Linköpings Universitet
Institutionen för datavetenskap (IDA)
UPP-gruppen 2015-11-01

Standard Template Library

Mål

Denna laboration går ut på att öva på att använda de givna klasser och algoritmer som finns i C++ standard template library.

STL i C++ är stort. Det finns ett antal olika klasser för att lagra data på olika sätt, och ett antal algoritmer för att behandla lagrad data (för det mesta användbara oavsett lagringsform). Laborationen behandlar enbart en liten del av detta, men så mycket att du sedan på egen hand bör kunna använda andra delar korrekt.

Läsanvisningar

- Strängar och strängfunktioner (`std::string`, `at()`, `substr()`, `insert()`, `erase()`, `<cctype>`)
- STL containrar (`std::vector`, `std::list`, `std::map`)
- STL iteratorer (`::iterator`, `::const_iterator`)
- STL algoritmer (`sort()`, `find()`, `copy()`, `for_each()`)
- Programkod som argument till funktioner
  - Funktionspekare (C)
  - Funktionsobjekt (::operator(), C++98)
  - Lambdafunktioner (C++11)
- Kommandoradshantering (argc, argv)
- Slumpgenerering (random_device, default_random_engine, uniform_int_distribution, etc.)

Uppgift: Tvättad ordlista

Bakgrund

Ett antal år tillbaka i tiden, innan smarttelefonen, försökte mobiltelefoner förutse vad du försökte tumma fram genom att använda ordlistor. Ibland lyckades de riktigt bra, men ibland blev det också minnesvärda historier (en tid efter pinsamma ursäkter till mottagaren).

Mindre kända är omständigheterna hur dessa ordlistor telefonerna använde skapades. Detta historia blir[1] kanske alltså en uppenbarelse. Historien tar sin början i det avlägsna landet Långtbortistan. Där är arbete båda bra och billigt. En ideal plats för ett telefonbolag att placera ordlisteavdelningen. I denna avdelning utvecklas ett program som samlar ord och ordstatistik från alla populära webbsidor. För att avgöra vilket språk ett ord är skrivet i används helt enkelt toppdomänen för sidan där ordet förekommer. Med denna samling av ord och användningsfrekvens kan ordlisteavdelnigen sätta ihop ordlistor som ger goda (eller på lite sikt åtminstone roliga) ordförslag.

Ett par inhemska långtbortistanier gör också sitt bästa att exkludera ord som inte är “rumsrena” (såvida de inte kan skapa roliga missförstånd så klart).

[1] Om det finns någon sanning i historien är det helt och hållet en tillfällighet.

Insamling och tvättning

Några tillväxtmarknader använder fortfarande den gamla typen av telefoner på grund av deras överlägsna räckvidd och batteritid. De behöver ett nytt ordsamlingsprogram. Det är det du ska skriva åt dem nu. Programmet ska läsa igenom en textfil för att ta fram *potentiella ord*. Alla potentiella ord är separerade med minst ett vitt tecken. Ett potentiellt ord kan tänkas innehålla "skräptecken" som normalt ingår i skriven text, till exempel parenteser, citationstecken, komma, punkt och annat som kan avsluta en mening. Vi delar upp dessa skräptecken i sådana som kan förekomma omedelbart *före* ett ord och omedelbart *efter* ett ord. Möjliga skräptecken i varje kategori är som följer:

- Omedelbart före ett ord kan förekomma citattecken och öppningsparenteser. Vi kallar detta inledande skräp: `"'(`
- Omedelbart efter ett ord kan förekomma interpunktationstecken, citattecken, apostrofer, och avslutande parenteser. Vi kallar detta avslutande skräp: `!?;,:."')`
- Omedelbart före det avslutande skräpet kan också förekomma en engelsk genitivändelse: `'s`
- Notera att skräptecken som förekommer inuti (med icke-skräp på vardera sida) ett ord är vare sig inledande eller avslutande.

Ditt program ska nu tvätta alla *potentiella ord* för att producera *tvättade ord*. Detta görs genom att ta bort allt inledande skräp, allt avslutande skräp och eventuell engelsk genitiv. Slutligen avgörs om ett *tvättat ord* är ett *giltigt ord* genom att undersöka om följande villkor uppfylls:

- Ordet innehåller enbart bokstäver (små eller stora) och bindestreck.
- Bindestreck förekommer endast inuti ordet (inte först eller sist) och alltid enskilt (inte flera i rad).
- Ordet innehåller minst tre tecken (kortare ord bedöms lönlösa att förutse).

När ett ord bedömts som *giltigt ord* ersätts alla versaler med gemener och läggs till i ordlista och statistik. Ogiltiga ord ignoreras helt enkelt.

Indata och utdata till programmet

Notera: Specifikationen på ord och ordtvätt från verkställande chefslångtbortistanier är mycket bestämd, och långtbortistanierna är pedanter som tar mycket allvarligt på minsta avvikelse från specifikation. Testa noga.

Ditt program ska fungera för godtycklig textfil, t.ex. en webbsida eller källkoden till ditt program (som kan vara intressanta testfall). Indatafilen ska anges på kommandoraden (felhanterat) och sökas igenom efter ord så som anges i avsnittet om insamling och tvättning. En andra kommandoradsparameter (efter filnamnet) anger vilken typ av utdata som begärs enligt följande lista (se även senare exempel):

-a Alla *giltiga ord* skrivs ut i alfabetisk stigande ordning följt av antalet gånger det förekom i filen. Tabellen ska vara formaterad i två raka kolumner. Först bokstaven i ord och sista siffran i nummer ska stå rakt över motsvarande tecken på nästa rad. Du måste anpassa kolumnbredden till längsta ordet och största siffran så allt får plats (ord vänsterjusteras, nummer högerjusteras).

-f Alla *giltiga ord* skrivs ut i fallande frekvensordning. Listan skrivs ut på kolumner som ovan, men har istället sista bokstaven i ord rakt över motsvarande bokstav på raden över (både ord och nummer högerjusteras). Du behöver förmodligen kopiera orden till en annan containertyp innan du sorterar.

-o Skriv alla *giltiga ord* i den ordning de förekom i indatafilen[2] Du ska sätta in radslut så att alla rader blir så långa som möjligt, men strikt kortare än N tecken[3] Radlängden N anges som tredje kommandoradsargument precis efter `-o`.
KRAV: Detta problem ska lösas med algoritmen `for_each`. Du behöver skapa antingen ett funktionsobjekt eller en lambdafunktion (prova båda varianter).

KRAV: Du ska använda STL containrar och algoritmer till allt och så långt det är möjligt. Överdriv hellre än att missa ett lärotillfälle.

På nästa sida ges några exempel på hur programmet är tänkt fungera. Du måste så klart själv skapa egna mer utförliga test för att säkerställa att allt du skriver fungerar som specificerat.

Frivilligt: Slumpad ordning

Lägg till en kommandoradsparameter `-s` som låter användaren skriva ut orden i slumpmässig ordning. Vill du göra det lite roligare kan du kika på `https://en.wikipedia.org/wiki/Parody_generator` eller `https://en.wikipedia.org/wiki/Dissociated_press`.

[2]Detta är för att teamet för kvalitetsgranskning lättare ska kunna läsa igenom innehållet i en fil utan att se t.ex. alla HTML-taggar.

[3]Hela ord som är längre än N tecken skrivs ut i sin helhet på sin egen rad.

En HTML-fil som exempel

```
<html>
<head>
<title> The page title! </title>
</head>
<body id="my-body"><h1>The Page: </h1><p>This is the page body. </p></body>
</html>
```

Körningar av programmet med exempelfilen

```
$ a.out
Error: No arguments given.
Usage: a.out FILE [-a] [-f] [-o N]

$ ./a.out example.html
Error: Second argument missing or invalid.
Usage: ./a.out FILE [-a] [-f] [-o N]

$ a.out example.html -a
body   1
page   3
the    2
title  1

$ a.out example.html -f
 page  3
  the  2
 body  1
title  1

$ a.out example.html -o 14
the page
title page
the page body

$ a.out example.html -o 9
the page
title
page the
page
body
```