AS-0.101 C/C++ -ohjelmoinnin peruskurssi / Tentti 6.9.2004/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, pisteitä voi ko. tehtävästä saada 4 pistettä.

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);
        }
```

C-kielen standardikirjaston merkkijonojen käsittelyfunktioissa on yleensä lähtökohtana se, että tila mahdolliselle tulosmerkkijonolle on valmiiksi varattuna. Esimerkiksi funktion strcpy (merkkijonon kopiointifunktio), jonka prototyyppi on

char * strcpy(char *destination, const char * source);

tapauksessa parametrin destination osoittamalla muistialueella tulee olla riittävästi tilaa merkkijonon source kopiolle. C-kielen standardikirjastosta myös puuttuu joitakin käteviä mm. Basic-kielestä tuttuja funktioita kuten osan ottaminen merkkijonosta lähtien tietystä merkkipaikasta ottamalla tietty määrä merkkejä siitä eteenpäin.

Kirjoita C-kielen merkkijonoille seuraavat kolme funktiota:

substring
allocstrcpy
allocstrcat

Funktiolle substring välitetään kolme parametriä: 1) merkkijono, josta otetaan osa, 2) sen merkkipaikan indeksi (numero), josta lähtien alastringi otetaan ja 3) alastringin pituus eli otettavien merkkien määrä. Funktio palauttaa osoittimen alastringiin. Funktio varaa alastringille juuri sopivan kokoisen tilan dynaamisesta muistista. Alkuperäinen stringi säilyy siis muuttumattomana.

Funktiolla allocstrcpy voidaan tehdä kopio merkkijonosta. Ero standardikirjastossa olevaan funktioon strcpy on siinä, että allocstrcpy varaa tilan kopiolle dynaamisesta muistista (juuri sopivan kokoisena). Funktiolla pitää olla edelleen kaksi parametriä, joista ensimmäisellä ilmaistaan kohdetta (destination) ja toisella lähdettä (source).

Funktio allocstrcat liittää toisen parametrin edustaman merkkijonon ensimmäisen parametrin edustaman merkkijonon perään. Tässäkin tapauksessa lähtökohta on se, että ei voida olettaa, että sen merkkijonon perässä, johon liitetään olisi tilaa ylimääräisille merkeille. Siksi funktio allocstrcat varaa juuri sopivan pituisen muistialueen tulosstringille. Samoin kuin standardifunktion strcat tapauksessa, merkkijono, jonka perään liitetään ei enää ole sellaisenaan erillisenä olemassa funktion käytön jälkeen.

Näyttääksesi kuinka kirjoittamiasi funktioita käytetään, **laadi pieni pääohjelma**, joka ensin lukee näppäimistöltä kaksi merkkijonoa (string1 ja string2). Sitten ohjelma tekee kopion ensimmäisestä luetusta merkkijonosta (copyString) dynaamiselle alueelle funktiolla allocstrcpy. Seuraavaksi ohjelma muodostaa uuden merkkijonon (leftString) 4:stä ensimmäisestä merkkijonon string2 merkistä funktiolla substring. Lopuksi ohjelma muodostaa tulosmerkkijonon (resultString) liittämällä ensimmäisen luetun merkkijonon kopioon copyString toisen luetun merkkijonon alun leftString.

Tällöin siis molemmat alunperin luetut merkkijonot pysyvät muistissa muuttumattomina.

Huomautus. Standardifunktion strcat prototyyppi on

char * strcat(char *destination, const char * source);

4.

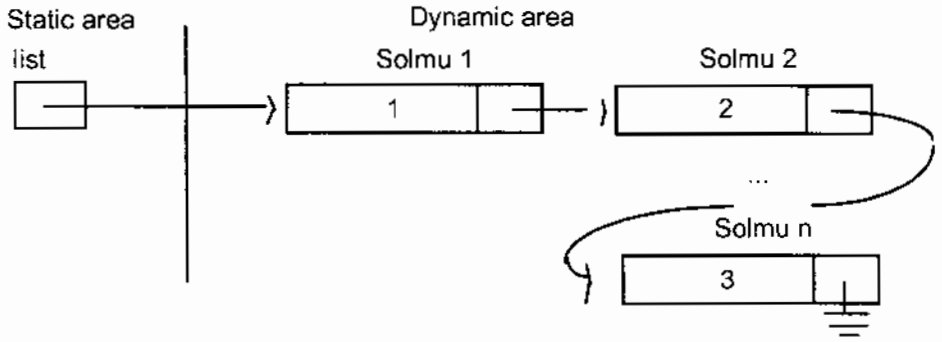
Kokonaislukujen listalle tehdään dynaamisesti linkattu toteutus siten, että koko listaa edustaa yksi osoitin ensimmäiseen dynaamisella alueella olevaan solmuun (katso kuva alla). **Kirjoita tarvittavat tietomäärittelyt** tätä listaa edustavalle tyypille Tlist. **Kirjoita sille lisäksi seuraavat kolme operaatiofunktiota:**

initialize, joka alustaa listaa edustavan osoittimen siten, että se sisältää NULL-osoittimen, joka edustaa tyhjää listaa.

delete_nth_node, joka poistaa n:nnen solmun listasta. Poistettavan solmun järjestysnumero (alkaa nollasta) annetaan parametrina.

split_list, joka muodostaa annetusta listasta kaksi listaa siten, että ensimmäisessä on alkuperäisen listan järjestysnumeroltaan parilliset solmut (siis nollas, toinen, neljäs jne) ja toisessa alkuperäisen listan parittomat solmut. Splittaus pitää tehdä siten, että vain osoittimia modifioidaan.. Uusia solmuja ei saa luoda muistiin eikä solmun datasisältöjä (kokonaislukuja) saa siirtää solmusta toiseen.

Huomautus 1. Erikoistapaukset pitää tietysti hoitaa asianmukaisesti (tyhjä lista, vain yksi alkio listassa jne tai poistetaan ensimmäinen alkio, viimeinen tai mikä tahansa alkio). NULL-osoitin edustaa tyhjää listaa.



5.

Vertaile dynaamisia muuttujia ja staattista muuttujia. Vertailussa tarkastellaan ominaisuuksia, käyttöä, mahdollisuuksia, rajoituksia jne. Staattisina muuttujina pidetään kaikkia muuttujia, jotka eivät ole dynaamisessa muistissa (kasassa, heapissä).