TDIU20 – Tentaregler

Inloggning

Logga in i tentasystemet genom att välja session ”exam system” och logga in med ditt vanliga LiU-ID. Välj inte att ha denna session som standardsession. Verifiera att dina uppgifter stämmer och förbered din tentaplats. Som vanligt är det inte tillåtet att ha väskor eller jackor vid sin skrivplats och mobiltelefoner ska ligga avstängda i jacka eller väska. Ta fram ditt LiU-kort och invänta tentavakt för att få ett engångslösenord.

Hjälpmedel

Följande får tas med på tentan:

- En bok om c++. För boken gäller följande regler:
  - Kommentarer/noteringar som direkt rör text och exempel på sidan i fråga får finnas i sidmarginalen.
  - Egna sidflikar för att enkelt kunna hitta t.ex. de olika kapitlen är tillåtna.
  - Inga extra ark eller lappar, lösa eller fastsatta, får finnas.
  - Tomma sidor, insidan av pärmarna, försättsblad, etc., får inte innehålla programkod.
- Ett A4-ark med valfritt innehåll på vardera sidor (går ej att ersätta med två enkelsidiga ark).
- Penna för att anteckna under tentan. Ni kommer förses med blanka papper

Följande får INTE tas med:

- Elektroniska hjälpmedel såsom miniräknare och mobiltelefon

Utloggning

När du är nöjd med ditt betyg (som står i tentaklienten) kan du avsluta och logga ut som vanligt i menyn. Klicka sedan på ok följt av knappen avsluta tentan. Observera att du när du gjort detta inte kan logga in igen. Lämna inte din plats innan vanliga inloggningsskärmen syns.

Tentaregler

Tentan består av fyra uppgifter klassificierade som grundnivå eller avancerad. För godkänt betyg på tentan krävs lösning av en uppgift på grundnivå. För betyg 4 krävs dessutom lösning av en avancerad uppgift och för betyg 5 krävs en grunduppgift samt båda avancerade uppgifterna.
För att en uppgift ska anses godkänd krävs följande:

- att man noga följt alla instruktioner och krav ställda i uppgiften
- din kod följer god programmeringsstil (se labseriens rättningsguide)
- att klasser har ett tydligt ansvar och funktioner har en väl definierad uppgift
- att din kod har bra inkapsling och resurshantering

Bonus från labserien

Om du har bonus från labserien minskas antalet avancerade uppgifter som krävs med ett, dvs för betyg 4 krävs endast en grunduppgift och för betyg 5 krävs en av varje. Bonusen är endast giltig det år den erhölls.

Frågor om uppgifter

Frågor om tentan i stort eller uppgiftspecifika frågor ska ställas via tenta-klienten. Detta för att vi ska ha en historik av konversationen samt för att vi ska kunna ge samma hjälp till olika studenter.

Systemfrågor

Om du har systemfrågor som t.ex. problem med tentaklienten eller terminalen räcker du upp handen så kommer en assistent och hjälper till.

C++ referens

Det finns tillgång till valda delar av `cppreference.com`. Du måste starta webbläsaren via menyn för att komma åt sidan.

Alias för kompilering

Det finns tre alias att använda sig av för kompilering med c++17:
`g++17` Kompilering utan varningar.
`w++17` Rekommenderas!
`e++17` Kompilering med alla varningar som fel.

Uppgift 1 - Grundnivå

Kopiera filerna `given_files/{uppgift1.cc,catch.hpp}` till din hemmapp. Lägg din lösning i filerna `date.cc` och `date.h`.

I ett projekt finns funktioner för att hantera datum. Dessa funktioner hittar du i den givna filen `given_files/uppgift1.cc`. Nu har projektgruppen bestämt sig för att skriva programmet objektorienterat. Det har blivit din uppgift att skriva om datumfunktionerna till en klass som representerar ett datum samt komplettera klassen med funktionen `tomorrow()`.

Funktionen `tomorrow()` utgår från klassinstansens datum och ändrar det till dagen efter. Varken parametrar eller returvärde behövs då. Din datumklass ska stödja skottår enligt den givna implementationen, dvs dagen efter 2012-02-28 är 2012-02-29 medan dagen efter 2013-02-28 är 2013-03-01.

När du gjort klassen ska du även skapa ett program som låter användaren mata in ett datum. Programmet ska kontrollera så detta datum är korrekt och sedan skriva ut det datum som inträffar 10000 dagar efter det angivna datumet. Utskrifterna ska helt överensstämma med nedanstående körexemplen, med två siffror för månad och dag. Du kan utgå från att användaren matar in enbart siffror separerade med två enskilda bindestreck på korrekt datumformat som i exemplet.

Klassen ska vara uppdelad i korrekt deklarations- och implementationsfil med bibehållen funktionalitet, och dessutom erbjuda funktionen `tomorrow()`. Du får om du behöver lägga till egna medlemmar utöver det som finns i givna koden på villkor att klassen är fortsatt väl felhanterad och inkapslad.

Speciellt viktigt är att tänka på placering av datamedlemmar och åtföljande överväganden med avseende på inkapsling, åtkomst, skydd och korrekt initiering.

Körexempel (användarinmatning i fet stil)

```
$ ./a.out
Enter a date: 2012-02-29
10000 days later: 2039-07-17

$ ./a.out
Enter a date: 2013-13-02
Invalid date, enter another date: 2013-12-42
Invalid date, enter another date: 1900-02-29
Invalid date, enter another date: 1987-04-13
10000 days later: 2014-08-29
```

Uppgift 2 - Grundnivå

Kopiera filerna `given_files/{color.cc,catch.hpp}` till din hemmapp. Lägg till din lösning i `color.cc`.

I många grafiska applikationer behöver vi hantera färger. Dessa kan representeras i datorn med tre tal som beskriver intensiteten i vardera av primärfärgerna röd, grön och blå. De tre primärfärgerna representeras traditionellt av ett heltal i intervallet 0-255. Ibland används istället tre flyttal i intervallet 0.0-1.0 (t.ex. inom OpenGL).

Högsta värdet representerar full intensitet och lägsta värdet representerar att den primärfärgen inte ingår alls. För att representera hur genomskinlig färgen ska vara används ett fjärde tal (alpha-kanalen) med samma typ och i samma intervall som de tre primärfärgerna där värdet noll representerar fullständig genomskinlighet.

Skapa en klass för att beskriva en färg enligt färgsystemet RGBA ovan (andra färgsystem är HSV ock CMYK). Använd heltalsrepresentation. När en klassinstans skapas ska initialvärden krävas för de tre primärfärgerna. Genomskinligheten ska kunna anges vid behov. Anges den inte antas färgen vara fullständigt kompakt (ogenomskinlig). Ett felaktigt värde på någon datamedlem ska generera undantag av typen `std::range_error` från `<stdexcept>`. Speciellt ska det även gå att skapa en gråskalefärg. Då anges enbart ett värde som initierar alla tre primärfärgerna. Kompilatorn ska förbjudas att automatiskt konvertera enskilda heltal till en färg.

En färg ska gå att skriva ut på valfri utström med operatorn för formaterad utmatning. En färg skrivs ut på formatet {R, G, B}. Till exempel skrivs vitt som {255, 255, 255} och svart som {0, 0, 0}.

Två färger ska kunna adderas med plus-operatorn. Den sammansatta färgen beräknas genom att medelvärdet av vardera primärfärg beräknas. Om {100, 200, 0} adderas med {50, 80, 120} blir då färgen {75, 140, 60}. Det ska även gå att addera en färg med ett heltalsvärde (gråskalevärde). Det ska ge samma resultat som om heltalsvärdet först konverteras till en gråskalefärg och sedan adderas.

Alla givna testfall måste fungera. Inga givna testfall får modifieras. Det är viktigt att använda medlemsfunktioner och konstruktorer för att undvika koduppprepning. Det är viktigt att implementera klassen med full inkapsling. Medlemsfunktioner vars syfte är att sätta eller hämta värdet på en medlem räknas i denna uppgift som brott mot inkapslingen.

Uppgift 3 - Avancerad nivå

Kopiera filerna `given_files/{uppgift3.cc,catch.hpp}` till din hemmapp. Lägg till din lösning i `uppgift3.cc`.

Ibland vill man skapa något i begränsad upplaga eller åtminstone kontrollera att man inte skapar orimligt många kopior. I filen `given_files/uppgift3.cc` finns klassen `Resource` som vi av någon anledning inte vill råka kopiera obegränsat. I denna uppgift ska du skapa en klass `Limited_Resource` för att begränsa antalet kopior.

Klassen har en datamedlem som pekar ut en `Resource` och en datamedlem som pekar ut en `Resource_Control` som i sin tur består av datamedlemmarna `max_copies` och `current_copies`.

Ett objekt `Limited_Resource` ska använda `Resource_Control` för att hålla reda på hur många kopior det finns av resursen. Om antalet objekt som pekas ut är lika med `max_copies` ska försök till kopiering generera undantaget `std::runtime_error`. Annars ska kopiering tillåtas. Flytt ska alltid tillåtas.

Nu gäller det att i de speciella medlemsfunktionerna i `Limited_Resource` tänka efter hur `Resource` ska kopieras, hur den utpekade `Resource_Control` ska modifieras och vilka kontroller som måste göras för att inte få för många kopior av `Resource`.

Uppgift 4 - Avancerad nivå

Kopiera filen `given_files/{uppgift4.cc}` till din hemmapp. Lägg till din lösning i `uppgift4.cc`.

Ett minispel som går ut på att vårda en växt har en klass `Plant` för att representera växten. Klassen består av datamedlemmarna `size` (växtens höjd i cm) samt `water`, `air` och `nutrient` som tillsammans beskriver växtens mående. Samtliga datamedlemmar är vanliga heltal. På filen `given_files/uppgift4.cc` finns ett antal medlemsfunktioner som avgör hur växten fungerar, och ett huvudprogram för att köra spelet.

I spelet ska finns ett antal olika `Powerup` som kan påverka en växt på olika sätt genom anrop av `Powerup::apply_to` med en `Plant` som argument. Klassen `Water` ska lagra en mängd vatten, och via `Plant::give_water`föra över detta till växten när `apply_to` anropas. `Nutrient_Water` fungerar som `Water` och har dessutom en mängd näring i vattnet. Slutligen gör `Sunlight` att växten ökar sin storlek via `Plant::grow`. En `Powerup` ska även ha en funktion för att skriva ut vilken typ av Powerup det är och hur mycket vatten och näring den innehåller (när så är relevant).

Din uppgift är att färdigställa implementationen av de tre klasserna `Water`, `Nutrient_Water` och `Sunlight`. Det är speciellt viktigt att få till en lösning som bygger på objektorienterade principer och ger enkla, oberoende och självständiga klasser. Minimera mängden datamedlemmar och if-satser.

Körexempel (användarinmatning i fet stil)

```
New powerup available: Sunlight
Apply? (y/n) y
Your plant is 2 cm high and healthy

New powerup available: 2 cups of water
Apply? (y/n) y
Your plant is 2 cm high and healthy

New powerup available: Sunlight
Apply? (y/n) y
Your plant is 3 cm high and healthy

New powerup available: 3 cups of water
Apply? (y/n) y
Your plant is 3 cm high and healthy

New powerup available: Sunlight
Apply? (y/n) y
Your plant is dead.
You scored 14 points!
```