Duggainformation

Duggan tjänar som individuell examination av kursmål för betyg 3.

Duggan har en uppgift och bedöms G eller U. Vi kräver tydlig, lättfattlig och underhållsbar kod, men vi kommer *inte* vara lika petiga som i labserien. En godkänd lösning ska fungera korrekt, lösningen använder lämplig klasshierarki, den saknar minnesläckor, den undviker koduppprepning, och lösningen har någorlunda namngivning och indentering.

Duggatid

Du har 120 minuter på dig att lösa duggan. Du med rätt till förlängd skrivtid har 160 minuters skrivtid. Rätt till förlängd tid beslutas av LiU centralt.

Ämnesinnehåll

Fokus på duggan kommer vara att skapa en minimal klasshierarki och få den att lösa en uppgift. Detta medför att god kunskap om syntax och semantik rörande klasser behövs.

1. Grunder: typer, variabler, satser, std::vector, funktioner, parametrar, argument
2. Klass: objekt/instans, konstruktor, inkapsling, datamedlem, medlemsfunktion, public, private
3. Arv: basklass, härledd klass, anrop av basklasskonstruktorkonstruktor, protected, anrop av basklassfunktion
4. Polymorfi: slicing, virtuell destruktor, dynamisk/statisk bindning, virtual, pure virtual, override
5. Minneshantering: allokering, avallokering, minnesläcka

Hjälpmedel

Under duggan råder standard tentamensregler. Du får ha med följande hjälpmedel:

- En bok om C++. För boken gäller följande regler:
  - Kommentarer eller 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.
  - Ingen programkod på tomma sidor, insidan av pärmarna, försättsblad, etc.
- Ett A4-ark. Valfritt innehåll på vardera sida. Går ej ersätta med två enkelsidiga ark.
- Penna för att anteckna. Du kan be tentamensvakt om kladdpapper.

Följande får INTE tas med:

- Elektroniska prylar. Till exempel miniräknare, mobiltelefon, hörlurar, tangentbord, smartklocka etc.

Inlogging

Tentamensvakt kontrollerar anmälan vid inpassage till salen. När datorns inloggningsskärm visar "tentamensläge aktivt" kan du logga in som vanligt med ditt LiU-ID och ditt privata lösenord och börja förbereda din programmeringsmiljö. Tentamensvakten kommer under duggan gå runt för att kontrollera legitimation och hjälpmedel.

Alias för kompilering

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

Frågor

Frågor ska ställas via studentklienten. Detta för att vi ska ha en historik av konversationen samt för att vi ska kunna ge samma hjälp till olika studenter. Räck upp handen om klienten, tangentbordet, musen eller datorn inte verkar fungera som de ska.

C++ referens

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

Inlämning

När du har en lösning som kompilerar utan varningar, kör utan minnesläckor och löser uppgiften skickar du in den via studentklienten. Din inlämning bedöms direkt[1] och du får svar via studentklienten. Om du får betyg G är det bara att spara, avsluta, logga ut, och framåt kvällen fira! Om du får komplettering läser du noga vad som inte fungerar, fixar till det efter bästa förmåga, och skickar in igen.

Utloggning

När du är nöjd med ditt betyg (som står i studentklienten) kan du avsluta duggan. Stäng alla program och spara alla filer. Sedan loggar du ut som vanligt i menyn. Lämna inte din plats innan vanliga inloggningsskärmen syns.

[1] Beroende på mängd samtidiga inskickningar kan det ta flera minuter att få svar, räck upp handen om du inte fått svar på 15 minuter.

Uppgift

Denna uppgift liverättas ej. Kontrollera noga att den fungerar och är välskriven innan du skickar in. Se nästa sida. Du kan fortfarande skicka in så många gånger du vill. Det senast inskicket bedöms.

I ett nytt äventyrsspel med 2D grafik ska en spelare kunna röra sig runt och interagera med objekt i spelet. I spelet ska det finnas minst två typer av spelobjekt som påverkar spelaren. Spelaren representeras av `struct Player` som finns given i filen `uppg6.cc`. Där finns även ett huvudprogram för att testa de skapade spelobjekten. Varken spelaren eller huvudprogrammet får modifieras. Alla positioner i spelet är heltal.

Utöver basklassen som representerar spelobjekt (`GameObject`), som du designar själv, ska du implementera två klasser:

Portal: En portal har en position på spelplanen och äger (ansvarar för) ett annat spelobjekt som fungerar som portalens destination. På ett portalobjekt ska det gå att anropa tre medlemsfunktioner:

- `void move_player_here(Player& p)`: Sätter spelarens position till portalens position.
- `bool collides(Player const& p) const`: Returnerar sant om spelarens position är samma som portalens position.
- `void affect(Player& p) const`: Sätter spelarens position till destinationens position ***och*** låter sedan destinationen utföra sin påverkan på spelaren.
- Kopieringskonstruktor och kopieringstilldelsningsoperator tas explicit bort (eller implementeras korrekt) eftersom denna klass ansvarar för ett annat objekt. Borttagning av en medlemsfunktion görs genom att deklarera funktionen och avsluta med `= delete`.

Bouncer: En studsare har en position på spelplanen och håller reda på sin studsförmåga (ett heltal). På ett studsobjekt ska det gå att anropa tre medlemsfunktioner:

- `void move_player_here(Player& p)`: Sätter spelarens position till studsarens position.
- `bool collides(Player const& p) const`: Returnerar sant om spelarens position är samma som studsarens position.
- `void affect(Player& p) const`: Multiplicerar spelarens hastighet med studsförmågan f och subraherar produkten från spelarens position. Matematiskt skulle det se ut så här:
  $p_x = p_x - v_x * f$
  $p_y = p_y - v_y * f$

Icke-funktionella krav:

- Det ska inte finnas koduppprepning. Gör lämplig basklass `GameObject`.
- Det givna testprogrammet ska fungera oförändrat. Gör lämpliga konstruktorer utifrån hur de anropas i testprogrammet.
- Det ska inte förekomma minnesläckor så som testprogrammet är skrivet. Gör lämplig(a) destruktor(er).

Funktionella krav: Korrekt körning av programmet ska endast ge 7 utskrifter av “ok” i terminalen.

Innan du skickar in

- Läs igenom hela uppgiften igen. Kontrollera att du löst alla detaljer.
- Kontrollera koden är korrekt indenterad och prydligt skriven.
- Kontrollera att huvudprogrammet inte är ändrat. Kopiera vid behov tillbaka koden från givna filen.
- Kontrollera att programmet fungerar och ger förväntad utdata.
- Kontrollera att klasser är väl inkapslade. Data ska bara vara åtkomligt av precis de objekt som behöver det.
- Kontrollera att ansvarsfördelning mellan basklass och härledda klasser är lämplig. Det ska inte finnas dataduplicering eller kodduplicering.
- Kontrollera med valgrind att du inte har minnesläckor.
- Kontrollera att kompileringen varken ger varningar eller fel vid kompilering med `w++17`.
- Om du ändrade något till följd av ovan kontroller: kontrollera allt igen.
- Om du är osäker på något eller inte lyckats lösa något, skicka in en fråga och skriv en kommentar i koden där du förklarar vad du vet om problemet.