TDP004 - Tentamen

2014-04-23

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:     kluring1
```

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 Plotta ASCII

I denna uppgift skall du visa att du behärskar användning av färdiga klasser och objekt från C++ standardbibliotek för att lösa ett problem.

Inledning

På filerna "given_files/smiley.txt" och "given_files/coordinates.txt" finns rader med två positiva heltal på varje. Du är nyfiken på vad det kan vara och bestämmer dig för att skriva ett program som tolkar varje rad som koordinater för tecknen i en ascii-figur.

Funktionella krav

Skriv ett program som låter användaren skriva in data motsvarande den på filerna och plottar detta. Tolka första talet på varje rad som kolumn och andra talet som rad. Antag att raderna i terminalen numreras med start på 1 och ökande nedåt, och kolumnerna numreras med start på 1 och ökande åt höger. Plotta sedan alla talpar som matas in till ditt program enligt denna tolkning. Koordinater som förekom i indata representeras med en "brädgård" (#), annars används blanktecken.

Ickefunktionella krav

- Du får inte lagra ohemult mycket data i datorns arbetsminne (inte mer än ungefär så många koordinater som matas in).
- Du ska använda std::vector och std::pair för att lagra alla koordinater.
- Du får inte göra några antaganden om i vilken ordning koordinater matas in, eller maximala värden på koordinaterna, de kan vara godtyckligt stora.
- Utskrifterna ska matcha körexemplen till fullo.

Tips och begränsningar

- Inga meddelanden till användaren behövs.
- Ingen felhatering behövs, indata förutsätts korrekt.

Körexempel (nedre delen av figuren klippt av utrymmesskäl)

```
./a.out < given_files/smiley.txt
                     ##########
                   ##          ##
      #####       ##              ##
     #     #    ##     ##    ##     ##
     #     #   ##      ##    ##       ##
     #    #  ##        ##    ##        ##
      #   #   #                        ##
    ############                       ##
   #            # ##             ##     ##
  ##            #  ##            ##     ##
 ##   ###########    ##        ##       ##
 #               #     #######         ##
 ##              #                    ##
```

Uppgift 2 Puzzle

I denna uppgift visar du att du kan lägga till rekursiv funktionalitet i givna klasser och objekt för att lösa ett problem, samt korrekt hantera minne och pekare.

Inledning

Ett litet spel består av en rad golvbrytare som antingen är “på” (1) eller “av” (0). Efter brytaren längst till höger finns en låst dörr som endast öppnas om alla brytarna är i läge “på”. Du startar längst till vänster, framför den första brytaren, och har som mål att komma igenom dörren genom att gå fram och tillbaka över brytarna. Varje gång du går på en brytare byter den läge från “på” till “av” eller vice versa.

Funktionella krav

Komplettera programmet “given_files/puzzle.cc” så det detekterar när spelet är vunnet korrekt. Spelet vinns när spelaren står på sista brytaren och går framåt om alla brytare då är “på”.

Ickefunktionella krav

- Din lösning ska vara rekursiv.
- Utöka givna klasser som lämpligt. Fria funktioner tillåts inte.
- Korrigera minneshanteringen. Minnesläckor tillåts inte.

Körexempel

```
Switches in "off" state show as '0'.
Switches in "on" state show as '1'.
You show as 'i' if you are on an "on" switch.
You show as 'o' if you are on an "off" switch.
You can enter many commands at once.
Press 'Enter' after your command(s).
1100011011: forward or back (f/b)? f
o100011011: forward or back (f/b)? f
0o00011011: forward or back (f/b)? b
i000011011: forward or back (f/b)? f
1i00011011: forward or back (f/b)? f
11i0011011: forward or back (f/b)? f
111i011011: forward or back (f/b)? f
1111i11011: forward or back (f/b)? f
11111o1011: forward or back (f/b)? f
111110o011: forward or back (f/b)? b
11111i0011: forward or back (f/b)? f
111111i011: forward or back (f/b)? f
1111111i11: forward or back (f/b)? f
11111111o1: forward or back (f/b)? f
111111110o: forward or back (f/b)? b
11111111i0: forward or back (f/b)? f
111111111i: forward or back (f/b)? f
You win!
```

Uppgift 3 Shift-vector

I denna uppgift visar du att du kan skapa klasser och objekt och hantera dem tillsammans med C++ standardbibliotek.

Inledning

Spelet 2048 består av ett 4x4 bräde som startar med två utslumpade tal, 2 eller 4. Du kan sedan skifta nummer upp, ned, höger eller vänster så länge brädet ändras. Efter varje drag slumpas en ny 2:a eller 4:a ut på en tom position.

I denna uppgift skall vi fokusera på hur skiftningen till vänster går till i en förenklad variant. Detta löser vi för en godtyckligt lång 1xN-vektor. Vi kallar operationen en "left_shift" av vektorn. Om några positioner är tomma (representeras med 0) så tas dessa bort och sätts in längst till höger. Om två intilliggande positioner är lika så blir den ena summan av de två och den andra tas bort och en tom position sätts in längst till höger. Intilliggande par längst åt vänster behandlas först. Tal som vid start av skiftningen separeras av tomrum slås inte ihop. Studera följande exempel:

```
[ 0 2 2 2 0 2 0 2 ] // Startvektor
[ 4 2 2 2 0 0 0 0 ] // Efter en vänsterskiftning
[ 4 4 2 0 0 0 0 0 ] // Efter en andra vänsterskiftning
[ 8 2 0 0 0 0 0 0 ] // Efter en tredje vänsterskiftning
```

Ickefunktionella krav

Implementera en klass "shift_vector" som representerar en skiftbar heltalsvektor. Klassen skall förutom grundfunktionalitet för en vektor ha en medlemsoperation som utför operationen "left_shift" enligt ovan. Utskriften för givna exempel skall matcha.

Funktionella krav

Skriv ett program som läser en rad heltal (eller till Ctrl-D) och lagrar talen i en "shift_vector". Ingen felhantering behövs. Skriv sedan ut och skifta vektorn så många gånger som antalet tal i den.

Tips och begränsningar

- Du får i detta fall ärva från standardklassen "vector" för heltal om du vill.
- Det beskrivna beteendet vi implementerar är inte helt överensstämmande med spelet 2048.

Körexempel (inledande raden är det som matas in)

```
 32   2   0   2   4   0   8  16  32  16  16 Ctrl-D
 32   2   0   2   4   0   8  16  32  16  16
 32   2   2   4   8  16  32  32   0   0   0
 32   4   4   8  16  64   0   0   0   0   0
 32   8   8  16  64   0   0   0   0   0   0
 32  16  16  64   0   0   0   0   0   0   0
 32  32  64   0   0   0   0   0   0   0   0
 64  64   0   0   0   0   0   0   0   0   0
128   0   0   0   0   0   0   0   0   0   0
```

Fyra upprepningar av sista raden bortklippta av utrymmesskäl.

Uppgift 4

I denna uppgift visar du att du kan skapa en generellt användbar klass.

Inledning

En studentassistent måste hantera både egna studier och undervisningsrelaterade uppgifter så som labrättning och lektionsförberedelse. Vår assistent har som mål att utföra alla aktiviteter som inte är fast schemalagda under dagtid (8-18). Vår assistent har ett script som sparar en dags aktiviteter till en enkel textfil (se "given_files/schedule1.txt") för senare bearbetning.

Ickefunktionella krav

- Implementera en klass som representerar en aktivitet.
- Din klass måste kräva full initiering när en instans skapas.
- Två aktiviteter ska vara jämförbara med den vanliga "<"-operatorn. En aktivitet som slutar innan en annan aktivitet börjar räknas då som mindre.
- Din klass ska även ha en medlemsfunktion "time_to" som beräknar hur många minuter det är från slutet av aktiviteten fram till en annan aktivitet börjar.
- Klassen ska ha en utmatningsoperator (<<) som skriver ut aktiviteten på samma format som aktiviteterna i "given_files/schedule1.txt".

Funktionella krav

Skriv ett program som testar alla delar av din klass till fullo. Ingen felhatering krävs.

Tips och begränsningar

- Läs nästa uppgift innan du börjar.

Uppgift 5

I denna uppgift visar du att du kan använda en egen klass och standardkomponenter i ett litet objektorienterat program.

Inledning

Se “Inledning” i föregående uppgift.

Funktionella krav

Skriv ett program som läser in alla aktiviteter från en fil, sorterar dessa, skriver ut aktivitetslistan och slutligen skriver ut de två aktiviteter med längst ej schemalagd tid emellan. Filnamnet skall anges på kommandoraden och programmet skall ge tydliga felmeddelanden om ingen fil anges eller om den inte kan öppnas.

Ickefunktionella krav

- Programmet skall vara objektorienterat med minst en egen klass för att representera en aktivitet.
- Använd standardkomponenter från STL för att lösa delproblem.
- Utskrifterna ska matcha körexemplen till fullo.

Tips och begränsningar

- Läs (läs lös) föregående uppgift innan du börjar.
- Du kan förutsätta att filen endast innehåller korrekta data separerade med godtyckligt antal blanksteg (starttid, bindestreck, sluttid och beskrivning).
- Antag att filen innehåller information om dagens start och slut.

Körexempel

```
./a.out given_files/schedule1.txt
8:00 - 8:00  Workday start
8:15 - 10:00  Lecture
10:15 - 12:00  Lesson
12:00 - 13:00  Lunch
14:00 - 15:00  Seminar
15:13 - 16:00  Group meeting
17:45 - 18:00  Daily worklog
18:00 - 18:00  Workday end
Largest consecutive block of unbooked time is between:
15:13 - 16:00  Group meeting
17:45 - 18:00  Daily worklog
```