Inlämningsuppgifter, EDAF30, 2015

Det finns två deluppgifter som båda ska lösas:

1. skriv ett program för att hantera bankkonton
2. skriv ett program som simulerar en kaninkapplöpning

1 Anvisningar för redovisning

Inlämningsuppgifterna ska redovisas med en kort rapport och de program som du har skrivit. Gör så här för att lämna in inlämningsuppgiften:

1. Arkivera lösningen i en .zip-fil, med rapport (som .pdf) och kompilerbar källkod (inkludera project/solutionfiler så att programmet går lätt att bygga i den utvecklingsmiljö du har använt). Eventuella indatafiler läggs på yttersta nivån i zip-filen. Källkoden till de båda uppgifterna ska vara tydligt åtskild, till exempel i separata underkataloger.
2. Skriv ett e-brev till `edaf30@cs.lth.se` och bifoga den arkiverade uppgiften (.zip-filen). Subjectraden i brevet ska se ut så här: `inl by id1 id2`, där id1 och id2 är de identiteter ni använde för att anmäla er till laborationerna. Normalt är det din StiL-identitet, till exempel `dat12xyz` och `elt11yzyx`. För dessa blir subjectraden: `inl by dat12xyz elt11yzyx`.
   Det är viktigt att subjectraden är korrekt, den används av vårt system för att identifiera vem som lämnat in uppgiften. Skriv även era namn i klartext i brevet så att vi enkelt kan lösa eventuella fel. Om du har glömt din användaridentitet kan du hämta den från kursens hemsida.

2 Krav på uppgiften

2.1 Rapport

Uppgiften ska redovisas med en kort (ett par sidor) rapport som översiktligt presenterar din lösning. För vardera deluppgift bör följande punkter diskuteras:

1. Övergripande design: beskriv centrala klasser och funktioner, och hur dessa är relaterade till varandra.
2. En kort användarinstruktion: hur använder man programmet?
3. Brister och kommentarer: Finns det något i lösning som du i efterhand anser borde gjorts annorlunda? Andra kommentarer?

Rapporten ska lämnas in som pdf-fil.

2.2 Program

Programmet ska naturligtvis fungera och lösa den angivna uppgiften. Provkör programmet ordentligt innan du lämnar in det.

Användargränssnitt och testprogram

Programmen i båda deluppgifterna ska vara interaktiva. För att kunna testa den inre logiken i sådana program utan att behöva manuellt testa via användargränssnittet, är det lämpligt att skilja användarinteraktionen från den centrala funktionaliteten i programmet.

För att möjliggöra detta ska programmet organiseras så att det finns ett testprogram, som använder samma gränssnitt mot resten av systemet som användargränssnittet, men kör igenom en mängd olika scenarier *som om en användare hade interagerat med systemet via användargränssnittet*.

Dettta betyder att det ska finnas två huvudprogram:

1. Ett huvudprogram med interaktiv användarinteraktion.
2. Ett huvudprogram med simulerad användarinteraktion enligt en uppsättning scenarier som testar programmets funktion. Dessa scenarier kan vara kodade som funktioner i testprogrammet. Kom ihåg att testa även felaktig inmatning från användaren.

Generella krav

Programkoden i lösningen ska uppfylla följande krav:

1. Programmet ska ha en vettig design.
2. Klasser och funktioner ska ha tydligt avgränsade uppgifter och ansvarsområden.
3. Programkoden ska vara lätt att följa och förstå.
4. Koden ska vara formatterad på ett sätt som underlättar läsning. Detta innebär en vettig indentering och att raderna inte är för långa.
5. Funktions- och variabelnamn ska vara väl valda och återspegla funktionens eller variabelns innebörd.

   En tumregel, både för design och läsbarhet, är att en funktion inte får vara längre än 24 rader eller ha fler än 5–10 lokala variabler. Det finns ibland goda skäl att göra ett undantag från detta, men ofta är det bättre att dela upp en lång, komplex, funktion i några enkla hjälpfunktioner. Varje funktion ska bara göra *en* sak, gör den flera – dela upp den.

3 Rättning

Vi rättar uppgifterna så snart vi hinner, normalt inom en arbetsvecka räknat från inlämningsdag. När uppgiften är rättad får du via e-brev besked om uppgiften är godkänd eller ej. I brevet finns också en kort sammanfattande kommentar om din lösning samt i de fall uppgiften inte är godkänd kommentarer om vad som behöver förbättras. Om uppgiften inte är godkänd ska du inom rimlig tid lämna in en förbättrad version.

Deluppgift 1

Bankkonton

Kortfattad beskrivning

Uppgiften går ut på att skriva ett program med ett textbaserat menysystem för hantering av bankkonton.

1 Programskrivning

Programmet består av två delar. En del som utgör en modell av registret och en del som utgör ett textbaserat menysystem.

1.1 Modellen för bankkontona

Data som lagras för varje konto är kontonummer (unikt), kontoinnehavare, typ av konto (t.ex. S=”sparkonto”, T=”transkationskonto”) och saldo. Implementationen bör innehålla en klass som representerar ett konto. I klassen ska det bl.a. finnas en konstruktor och funktioner för att uppdatera respektive hämta värdet för respektive fält.

I klassen som representerar samlingen av kontona skall finnas funktioner för att inläsning och utskrift från och till fil. Funktioner skall också finnas för att hantera de kommandon som kan ges i det textbaserade menysystemet.

1.2 Textbaserat menysystem

De kommando som skall finnas är:

- Skapa konto
- Insättning av belopp till konto
- Uttag av belopp från konto
- Fråga om saldo för konto
- Lista information för alla konton
- Lista information för alla konton för en viss kontoinnehavare
- Avsluta (ta bort) ett existerande konto
- Modifiera ett existerande konto
- Avsluta programmet

Programmet ska vara bekvämt att använda. Det ska inte krascha om användaren skriver in konstiga värden under inmatningen.

2 Hantering av textfil i programmet

indata

Kunddata lagras mellan körningar i en datafil. Filen placeras lämpligen i projektkatalogen. Vid uppstart läses datafilen in till programmet och vid varje kommando som lägger till, tar bort eller uppdaterar ett konto skall informationen i datafilen uppdateras.

Uppdatering av fil

Det är tillåtet att använda en textfil som datafil och att vid varje uppdatering av filen skriva över den gamla filen helt. I så fall skall den gamla filen sparas som en backup för att gardera sig mot dataförlust om programmet kraschar under sparningen av filen. Funktionerna `rename()` och `remove()` i `<cstdio>` kan användas för detta.

Testdata vid inlämning

En datafil med lämpligt innehåll för testning skall bifogas vid inlämningen av uppgiften.

Deluppgift 2

Kaninkapplöpning

Kortfattad beskrivning

Uppgiften går ut på att skriva ett program som simulerar en kapplöpning mellan kaniner.

1 Programskrivning

Programmet består av två delar. En del som utgör en modell av registret och en del som utgör textbaserad in- och utmatning i ett kommandofönster.

1.1 Modellen för kapplöpningen

Ett antal kaniner som skall delta i tävlingen. Det är gemensam start och tävlingsbanan är 10 meter lång.

Implementationen bör innehålla en klass som representerar en kanin. Varje sekund kan kaninen göra endera av följande (med ”default” sannolikhet inom parenteser): Hoppa framåt (40 %), stå stilla och beta av gräset (40 %) eller hoppa bakåt (20 %). Hur långt varje hopp är dras likformigt i ett intervall (default [20 cm .. 40 cm]). De olika sannolikheterna och undre/övre hopplängd lagras per individ så att kaninerna kan ha olika egenskaper. (Att kaninerna verkligen har olika egenskaper behöver däremot inte implementeras.)

Kaninerna lagras i lämplig containerklass från klassbiblioteket i C++. En funktion skall finnas för att uppdatera ställningen med en sekund. Det behövs också funktioner för att sätta upp loppet och för att läsa av ställningen.

1.2 Textbaserat gränssnitt

Programmet skall fråga efter hur många kaniner som skall delta i tävlingen.

Simuleringen avslutas när en kanin kommit i mål och vinnaren skrivs ut. Hur långt varje kanin har kommit skall också presenteras, sorterat efter avstånd till mål.

Programmet ska vara bekvämt att använda. Det ska inte krascha om användaren skriver in konstiga värden.