Inlämningsuppgift #

”Projekt rapport”

Grundläggande programmering

Felix Ahlcrona

921216-1252

ST 201X

# Inledning

Uppgiften går ut på att skapa ett roulette program i valfri C++ miljö. Spelet kommer vara textbaserat och användaren kommer använda inputs för att ta sig vidare i spelet. Själva spelet går ut på att användaren presenteras med en startsumma på 1000, användaren kommer sedan välja mellan att satsa 100, 300 eller 500 efter satsad summa så väljer användaren att antingen välja att spela på svart eller röd färg eller spela på siffra. Användaren presenteras sedan med resultatet och har möjlighet att spela igen eller avsluta spelet. När det gäller lösningsförslag så har det funnits lite olika idéer från start, att börja modellera ett flödesdiagram gav en väldigt bra överblick över hur programmet kommer att fungera och vilka åtgärder som kommer behövas göra. Är flödesdiagrammet logiskt och fungerar så gäller det bara att förstå varje del som en funktion och översätta det i fungerande kod.

# Problembeskrivning

Vid start av uppgift så fanns det en del problem som behövdes tid för att tänkas igenom hur de skulle lösas på ett fungerande sätt. Allteftersom programmet blev mer utvecklat så hittades det mer delproblem som inte var hittade eller genomtänkta från början. Vissa delproblem kändes svåra i början men efter funderat på exakt vad problemet är och exakt vad man faktiskt försöker koda så inser man att det är relativt lätt att lösa, det gällde bara att tänka väldigt logiskt och förstå exakt vad de man gör.

# Delproblem

* **Inmatning av satsat värde.**

Här behöver användaren mata in det värde som användaren väljer att satsa på, beroende på vilka antagande man gör här så kan det bli olika mycket problemområden. Av de antagandes som har gjort så är det första problemet att summan som användaren matar in måste stämma överens med de värden som anges i uppgiftsbeskrivningen dvs 100,300 eller 500.

Det andra delproblemet har med den totala spelsumman att göra. Väljer användaren att fortsätta spela och börjar till slut få ont om pengar så ska det göras en kontroll att användaren faktiskt har de saldot som krävs för att fortsätta spela. Denna kontroll kommer även behöva upprepas så länge som användaren väljer ett felaktigt saldo samt inte krascha.

Användaren ska t ex inte kunna spela för 300kr om saldot ligger under det valda beloppet.

* **Inmatning av speltyp**

Det här problemet liknar det första problemet till strukturen dock kommer det behövas en helt annan lösning till problemet.

Efter användaren har valt en acceptabel satsning som ligger inom saldots ram så kommer användaren ha möjlighet att välja mellan att spela på siffra eller färg.

Programmet måste då alltså kunna identifiera och skilja på svaren samtidigt göra olika vägval beroende på vad användaren väljer. Den största utmaningen i det här problemet var just att kunna initiera en av de speltyperna som användaren väljer.

* **Kontroll av jämna eller ojämna nummer**

Väljer användaren att spela med färg så kommer det behöva göras en kontroll om siffran man får är jämn eller ojämn och beroende på vilken färg man har valt så förlorar man eller vinner pengar. De flesta problemen här har redan lösts genom liknande exempel i tidigare problem dock så återstår det ett större problem vilket var hur programmet vet att siffran är jämn eller ojämn. Det här problemet har varit en av de större i hela programmet.

* **Kontroll av exakt siffra**

Det här delproblemet är liknande det av ojämna och jämna siffror dock ska programmet jämföra om siffran som användaren har valt att satsa på stämmer överens med siffran som slumpas fram. Gissar användaren på rätt siffra kommer användaren erhållas en vinst som är tio gånger av det satsade beloppet.

* **Generera en boll**

Vid första anblick var det här problemet förvirrande eftersom det inte fanns något exempel på hur problemlösningen såg ut. Det visade sig dock vara väldigt enkelt. Problemet var alltså att få programmet att slumpa fram en siffra som kommer användas som bollen man spelar med. Denna ”boll” ska sedan jämföras med med vad användaren har valt.

* **Möjlighet att kunna spela igen**

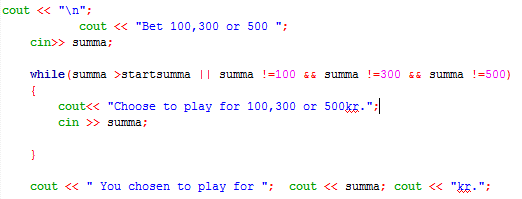
Det här problemet var väldigt högt prioriterat från start, eftersom koden läses av kompilatorn uppifrån och ner så behövs det på något sätt lyckas hoppa tillbaka till början av programmet. Hade inte det här lösts hade programmet varit mer eller mindre odugligt.

* **Möjlighet att kunna avsluta spelet**

Ett av kraven i programmet var även att man ska kunna ha möjligheten att avsluta spelet när som helst.

# Lösningsdesign

* **Inmatning av satsat värde.**

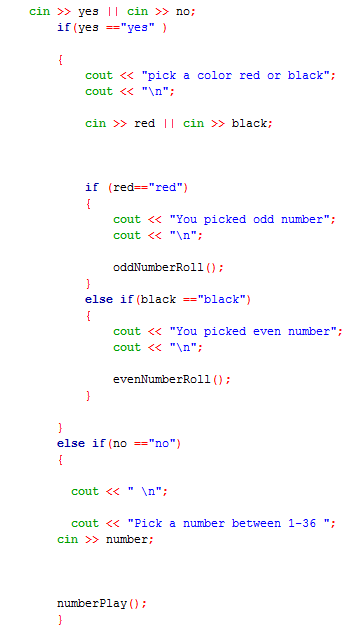


Lösningen var att kunna jämföra talet som matas in med de värden som tidigare har angetts.

Programmet måste då endast kunna ta emot dessa inmatningar samtidigt som det inte kraschar.

Användaren ska t ex inte kunna mata in siffran 301 eller 99 utan endast de siffror som anges. Det första koden gör är att kollar som summan man satsar är större en det totala saldot att spela för. Det blev fel och skulle egentligen inte heta startsumma, det är alltså inte startsumman den jämför med utan alltid totalsumman. Att använda sig av != operator så kan man enkelt urskilja alla andra siffror samt användandet av || och && gör att man inte behöver upprepa while loopen flera gånger utan kan skriva allt i en while loop.

* **Inmatning av speltyp**

****

Att skapa en input funktion där användaren får välja mellan att skriva ”yes” eller ”no” för att sedan utföra en händelse i en loop beroende på vilket svar användaren ger kändes som en enkel lösning på problemet. Input värdet kommer sen jämföras med en sträng som ligger globalt och inte i funktionen. Det blev rörigt att ha all kod i main, här används funktionsanrop på tre funktioner beroende på vad man väljer att spela istället för ha all kod i main.

* **Kontroll av jämna eller ojämna nummer**

Som nämnt i tidigare lösning så kommer användaren välja en av de tre speltyperna och kommer i det här exemplet spela på färgen svart dvs jämnt nummer. Funktionen kommer kallas och det väsentliga i den här funktion är denna kod.

Bollen som har slumpats fram kommer kolla om talet som har blivit delat har någon rest, är resten 0 så kommer loopen sättas igång. Eftersom problemet var att det fanns två typer av nummer, ojämna och jämna så måste det finnas någon typ av funktion som kan kolla just det. Det är givet att någon form av selektion behövs för att kunna bevisa ett påstående. If satser fungerade perfekt för det här ändamålet.

För ojämna tal så kommer en liknande funktion köras, skillnaden är om det blir rest över från divideringen så kommer denna loop köras.



Lösningen var först tänkt att göras genom Arrayer. T ex

if(randomNumber = OddNumbers[18]) dock så blev det problem med lösningen.

* **Kontroll av exakt siffra**

Väljer användaren att spela på siffra kommer denna kontroll göras, siffran som slumpas fram av siffergeneratorn kommer jämföras med siffran som användaren väljer att mata in via input.

Bollen som slumpas fram kommer jämföras med vad användaren valde att satsa på.

****

Problembeskrivningen är väldigt lik de förra funktionerna, då just lösningen till problemet kommer behöva någon form av selektion för att kunna lösas. Återigen så är If satsen utmärkt för att lösa det här problemet.

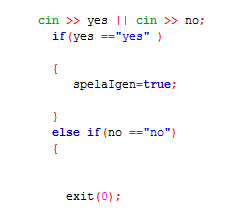
* **Generera en boll**

Den här lösningen gavs delvis till oss från uppgiftsbeskrivningen. Problemet var egentligen hur exakt det skulle implementeras dock så blev det logiskt att det enda man faktiskt behövde göra var att använda sig av den variabeln man skapade genom bollgeneratorn. Denna variabel kommer användas flera gånger i programmet som en siffra som slumpas fram, den kommer sedan jämföras med jämna/ojämna nummer samt om användaren väljer att spela på nummer.

* **Möjlighet att kunna spela igen/avsluta spel**

Den här lösningen ändrades några gånger då hela strukturen av programmet ändrades då inte hela programmet låg i main längre. Eftersom det inte är bra att använda sig av ”go to” så behövdes det på något sätt kunna upprepa koden om användaren valde att spela igen. Programmet kommer fråga användaren om den vill spela igen, väljer användaren att spela igen så kommer en boolean få sitt värde sant. Den här boolean jämförs med sant eller falskt i en while loop som ligger omslutet kring main. Dvs kommer ”spelaIgen” att vara sann kommer while loopen sättas igång och spelet börjar om.

Väljer användaren att säga nej så kommer programmet avslutas, eftersom det här ligger i en void funktion så går det ej att använda sig av return 0 ; då det inte går att returnera något i void.



Att använda sig av boolean är väldigt enkelt och genom att kunna ändra värdet för boolean så går det att skapa flera olika beslut beroende på värdet.

# 

# Antaganden och krav

Det fanns lite antagande att göra i uppgiften, man ska kunna avsluta spelet. Betyder det att användaren ska ha möjlighet att kunna avsluta var som helst i spelet eller efter varje spelad omgång. I det här programmet så är det bara möjligt att avsluta efter resultat dvs en spelomgång.

Det går också att tolka den totala summan på olika sätt, vill man att programmet listar all vinst och förlust som användaren har gjort eller endast den aktuella totalsumman. I det här programmet visas alltid det totala saldot användaren har vid varje tillfälle.

Det går inte att säga hur mycket pengar användaren förlorar om den förlorar, här får ett antagande göras och det lär endast vara det satsade beloppet.

Ännu ett antagande är om användaren har fått slut på pengar genom förlust, ska den kunna fylla på kontot med pengar att spela för igen? Eller ska det helt enkelt avsluta programmet av sig själv. I det här programmet kommer användaren informeras om att det är slut på pengar och spelet kommer avslutas.

# Diskussion

Det finns förmodligen brister i de flesta program och det här programmet är givetvis inget undantag. En av bristerna är att när användaren har 0 på saldot kan användaren välja att spela igen, dock så kommer programmet avslutas när programmet börjar om för det görs en kontroll om det finns pengar att spela för, gör det inte det så avslutas spelet. Tekniskt sett så fungerar det då man inte kan spela utan pengar, problemet är mer att den kontrollen borde göras innan användaren har chans att välja spela igen.

Det fattas även skydd mot felskrivningar i programmet t ex skulle användaren välja att skriva in en bokstav istället för en siffra när användaren ska satsa pengar så kommer det fastna i en loop. Så skydd mot felskrivningar finns inte riktigt dock så finns de viktiga delarna som kontroll av rätt belopp matas in och att pengar faktiskt finns för att kunna spela.

Programmets output är inte heller perfekt, det kan fattas mellanslag i output texter och några radavslut, det hade gett en bättre upplevelse för användaren om allt var perfekt dock så fanns det inga detaljerade designkrav så fokus har inte legat på designen.

Det finns även några variabel namn som kan namnges bättre som t e x summa, som kunde visa tydligare att det är summan användaren väljer att satsa.

Det är svårt att påpeka några styrkor i programmet då det är skapat för möta kraven som har ställts och lyckas uppfylla det, dock så finns det fördelar att programmet är skrivet i funktioner och inte rakt upp och ner i Main vilket gör att felsökningen kan begränsas inom funktioner och det går att återanvända koden man har skrivit samt massa andra fördelar. Hade det funnits behov att utöka funktionaliteten i framtiden så kommer funktionerna ligga som en bra grund i programmet.

# Källkod

#include <iostream>

#include <string>

#include <ctime>

#include <time.h>

#include <algorithm>

using namespace std;

int summa; // Här deklarerar jag en variabel av datatypen int som ska hålla i den summan jag satsar.

int startsumma =1000; // Den här variabeln håller i startsumman och jag har initierat 1000.

int vinst; // Den här variabeln har jag för att räkna ut resultatet av det jag satsar och vinner/förlorar

int number ; // Satsar en siffra som sen kommer att jämföras med siffran som slumpas

bool spelaIgen =true; // Deklarerar en boolean som man använder för starta en spelomgång eller ej.

string red ("red"); // Variabel som jag använder som input

string black ("black");

string yes ("yes");

string no("no");

int randomNumber = 1 + (rand() % 36); // Genererar en siffra mellan 1 och 36.

void evenNumberRoll() // Funktion som kommer slumpa fram en siffra och jämföra om bollen är en jämn siffra.

{

int randomNumber = 1 + (rand() % 36); // Genererar en siffra mellan 1 och 36.

cout << "the ball is " << randomNumber << "."; // Visar vad för boll man fick.

if( randomNumber%2 == 0) // Om resten på siffran som har delats är lika med noll, dvs ingen siffra blir över så är det här sant.

{

vinst = summa+summa+startsumma; // Användaren dubblar sin vinst och den läggs på den totalasumman.

cout << "You won! You have : "; cout << vinst; cout << " to play for!";

cout << "\n";

cout << "Do you want to play again? 'yes'/'no'";

startsumma=vinst; // Eftersom startsumman inte ska gå tillbaka till 1000 varje spelomgång.

}

else {

vinst = startsumma-summa; // förlorar användaren så dras det satsade värdet av från den totala summan.

cout << "You lost! Your have :"; cout << vinst; cout << "left";

cout << "\n";

cout << "Do you want to play again? 'yes'/'no'";

startsumma=vinst; // Eftersom startsumman inte ska gå tillbaka till 1000 varje spelomgång.

}

cin >> yes || cin >> no; // Användaren väljer "yes" eller "no" för att spela igen eller stänga av spelet.

if(yes =="yes" )

{

spelaIgen=true; // Boolean blir true och while loopen i main blir sann och kommer köras igen.

}

else if(no =="no")

{

exit(0); // Anger användaren "no" så stängs spelet av, det går inte att använda return 0; eftersom funktionen är void.

}

}

void oddNumberRoll() // Funktion som kommer jämföra om siffran är ojämn.

{

int randomNumber = 1 + (rand() % 36);

cout << "the ball is " << randomNumber << ".";

cout << "\n";

if( randomNumber%2 != 0) // Om det blir rest över efter siffran har delats så kommmer denna if sats köras.

{

vinst = summa+summa+startsumma;

cout << "You won! You have : "; cout << vinst; cout << " to play for!";

cout << "\n";

cout << "Do you want to play again? 'yes'/'no'";

startsumma=vinst;

}

else {

vinst = startsumma-summa;

cout << "You lost! Your have :"; cout << vinst; cout << "left";

cout << "\n";

cout << "Do you want to play again? 'yes'/'no'";

startsumma=vinst;

}

cin >> yes || cin >> no;

if(yes =="yes" )

{

spelaIgen=true;

}

else if(no =="no")

{

exit(0);

}

}

void numberPlay() // Den här funktionen kallas om användaren väljer att spela på nummer.

{

cout << "the ball is " << randomNumber << ".";

cout << "\n";

if(randomNumber==number) // Om siffran som har slumpats fram av generatorn stämmer överens med det nummer som användaren har gett som input så körds den här if sastsen

{

cout << "\n";

vinst =summa\*10 + startsumma;

cout << "Correct number! You have : "; cout << vinst; cout << " to play for!";

cout << "\n";

cout << "Do you want to play again? 'yes'/'no'";

startsumma=vinst;

}

else

{

cout << "\n";

vinst=startsumma-summa;

startsumma=vinst;

cout << "You lost! Your have :"; cout << vinst; cout << "left";

cout << "\n";

cout << "Do you want to play again? 'yes'/'no'";

}

cin >> yes || cin >> no;

if(yes =="yes" )

{

spelaIgen=true;

}

else if(no =="no")

{

exit(0);

}

}

int main()

{

srand(time(0));

while (spelaIgen == true ) // Den här while loopen ligger runt hela main och kommer påverkas om användaren väljer att spela igen.

{

if(startsumma==0) // Här kollar man om det totala saldot är 0, är det helt slut så finns det ingen möjlighet att spela vidare

{

cout << ". Sorry you are out of money.";

return 0; // Finns det inte pengar avslutas programmet.

}

cout << "\n";

cout << "Bet 100,300 or 500 ";

cin>> summa;

while(summa >startsumma || summa !=100 && summa !=300 && summa !=500) // Här kollas det om summan man satsar är större en saldot och summan inte är något annat än 100 ,300 eller 500. Stämmer inte det så blir användaren tvungen att inmata summan tills det stämmer.

{

cout<< "Choose to play for 100,300 or 500kr.";

cin >> summa;

}

cout << " You chosen to play for "; cout << summa; cout << "kr.";

cout << "\n";

cout << "Do you want to play with color enter 'yes' or number enter 'no'";

cin >> yes || cin >> no; // Användaren väljer yes för att spela med färg och no för att spela med siffra.

if(yes =="yes" ) // Stämmer inputen yes med strängen "yes" så kommer man sedan få frågan om att välja färg.

{

cout << "pick a color red or black";

cout << "\n";

cin >> red || cin >> black;

if (red=="red")

{

cout << "You picked odd number";

cout << "\n";

oddNumberRoll(); // Skriver användaren in red som input så kommer detta funktionsanrop kallas.

}

else if(black =="black")

{

cout << "You picked even number";

cout << "\n";

evenNumberRoll();

}

}

else if(no =="no")

{

cout << " \n";

cout << "Pick a number between 1-36 ";

cin >> number;

numberPlay();

}

}

}