Kursnamn: Grundläggande programmering med C++

Kurskod: IT120G

Program: Webbutveckling (WEBUG)

Namn: Oscar Gustavsson

Användarnamn: a21oscgu

Mail: a21oscgu@student.his.se

Inlämningsuppgift 1

”Kasta tärning”

1. Problembeskrivning

Den här rapporten handlar om programmet ”Kasta tärning” som skapats med hjälp av LucidChart och Microsoft Visual Studio. C++ programmet som har skapats går i grund och botten ut på att användaren ska kunna spela ett tärningsspel med datorn. Innan tärningsspelet börjar ska användaren kunna satsa pengar, för att senare i spelet kunna vinna eller förlora pengar. För att kunna spela spelet behöver användaren, enligt de redan fördefinierade kraven, kunna satsa pengar, vilket leder till det första delproblemet.

* 1. Programmets start: inmatningen

Programmets första delproblem är att programmet kräver någon form av insättningsmetod. Detta delproblemet bär också med sig ett flertal mindre delproblem, vilka kan beskrivas med hjälp av ett antal frågeställningar. Vad händer ifall användaren inte vill sätta in pengar? Vad händer ifall användaren försöker sätta in mer än 5000 kr? Har insättningsmetoden också en minimigräns? Får användaren veta hur mycket pengar som är möjliga att vinna i spelet? Vad händer ifall användaren matar in fel tecken under insättningsfasen, exempelvis ett negativt tal? Vad händer ifall användaren anger fel värde vid satsningen av pengar? Kan användaren spela spelet utan att varken sätta in pengar eller satsa pengar?

* 1. Programmets kärna: tärningsspelet

Det andra större delproblemet i C++ programmet är tärningsspelet och detta är den allra viktigaste funktionen i programmet. Användaren ska kunna utmana datorn i ett tärningsspel och spelet går ut på att datorn och användaren slår två tärningar var. Den som får högst summa vinner rundan och den som vinner bäst utav tre rundor, alltså två stycken rundor, har vunnit tärningsspelet och erhåller därefter vinstpengarna. De frågeställningar som bäst sammanfattar detta delproblemet är följande. Hur ska tärningarna och resultatet representeras på skärmen? Kommer användaren att kunna interagera med datorn för att slå tärningarna? Hur ska användaren kunna spela mer än en runda i spelsessionen? Ska användaren kunna avsluta spelet mitt i en spelsession?

* 1. Programmets slut: utmatningen

Det tredje större delproblemet i programmet ”Kasta tärning” är spelets resultat och utdata. Tärningsspelet behöver interagera med spelaren inte enbart under spelets slut, utan även genom hela spelets gång, för att användaren ska kunna förstå vad programmet gör i realtid. Ifall exempelvis användaren matar in fel värden vid insättningen av pengar, ska programmet berätta det, genom ett meddelande och därefter låta användaren göra om steget. Ett annat exempel är i de fall då användaren väljer att avsluta programmet. Då behöver användaren få reda på vad som händer med deras resterande pengar. Dessutom behöver särskilda heltalsvariabler beräknas innan det matas ut på fönstret. Heltalsvariabler såsom spelomgångens resultat, vinstpengar, saldo, med mera, behöver beräknas av programmet och slutligen visas. Frågeställningarna som kan sammanfatta detta delproblemet är följande. Hur mycket statistik behöver användaren få tillgång till? Vilken statistik är nödvändig att användaren får tillgång till? Hur ska de satsade pengarna beräknas? Vilken information är nödvändig för användaren att få, då användaren exempelvis väljer att avsluta spelet? Vilken information vore istället onödig för användaren att få ta del av?

* 1. Programmets slut: spela mer

Det fjärde stora delproblemet i programmet ”Kasta tärning” är programmets alternativ innan programmet avslutas. Enligt de fördefinierade kraven ska användaren ha möjlighet till att spela mer än en spelomgång. Dessutom ska användaren inte behöva sätta in pengar vid ny spelomgång, ifall det redan finns tillräckligt med pengar innestående på spelkontot. På grund av detta kan flera frågeställningar ställas till utvecklaren. Måste användaren spela mer än en omgång? Kan användaren spela ett oändligt antal omgångar? Ska användaren ha möjlighet till att sätta in mer pengar även om användaren redan har mer än 5000 kr på spelkontot? Vad händer ifall användaren råkar fortsätta spelet på ny omgång, då avsikten istället var att avsluta spelet? Dessa frågeställningar som nämnts besvaras härefter i rapporten.

1. Antaganden och krav
   1. Programmets start: inmatningen

De krav som redan har definierats för programmets start är för det första att användaren endast får sätta in max 5000 kr om gången. Det andra redan definierade kravet är att användaren ska få se när betalningsmoment sker i programmet.

Ett eget antagande som har ställts är att användaren inte ska kunna spela tärningsspelet utan att först sätta in pengar och satsa pengar. Detta innebär att användare som inte sätter in pengar inte kan spela tärningsspelet ”Kasta tärning”. Användare som inte sätter in pengar kan enbart starta programmet och avsluta det, eftersom inga andra alternativ är tillgängliga. Ett krav som ställts på programmet är att ifall en användare försöker sätta in mer än 5000 kr, eller av misstag råkar sätta in mer än 5000 kr, ska programmet avbryta och låta användaren göra om sin insättning eller satsning. Detta gäller även ifall användaren försöker sätta in mindre än 100 kr eller ett negativt belopp (t.ex. -3000 kr). Ett till eget definierat krav är ifall användaren ångrar sin insättning. Ifall användaren ångrar sin insättning ska användaren kunna avbryta spelet och föra pengarna tillbaks till användaren innan spelet avslutas.

För att användaren ska veta hur mycket pengar man kan vinna i spelet, har kravet ställts på programmet att det i början av programmet meddelar användaren om att det är möjligt att vinna dubbelt så mycket pengar som har satsats. Detta sker såklart innan respektive spelomgång. Ifall användaren vinner erhålls då både den egna insatsen samt datorns insats, vilket kommer att bli dubbelt så mycket som användarens egna insats. Ett till antagande är därför att datorns satsade belopp alltid kommer att vara lika högt som användarens satsade belopp. Ett ytterligare antagande som ställts är att programmet ska avbryta användaren vid satsningen av pengar, ifall användaren inte satsar 100 kr, 300 kr eller 500 kr. I sådana fall där användaren gör fel, får användaren göra om föregående steg. Det sista kravet i programmets start är att programmet även kommer att stoppa användaren ifall användaren försöker satsa mer pengar än vad som finns på spelkontot.

* 1. Programmets kärna: tärningsspelet

De fördefinierade kraven på tärningspelet är för det första att användaren ska kunna utmana datorn i ett tärningspel. Det andra kravet är att användaren och datorn ska få kasta två tärningar åt gången, i tre rundor. Det tredje fördefinierade kravet är att den som får högst summa av tärningskastet vinner rundan. Det fjärde kravet är att det är bäst av tre rundor som gäller vilket också innebär att ifall någon vinner två rundor i rad, behöver inte den tredje rundan spelas. Det femte kravet är att användarens och datorns tärningskast måste redovisas varje omgång och det sjätte kravet är att det måste framgå tydligt vem som vann spelet totalt och för respektive runda. Dessutom finns det ett krav på att användaren ska välja själv när programmet ska avslutas.

Ett antagande som valts i programmet är att ifall det blir lika, ska ingenting mer hända än att ett meddelande om just det, syns i fönstret. Ingen kommer att vinna rundan när det blir lika men användaren kommer att behöva spela en extra runda för att kompensera för den oavgjorda rundan. Detta innebär att en spelomgång i själva verket kan bestå av fyra eller mer rundor beroende på hur många rundor som istället blir oavgjorda. Ett till eget antagande på programmet är att det ska fungera att ta ut sina pengar och avsluta programmet mitt i en spelomgång. Fördelen med detta antagande är att användaren själv får större valmöjligheter. Användaren kan då avsluta spelet ifall användaren exempelvis ångrar sin satsning eller insättning. Nackdelen med detta antagande, utifrån ett utvecklarperspektiv, är att användaren själv kan välja att avsluta spelet när det är datorn som leder spelomgången med en runda och därefter kan användaren påbörja ett nytt spel. Det sista kravet som ställts i den här delen av programmet är att användaren ska kunna interagera med datorn för att tärningarna ska slås. Användaren får mata i en etta ,”1”, i fönstret för att slå tärningarna. Tärningarnas värde syns därefter på skärmen tillsammans med datorns tärningar.

* 1. Programmets slut: utmatningen

Det första kravet, som redan har definierats för programmets utmatning, är ifall användaren vinner en spelomgång (obs! Inte detsamma som att användaren vinner en runda. En spelomgång definieras som ett spel där bäst av tre rundor gäller). Ifall användaren vinner en spelomgång ska användaren erhålla summan av både användarens och datorns insats, men vid förlust ska användarens egna insats förloras och istället överföras till datorn. Det andra kravet är att programmet ska skriva ut aktuell vinst samt den totala vinstsumman ifall användaren lyckas vinna en spelomgång. Det tredje redan fördefinierade kravet är ifall datorn vinner en spelomgång. Då ska programmet skriva ut ett tröstmeddelande och den totala vinstsumman som finns kvar.

Det första egna definierade kravet är att alla resterande pengarna, på användarens spelkonto, ska överföras från spelkontot till användaren varje gång, innan användaren avslutar spelet. Det första definierade antagandet, när det gäller utmatningen, är att alla användare säkerligen inte är intresserade av all statistik som kan loggas från spelomgångarna. Dock finns möjligheten för användaren att se statistiken, efter varje spelad omgång.

* 1. Programmets slut: spela mer

De krav som redan har definierats för programmets absolut sista fas är för det första att användaren ska få ha möjligheten till att spela hur många omgångar som helst. På grund av detta kravet är det första egna definierade antagandet att användaren måste ha tillräckligt med pengar på spelkontot för att kunna spela hur många gånger som helst. Det andra fördefinierade kravet vid programmets sista fas är att användaren aldrig ska kunna sätta in mer än 5000 kr vid en insättning. Detta innebär givetvis att användaren ska ha möjligheten till att sätta in mer pengar på spelkontot, inför varje ny spelomgång (om så önskas av användaren). Ett till krav som har ställts på programmet är att en användare som har tillräckligt med pengar på spelkontot, aldrig ska bli tvingad till att sätta in mer pengar. Dock ska användaren få möjligheten att sätta in mer pengar ifall det är det som önskas utav användaren.

1. Lösningsdesign i C++

För att lösa de frågeställningar och krav som tidigare ställts har ett flertal tekniker använts i det byggda programmet ”Kasta tärning” och något att notera är att programmet matar ut meddelanden till användaren vid varje delmoment som sker i spelet.

* 1. Insättning

För att sätta in pengar används en if-else-sats. Om användaren har mindre än 100 kronor på spelkontot, behöver användaren sätta in pengar. Annars ifall användaren redan har 100 kronor eller mer på spelkontot, vilket kan vara möjligt efter spelomgång 1, kommer programmet enbart att fråga användaren ifall användaren vill sätta in mer pengar. Inuti denna övergripande if-else-satsen befinner sig två while-loopar, med flera if-else-satser (och if-else if-else-satser för flera olika alternativ, se bilaga 1). Alla if-else-satser används för att programmet ska kunna interagera korrekt vid användarens insättning. While-looparna används för att användaren ska kunna göra om sin insättning ifall antingen programmet upptäcker ett fel vid insättningen eller ifall användaren ångrar sig (se bilaga 2).

* 1. Satsning

För att satsa pengar används en while-loop som avbryts ifall användaren antingen satsar ett godkänt belopp eller ifall användaren inte väljer att satsa något belopp. Inuti while-loopen används if-else-satser (och if-else if-else-satser för flera olika alternativ) för att programmet ska kunna interagera korrekt med användaren (se bilaga 3).

* 1. Tärningsspel

Programmets tärningsspel består av en while-loop som avbryts antingen ifall någon vinner spelomgången eller ifall användaren väljer att avsluta spelomgången (se bilaga 4). Inuti while-loopen används ett antal if-else-satser (och if-else if-satser för flera olika alternativ) för att programmet ska kunna skilja mellan vinst, förlust och avgjorda rundor. En fördefinierad randomfunktion (srand) har använts för att få tärningarnas värde att slumpa.

* 1. Resultat, uttag och beräkningar

För att användaren ska kunna se resultat används enbart if-else- och if-else if-satser. Ifall användaren vinner beräknas först flera variabler, innan resultatet syns på fönstret. Det som beräknas är antalet poäng, antalet spelade spel, antalet vunna spel, antalet förlorade spel, antalet vunna pengar vid den här spelomgången, det totala antalet vunna pengar samt användarens och datorns uppdaterade saldo (se bilaga 5). Användaren får även, med hjälp av en if-else-sats, alternativet att se fullständig statistik ifall det önskas (se bilaga 6). Ifall datorn vinner spelomgången sker samma typ av händelseförlopp.

* 1. Spela mer

För att användaren ska kunna spela mer, har en do-while- loop använts. Den här loopen sträcker sig över all källkod inom main()-funktionen (loopen startar på rad 23 och slutar på rad 290). Spelet kommer alltså att köra från rad 23 om och om igen tills villkoret ”playOneMoreGame” har blivit falskt. Detta villkoret blir falskt ifall användaren anger det i slutet av spelet. En while-loop med en if-else if-sats används för att kontrollera vad som sker efter spelets resultat (se bilaga 7). Antingen spelar användaren en till spelomgång, eller avslutas spelet och pengarna förs över till användaren. Ifall det inte finns pengar på spelkontot avslutas spelet utan att överföra några pengar till användaren. Om användaren väljer att spela en till omgång, nollställs flera variabler för att nästa omgång ska vara fullt spelbar.

Diskussion

En svaghet i källkoden är att alla variabler initieras på samma plats i programmet (se bilaga 8). Variablerna kan inte användas av andra funktioner i programmet men eftersom det endast finns en funktion i programmet (main) spelar det inte alls någon stor roll. Den enda fördelen med variablernas placering är att dem alla är samlade och initierade på samma plats i programmet.

En till svaghet i programmet är att det inte finns någon åldersgräns på programmet. Det ända som krävs för att kunna spela är att användaren kan sätta in pengar, vilket inte minderåriga bör göra.

En styrka med detta programmet är att jag redan i analysfasen har utgått från att göra programmet så felfritt som möjligt. Under insättningen och satsningen av programmet ska användaren mata in utvalda värden. Ifall användaren matar in en sträng där endast heltal accepteras, korrigerar programmet användaren och låter istället användaren få gå tillbaka och göra om sin imatning. För att programmet ska klara av det här, har två funktioner implementerats. Funktionerna är ”std::cin.clear();” och ”std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');” (se bilaga 9). Den första funktionen (cin.clear()) upphäver det feltillstånd som inmatningsströmmen hamnat i. Den andra funktionen (cin.ignore()) hoppar över ett antal tecken, vilket anges som första parameter i funktionen, ända tills funktionen stöter på ett tecken som är lika med den andra parametern eller tills fil-slutet (EOF) påträffas. Den första parametern anger det maximala antalet tecken som kan ignoreras.

Bilagor

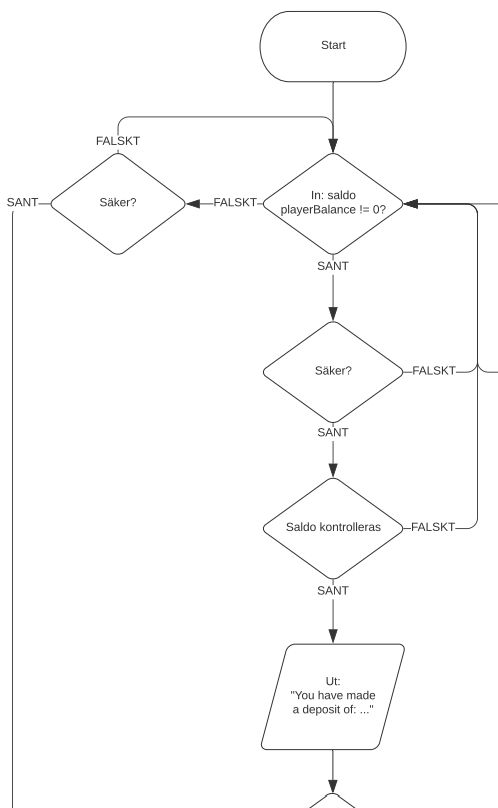
if (playerBalance < 100) // Användaren måste sätta in pengar här ifall det inte redan finns 100 kr på spelkontot.

{

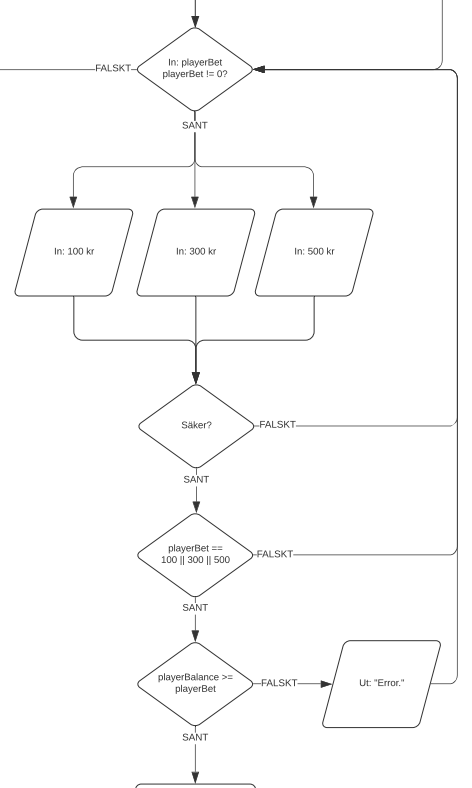
while (playerBalanceOk == 0) // När playerBalanceOk är sant går spelet vidare.

{

*Bilaga 1: Bilagan visar den första if-satsen och den första while-loopen, med dess villkor, som krävs för programmets första del: insättningen av pengar.*



*Bilaga 2: Bilden visar flödesschemat över programmets första del: insättningen. Det som sker är följande: ifall saldot är över 0 kr och ifall användaren är säker på sin insättning samt att saldot är giltigt, kommer insättningen att gå igenom. Ifall användaren inte sätter in pengar kommer tärningsspelet att avslutas (detta syns inte helt i bilden).*



*Bilaga 3: Bilden visar en del utav flödesschemat som använts för att konstruera programmets andra del: satsningen.*

*Bilaga 4:* *Bilagan visar den första while-loopen, med dess villkor, som krävs för att användaren (eller datorn) ska kunna vinna tärningsspelet.*

std::cout << "\nCurrent game ends.\n";

if (playerScore == 2) // Ifall användaren vann spelomgången (bäst av tre rundor).

{

gamesWon++;

playerBalance += cpuBet;

cpuBalance -= cpuBet;

prize = playerBet + cpuBet;

totalWinnings += prize;

// Spel.

// Brist: Programmet fungerar inte korrekt ifall användaren matar in en input som exempelvis dem här: "-100" "777siffrorochsedanbokstäver".

while (((playerScore | cpuScore) < 2) && (playerRoundExitConfirmed != 1))

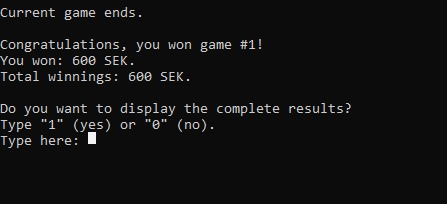
{

std::cout << "\nBest of three. In order to win you'll have to win 2 rounds." << "\nDo you want to start round #" << (currentRound + 1) << "?\nType \"1\" (yes) or \"0\" (no).\nType here: ";

std::cin >> playNextRoundConfirmed;

if (playNextRoundConfirmed == 0)

*Bilaga 5: Bilagan visar en del beräkningar som sker ifall användaren vinner tärningsspelet.*



*Bilaga 6: Bilden visar alternativet att se den fullständiga statistiken i tärningsspelet.*

*Bilaga 7: Här syns källkoden till spelets avslutande del.*

while (playerExitConfirmed\_2 == 0)

{

std::cout << "\nDo you want to play again?\nType \"1\" (yes) or \"0\" (no).\nType here: ";

std::cin >> playOneMoreGame;

std::cout << "\nAre you sure?\nType \"1\" (yes) or \"0\" (no).\nType here: ";

std::cin >> playerExitConfirmed\_2;

}

if (playOneMoreGame == 1)

{

playerBalanceOk = 0, playerBalanceConfirmed = 0, playerDepositMoreMoneyConfirmed = 0, playerDepositMoreMoneyBalance = 0,

playerExitConfirmed\_1 = 0, playerBet = 0, cpuBet = 0, PlayerBetOk = 0, playerBetConfirmed = 0, playNextRoundConfirmed = 0,

currentGame = 0, playerScore = 0, cpuScore = 0, playerRollsDice = 0, playerDice\_1 = 0, cpuDice\_1 = 0, playerRoundExitConfirmed = 0,

roundsPlayed = 0, playerExitConfirmed\_2 = 0, playerDepositMoreMoneyOk = 0; // Variabler nollställs

}

else if (playOneMoreGame == 0 && playerBalance > 0)

{

std::cout << "\nYour balance, " << playerBalance << " SEK, will be transferred to you.\nThe game ends.\n";

}

else if (playOneMoreGame == 0 && playerBalance == 0)

{

std::cout << "\nYour balance is empty. No money will be transferred to you.\nThe game ends.\n";

}

*Bilaga 8: Bilagan visar de initierade variablerna i programmets start.*

if (std::cin.fail()) // Kontrollerar så att användaren inte har matat in t.ex. en string med bokstäver.

{

std::cout << "\nError.\nWrong input. Try again.\n";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

}

// Variabler initieras (deklareras och tillges värden)

// Variabler: Insättning av pengar.

int playerBalance = 0, playerBalanceOk = 0, playerBalanceConfirmed = 0, playerDepositMoreMoneyConfirmed = 0, playerDepositMoreMoneyBalance = 0, playerExitConfirmed\_1 = 0, playerDepositMoreMoneyOk = 0;

// Variabler: Satsning av pengar.

int playerBet = 0, cpuBet = 0, cpuBalance = 100000, PlayerBetOk = 0, playerBetConfirmed = 0;

// Variabler: Spel.

int playerScore = 0, cpuScore = 0, playNextRoundConfirmed = 0, playerRollsDice = 0, playerDice\_1 = 0, playerDice\_2 = 0, totalNumberPlayerDice = 0, cpuDice\_1 = 0, cpuDice\_2 = 0, totalNumberCpuDice = 0, currentRound = 0, roundsPlayed = 0, playerRoundExitConfirmed = 0, playOneMoreGame = 1, playerExitConfirmed\_2 = 0;

// Variabler: Resultat.

int prize = 0, totalWinnings = 0, currentGame = 0, gamesPlayed = 0, gamesWon = 0, gamesLost = 0, displayResultsConfirmed = 0;

*Bilaga 9: Bilagan visar koden som kontrollerar att användarens inmatning är av rätt datatyp.*