TDP004 - Tentamen

2017-12-01

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 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 tentamenstidens 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. Stroustrup) Ett A4-ark med egna anteckningar

Information

Betygssättning vid tentamen

Tentamen består av ett antal uppgifter på varierande nivå. Uppgifter som uppfyller specifikationen samt 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 och . *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**	**Tillgodo**
4 h	3	5	inget
2.5 h	2	4	inget
4 h	2	3	inget
4 timmar	1	-	löst uppgift

Tabell 1: Betygsättning vid förtentamen (dugga), 4 uppgifter ges

Tid	**Lösta uppgifter**	**Betyg**
3 h $+B$	3	5
4 h $+B$	4	5
4 h $+B$	3	4
2 h $+B$	2	4
5 timmar	2	3

Tabell 2: Betygsättning vid sluttentamen, 5 uppgifter ges

Bonustid ($+B$)

Bonustid ges endast vid första ordinarie tentamen. Varje bonus som samlats in i kursen ger 5 minuter extra för gränserna till högre betyg (symboliseras med B i tabell).

Tillgodoräknanden

Tillgodoräknanden gäller endast under den första ordinarie tentamen i samband med kursen. Om du på förtentamen endast lyckas lösa en uppgift kan du tillgodoräkna motsvarande uppgift på sluttentamen (se tabell 1). Du har då bara en uppgift kvar till betyg 3. För högre betyg (om du löser mer än en uppgift på förtentamen) kan du inte tillgodoräkna något. Om du på sluttentamen löser en andra uppgift snabbt och vill sikta på högre betyg kan du lösa den tillgodoräknade uppgiften "igen", men den räknas endast mot högre betyg, **inte** som andrauppgift för betyg 3.

Inloggning

Innan du loggar in ska du välja "Exam system" från sessionsmenyn i nedre vänstra hörnet av inloggningsrutan. Därpå loggar du in med dina LiU inloggningsuppgifter. Du kommer nu in i tentainloggningen och ska börja med att välja språk. Ta den flagga som ser minst Engelsk ut så blir det Svenska (valet spelar ingen roll för dig som kan båda språken). Följ instruktionerna på skärmen 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.

Skrivbordsmiljön

När du kommit in i tentasystemet har du en normal skrivbordsmiljö (Mate-session i Linux Mint) med grön skrivbordsbakgrund. Efter en stund kommer även din kommunikationsklient att dyka upp automatiskt. Startmenyn är nedskalad till enbart det som examinator bedömt relevant. Andra program kan fortfarande finnas tillgängliga genom att starta dem från en terminal. Observera att en del program som använder nätverkstjänster inte fungerar normalt, eftersom nätverket inte är åtkomligt.

När du är inloggad är det viktigt att du har tentaklienten igång hela tiden. Om den inte dykt upp fem minuter efter inloggning och inte heller när du startar den manuellt från menyn (fisken) tar du kontakt med assistent eller vakt i sal.

Terminalkommandon

`e++17` används för att kompilera med "alla" varningar *som fel*.
`w++17` används för att kompilera med "alla" varningar. **Rekommenderas.**
`g++17` används för att kompilera `utan` varningar.
`valgrind --tool=memcheck` används för att leta minnesläckor.

C++ referenssidor

På tentan har du *experimentiell* tillgång till referenssidorna på `http://www.cppreference.com/` via webbläsaren Chrome. Starta `chromium-browser` i terminalen eller välj lämpligt alternativ från startmenyn. Observera att allt utom referenssidorna är avstängt. Om du inte kan komma åt en sida du tycker hör till referenssidorna (som kanske blockerats av misstag) kan du skicka ett meddelande via tentaklienten. Då lösningen är på experimentell nivå kan det hända att problem uppstår, t.ex. att proxyn inte går att nå. Tag då hjälp av assistent i sal.

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

När dina uppgiftsbetyg och ditt slutbetyg i kommunikationsklienten stämmer överens med det du förväntar och du är nöjd, eller när tentamenstiden är slut, är det dags att logga ut. Hinner du inte se ditt betyg får du höra av dig till examinator via epost efter tentamen.

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 – Klasshierarki

På filen `given_files/game.cc` finns ett program som ritar ut en tänkt spelvärd. Ditt jobb är att skapa en klasshierarki för att representera objekt att rita ut i spelet. Det ska finnas två konkreta klasser, `Rock` och `ZigZag` som har olika symboler och rörelsemönster. De ska ha en gemensam abstrakt basklass `GameObject` some du ska skapa. Samtliga spelobjekt har en position (av den givna typen `Coord`) och en symbol. Följande funktioner ska finnas för samtliga spelobjekt:

Coord position() Ger objektets nuvarande position

char symbol() Ger objektets symbol för utritning

void update() Uppdaterar objektets position enligt nedanstående beskrivning

De olika spelobjekten rör sig i olika mönster. Ett objekt av typen `Rock` förflyttar sig diagonalt ett steg varje uppdatering (i ökande x- och y-koordinat) medan `ZigZag` rör sig i ett sicksackmönster, dvs x-koordinaten ökar konstant med 1 medan y-koordinaten växlar mellan att öka med 1 och minska med 1. Spelvärlden ska applicera så kallad wrap-around-teknik, dvs när ett objekt kommer utanför någon gräns ska det börja om på andra sidan. Om vi antar att spelvärlden har 26 rutor i x-led (indexerade 0-25) och ett objekt av typen `Rock` står på position (25,14) ska nästa position vara (0,15).

`Rock` ska symboliseras med bokstaven `O` och `ZigZag` med `X`.

Du behöver inte ta hänsyn till eventuella kollisioner mellan objekten.
Skriv din kod på två nya filer `game_objects.h` och `game_objects.cc`.
Filen `given_files/game_settings.h` innehåller hjälpfunktioner för att hantera wrap-around som du kan använda dig av. För att kompilera din kod kan du köra följande kommando från din hemkatalog

```
make -f given_files/game.make
```

Uppgift 2 – Tidtagning

Ofta när man vill undersöka effektiviteten av kod vill man veta hur snabb (eller långsam) en viss funktion är. I denna uppgift ska du skapa en klass `Timer` som håller koll på hur länge det funnits och skriver ut denna information när det destrueras. Man kan då skapa ett `Timer`-objekt i början av funktionen och sedan skrivs tiden det tog att köra funktionen ut i slutet. För att lösa detta ska du använda standardbibliotekets funktionalitet från `<chrono>`. Nedan beskrivs väl valda delar som kan vara till hjälp:

steady_clock En klocka, håller koll på tiden sedan den startades och nuvarande tid kan fås med den statiska medlemsfunktionen `now()` (av typen `time_point`)

duration En typ som representerar skillnaden mellan två `time_point`-objekt. Finns i flera varianter och man kan använda `duration_cast` för att omvandla mellan dem. Har medlemsfunktion `count` för att få ut värdet.

Nedan ges ett kort exempel på tidtagning:

```
#include <chrono>
#include <iostream>
using namespace std;
using namespace std::chrono; // tidshantering ligger separat
int main()
{
    auto tp = steady_clock::now();
    // some complicated calculations we want to time
    // ...
    cout << "Total time: "
         << duration_cast<milliseconds>(steady_clock::now() - tp).count();
}
```

Följande medlemsfunktioner ska `Timer` ha:

tick() Nollställer den interna klockan

tock() Ger antal millisekunder sedan start eller senaste nollställning

Konstruktor Ska ta emot en sträng (beskrivning) som ska skrivas ut innan åtgången tid vid destruktion. Ska även kunna anropas utan sträng.

Destruktor Skriver ut eventuell beskrivning samt tiden objektet funnits (eller tiden sedan senaste omstart) på `cout`.

Det ska inte vara tillåtet att kopiera eller flytta `Time`-objekt.

Skapa även ett litet testprogram som testar funktionaliteten av din klass.

Uppgift 3 – Viskleken

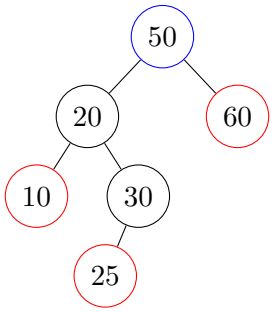
I leken ”Viskleken” börjar en person genom att viska en mening i någon annans öra. Denna ska sedan viska det vidare till någon annan och efter flera led ska den sista berätta vad den hört. I denna uppgift ska du, med hjälp av standardbiblioteket, simulera en kort session (med endast tre deltagare) av viskleken.

1. Läs in en samling ord från `cin` till en lämplig sekvenscontainer (senare kallad `words`). Detta motsvarar första viskningen.
2. Första mottagaren är glömsk och tappar därför slumpmässigt bort en sjundedel av orden i meddelandet. Simulera detta enligt följande algoritm:
   (a) Skapa en `vector<size_t>` och fyll denna med alla index från `words`, dvs värdena 0..`words.size()-1`.
   (b) Blanda index-vektorn slumpmässigt
   (c) Räkna ut hur många värden som motsvarar en sjundedel (detta värde kallas N nedan)
   (d) Sortera de N första talen i index-vektorn i sjunkande ordning
   (e) För de N första talen i vektorn, ta bort elementet på det indexet i words
3. Nästa person har lite dålig hörsel och tappar lätt stavelser ur långa ord. Detta ska utföras för varje ord i `words`:
   (a) Gör om ordet till endast små bokstäver.
   (b) Om ordet är längre än 7 tecken: hitta den sista vokalen i ordet och plocka bort denna vokal och alla efterföljande tecken. Du kan anta att det endast är engelska ord (med vokalerna `"aeiouy"`). Du behöver inte ta hänsyn till eventuella interpunktionstecken.
4. Skriv ut varje ord i `words` följt av ett mellanslag på cout.

Uppgift 4 – Träd

Ett binärt träd är en länkad datastruktur där varje nod innhåller pekare till två andra noder. Man brukar säga att en nod är förälder till två andra noder som ligger till vänster eller höger. Roten ligger överst och de noder som inte har barn kallas lövnoder. Se figur 1 för exempel.

Ett binärt sökträd är ett binärt träd med bivillkoret att en nods vänstra delträd alltid är mindre än noden och höger delträd alltid har värden som är större.



Figur 1: Binärt sökträd, rotnoden är blå och lövnoder är röda

I filerna `given_files/Tree.h` och `given_files/Tree.cc` finns en implementation av ett binärt sökträd, det saknas dock en viktig funktion; `insert`. Din uppgift är att skapa funktionen `insert` som ska stoppa in ett nytt värde i trädet. Trädet ska såklart fortfarande vara ett binärt sökträd efter insättning. Funktionen `print` kommer alltid skriva ut värdena i sorterad ordning om du gjort rätt. Du kan anta att vi endast stoppar in unika värden i trädet.

TIPS: Eftersom vi har unika värden kommer ett nytt värde **alltid** läggas in som en lövnod.

Filen `given_files/tree-test.cc` innehåller ett litet testprogram som bör skriva ut värdena 1-7 i ordning om du gjort rätt. Ett tips kan också vara att använda valgrind för att undersöka eventuell minnesläcka.