Programmeermethoden 2015
Tweede programmeeropgave: Geheim

De *tweede* programmeeropgave van het vak **Programmeermethoden** in het najaar van 2015 heet *Geheim*; zie ook het **vierde werkcollege**, **vijfde werkcollege** (de betreffende WWW-bladzijde bevat handige tips evenals testfiles — te zijner tijd!) en **zesde werkcollege**, en lees geregeld deze pagina op WWW.
Spreek/Vragenuur in zalen 302 ... 309: dinsdag 22, woensdag 23, donderdag 24 september, dinsdag 29, woensdag 30 september, donderdag 1, dinsdag 6, woensdag 7, donderdag 8 en vrijdag 9 oktober 2015, van circa 15:30 tot 17:00 uur.

I&E-studenten (Den Haag) Vragenmiddag in zaal Paleistuin/Malieveld op donderdag 8 oktober 2015, 14:45-17:30 uur.

Staan er geheime boodschappen in teksten? En wat voor getallen staan in die boodschappen? Deze vragen gaan we beantwoorden!

Aan de gebruiker wordt eerst gevraagd of hij/zij een boodschap wil coderen of juist ontcijferen. Daarna geeft hij/zij één of twee filenamen: bij het ontcijferen één, bij het coderen twee. Verder geeft hij/zij in het geval van ontcijferen nog een geheim getal: *k* (geheel, minstens 1, hoogstens 20). Bij het coderen krijgt hij/zij deze waarde juist te zien. Stel eenvoudige vragen om deze gegevens van de gebruiker te weten te komen. Het programma leest dan eenmalig de opgegeven invoerfile, en schrijft de uitvoer symbool voor symbool op de juiste wijze weg naar de uitvoerfile (bij het coderen) of naar het scherm (bij het ontcijferen). Elk symbool uit de invoerfile mag en moet precies één maal (met `invoer.get (...)`) gelezen worden.

Een geheime boodschap, evenals de bijbehorende codering, bestaat uit symbolen uit de ASCII-tabel vanaf nummer 32 (spatie) tot en met 125, en "LineFeed"s (LF, \n, 10). Als per ongeluk ook "CarriageReturn"s (CR, \r, 13) worden gebruikt is dat niet erg. De volgorde van de symbolen in de boodschap is dezelfde als de volgorde van de desbetreffende symbolen in de originele file. Maar wanneer hoort een symbool tot de boodschap? Lopend door de tekst houden we bij hoeveel klinkers we zagen. Het eerste symbool na precies *k* klinkers is deel van de boodschap. Daarna wordt de teller weer op 0 gezet (ook als het symbool zelf een klinker is). Verder klapt het symbool % (als het niet in de boodschap staat) om tussen het tellen van klinkers en medeklinkers, waarbij de teller meteen weer op 0 wordt gezet. Regelovergangen ("LineFeed"s; als ze niet in de boodschap staan) zetten de teller ook op 0, en vanaf dan worden weer klinkers geteld. We definiëren als klinker: a, e, i, o en u (kleine letters). Medeklinkers zijn de andere kleine letters.

Verder moet het programma getallen in de geheime boodschap detecteren en bepalen of dit een wellicht **Lychrel-getallen** zijn. Na afloop worden het kleinste en grootste getal afgedrukt, en hun gemiddelde — afgerond naar het dichtstbijzijnde gehele getal. Voorkomende getallen hebben maximaal vier cijfers. We doen alleen positieve (> 0) gehele getallen. Zo bevat de boodschap `123abcd-"qqq 5"+++uvw-7.88ddd//vb5656p` wat ons betreft de gehele getallen 123, 5, 77, 88 en 5656. De boodschap eindigt niet op een cijfer.
Op het scherm wordt naast het desbetreffende getal in de boodschap tussen haakjes afgedrukt wat het aantal iteraties is om tot een palindroom te komen (voor 545 is dit 0, voor 113 is dit 1), of het nummer van de iteratie waarvan het resultaat boven `INT_MAX` (gebruik `include <climits>`) uitkomt (voor 196 is dit (waarschijnlijk) 28). Als dit laatste gebeurt, wordt dit erbij vermeld, bijvoorbeeld als "196(18;overflow)".

Voor het coderen moet een random-generator worden gebruikt (zie de vorige programmeeropgave). Genereer steeds random karakters, en als het moment daar is: neem het volgende karakter uit de te coderen boodschap. Het algoritme lijkt erg veel op dat voor het ontcijferen!

Ter verdere inspiratie met voorbeeldfiles, zie het **vijfde werkcollege**, en een tweetal kleine voorbeelden:

- Met *k* gelijk aan 2: tekst `xxxaxxaGbbaaraaoaauQQaBBic%aaneeeBrhtto` levert boodschap `Groucho`.
- Met *k* gelijk aan 3: tekst `a55aa1eee2iii3xyuu6uaz` levert boodschap `123(1)a`.

Let op: files van websites kopiëren door met rechter muisknop op de links te klikken, anders (met markeer-

opy-paste) gaan spaties/tabs wellicht fout!

Opmerkingen

- We nemen aan dat de gebruiker zo vriendelijk is verder geen fouten te maken bij het invoeren van gegevens.
- Gebruik zonodig de regelstructuur: elke regelovergang in een bestand bestaat uit een LineFeed (in UNIX) of een CarriageReturn gevolgd door een LineFeed (in Windows). Normaal gesproken gaat dit "vanzelf" goed. We nemen aan dat er voor het EndOfFile-symbool (wat dat ook moge zijn) een regelovergang staat.
- Alleen voor de namen van de files mag een array (of string) gebruikt worden; voor het lezen en verwerken van de tekst is slechts het huidige karakter en enige kennis over de voorgaande karakters nodig — zie boven. Alleen de headerfiles `iostream` en `fstream` mogen gebruikt worden (en `string` voor de filenamen; denk in dat geval aan het gebruik van `c_str`; en `climits` voor `INT_MAX`). Uit een file mag alleen met `invoer.get (...)` gelezen worden, vergelijk Hoofdstuk 3.7 uit het dictaat, gedeelte "aantekeningen bij de hoorcolleges". Er staan zo weinig mogelijk `get`'s in het programma: vergelijk het voorbeeldprogramma uit dit hoofdstuk (daar staat twee keer `get`, één maal vóór de loop, uiteraard). Karakters mogen niet worden teruggezet in de oorspronkelijke file.
  Schrijf zelf functies die testen of een karakter een cijfer is, etcetera. Er mogen geen andere functies dan die uit `fstream` gebruikt worden, en `c_str`.
- Denk aan het infoblokje dat aan begin op het scherm verschijnt. Gebruik enkele zelfgeschreven geschikte functies, bijvoorbeeld voor infoblokje, inlezen gegevens van de gebruiker, omkeren van het getal en testen of iets een klinker is (zie de tips bij het vijfde werkcollege). Gebruik in het infoblokje minstens één for-loop om een kader met (bijvoorbeeld) sterretjes te produceren. Globale variabelen zijn streng verboden. Ruwe indicatie voor de lengte van het C++-programma: circa 200 regels.

Uiterste inleverdatum: vrijdag 9 oktober 2015, 17:00 uur.
Manier van inleveren:

1. Digitaal de C++-code inleveren: stuur een email naar `pm@liacs.leidenuniv.nl`.
   Stuur geen executable's, lever alleen de C++-file digitaal in! Noem deze bij voorkeur zoiets als `ruttesamsom2.cc`, dit voor de tweede opdracht van het duo Samsom-Rutte. De laatst voor de deadline ingeleverde versie wordt nagekeken.
2. En *ook* een papieren versie van het verslag (inclusief de C++-code) deponeren in de speciaal daarvoor bestemde doos "Programmeermethoden" in de postkamer van Informatica, kamer 156 van het Snellius-gebouw.
   
   Haagse studenten mogen de `pdf`-file meesturen.
   Overal duidelijk datum en namen van de (maximaal twee) makers vermelden, in het bijzonder als commentaar in de eerste regels van de C++-code. Lees bij het zesde werkcollege hoe het verslag eruit moet zien en wat er in moet staan.

De gebruiken compiler: als hij maar C++ vertaalt; het programma moet in principe zowel op een Linux-machine (met `g++`) als onder Windows met Code::Blocks draaien. Test dus in principe op beide systemen! Normering: layout 2; commentaar 2; overzichtelijkheid/modulariteit/functies 2; werking 4. Eventuele aanvullingen en verbeteringen: lees deze WWW-bladzijde: `www.liacs.leidenuniv.nl/~kosterswa/pm/op2pm.php`.