Ohjelmointi 1 lopputyö

Pankkiautomaatti

Johan Sääskilahti

Ongelma:

Kurssin lopputyön tehtävänä oli luoda simulaatio pankkiautomaatista, josta löytyy pankkiautomaatin toimintoja: (1) rahan nostaminen 20 € ja 50 € seteleinä, (2) tilin saldon tarkastelu, (3) tilin tapahtumien tarkastelu. Pankkiautomaatissa täytyi olla valikoita, joissa käyttäjä pystyi navigoimaan tai lopettamaan ohjelman käytön. Tilin tietokantana toimi tekstitiedosto, jonka alkunimeksi otettiin käyttäjän tilinumero ja josta pin-koodi ja saldo haettiin. Tiedosto rakentui siten, että ensimmäiseltä riviltä löytyi käyttäjän pin-koodi ja toiselta riviltä saldo. Kurssilla tehtiin pienempiä ohjelmointitehtäviä ja kahdeksanosainen kotitehtävä, tämä kokonaisuus, johon kuului:

(1) Kurssin alussa tehty alkeellinen esseemäinen vaatimusmäärittely.

(2) Pankkiautomaatin alkeellinen käyttöliittymä algoritmi, joka kirjoitettiin pseudokoodilla.

(3) käyttöliittymän pseudokoodin ohjelmoiminen C-kielellä.

(4) Vähän kehittyneemmän version ohjelmointi, jossa käytettiin funktioita.

(5) Nostovalikon algoritmin ohjelmoiminen, joka jakoi oikean määrän 20 € ja 50 € seteleitä priorisoiden suurempia seteleitä.

(6) Tilin pin-koodin, tilinumeron ja saldon lukeminen tiedostosta ja saldon muokkaaminen.

(7) Ohjelman testaus ja siihen liittyvän suunnitelman tekeminen, sekä testiaineiston kerääminen ja dokumentointi

(8) Työselostus kokonaisuudesta, toisin sanoen tämä dokumentti.

Ratkaisu:

Ohjelmoimani automaatti koostuu neljästä tiedostosta, joita ovat: (1) “*atm.c”* suoritettava pääohjelma, (2) *“funktiot.c”* apufunktiot erillisenä kirjastotiedostona, (3) “*12345.acc*” tilin tietokanta tiedostona, joka toimii tilin tietokantana, (4) *“headers.h”* josta löytyy

apufunktioiden esittelyt/prototyypit - tarkemmat ja teknisemmät toteutustiedot tiedostojen sisäisinä kommentteina.

(1) atm.c

Pääohjelma josta löytyy ainoastaan käyttäjätietue ja main() funktio. Käyttäjätietueesta löytyy käyttäjän tilinumero, saldo sekä pin-koodi.

Ohjelma pyytää ensin tilinumeroa, jonka jälkeen siirrytään kysymään tilinumeron pin-koodia, jonka oikeellisuus tarkistetaan tilin tietokannasta, toisin sanoen tiliin liitetystä tiedostosta. Kun käyttäjä on kirjautunut onnistuneesti, ohjelma siirtyy valintavalikkoon. Valintavalikossa käyttäjältä pyydetään valintasyötettä ja käyttäjä voi valita yhden viidestä toiminnosta: (1) rahan nosto tililtä, (2) tilin saldon tarkastelu, (3) tilin tapahtumien tarkastelu, (4) tilille rahan tallettaminen - joka on pääasiassa luotu simulaatiota varten, (5) toiminnan lopettaminen. Jokaisen valinnan jälkeen kutsutaan *valiValikko()* funktiota joka kysyy haluaako käyttäjä jatkaa ohjelman käyttöä. Ohjelman selvyyden vuoksi olisi ollut parempi toteuttaa tilinumeron sekä pin-koodin kysyminen funktioina.

(1) Nostovalikossa käyttäjä pystyy nostamaan rahaa tililtä kymmenen euron välein 20 € ja 50 € seteleinä jos tilillä on tarpeeksi rahaa.
(2) Saldovalikossa käyttäjä voi tarkastella saldoaan joka on haettu tietokannasta.

(3) Tapahtumavalikossa on listattu tilin pseudotapahtumia, jotka haetaan näennäisesti tietokannasta. Esimerkiksi tapahtumien tarkastelu ei onnistu, jos tietokantaan ei saada yhteyttä; toisin sanoen tiedoston käsittely epäonnistuu.

(4) Talletusvalikossa tilille voi tallettaa rahaa. Talletusvalikossa ei ole juurikaan ehtoja ja se on ohjelmoitu enemmänkin testaajaa varten, ettei tiedostoa tarvitse manuaalisesti muuttaa.

(5) Lopettaa toiminnot.

(2) funktiot.c

Tiedosto joka toimii kirjastona ja josta löytyy pääohjelmassa käytetyt funktiot.

Void lueMerkkijono(char merkkijono, int pituus), int lueKokonaisluku(void), double lueReaaliluku(void)

Ensiksi löytyy kolme funktiota (*lueMerkkijono(), lueKokonaisluku()* ja *lueReealiluku()*) joita kutsuessa voi turvallisesti ottaa käyttäjäsyötteen. Funktiot on haettu Oulun yliopiston järjestämän Ohjelmointi 1 kurssin vapaasti käytettävistä apuohjelmista, mutta niitä on modifoitu hieman printtien suhteen; tarkempaa tietoa tiedoston sisäisessä dokumentaatiossa. *LueMerkkijono()* parametreinä taulukko johon merkkijono luetaan, sekä taulukon pituus. Funktio *lueKokonaisluku()* palauttaa syötetyn kokonaisluvun int:nä ja *lueReaaliluku()* palauttaa syötetyn luvun doublena.

Void lueRoskat(void)

Funktio tyhjentää puskurin ja on haettu Oulun yliopiston Ohjelmointi 1 kurssimateriaalista.

*Int haePin(char *tiedosto, char *validPin)*

lukee pin-koodin tiedostosta ja palauttaa sen muokkaamalla argumentiksi annettua merkkijonoa. Jos luku onnistuu funktio palauttaa 1 ja sen epäonnistuessa se palauttaa 1. Koska pin löytyy tiedoston ensimmäiseltä riviltä, merkkejä luetaan ja tallennetaan muokattavaan taulukkoon ensimmäiseen rivinvaihtoon asti. Parametreinä tiedosto josta pin haetaan sekä taulukko johon pin tallennetaan.

*Double haeSaldo(char *tiedosto)*

lukee saldon tiedoston toiselta riviltä ja palauttaa sen doublena jos tiedostoon saadaan yhteys; muutoin virhetilanteessa palauttaa –1 ja printtaa virheviestin. Funktio lukee tiedoston merkkejä tyhjään, kunnes eteen tulee rivinvaihto ja tämän jälkeen tallentaa toisen rivin saldo muuttujaan. Funktio lukee merkkejä tiedoston loppuun asti jos tiedostoon lisätään tietoa, täytyy silmukan ehto muuttaa lopettaakseen toiseen rivinvaihtoon asti. Parametrinä tiedosto josta saldo haetaan.

*Int muokkaaSaldoa(char *tiedosto, char *pin, double uusiSaldo)*

nimensä mukaisesti muokkaa saldoa tiedostosta. Koska saldo löytyy toiselta riviltä, funktio kirjoittaa tiedostoon pin-koodin ensimmäiselle riville ja saldon toiselle. Päädyin tähän ratkaisuun koska ohjelman kuormitus on vielä vähäistä eikä tiedostossa ei ole paljoa kirjoitettavaa tietoa, mutta kunhan se ostetaan laajempaan käyttöön, täytyy nopeutta ajatellen keksiä jokin uusi ratkaisu.

*Void jakaaSetelit(char *maarat, int summa)*

funktiota kutsutaan nostovalikon sisältä ja se laskee setelien määrät summan mukaan priorisoiden suuria seteleitä. Funktion parametreinä

*Double nostoValikko(char *tiedosto, char *pin)*

toimii valikkona, kun käyttäjä haluaa nostaa tililtä rahaa. Se kysyy käyttäjältä nostosumman – joka on 20 €, 40 €, tai 50 € suurempi luku kymmenen euron välein aina 1000 € asti – jonka jälkeen se kutsuu *jakaaSetelit()* funktiota jakamaan summan mahdollisimman suuriin seteleihin sekä laskemaan niiden määrät. Nostovalikon algoritmi on todella sotkuinen, joka johtuu deadlinea edeltävän ajan puutteesta; tästä syystä esitän syvät pahoittelut. Funktion parametreinä ovat tiedosto josta tiedot haetaan ja pin saldon muokkausta varten. Funktion sisältä kutsutaan *muokkaaSaldoa()* funktiota joka muokkaa saldon ja itse *nostoValikko()* palauttaa uuden saldon doublena.

*Void tapahtumaValikko(char *tiedosto)*

tulostaa liudan pseudotapahtumia jotka se näennäisesti hakee tietokannasta. Jos tietokantaan ei saada yhteyttä tulostuu virheviesti. Parametrinä tiedosto josta tapahtumat näennäisesti haetaan; tällä tarkistetaan ainoastaan että tietokantaan saadaan yhteys.

*Void saldoValikko(char *tiedosto)*

Hakee saldon tiedostosta kutsumalla *haeSaldo()* funktiota ja tulostaa sen nykyisen arvon. Parametrinä tiedosto josta saldo haetaan.

*Void talletusValikko(char *tiedosto, char *pin)*

Funktio jonka avulla tilille voi tallettaa rahaa ja joka on tehty pääasiassa testaajaa varten, ettei tiliä tarvitse mennä manuaalisesti sörkkimään vaan sen voi tehdä ohjelmalla. Parametreinä tiedosto jonne summa talletetaan sekä pin jota tarvitaan kutsuttaessa *muokkaaSaldoa()* funktiota.

Int valintaValikko(void)

Valikko joka auttaa käyttäjävalinnan ottamisessa toiminnon jälkeen. Kysyy haluaako käyttäjä palata käyttämään ohjelmaa jolloin funktio palauttaa (1), tai haluaako käyttäjä lopettaa ohjelman jolloin funktio palauttaa (2).

Void virheValikko(void)

Valikko joka ilmoittaa olemaan yhteydessä asiakaspalveluun, mikäli tietokantaan ei saada yhteyttä.

(3) 12345.acc

Tiedosto joka toimii tilin tietokantana. Tiedostoon on tallennettu pin-koodi ensimmäiselle riville, sekä saldo toiselle riville.

(4) Headers.h

Tiedostosta löytyy funktioiden esittelyt.

Käyttöohjeet:

Kun ohjelma käynnistetään, se kysyy käyttäjältä tilinumeroa, joka on simulaation tapauksessa *12345*. Tämän jälkeen ohjelma pyytää pin-koodia ja vertaa sitä tilin tiedostosta löytyvän oikean pin-koodin kanssa; oikea pin on *1122*. Tämän jälkeen käyttäjältä pyydetään toiminnan valintaa numeroarvona, jotka ovat: 1. rahan nosto tililtä, 2. saldon tarkastelu, 3. tilin tapahtumien tarkastelu, 4. rahan talletus tilille, 5. ohjelman lopettaminen.

1. Jos valitaan rahan nosto tililtä, käyttäjä voi nostaa tililtä 20 €, 40 € ja 50 € ylöspäin kymmenen euron välein aina 1000 € asti. Ohjelma ilmoittaa, jos syöte on virheellinen tai jos tietokantaan ei saada yhteyttä. Jos rahaa nostetaan onnistuneesti, käyttäjä voi valita: 1. ohjelman käytön jatkaminen tai 2. ohjelman lopettaminen.
2. Jos valitaan saldon tarkastelu, tilin tämänhetkinen saldo tulostetaan näytölle. Ohjelma ilmoittaa jos tietokantaan ei saada yhteyttä. Kun saldo on haettu ja tulostettu onnistuneesti, käyttäjä voi valita: 1. ohjelman käytön jatkaminen tai 2. ohjelman lopettaminen

3. Jos valitaan tapahtumien tarkastelu ja tietokantaan saadaan yhteys – muutoin tulostuu virheviesti – tulostaa valikko tilin menneet tapahtumat. Tapahtumien tarkastelun jälkeen käyttäjä voi valita: 1. ohjelman käytön jatkaminen tai 2. ohjelman lopettaminen.
4. Jos valitaan rahan talletus tilille, kysytään käyttäjältä rahamäärää, jonka hän haluaa tallettaa. Rahamäärän täytyy olla suurempi kuin nolla, eikä muita rajoituksia ole. Talletuksen jälkeen käyttäjä voi valita 1. ohjelman käytön jatkaminen tai 2. ohjelman lopettaminen.
5. Jos valitaan toimintojen lopetus, ohjelman suoritus loppuu.

Kokemukset:

Pankkiautomaatin ohjelmoiminen oli itselle mielekästä, jos ei ota huomioon aikapainetta. Oli mukava ohjelmoida suurempaa projektia verrattuna kurssin minitehtäviin, jotka täytyi olla aina lähes pilkuntarkasti oikein: vaikka niidenkin tekeminen oli ihan nautittavaa. Olin kurssilla hieman jäljessä erinäisistä syistä ja siksi teinkin lopputyön melkoisessa sprintissä ja joka heijastuu koodin heikkoon laatuun. Jos aikaa olisi vaikkapa muutama päivä enemmän refaktorointiin, niin koodista voisi saada huomattavasti selvemmän. Esimerkiksi tilinumeron ja pin-koodin syöttö olisi kannattanut toteuttaa erillisinä funktioina, nostovalikon muokkaaminen koska tällä hetkellä se on suorastaan kamala, sekä yleisen tiedonsiirron selkeyttäminen. Minulla oli onneksi apuna lyhyt ohjelmointitausta Pythonilla, joka helpotti työn tekemistä paljonkin. Ohjelman monimutkaistuessa oli hieman haastavaa pitää se kasassa ja johdonmukaisena, jonka takia koodin refaktorointiin ja editointiin kului paljon aikaa. Ohjelmassa huomasi että kirjoittaessa uutta koodia, varsinkin toimintojen funktioittaminen, se ei välttämättä integroitunut koodiin virheettömästi ja kun virheet korjasi; loi tämä korjaus taas uusia bugeja.

Työn ansiosta opin C-kielen perusteet, muistinhallintaa sekä perusteet ohjelmiston ja laitteiston välisestä kommunikaatiosta. Opin ohjelman jakamisesta moneen eri tiedostoon ja opin kuinka tärkeää on pitää ohjelma selkeänä.