AS-0.101 C/C++ -ohjelmoinnin peruskurssi / Tentti 6.5.2003/Hannu Laine

Vastaa neljään kysymykseen! Tentin arvosteluasteikko on 0 – 24 pistettä. Kaikkien kysymysten painoarvo on sama (6 pistettä/tehtävä). **Lähtökohtana on, että kysymyksessä 2 käytetään C++-kieltä ja muissa C:tä.** Jos kuitenkin kysymyksessä 2 taulukko on toteutettu oikeaoppisesti abstraktina tietotyyppinä C:llä, voi ko. tehtävästä saada 4 pistettä.
Palauta jokainen tehtävä erillisellä konseptilla!

1.

Kirjoita ohjelma, joka lukee mittalaitteelta saatuja mittaustuloksia (desimaalilukuja) näppäimistöltä. Mittaustulokset tasoitetaan (suodatetaan) siten, että lasketaan aina kolmen viimeisen mittauksen ns. liukuva keskiarvo. Täten

$$m_{1tas} = m_1$$

$$m_{2tas} = (m_1 + m_2)/2.0$$

$$m_{3tas} = (m_1 + m_2 + m_3)/3.0$$

$$m_{4tas} = (m_2 + m_3 + m_4)/3.0$$

jne

Yllä m_{itas} on i:s tasoitettu mittaus ja m_i on i:s tasoittamaton mittaus.

Ohjelma siis lukee tasoittamattomia mittauksia m_i, tasoittaa mittausarvot ylläkerrotulla tavalla ja tallettaa tasoitetut tulokset reaalilukutaulukkoon. Kun tasoitettuja mittauksia on saatu 50 kpl, ohjelma laskee niiden keskiarvon. Seuraavaksi ohjelma tulostaa suodatetuista mittausarvoista ne, jotka ovat yli 70% lasketusta keskiarvosta. Lopuksi ohjelma tulostaa kuinka/monta tällaista mittausta oli.

Huomautus 1. Tehtävä pitää funktioilla jakaa tehtäväkuvauksen mukaisiin tai muuten järkeviin osiin ja tietojen välitys on tehtävä parametreilla. Globaaleja muuttujia ei saa käyttää.

Huomautus 2. Syöttötiedoille ei tarvitse tehdä mitään järkevyystarkastuksia.

Huomautus 3. Ohjelman ei tarvitse tulostaa ohjelman käyttäjälle mitään kehotteita tai ohjetekstejä.

2.

Käyttäjälle halutaan tarjota uusi C-kielen normaalia taulukkoa parempi floating point lukujen taulukko. Parannetut ominaisuudet ovat:
1) Taulukon alkiot ovat aina dynaamisella alueella, mutta käyttäjän ei tarvitse ymmärtää pointtereista tai dynaamisesta muistinvarauksesta mitään, vaan hänellä on yksinkertainen interface taulukon käyttöön.
2) Taulukon koko voidaan määrätä ajonaikana (kuitenkin ennen kuin sinne aletaan tallentaa tietoa).

3) Virheellinen indeksointi ei aiheuta vakavia seurauksia ja taulukon käyttäjä voi testata indeksoinnin onnistumisen.

Taulukolle tarvitaan seuraavat operaatiot:

konstruktori
PutItem
GetItem
IndexingSucceeded
destruktori

Konstruktorissa varataan taulukolle tilaa siten, että siinä on paikka parametrina annetulle määrälle alkioita. Funktiolla PutItem voidaan tallettaa parametrina annettu luku parametrina annettuun paikkaan (indeksi). Funktiolla GetItem taulukosta voidaan hakea tietoalkio parametrina annetusta paikasta i. Funktiolla IndexingSucceeded, voidaan testata edellisen indeksointioperaation onnistuminen eli funktiokutsujen PutItem tai GetItem onnistuminen. Funktio IndexingSucceeded palauttaa arvon 1, jos indeksointi onnistui ja 0, jos indeksointi ei onnistunut eli jos funktiokutsussa GetItem tai PutItem oli indeksinä sopimaton arvo (negatiivinen tai suurempi arvo, kuin taulukon koko edellyttää). Destruktori vapauttaa taulukolle varatun tilan.

Toteuta C++:lla luokka Tarray (tai vaihtoehtoisesti C:llä abstraktina tietotyyppinä sama asia), joka täyttää yllämainitut vaatimukset ja jolla on yllämainitut operaatiot ts. **kirjoita luokan Tarray luokkamäärittely ja toteuta sen yllämainitut jäsenfunktiot.**

Huomautus 1. Jäsenfunktiot PutItem ja GetItem saa toteuttaa myös indeksointioperaattorina.

Jotta tässä tarkoitetun taulukon idea tulisi selväksi, alla oleva esimerkki näyttää, kuinka sitä voitaisiin käyttää.
//Esimerkkiohjelma luokan Tarray käytöstä

```
void main (void) {
        Tarray array(3);
        float item;
        int i;
        for (i = 0 ; i < 5 ; i++) {
                array.PutItem(i, 10+i); //(luku 10+i paikkaan i)
                if(!array.IndexingSucceeded())
                        printf("\n Väärä indeksointi"); //2 kertaa
        }
        for (i = 0 ; i < 5 ; i++) {
                array.GetItem(i, &item);
                if(!array.IndexingSucceeded())
                        printf("\n Väärä indeksointi"); //2 kertaa
                else
                        printf("\n Taulukosta saatiin %f", item);
        }
}
```

3.

Kirjoita funktio, joka saa parametrina merkkitaulukon ja kokonaisluvun, joka ilmoittaa montako merkkiä taulukon alusta käsitellään. Funktio asettaa taulukon merkkeihin pariteettibitin parillisen pariteetin mukaan.

Merkit ovat 7 bittisen standardi ASCII-koodiston mukaisia. Tämä merkitsee, että merkkikoodin bitit ovat bittipositioissa 0, 1, 2, ... 6. Bittipositiossa 7 on alunperin täysin satunnaista sisältöä, eli siinä voi olla 0 tai 1 täysin muista biteistä riippumatta. Funktio asettaa bittipositioon 7 pariteetin aina parillisen pariteetin mukaan. Tämä merkitsee, että jokaisessa taulukon merkisssä on aina parillinen määrä bittejä, kun lasketaan kaikki kahdeksan bittipositiota

4.

Kaksipäiseksi jonoksi (deque) sanotaan tietosäiliötä, jolla on seuraavat operaatiot:

initialize
addToFront (lisää alkuun)
addToRear (lisää loppuun)
removeFromFront (ottaa alusta)
removeFromRear (ottaa lopusta

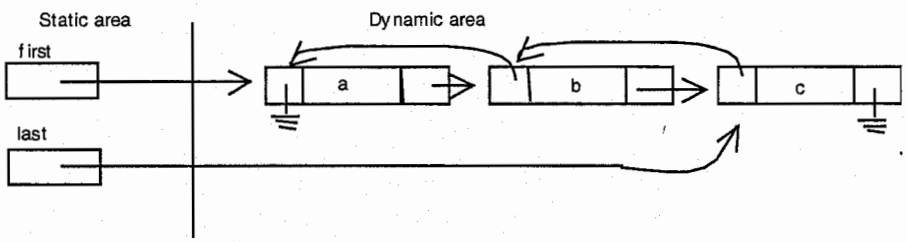
Eräs kätevä toteutustapa tällaiselle kaksipäiselle jonolle on kaksoislinkitetty lista. Alla oleva kuva valaisee asiaa. Kuvassa nähdään tilanne muistissa, kun kaksipäiseen merkkijonoon on viety merkit seuraavilla operaatioilla:

Lisää b alkuun (tai loppuun, koska ensimmäisellä lisäyksellä ei väliä, onko loppuun vai alkuun).
Lisää a alkuun.
Lisää c loppuun.

Toteuta kaksipäinen merkkijono abstraktina tietotyyppinä tällä periaatteella, ts. **kirjoita tietotyypin Tdeque määrittely ja toteuta sen yllämainitut operaatiofunktiot.**

Tietotyyppi Tdeque sisältää siis vain kaksi osoitinta, toinen dynaamisella alueella olevaan ensimmäiseen solmuun (alku) ja toinen viimeiseen (loppu).

Huomautus 1. Erikoistapaukset pitää tietysti hoitaa asianmukaisesti (tyhjä jono, vain yksi alkio jonossa, jne).



5.

Selvitä mitä tarkoitetaan geneerisellä ohjelmoinnilla ja millä tavoilla sitä voidaan käytännössä toteuttaa C-kielellä (tai vaihtoehtoisesti C++-kielellä).