TDP004 - Tentamen

2014-08-26

Regler

- All kod som skickas in för rättning ska kompilera och vara väl testad.
- Inga elektroniska hjälpmedel får medtas. Mobiltelefon ska vara avstängd och ligga i jacka eller väska.
- Inga ytterkläder eller väskor vid skrivplatsen.
- Student får lämna salen tidigast en timme efter tentamens start.
- Vid toalettbesök eller rökpaus ska pauslista utanför salen fyllas i.
- All form av kontakt mellan studenter under tentamens gång är strängt förbjuden.
- Böcker och anteckningssidor kan komma att granskas av assistent, vakt eller examinator under tentamens gång.
- Frågor om specifika uppgifter eller om tentamen i stort ska ställas via tentasystemets kommunikationsklient.
- Systemfrågor kan ställas till assistent i sal genom att räcka upp handen.
- Endast uppgifter inskickade före tentamnenstidens slut rättas.
- Ingen uppgift kan kompletteras under tentamens sista kvart.
- En uppgift kan som regel kompletteras tills den är antingen "Godkänd" eller "Underkänd". En uppgift bedöms som "Underkänd" om ingen markant förbättring skett sedan tidigare inlämning.
- Kompilerande kod, fullständig kravuppfyllnad och följande av god stil och goda konventioner enligt god programmeringssed är krav för att en uppgift ska bedömas "Godkänd".

Hjälpmedel	En C++-bok (t.ex. C++ Primer 5th ed.) En A4-sida med egna anteckningar

Information

Betygssättning vid tentamen

Tentamen består av fem uppgifter på varierande nivå. Uppgifter som uppfyller specifikationen och följer god sed och konventioner ges omdömet "Godkänd". Annars ges omdömet "Kompletteras" eller "Underkänd". Tentamen kräver två godkända uppgifter för betyg 3. Alla betygsgränser ses i tabell 1. *För betyg 3 har du alltid hela tentamenstiden, Varken mer eller mindre.* (För student som från LiU fått rätt till förlängd skrivtid förlängs betygsgränserna i proportion till den förlängda skrivtiden.)

Tid	Lösta uppgifter	Betyg
3 h ±	3	5
4 h ±	4	5
4 h ±	3	4
5 timmar	2	3

Tabell 1: Betygsättning vid tentamen

Bonus och tillgodoräknanden (±)

Gäller inte vid denna tentamen.

Inloggning

Logga in på tentamenskontot med följande användaruppgifter:

```
Användarnamn: examx
Lösenord ges på whiteboard.
```

Följ menyvalen så långt det går tills du ska mata in ett engångslösenord. Tag fram ditt LiU-kort och visa det för assistent eller vakt i sal för att få detta lösenord.

Skrivbordsmenyn

Genom att högerklicka på skrivbordet (bakgrunden) får du fram skrivbordsmenyn. Där finns alternativ för att starta tentaklienten, en terminal, emacs och (som experimentell beta) textbaserad tillgång till cplusplus.com referenssidor.

När du är inloggad är det viktigt att du direkt startar tentaklienten genom att högerklicka på bakgrunden och välja "tentaklient" i menyn.

Terminalkommandon

w++11 används för att kompilera med "alla" varningar.

g++11 används för att kompilera utan varningar.

w++11filter används för att köra *w++11* med gccfilter.

g++11filter används för att köra *g++11* med gccfilter.

bcheck ./a.out används för att leta minnesläckor.

cppreference startar en terminal med textbaserad tillgång till cplusplus.com referenssidor.

C++ referenssidor

Detta är en "experimentell beta" som provades första gången på duggan i TDP004 2013-12-09 och ger textbaserad tillgång till C++ referenssidor på cplusplus.com. Det går bara att nå referenssidorna på cplusplus.com, forumsidorna liksom resten av internet är avstängt. Om det inte fungerar hänvisas du till boken.

Använd piltangenterna och enter för att navigera. Numeriska tangentbordets "Page up" och "Page down" scrollar upp eller ned (toggla "Numlock" om det inte fungerar). Tryck "Esc" för att få fram en meny, och "/" för att söka på en sida. Mer kommandon finns i hälpmenyn under alternativet "Keys".

Givna filer

Eventuella givna filer finns i katalogen "given_files". Denna underkatalog är skrivskyddad, så det är ingen risk du råkar ändra på dessa filer. Skrivskyddet gör dock att du måste kopiera in de givna filer du vill använda till tentakontots hemkatalog. Hur du listar och kopierar filer ska du kunna. Hemkatalogen står du i från början, och du kommer alltid tillbaka till den genom att bara exekvera kommandot "cd" i terminalen. Hemkatalogen heter alltid "/home/student_tilde" om du undrar.

Avslutning

Avsluta alla öppna program och tryck på knappen märkt "Exit" i menyn längst ner på skärmen och välj "ok". Vänta ett tag och tryck sedan på knappen "Avsluta tentamen" när det är möjligt. När detta är gjort är det omöjligt att logga in igen. Lämna inte datorn förrän du ser den vanliga inloggningsskärmen med blå bakgrund. Anmäl till assistent eller vakt om inloggningsskärmen inte dyker upp inom en minut så åtgärdar vi problemet.

Uppgift 1 IPv6

I denna uppgift skall du visa att du kan skapa och ärva klasser enligt alla konstens regler.

Inledning

Alla enheter anslutna till Internet har ett unikt identifieringsnummer kallat IP-adress. Traditionellet består den av fyra 8-bitars tal. Det ger drygt 4 miljarder adresser och är inte på långa vägar tillräckligt för The Internet of Things, när allt ska kopplas upp. Därför består en adress i IPv6 av åtta 16-bitars tal. Nu finns adresser så det räcker till alla atomer på jordens yta. Det är lite mer rimligt än den gamla versionen! En IPv6-adress skriv ut med åtta 4-siffriga hexadecimala tal separerade med sju kolon. I full version skrivs alla nollor ut. I förkortad version slopas alla inledande nollor.

```
Fullt exempel: 2001:0db8:0000:0000:0000:0000:1428:07ab
Förkortat:     2001:db8:::::1428:7ab
```

Funktionella krav

Skriv en klass som representerar en IPv6-adress. Ett huvudprogram för att testa klassen är givet och skall fungera omodifierat.

- Det skall finnas en konstruktor som tar emot en sträng i antingen full eller förkortad form (båda formerna skall klaras). Konstruktorn måste garantera att klassinstansen som skapas innehåller en korrekt adress. Om något av de åtta talen som utgör en adress saknas eller är felaktigt ersätts det helt enkelt med noll.
- Det skall finnas en utskriftsoperator som skriver ut adressen i full form.
- Det skall finnas en subklass, IPv6_Abbr, som är identisk i allt utom att utskrift sker i förkortad form.

Ickefunktionella krav

- Du ska lagra de åtta talen som utgör en adress i en standardkontainer; std::array, std::vector eller std::list och använda C++ fullt ut för att undvika duplicerad kod.
- Du ska använda polymorfi för att avgöra vilken utskrift som ska ske.
- Utdata från ditt program skall exakt överensstämma med givet exempel.

Körexempel (indata i kursiv stil)

```
2001:0db8:0000:0000:0000:0000:1428:07ab
2001:0db8:0000:0000:0000:0000:1428:07ab  <==>  2001:db8:::::1428:7ab
2001:db8:::::1428:7ab
2001:0db8:0000:0000:0000:0000:1428:07ab  <==>  2001:db8:::::1428:7ab
07ab:0000:0db8:0000:1428:0000
07ab:0000:0db8:0000:1428:0000:0000:0000  <==>  7ab::db8::1428:::
::
0000:0000:0000:0000:0000:0000:0000:0000  <==>  :::::::
7ab::db8::1428:
07ab:0000:0db8:0000:1428:0000:0000:0000  <==>  7ab::db8::1428:::
pqr:0:5:tre:76:
0000:0000:0005:0000:0076:0000:0000:0000  <==>  ::5::76:::
```

Uppgift 2 MasterMind

I denna uppgift visar du att du kan använda C++ funktioner och standardkomponenter för att fullborda ett litet spel.

Inledning

MasterMind är ett enkelt spel som går utt på att motståndaren skall lista ut din hemliga färgkod genom strategiska gissningar. Efter varje gissning måste du uppge hur nära det var enligt reglerna.

Funktionella krav

Skriv ett enkelt MasteMind-liknande spel. Programmet skall slumpa fram en 4-siffrig hemlig kod och därpå låta användaren gissa. Gissningar som innehåller fler eller färre än 4 tecken ignoreras. Efter varje gissning skriver programmet ut hur många av siffrorna som har både rätt värde och står på rätt plats, samt hur många av de övriga som har rätt värde men står på fel plats. När gissningen till slut är helt korrekt skriver programmet ut hur många gissningar som krävdes för att vinna.

Exempel: Om den hemliga koden är 1234 och gissningen är 1332 så har sifferposition ett och tre i gissningen rätt värde på rätt plats, och sista siffran i gissningen har rätt värde men sitter på fel position. Sifferposition två är helt fel. Notera att om gissningen varit 2022 skulle fortfarande enbart en av tvåorna i gissningen vara rätt siffra på fel plats, de övriga tre siffrorna skulle vara helt fel.

Ickefunktionella krav

- Du skall skriva en funktion som tar reda på hur många siffror i en gissning som är helt korrekta.
- Du skall skriva en annan funktion som tar reda på hur många av de övriga siffrorna i en gissning som enbart sitter på fel plats.
- För att underlätta testning skall ditt program börja med att skriva ut den hemliga koden.
- Utdata från ditt program (givet samma kod och indata) skall överensstämma med körexempel, men ske på nästa rad med 5 blankstegs indrag.

Körexempel (indata i kursiv stil)

```
9902
A secret number is generated, enter your guesses:
1234     0 correct and 1 in wrong place!
5678     0 correct and 0 in wrong place!
2222     1 correct and 0 in wrong place!
9999     2 correct and 0 in wrong place!
9329     1 correct and 2 in wrong place!
4299     0 correct and 3 in wrong place!
9912     3 correct and 0 in wrong place!
9902     4 correct and 0 in wrong place!
Correct! You won in 8 rounds!
```

Utskrifterna i exemplet är uppflyttade en rad för att spara plats.

Uppgift 3 Egenheter med 999

I denna uppgift visar du bland annat att du kan hantera kommandoradsargument och fånga standardfel.

Inledning

En viss tidskrift avslöjar på pysselsidorna en intressant matematisk egenhet. Det påstås att du alltid kommer att få summan 999 om du tar ett tal, vilket som helst, multiplicerar det med 999 och sedan delar upp produktens siffror i grupper om tre och slutligen adderar grupperna. Vi provar:

```
        67 * 999 =          66 933                    66 + 933 = 999
     1 045 * 999 =       1 043 955                 1 + 43 + 955 = 999
33 895 899 * 999 = 33 862 003 101           33 + 862 + 3 + 101 = 999
```

Det verkar faktiskt stämma! Hur är det möjligt?

Funktionella krav

Skriv ett program som testar denna matematiska egenhet (motbevisar den). Ditt program ska ta in exakt ett heltal N via kommandoraden. Sedan skall de N första talen där egenheten inte stämmer skrivas ut. Utdata skrivs med ett tal per rad, inklusive både produkt och summa. Om N saknas eller är felaktigt skall undantag fångas och fel hanteras enligt körexempel.

Ickefunktionella krav

- Du måste använda *std::stoi* för att konvertera kommandoradsargumentet till heltal.
- Du får inte skriva all kod i huvudprogrammet, minst en lämplig funktion måste användas.

Tips och begränsningar

- Notera att produkten i exemplet är för stor för att rymmas i en vanlig heltalsvariabel. Du behöver *INTE* hantera tal större än *intmax* i ditt program. Vi testar enbart med små N.
- Du kanske kommer att notera en egenhet relaterad till summan för de tal där ursprungliga egenheten inte stämmer.
- Formatet på utskrifterna ska överensstämma exakt med körexempel, men det är okej att använda tusentalsavgränsare.

Körexempel (kommandoraden visas med inledande dollartecken)

```
$ a.out
Usage: a.out COUNT
$ a.out abc
ERROR: 'abc' is not a number
$ a.out 10 20
Usage: a.out COUNT
$ a.out 100001
// To save paper the first 100000 lines are skipped here
// Your program should of course print them all
33896896 * 999 = 33862999104 -> 1998
```

Uppgift 4 Kjellsortering

I denna uppgift visar du att du kan hantera STL och filer.

Inledning

Källsortering är viktigt och bra för miljön. Kanske går det att skriva ett program som hjälper oss källsortera? Det finns ju redan en sorteringsalgoritm som åtminstone låter passande: shellsort! I denna uppgift skall vi dock begränsa oss till ett program som bland annat kan Kjell-sortera.

Funktionella krav

Skriv ett program som frågar efter namnet på en fil och förnamnet på en person. Programmet kommer att utgå från att filen innehåller rader med exakt ett förnamn och ett efternamn; ett helt namn per rad. Läs in alla rader från filen där förnamnet matchar indata till programmet och skriv sedan ut alla matchande namn i sorterad ordning.

Ickefunktionella krav

- Du måste lösa uppgiften genom att använda STL-behållare och STL-algoritmer så långt det är möjligt.
- Utdata måste överensstämma med körexempel.

Tips och begränsningar

- Du kan anta att varje rad i filen innhåller exakt två strängar.
- Det är behändigt om indata till programmet kan ges på kommandoraden, men inget krav.

Körexempel (i detta fall används kommandoradsargument)

```
$ a.out given_files/one_name.txt Kjell
Kjell Kjellsson

$ a.out given_files/one_name.txt Max

$ a.out given_files/names.txt Anna
Anna AaaJustOnce
Anna Adolfsson
Anna Carlsson
Anna Henriksson
Anna Isaksson
Anna Lundberg
Anna Magnusson
```

Uppgift 5 Tusentalsavgränsare

I denna uppgift visar du att du kan skapa en liten programkomponent som är återanvändbar i andra program.

Inledning

Riktigt stora tal är svåra att läsa och få grepp om när de skrivs ut utan tusentalsavgränsare, och olika länder har olika standarder för hur tusental avgränsas.

Funktionella krav

Skriv en programkomponent som kan användas för att skriva ut riktigt stora tal med tusentalsavgränsare. Både positiva och negativa tal skall hanteras. Tusentalsavgränsare skall kunna anges, men om den inte anges skall blanktecken användas som standard.

Givet är ett testprogram som läser in heltal tills användaren väljer att avsluta inmatningen med Ctrl-D. Användaren skall kunna mata in både positiva och negativa heltal som är riktigt stora, upp till 18 decimala siffror.

Ickefunktionella krav

- Det skall vara en programkomponent (t.ex. en klass, eller en funktion), inte flera.
- Utdata måste överensstämma med körexempel.

Tips och begränsningar

- Datatypen “signed long long int” räcker.
- Användning av “using” eller “typedef” uppmuntras.
- Skulle programkomponenten kunna passa i någon annan av uppgifterna?

Körexempel (indata i kursiv stil)

```
Enter numbers to format, finish with Ctrl-D
0
'0' == "0"
-1
'-1' == "-1"
12624
'12 624' == "12.624"
-213
'-213' == "-213"
123456
'123 456' == "123.456"
-876254
'-876 254' == "-876.254"
123456789012345678
'123 456 789 012 345 678' == "123.456.789.012.345.678"
Done.
```