Tentamen Algoritmiek (I00009)

maandag 15 januari 2007, 13:30 - 15:30 uur, E0003

- Deze toets is gesloten boek. Je mag geen literatuur of PC gebruiken.
- De toets bestaat uit drie opgaven die in zwaarte verschillen. Aan het begin van iedere opdracht staat aangegeven hoeveel punten je maximaal kunt halen. Onderdelen worden eveneens verschillend gewaardeerd. Het aantal punten dat je maximaal per onderdeel kunt halen staat in de kantlijn vermeld.
- Schroom niet vragen te stellen over eventuele onduidelijkheden in de opgaven.
- Ga niet overhaast te werk en lees de opgaven nauwkeurig. Begin met de opgaven die je het eenvoudigst lijken. Denk eerst goed na. Hou je antwoorden kort en controleer ze voor je het werk inlevert.

1 Klassen en Rijen: vliegtuigstoelen reserveren **totaal 45 punten**

In een stoelreservingssysteem worden vliegtuigstoelen aan passagiers gekoppeld. In deze opdracht ga je een eenvoudig reserveringssysteem ontwerpen en implementeren in C++.

Voor het aantal beschikbare stoelen gebruiken we de constante

```
const int Capaciteit = 120;
```

Voor het opslaan van reizigersinformatie gebruiken we een rij van `Reizigers`. Van iedere reiziger houden we enkel zijn/haar naam bij en wel als een rij van tekens (**char**s). Dit leidt tot de volgende declaraties:

```
const int Naamlengte = 12;
typedef char Reiziger [NaamLengte];
```

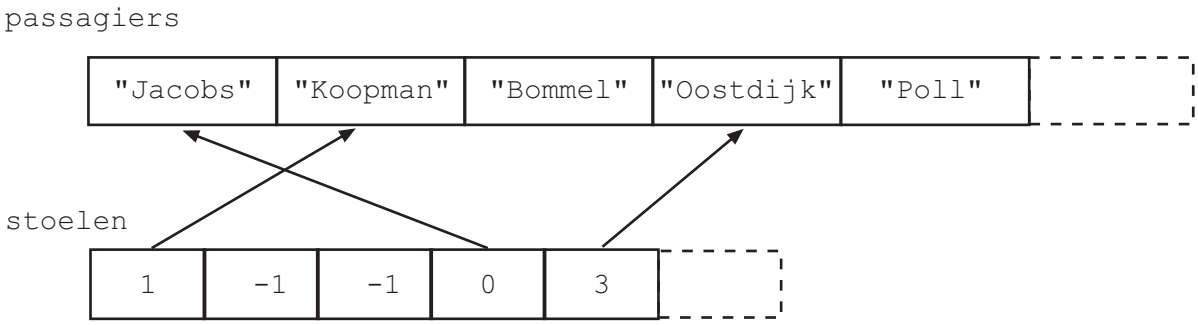
De bezetting van de stoelen houden we bij in een rij van integers.

Zowel de rij van reizigers als die van de stoelen houden we verborgen door ze als **private** variabelen in een klasse op te nemen. Dit leidt tot de volgende klassedefinitie.

```
class Reservering
{
private:
    Reiziger passagiers [Capaciteit];
    int stoelen [Capaciteit];

    int zoekVrijeRaamStoel  ();
    int zoekVrijeStoel      ();

public:
    Reservering ();

    void toon ();
```

```
    void plaatsPassagiers ();

} ;
```

In de rij van stoelen worden lege stoelen aangegeven met de waarde -1. Is stoel met nummer `i` bezet, laten we aannemen met persoon p, dan bevat `stoelen[i]` de *index* van p in de rij `passagiers`. Voorbeeld:



Ga zelf na dat `passagiers[stoelen[i]]` de naam van de reiziger is, die op stoelnummer `i` zit, tenminste als `stoelen[i]` != -1. Opmerking: het gebruik van *index-rijen*, zoals de stoelenrij van deze opgave, is in practicumopdracht 7 over het verenigingsinformatiesysteem ook al aan bod gekomen.

Verder nemen we aan dat iedere rij in het vliegtuig precies 6 stoelen bevat, waarbij de buitenste stoelen zich aan 'n raam(pje) bevinden. Dus rij 1 bestaat uit stoelen 0 t/m 5, rij 2 uit 5 t/m 11 etc. Van deze rijen zijn 0,5,6,11 etc. raamstoelen.

We veronderstellen dat in de constructor van bovenstaande klasse de passagiersrij geheel gevuld wordt en alle stoelen op -1 gezet om aan te geven dat ze (nog) onbezet zijn.

Implementeer nu de volgende operaties.

10 punten **1.a**) De methode `toon` die de bezetting van de vliegtuigstoelen weergeeft. De namen van de passagiers dienen rij voor rij in 6 kolommen te worden weergegeven. Je dient bij het afdrukken rekening te houden met de mogelijkheid dat stoelen onbezet zijn. **Hint:** met

```
cout << setw (breedte) << iets;
```

kun je ervoor zorgen dat voor het weergeven van `iets` precies `breedte` tekens worden gebruikt.

10 punten **1.b**) De (privé) methode `zoekVrijeStoel` die de eerste de beste lege stoel in het vliegtuig oplevert geven we voor.

```
int Reservering :: zoekVrijeStoel ()
{
     for (int i = 0; i < Capaciteit; i++)
         if (stoelen [i] == -1)
            return i;
     return Capaciteit;
}
```

Zoals je ziet, levert deze functie het stoelnummer op van de eerstvolgende vrije stoel. De waarde `Capaciteit` wordt gebruikt om aan te geven dat er niets meer vrij is. Definieer nu zelf de methode `zoekVrijeRaamStoel` die op zoek gaat naar de eerstvolgende vrije stoel die zich aan 'n raam bevindt.

15 punten **1.c**) Definieer de publieke methode `plaatsPassagiers` die alle passagiers aan stoelen koppelt. De volgorde waarin passagiers geplaatst worden komt overeen met volgorde in de rij die we gebruiken om de passagiers in op te slaan. Dus eerst wordt `passagiers`[0] geplaatst, daarna `passagiers`[1] etc.

Zolang er nog raamstoelen zijn moeten deze als eerste gevuld worden. Daarna komen de ander vrije plaatsen aan de beurt. Je hoeft geen rekening te houden met 'overboeking': er zijn voldoende stoelen om iedere passagier 'n plek te geven.

10 punten **1.d**) Definieer tot slot de `main`-functie waarin een `Reservering`-variabele gedeclareerd wordt, vervolgens worden passagiers geplaatst en en de bezetting van het vliegtuig weergegeven.

2 Teksten: geheimschrift

totaal 30+5 punten

Cryptografie of geheimschrift houdt zich bezig met technieken voor het zodanig *versleutelen* van te verzenden informatie, dat het voor een *cryptoanalist*, een persoon die toegang heeft tot het kanaal tussen zender en ontvanger, en dus als het ware 'mee kan luisteren', onmogelijk is om tegen aanvaardbare inspanning uit de getransporteerde data af te leiden welke informatie er door de zender was verzonden. Cryptografie wordt voornamelijk gebruikt om gegevens over te dragen die onderweg niet leesbaar mogen zijn door andere partijen. Enkel de zender en ontvanger beschikken over de juiste *sleutel* om de gegevens terug om te zetten in hun originele vorm.

Onder *encryptie* verstaat men het coderen (versleutelen) van gegevens op basis van een bepaald algoritme. Deze versleutelde gegevens kunnen nadien weer ontcijferd (gedecodeerd) worden zodat men de originele informatie weer terugkrijgt. Dit proces wordt *decryptie* genoemd.

In deze opgave ga je een encryptiemethode implementeren die bekend staat onder de naam *monoalfabetische substitutie*. Het idee hierbij is dat je alle tekens uit het oorspronkelijke bericht vervangt volgens een van tevoren opgestelde substitutietabel. Zo'n tabel kent dus aan ieder teken een nieuw teken uit het alfabet toe. Het moge duidelijk zijn dat zonder die tabel het moeilijk is om uit het versleutelde bericht het oorspronkelijke bericht te reconstrueren.

Stel je gebruikt de volgende substitutietabel (hetgeen eigenlijk de sleutel van het encryptiealgoritme is)

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
W Q A E D C X S Z R F V B G T Y H N J U I K L O P M
```

Als we nu de oorspronkelijke tekst versleutelen vervangen we gewoon de D door E, de I door Z enz. Voor de tekst

```
D I T I S Z E E R G E H E I M
```

levert dit de volgende geëncrypte boodschap op

```
E Z U Z J M D D N X D S D Z B
```

Ga dit zelf na!

Voor ons algoritme maken we gebruik van de volgende structuur waarmee we iedere substitutiepaar representeren.

```
struct Subst
{   char origineel, resultaat;
} ;
```

De intentie is om het teken `origineel` te vervangen door het teken `resultaat`. De substitutietabel is nu eenvoudigweg een rij van `Subst`-elementen, in C++:

```
const int TabelGrootte = ...; // Vul hier een geschikte waarde in.
Subst substitutietabel [TabelGrootte] = { ... }; // ... en hier alle substitutieparen.
```

15 punten **2.a**) Implementeer de functie

```
void versleutel (char tekst [])
```

die ieder teken van het (als C-string) meegeven bericht genaamd `tekst` vervangt volgens bovenstaande globaal gedeclareerde `substitutietabel`.

8 punten **2.b**) Wat is de complexiteit van dit algoritme, uitgedrukt in de lengte van bericht en de grootte van de tabel? Je moet dus rekening houden met het feit dat én de lengte van het bericht én de tabelgrootte *niet* vastliggen. Motiveer je antwoord!

7 punten **2.c**) Geef (kort maar duidelijk) aan hoe je algoritme moet aanpassen zodat het gebruikt kan worden voor decryptie: het omzetten van versleutelde bericht in het leesbare originele bericht.

5 punten **2.d**) **Bonus**: dit onderdeel hoef je niet te maken, maar het kan je wel extra punten opleveren. Weet je een slimme manier om de substitutietabel anders te representeren waarmee je de complexiteit van het versleutelalgoritme kun verbeteren? Zo ja, licht dit dan toe. Zo nee, dan krijg je voor dit onderdeel geen punten.

3 Recursie: een mysterieuze functie **totaal 15 punten**

Gegeven de volgende functie:

```
int doe_iets ( const int x, const int y )
{
    if ( x <= 0  &&  y <= 0 )
        return  - 1 ;
    else if ( x > y )
        return  x - y * doe_iets ( x - 2, y ) ;
    else
        return  x + y * doe_iets ( x, y - 2 ) ;
}
```

15 punten **3.a**) Bepaal de waarde van de volgende aanroep:

```
doe_iets ( 3, 4 )
```

Laat duidelijk (stap voor stap) zien, hoe je tot je antwoord komt.