Tampereen teknillinen yliopisto
Ohjelmistotekniikan laitos

OHJ-1101
Ohjelmointi Ie

Harjoitustyö
Ataxx
Välipalautus

Tuomo Turunen

9.1.2008 13:15 TB211

Petteri Aimonen (205441)
aimonen
petteri.aimonen@tut.fi

1 Ohjelman rakenne

1.1 Ohjelman tietorakenteet

Pelilauta tallennetaan 7x7 taulukkona, jonka alkioina enumeja. Taulukossa ensimmäinen indeksi on x-koordinaatti, ja jälkimmäinen y-koordinaatti.

```
const int LAUDAN_KOKO = 7;
enum Nappula {TYHJA, VALKOINEN, MUSTA};
Nappula pelilauta[LAUDAN_KOKO][LAUDAN_KOKO] = {{TYHJA}};
```

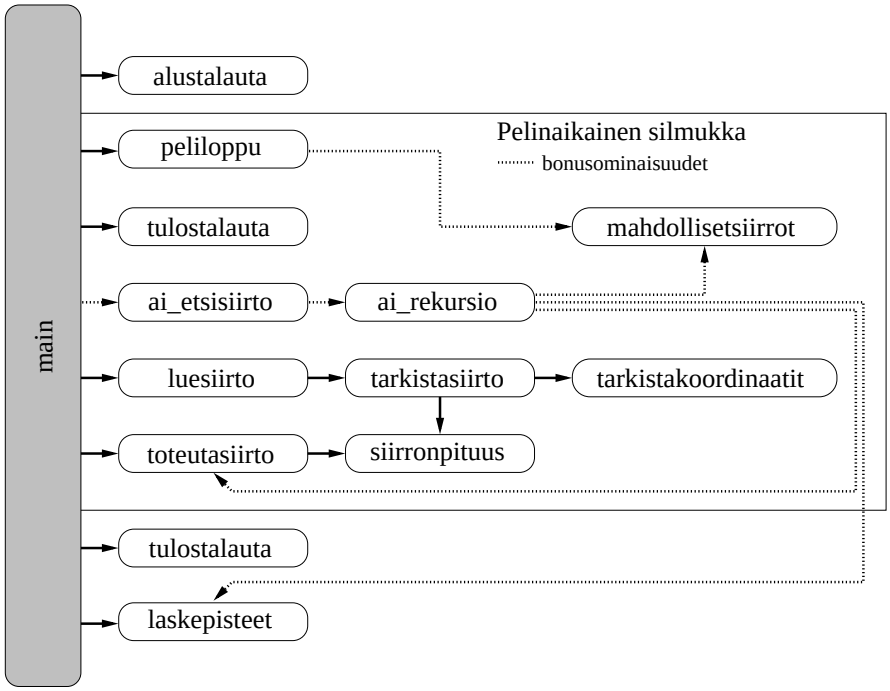
Siirrot kuvataan structilla, jossa on lähtö- ja kohdekoordinaatit. Structissa koordinaatit on tallennettu taulukon indeksoinnin mukaan, eli pienin arvo on 0 ja suurin **LAUDAN_KOKO** - 1.

```
struct Siirto {
    int alkuX, alkuY;
    int loppuX, loppuY;
};
```

Replay-komennot tallennetaan pääohjelmassa määriteltyyn ostringstreamiin sitä mukaa kun komentoja syötetään. Stringistä ne voidaan sitten tallentaa lopuksi tiedostoon, tai jättää tallentamatta.

```
ostringstream replaykomennot("");
```

1.2 Funktioiden kutsukaavio



2 Funktioiden kuvaukset

2.1 Pääohjelma

Ennen peliä pääohjelma kysyy pelaajien nimet, alustaa vuoroksi valkoisen pelaajan ja alustaa pelilaudan **alustalauta**-funktiolla. Pelaajien nimet tallennetaan replaykomentoihin.
Pelin aikana pääohjelma tekee loopissa seuraavat asiat:

1. Tarkistetaan onko peli päättynyt funktion **peliloppu** avulla.
2. Tulostetaan vuorossa olevan pelaajan nimi sekä laudan tilanne.
3. Jos vuorossa oleva pelaaja on tietokonepelaaja, etsitään siirto funktiolla **ai_etsisiirto**. Muussa tapauksessa luetaan siirto. Jos **luesiirto** palauttaa falsen, lopetetaan peli.
4. Toteutetaan siirto.
5. Tallennetaan siirto replaykomentoihin.
6. Vaihdetaan vuorossa olevaa pelaajaa.

Pelin jälkeen tulostetaan lauta ja tarkistetaan tilanne **laskepisteet**-funktiolla. Tulostetaan viesti voittajasta/tasapelistä.
Lopuksi kysytään replay-tiedoston tallennusta. Tiedostoon tallennetaan **replaykomennot**-muuttujan sisältö.

2.2 Funktio: alustalauta

Esittely:
`void alustalauta(Nappula pelilauta[][]);`
Parametrit:
Parametri **pelilauta** on taulukko, johon pelilauta alustetaan.
Paluuarvo:
Funktio ei palauta arvoa.
Toiminta:
Aluksi lauta käydään läpi, ja asetetaan kaikkiin paikkoihin tyhjä. Sitten kulmapaikkoihin asetetaan nappulat, vasempaan alakulmaan ja oikeaan yläkulmaan MUSTA, ja vasempaan yläkulmaan ja oikeaan alakulmaan VALKOINEN.
Käyttö:
Pääohjelma kutsuu funktiota ennen pelin alkua.

2.3 Funktio: tulostalauta

Esittely:
`void tulostalauta(const Nappula pelilauta[][]);`
Parametrit:
Parametri **pelilauta** on taulukko, jossa on pelilaudan tilanne siirtojen tarkistamista varten.
Paluuarvo:
Funktio ei palauta arvoa.
Toiminta:
Aluksi tulostetaan yläakseli. Sen jälkeen taulukko käydään riveittäin läpi, joka rivin alkuun tulostetaan rivinumero ja sen perään laudan paikkoja vastaavat symbolit.
Käyttö:
Pääohjelma kutsuu funktiota ennen siirtoa, sekä pelin loputtua.

2.4 Funktio: luesiirto

Esittely:
`bool luesiirto(const Nappula pelilauta[][], Nappula vuoro, Siirto &tulos);`

Parametrit:
Parametri **pelilauta** on taulukko, jossa on pelilaudan tilanne siirtojen tarkistamista varten.
Parametri **vuoro** on vuorossa olevan pelaajan väri (arvo TYHJA ei ole sallittu).
Parametri **tulos** on Siirto-tyyppinen struct, johon luetut koordinaatit tallennetaan.

Paluuarvo:
Jos syötettiin negatiivinen lähtöruudun x-koordinaatti (= pelin lopetus), palautetaan **false**. Muissa virhetilanteissa tulostetaan virheilmoitus ja kysytään koordinaatteja uudelleen. Onnistuneen syötteen jälkeen palautetaan **true**.

Toiminta:
Koordinaatit kysytään käyttäjältä **cin** ja **cout**-virtojen kautta. Kun koordinaatit on luettu, ne tallennetaan taulukon indekseiksi muutettuina Siirto-structiin. Jos lähtöpaikan x-koordinaattia luettaessa tulos on negatiivinen, palautetaan heti **false**. Muutoin luettu siirto välitetään pelilaudan ja vuoron kanssa funktiolle **tarkistasiirto**.

Jos siirto oli laillinen, palautetaan **true**. Muussa tapauksessa **tarkistasiirto** on tulostanut virheilmoituksen, joten riittää kysyä koordinaatteja uudelleen.

Käyttö:
Pääohjelma kutsuu funktiota loopissa, kunnes peli loppuu.

2.5 Funktio: tarkistasiirto

Esittely:
`bool tarkistasiirto(const Nappula pelilauta[][], Nappula vuoro, const Siirto &siirto);`

Parametrit:
Parametri **pelilauta** on taulukko, jossa on pelilaudan tilanne.
Parametri **vuoro** on vuorossa olevan pelaajan väri (arvo TYHJA ei ole sallittu).
Parametri **siirto** on tarkistettava siirto.

Paluuarvo:
Jos siirto oli laillinen senhetkisessä tilanteessa, palautetaan **true**. Muutoin palautetaan **false**.

Toiminta:

1. Tarkistetaan, että lähtö- ja kohdekoordinaatit ovat laudalla kutsumalla funktiota **tarkistakoordinaatit**. Elleivät olleet, tulostetaan virhe aiheesta ja palautetaan **false**.
2. Tarkistetaan, että lähtöruudussa on nappula.
3. Tarkistetaan, että lähtöruudussa oleva nappula on oma.
4. Tarkistetaan, että kohderuutu on tyhjä.
5. Tarkistetaan, että funktion **siirronpituus** laskema siirtymä kohderuutuun on enintään 2 ruutua (= hyppy).
6. Jos virheitä ei havaittu, palautetaan **true**

Käyttö:
Funktio **luesiirto** kutsuu funktiota siirron tarkistamiseksi.

2.6 Funktio: tarkistakoordinaatit

Esittely:
`bool tarkistakoordinaatit(int x, int y);`

Parametrit:
Parametrit **x** ja **y** ovat koordinaatit, joiden kuuluminen pelilautaan tarkistetaan.

Paluuarvo:
Jos sijainti on pelilaudalla, palautetaan **true**. Muutoin palautetaan **false**.

Toiminta:
Kumpaakin koordinaattia verrataan alarajaan **0** ja ylärajaan **LAUDAN_KOKO** - 1.

Käyttö:
Funktio **tarkistasiirto** tarkistaa lähtöpaikan ja kohdepaikan koordinaatit tällä funktiolla.

2.7 Funktio: siirronpituus

Esittely:
`int siirronpituus(const Siirto &siirto);`

Parametrit:
Parametri **siirto** on siirto, jonka pituus lasketaan.

Paluuarvo:
Paluuarvo on siirron pituus siirron tyypin mukaan seuraavasti: **1** = kloonaus, **2** = hyppy, **muut arvot** = laiton siirto.

Toiminta:
Paluuarvo on suurempi x-koordinaattien ja y-koordinaattien erojen itseisarvoista. Eli x-ero -2, y-ero 1 tuottaa pituuden 2.

Käyttö:
Funktio **tarkistasiirto** käyttää tätä funktiota siirron laillisuuden määrittämiseen ja **toteutasiirto** siirron tyypin määrittämiseen.

2.8 Funktio: toteutasiirto

Esittely:
`void toteutasiirto(Nappula pelilauta[][], const Siirto &siirto);`

Parametrit:
Parametri **pelilauta** on taulukko, jossa on pelilaudan tilanne.
Parametri **siirto** on suoritettava siirto.

Paluuarvo:
Funktio ei palauta arvoa.

Toiminta:
Siirto oletetaan lailliseksi. Lähtöruudussa ollut arvo kirjoitetaan kohderuutuun. Jos siirron pituus on 2 yksikköä, tyhjennetään lähtöruutu. Vierekkäiseen ruutuun siirryttäessä nappula kloonautuu. Sitten vaihdetaan kaikki kohderuudun ympäristössä olevat ei-tyhjät paikat siirretyn nappulan värisiksi.

Käyttö:
Pääohjelma kutsuu funktiota siirron lukemisen jälkeen.

2.9 Funktio: peliloppu

Esittely:
`bool peliloppu(Nappula vuoro, const Nappula pelilauta[][]);`
Parametrit:
Parametri **vuoro** on vuorossa oleva pelaaja.
Parametri **pelilauta** on taulukko, jossa on pelilaudan tilanne.
Paluuarvo:
Jos pelilauta on täysi, sillä on vain yhden pelaajan nappuloita, tai nykyisellä pelaajalla ei ole siirtomahdollisuutta, palautetaan **true**. Muutoin palautetaan **false**.
Toiminta:
Mahdollisten siirtojen määrä tarkistetaan funktiolla **mahdollisetsiirrot**. Jos mahdollisia siirtoja on 0, palautetaan heti **true**.
Tämän jälkeen pelilauta tutkitaan järjestyksessä läpi. Arvo **false** palautetaan heti, kun on löytynyt ainakin yksi kummankin pelaajan nappula, sekä yksi tyhjä ruutu. Ellei laudalta löydy jotain näistä, palautetaan läpikäynnin jälkeen **true**.
Käyttö:
Pääohjelma kutsuu funktiota siirron jälkeen laudan tilanteen tarkistamiseksi.

2.10 Funktio: laskepisteet

Esittely:
`void laskepisteet(int &valkoinen, int &musta, const Nappula pelilauta[][]);`
Parametrit:
Parametrit **valkoinen** ja **musta** ovat viittaukset, joiden kautta laskennan tulos tallennetaan.
Parametri **pelilauta** on taulukko, jossa on pelilaudan tilanne.
Paluuarvo:
Funktio ei palauta arvoa.
Toiminta:
Pelilauta käydään järjestyksessä läpi, ja mustat ja valkoiset nappulat lasketaan.
Käyttö:
Pääohjelma kutsuu funktiota pelin päätyttyä voittajan selvittämiseksi.

2.11 Funktio: mahdollisetsiirrot

Esittely:
`int mahdollisetsiirrot(std::vector<Siirto> &siirrot, Nappula vuoro, const Nappula pelilauta[][]);`
Parametrit:
Parametri **siirrot** on vektori, johon löydetyt siirrot tallennetaan.
Parametri **vuoro** on pelaaja, jonka siirtoja etsitään.
Parametri **pelilauta** on taulukko, jossa on pelilaudan tilanne.
Paluuarvo:
Paluuarvo on löydettyjen siirtojen lukumäärä.
Toiminta:
Pelilauta käydään järjestyksessä läpi. Jokaisen löydetyn ko. pelaajalle kuuluvan nappulan kohdalla tutkitaan siitä joka suuntaan 2 nappulaa. Jokaista tältä alueelta löytynyttä tyhjää ruutua kohti tallennetaan siirto vektoriin ja kasvatetaan laskuria.
Käyttö:
Funktio **peliloppu** tarkistaa tällä funktiolla, voiko pelaaja tehdä siirtoja. Tekoäly käyttää tätä funktiota mahdollisten siirtojen hakemiseen.

2.12 Funktio: ai_etsisiirto

Esittely:
`void ai_etsisiirto(Siirto &tulos, Nappula vuoro, const Nappula pelilauta[][]);`

Parametrit:
Parametri **tulos** on Siirto-structi, johon paras siirtovaihtoehto tallennetaan.
Parametri **vuoro** on pelaaja, jonka siirtoa etsitään.
Parametri **pelilauta** on taulukko, jossa on pelilaudan tilanne.

Paluuarvo:
Funktio ei palauta arvoa.

Toiminta:
Funktio kutsuu funktiota **ai_rekursio** sopivalla alkusyvyydellä, ja löytynyt siirto palautetaan suoraan.

Käyttö:
Pääohjelma käyttää funktiota, jos toinen tai molemmat pelaajat ovat tekoälyn ohjaamia.

2.13 Funktio: ai_rekursio

Funktio etsii parhaan mahdollisen siirron. Periaatteena on käydä siirrot läpi, ja jokaisen siirron kohdalla tutkia, mitä vastapelaaja tekisi sen jälkeen. Kun on tutkittu **syvyys** seuraavaa siirtoa, siirtoketjun hyvyys määritetään pelilaudan pistetilanteesta.

Esittely:
`int ai_rekursio(Siirto &tulos, Nappula vuoro, const Nappula pelilauta[][], int syvyys);`

Parametrit:
Parametrit **tulos**, **vuoro** ja **pelilauta** ovat samat kuin funktiossa **ai_etsisiirto**.
Parametri **syvyys** rajoittaa haun syvyyttä. Jos arvo on 0, rekursio pysähtyy.

Paluuarvo:
Paluuarvo kuvaa löydetyn siirron hyvyyttä siten, että suurempi arvo on parempi siirto ko. pelaajan kannalta.

Toiminta:
Mahdolliset siirrot haetaan funktiolla **mahdollisetsiirrot**. Jos syvyys on 0 tai mahdollisia siirtoja ei ole, lasketaan nappulat funktiolla **laskepisteet**. Palautettava arvo on se, montako nappulaa enemmän pelaajalla **vuoro** on kuin vastapelaajalla.

Jokainen mahdollinen siirto käydään läpi seuraavasti:

1. Muodostetaan kopio pelilaudasta.
2. Toteutetaan siirto pelilaudan kopioon.
3. Kutsutaan **ai_rekursio**:ta vastapelaajan vuorolla, pelilaudan kopiolla ja yhtä pienemmällä syvyydellä.
4. Paluuarvo on tulostilanteen hyvyys vastapelaajan kannalta, eli sen vastaluku on haluttu hyvyysluku.
5. Jos toteutettu siirto tuotti omalta kannalta parhaan tilanteen tähän mennessä, tallennetaan se muuttujaan **tulos**, ja sen hyvyys väliaikaismuuttujaan.

Lopuksi löytyneen parhaan siirron hyvyys palautetaan.

Käyttö:
Funktio **ai_etsisiirto** käyttää tätä funktiota varsinaiseen siirron etsintään.