tekstinkäsittelyohjelmien toteuttamisessa yleensä suosittavat vaativammat rakenteet, kuten linkitetyt listat, jo sanelevat sen, ettei ohjelmia voi luoda alkeellisimmilla ohjelmointikielillä.

Editorien toteuttamiseen tutustuttiin lähemmin tekemällä itse ASCII-editori Kajaanin ammattikorkeakoululle. Editori toteutettiin C-kielellä, vaikka sen yksinkertaisuuden vuoksi se olisi voitu toteuttaa myös suoraan konekielellä. Editori toimi tarkoituksensa mukaisesti, eikä testausvaiheessa ilmennyt ongelmia.

Työn suorittamisessa ei ilmennyt suuria ongelmia. Tiedon kerääminen oli yllättävän aikaa vievää ja vaikeaa, koska asiasta ei ole niin paljon tietoa kuin voisi kuvitella.

LÄHDELUETTELO

1. Valtiotieteellisen tiedekunnan tieto- ja viestintätekniikan johdantokurssi. Päivitetty syksyllä 2003. [www-dokumentti]. http://www.valt.helsinki.fi/kurssit/tvt/johdanto/tekstinkasittely.htm
2. Lingvistisen atk:n perusteet kurssimateriaalia, tekijät Risto Widenius, Jyrki Niemi. Helsingin yliopisto, yleisen kielitieteen laitos. Päivitetty 2.12.1994. [www-dokumentti]. http://www.ling.helsinki.fi/~janiemi/ctl531/moniste/node7.html
3. Johdatus työvälineohjelmistoihin, Helsingin yliopiston avoin yliopisto. Sisällöstä vastaa tietojenkäsittelytieteen lehtori Kai Korpimies. Päivitetty 10.9.2002.[www-dokumentti]. http://www.avoin.helsinki.fi/kurssit/tkttityo/avoin/
4. Tietotekniikan peruskurssi Tampereen yliopisto. Tekijät Kirsti Ala-Mutka, Matti Rintala, Vespe Savikko ja Jarmo Palviainen. Päivitetty 4.2.2002. [www-dokumentti]. http://www.cs.tut.fi/etaopetus/titepk/index.html#sisallys
5. Dz Soft Perl Editor tekijä ja ylläpitäjä Rich Mellor. Päivitetty 18. joulukuuta 2003.[www-dokumentti]. http://www.rwapservices.co.uk/reviews/DzPerl.html
6. Pine-sähköpostiohjelman käyttöohjeet. Päivitetty 27.10.1997. [www-dokumentti]. http://support.evtek.fi/EN/INST/NETWORK/pine.html
7. Paul Grahamin kotisivut. Päivitetty toukokuussa 2001. [www-dokumentti]. http://www.paulgraham.com/rootsoflisp.html
8. Eero Hyvönen, Jouko Seppänen: Lisp-maailma. Helsinki: Kirjayhtymä, 1987. ISBN 951-26-2834-1.
9. Timo Knuutila, Funktionaalinen ohjelmointi. Päivitetty 8.1.2004 [www-dokumentti]. http://guardian.cs.utu.fi/knuutila/Courses/okp/luennot/funktionaalinen.pdf

10. Lineaariset perustietorakenteet, päivitetty 1.2.2002. [www-dokumentti]. http://www.cs.hut.fi/Opinnot/T-106.250/materiaali/book/Linkitetty_lista.html
11. Donald E. Knuth: The Art of Computer Programming Fundamental Algorithms. 3. painos syyskuu 1997. ISBN 0-201-89683-4.
12. Kauko Kolehmainen: C++ expert algoritmeja ja ohjelman ratkaisuja. Helsinki: IT Press, 1999. ISBN 951-826-044-3.
13. Lappeenrannan teknillinen yliopisto, tietorakenteet ja c-kieli. Tekijä Jarno Mielikainen. Päivitetty 22.2.2004. [www-dokumentti]. http://www.it.lut.fi/kurssit/03-04/010533000/moniste/node18.html

LIITTEET

LIITE A Editorin ominaisuudet taulukko.

LIITE B Lähdekoodi.

LIITE C Editorin vuokaavio.

LIITE A

Taulukko 1. Editorin toiminnot.

Komento	Kuvaus	Kommentti
i	Insert	Tämän jälkeen tulevat merkit kirjoitetaan kursorista alkaen.
p	Print	Kirjoitetaan koko rivi, jolla kursori on.
?	Show Character Pointer	Näytetään merkkipointterin paikka.
<	Move Left	Siirretään merkkipointteria vasemmalle yksi pykälä.
>	Move Right	Siirretään merkkipointteria oikealle yksi pykälä.
t	Top	Siirretään merkkipointteri tekstin alkuun.
b	Bottom	Siirretään merkkipointteri tekstin loppuun.
d	Down	Siirrytään seuraavan rivin alkuun.
u	Up	Siirrytään edellisen rivin alkuun.
;	Start of line	Siirretään merkkipointteri rivin alkuun.
.	End of line	Siirretään merkkipointteri nykyisen rivin loppuun.
f	Find	Etsi annettu merkkijono ja siirrä merkkipointteri merkkijonon alkuun.
x	Delete	Poista merkkipointterin edellä oleva merkki.
q	Quit	Poistutaan editorista.
=	Statistics:	Tulostaa joko tekstin koon tai jäljellä olevan RAM-muistin koon tavuissa.
n	Move	Siirretään tekstiä eteenpäin, käyttäjän syöttämän luvun verran.
h	Help	lyhyt komentojen lista.

LIITE B/1

```
/* ASCII editori tti0s3 Janita Karppinen */

#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>
#define TAULUKON_PITUUS 4000     /*Tässä määritellään tekstialueen
                                        koko*/
#define TRUE 1
#define FALSE 0
#define LUE_MERKKI lue_sarjaportista()
int i,j,p,l,tavut;

char toiminto=0;
char taulukko[TAULUKON_PITUUS] = "";
char *pointteri=NULL;
char *pointteri_taulukon_loppuun;


/*********************************************************************
           Määritellään merkki kerrallaan tulostava metodi,
           jotta sarjaportin kautta tapahtuva liikenne onnistuisi
*********************************************************************/


void merkki_print(char* pointteri)
{
    int i;

    for (i=0;i<(int)strlen(pointteri);i++)
    {

                Kirjoita_sarjaporttiin(pointteri[i]);
    }
}


void tulosta_yksi_merkki( char merkki )
{
           char temp[2];

           temp[0] = merkki;
           temp[1] = 0;                    /* Lopetusmerkki */
           merkki_print(temp);   /* Tulosta merkki ja lopetusmerkki 0*/
}
```

LIITE B/2

```
/*************************************************************************
************************
           Tulostus-metodi tulostaa taulukon sisällön näytölle.
           Tulostus lopetetaan tekstin loppuun eli tyhjää ei tulosteta.

*************************************************************************
**********************/



void tulostus( void )
{
     char temp[2];

     for(i=0;i<TAULUKON_PITUUS;i++)
     {
           if(taulukko[i]==0)
           {
                     break;
           }
           else
           {
                     temp[0]=taulukko[i];
                     temp[1]=0;
                     merkki_print(temp);
           }
     }
}
```

LIITE B/3

```
/*************************************************************************
************************************
Kirjoitus-metodilla käyttäjä kirjoittaa taulukkoon. Kirjoitus
lopetetaan kun käyttäjä painaa kahdesti enter-näppäintä. Kun käyttäjä
painaa kerran enter-näppäintä, asetetaan rivinloppumerkki. Käyttäjälle
ilmoitetaan mikäli kirjoitus tila on loppunut eli ollaan saavuttu
taulukon loppuun.
*************************************************************************
************************************/


void kirjoitus( char *taulukko )
{
      char uusi_merkki;
      char vanha_merkki;
      char poistu;
      uusi_merkki = 0;            /*nollataan uusi merkki*/
      poistu=FALSE;

      while ( poistu==FALSE && *pointteri!=0x03)       /* ollaan tässä
                                                       niin kauan kunnes
                                                       käyttäjä painaa 2
                                                       kertaa ENTER tai
                                                       tullaan taulukon
                                                       loppuun*/
      {
            uusi_merkki = LUE_MERKKI;        /* Luetaan uusi merkki */

            if(vanha_merkki=='\r' && uusi_merkki=='\r')
            {
                        poistu=TRUE;
            }

            vanha_merkki=uusi_merkki;

            if(uusi_merkki=='\r')
            {
                        uusi_merkki=0x0D;     /*Asetetaan rivinloppumerkki
                                              kun painetaan enteriä*/
            }

            if (poistu==FALSE)    /* Onko painettu enteriä kahdesti. Jos
                                  on niin ei tallenneta sitä taulukkoon
                                  vaan poistutaan while-
                                  silmukasta */
            {
                  *pointteri = uusi_merkki;
                                  /*Tallennetaan uusi merkki taulukkoon*/
                  tulosta_yksi_merkki(*pointteri);      /*Tulostetaan
                                                        merkki näytölle*/
                  pointteri ++;                /*Liikutetaan pointteria*/
            }
      }
```

LIITE B/4

```
     if(*pointteri == 0x03)      /*tarkistetaan ettei olla taulukon
                                 lopussa*/
     {
          merkki_print("\nTilaa ei ole!\n");          /*ilmoitetaan
                                                      että tekstiä ei
                                                      voi enää lisätä,
                                                      koska tila
                                                      loppu*/
     }

}



/*************************************************************************
          Vasemmalle-metodilla siirretään kursoria yksi pykälä
          vasemmalle. Mikäli ollaan taulukon alussa, se ei ole
          mahdollista.
*************************************************************************/


int vasemmalle( void )
{
     if( pointteri != &taulukko[0])          /*Varmistetaan ettei olla
                                             editorin alussa*/
     {
          if( *(pointteri-1)!=0x0D)
          {
                    pointteri--;
                    return TRUE;
          }
     }
     return FALSE;

}
```

LIITE B/5

```
/*********************************************************************
           Oikealle-metodilla siirretään kursoria yksi pykälä oikealle.
           Mikäli ollaan taulukon lopussa, se ei ole mahdollista.
*********************************************************************/


int oikealle( void )
{
     if( *pointteri != 0x03)                /*Varmistetaan ettei olla
editorin lopussa*/
     {

           if( *(pointteri+1)!=0x0D)        /* Jos osoitin ei ole rivin
lopussa siirretään kursoria
                                            oikealle*/
           {
                pointteri++;
                return TRUE;
           }

     }
     return FALSE;
}



/*********************************************************************

           Tekstinalkuun-metodilla siirretään kursori tekstin alkuun.

*********************************************************************/


void tekstinAlkuun(  )
{

      pointteri=&taulukko[0];

}
```

LIITE B/6

```
/*********************************************************************

          Tekstinloppuun-metodilla siirretään kursori tekstin loppuun.

*********************************************************************/


void tekstinLoppuun( )
{
      for(i=0;i<TAULUKON_PITUUS; i++)
      {

           if(taulukko[i]==0);
           {
                     pointteri=&taulukko[i];
                     break;               /*Hypätään pois silmukasta
kun
                                          tekstinloppu on löytynyt*/
           }
      }
}



/*********************************************************************

          Rivinalkuun-metodilla siirretään kursori rivin alkuun.

*********************************************************************/

void rivinAlkuun(  )
{
      while (*pointteri != 0x0D && pointteri > taulukko)
           /*Odotellaan kunnes saavutetaan rivinloppumerkki ja
           varmistetaan ettei olla taulukon alussa*/
      {
           pointteri--;
      }

      pointteri++;    /*Lisätään pointterin arvoa jolloin tullaan rivin
                        alkuun*/
}
```

LIITE B/7

```
/*********************************************************************

          Rivinloppuun-metodilla siirretään kursori rivin loppuun.

*********************************************************************/


void rivinLoppuun(  )
{
          while               (*pointteri!=0x0D               &&
pointteri<&taulukko[TAULUKON_PITUUS-1]
          && *pointteri!=0 && *pointteri != 0x03)

                    /*Ollaan tässä kunnes saavutetaan rivinloppu ja
                    varmistetaan ettei olla taulukon lopussa. Ja ei
                    olla tekstinlopussa eikä editorin loppumerkin
                    kohdalla. */
      {
          pointteri++;
      }

}


/*********************************************************************
**************************
          Alas-metodilla siirretään kursori Seuraavan rivin alkuun.

*********************************************************************
*************************/

void alas( )
{
          void rivinLoppuun();

          if( *pointteri != 0x03)          /*Varmistetaan ettei olla
editorin lopussa*/
          {
                    pointteri++;
          }
}
```

LIITE B/8

```
/*********************************************************************

          Ylos-metodilla siirretään kursori edellisen rivinalkuun.

*********************************************************************/

void ylos(  )
{
                              void rivinAlkuun();/*Siirrytään
rivinalkuun, mikäli ei olla
                                                 taulukon alussa*/


          if( pointteri != taulukko)
      {
               pointteri--;
               rivinAlkuun();
      }

}


/*********************************************************************

      Etsi-metodilla etsitään käyttäjän antamaa merkkijonoa. Käyttäjä
      syöttää merkkijonon, jonka jälkeen aletaan etsiä taulukosta
      vastaavaa merkkijonoa. Kun vastaava löytyy, asetetaan kursori
      sanan kohdalle. Jos sanaa ei löydy, ilmoitetaan käyttäjälle ettei
      sanaa löytynyt.

*********************************************************************/

void etsi( char taulukko[] )
{
      int i,j;
      char temp[30];
      char loytyi;
      char uusi_merkki;
      uusi_merkki = 0;
      i=0;            /*Nollataan temp puskurin alkuosoite*/

      while(uusi_merkki != '\r' && i<30)    /*Odotetaan että käyttäjä
                                            painaa enter tai merkkijonon
                                            pituus ylittää 30 merkkiä*/
      {
           uusi_merkki=LUE_MERKKI;

           if(uusi_merkki != '\r')
           {
```

```
            temp[i]=uusi_merkki;
            tulosta_yksi_merkki(temp[i]);
            i++;
            temp[i]=0;
        }
    }
```

LIITE B/9

```
/*etsittävä merkkijono on temp-taulukossa, nyt etsitään tekstiä
taulukosta*/


    for(i=0;i<TAULUKON_PITUUS; i++)
    {
        if(taulukko[i]==temp[0])        /*Etsitään etsityn sanan
                                        ensimmäistä kirjainta*/
        {
            loytyi=TRUE;
            for(j=0; j<(int)strlen(temp); j++)
            {
                if(taulukko[i+j]==temp[j])      /*Verrataan
                                                seuraavia
                                                kirjaimia
                                                toisiinsa*/
                {
                }
                else
                {
                    loytyi=FALSE;    /*Jos kirjaimet eivät ole
                                            samat
                                        asetetaan arvo FALSE*/
                }
            }



            if(loytyi==TRUE)
            {
                pointteri=&taulukko[i];         /*Asetetaan
                                        pointteri  haettavan sanan
                                                kohdalle*/
                merkki_print("\nHakemasi sana loytyi!\n");
                return;

            }

        }
        else
        {
           if(taulukko[i]==0) /*Jos taulukon arvo on nolla, ollaan
                            tekstin lopussa ja etsiminen
                                        lopetetaan*/
                break;
```

```
            }
      }
      merkki_print("\nHakemaasi sanaa ei loytynyt!\n");
}
```

LIITE B/10

```
/*********************************************************************
          Poista-metodilla poistetaan edellinen merkki taulukosta.
          Kuitenkaan rivinloppumerkkiä ja taulukonloppumerkkiä ei saa
          poistaa.
*********************************************************************/

void poista( char taulukko[] )
{
          if(pointteri != taulukko)   /*Varmistetaan     ettei
olla editorin
                                                    alussa*/
      {
          if((*(pointteri-1))==0x0D)       /*Tarkistetaan ettei
                                           kyseessä ole
                                           rivinloppumerkki, koska sitä
                                           ei saa poistaa*/
          {
              *(pointteri-2)=' ';
                    memmove((pointteri-2), pointteri ,
                              pointteri_taulukon_loppuun-
(pointteri));
              pointteri-=2;     /*Siirretään kursoriakin*/
          }
          else
          {
              *(pointteri-1)=' ';
              memmove((pointteri-1), pointteri ,
                    pointteri_taulukon_loppuun-(pointteri));
              pointteri --; /*Siirretään kursoriakin*/
          }
      }
}
```

```
/*************************************************************************

           Tila-metodilla tarkistetaan jäljellä olevan tilan määrä ja
           ilmoitetaan se käyttäjälle.

*************************************************************************/


void tila( char taulukko[] )
{
      char temp[30];
      tavut=0;
      for(i=0;i<TAULUKON_PITUUS;i++)
      {
```



```
           if(taulukko[i]==0)    /*lasketaan kuinka monen alkion arvo on
                                       nolla*/
           {
                     tavut++;
           }
      }

      sprintf(temp, "\nVapaata tilaa %d tavua\n", tavut);
      /*Sijoitetaan tulostettava teksti temp taulukkoon, jotta se
      voidaan tulostaa merkki kerrallaan*/

      merkki_print(temp);

}



/*************************************************************************

           Siirra-metodilla siirretään tekstiä eteenpäin käyttäjä
           syöttämän luvun verran.

*************************************************************************/

void siirra( char taulukko[] )
{
      char temp[10];
      int value;
      char uusi_merkki;
      int i;
      uusi_merkki = 0;
      i=0;            /*Nollataan temp puskurin alkuosoite*/
```

```
while(uusi_merkki != '\r' && i<30)    /*Odotetaan että käyttäjä
                                      painaa enter tai yli 30
                                      merkkiä*/
{
     uusi_merkki=LUE_MERKKI;
     if(uusi_merkki != '\r')
     {
               temp[i]=uusi_merkki;
               tulosta_yksi_merkki(temp[i]);
               i++;
               temp[i]=0;
     }
}

value = atoi(temp);         /*Muutetaan temp-taulukossa oleva
                            numeroteksti luvuksi atoi-metodilla*/
```

LIITE B/12

```
      memmove(pointteri+value, pointteri ,  pointteri_taulukon_loppuun-
                     (pointteri+value));

      /*minne kopioidaan, mistä kopioidaan, kuinka monta tavua
      kopioidaan. Tässä lasketaan kuinka monta tavua on
      kopiointikohdasta taulukon loppuu. Laskun jälkeen osataan
      kopioida aina oikea määrä tavuja */


      memset(pointteri,' ',value);

      for(i=0;i<TAULUKON_PITUUS;i++)
      {
           tulosta_yksi_merkki(taulukko[i]);
      }

}


/*************************************************************************

           Ohjeet-metodilla tulostetaan käyttäjälle lista valittavista
           olevista toiminnoista.

*************************************************************************/


void ohjeet()
{
           merkki_print("i insert\n");
           merkki_print("p print\n");
           merkki_print("? Show character pointer\n");
           merkki_print("< Move left\n");
           merkki_print("> Move right\n");
           merkki_print("t Top of text\n");
           merkki_print("b Bottom of text\n");
           merkki_print("d Down\n");
           merkki_print("u Up\n");
           merkki_print("; Start of line\n");
           merkki_print(". End of line\n");
           merkki_print("f Find\n");
           merkki_print("x Delete\n");
           merkki_print("n Move\n");
           merkki_print("q Quit\n");
           merkki_print("= Statistic\n");
           merkki_print("\n");
}
```

LIITE B/13

```
/*************************************************************************

          Main-metodi sisältää valikon sekä muuttujien määrittelyt.

*************************************************************************/

void main( void)
{
      char toiminto;
      memset(taulukko, 0, TAULUKON_PITUUS);           /*Asetetaan
                                              aluksi kaikkien taulukon
                                              alkioiden arvoksi nolla*/

      pointteri=&taulukko[0];                  /*Määritellään että
                                              pointteri osoittaa taulukon
                                              alkioon*/

      pointteri_taulukon_loppuun = &taulukko[TAULUKON_PITUUS-1];

      *pointteri_taulukon_loppuun=0x03; /*Asetetaan taulukon
                                              loppumerkki*/
      toiminto = 0;
      merkki_print("Anna toiminto\n");

      while(toiminto!='q')
      {
           toiminto=getch();

           switch(toiminto)
           {
                      case 'p':
                            tulostus();
                            break;
                      case 'i':
                            kirjoitus(taulukko);
                            break;
                      case '<':
                            vasemmalle();
                            break;
                      case '>':
                            oikealle();
                            break;
                      case 't':
                            tekstinAlkuun(taulukko);
                            break;
                      case 'b':
                            tekstinLoppuun(taulukko);
                            break;
                      case 'd':
                            alas(taulukko);
                            break;
```

LIITE B/14

```
                        case 'u':
                              ylos(taulukko);
                              break;
                        case ';':
                              rivinAlkuun(taulukko);
                              break;
                        case '.':
                              rivinLoppuun(taulukko);
                              break;
                        case 'f':
                              etsi(taulukko);
                              break;
                        case 'x':
                              poista(taulukko);
                              break;
                        case 'q':
                              break;
                        case '=':
                              tila(taulukko);
                              break;
                        case 'n':
                              siirra(taulukko);
                              break;
                        case 'h':
                              ohjeet();
                              break;
                        default:
                              merkki_print("LAITON KOMENTO!\n");
                              break;

                  }

            }
      }
```

LIITE C

Vuokaavio