C++ programmering

Abukar Ahmed

19990901-9599

VT/HT 2018-19

Inlämningsuppgift 2

”Enarmad bandit”

1 Introduktion 3

2 Problembeskrivning 3

3 Antaganden och krav 3

4 Lösningsdesign 3

5 Diskussion 8

6 Källkod 8

7 Eventuellt referenser 14

# Introduktion

Den här uppgiften handlar om att göra en satsnings spel med hjälp av c++. Uppgiften går ut på att användaren ska lägga in pengar sedan satsa pengar. När den har gjort det så ska en tre gånger tre ruta slumpas fram av symboler. Det man vill då är att symbolerna blir lika i så många rader som möjligt.

# Problembeskrivning

De första problemet som man måste lösa för att skapa programmet är att göra funktioner. Det är ett krav att göra tre olika funktioner i programmet. Det är även ett krav att en av de funktionerna är en parameter som betyder att den tar emot indata och en return som returnerar data. Sedan är andra problemet att spelplanen måste skrivas ut varje gång spelaren satsar pengar. Spelplanen måste vara en tvådimensionell array med slumpmässiga symboler. Det tredje problemet är att sätta vinstsummorna. Om spelaren får en rad av samma symboler så får den insatserna två gånger sen ökar siffran för varje rad. Om det är två rader så får den gånger tre, vid tre rader så får den gånger fyra, fyra rader så får den gånger fem, fem rader så får den gånger sju och blir det en jackpot och spelaren får fullt spelfält så får den gånger tio. Det fjärde problemet är att spelaren ska kunna se vinstsumma tydligt efter varje omgång. Det femte problemet är att spelaren ska själv bestämma hur många gånger den kör. Spelet ska kunna gå att avsluta när som helst.

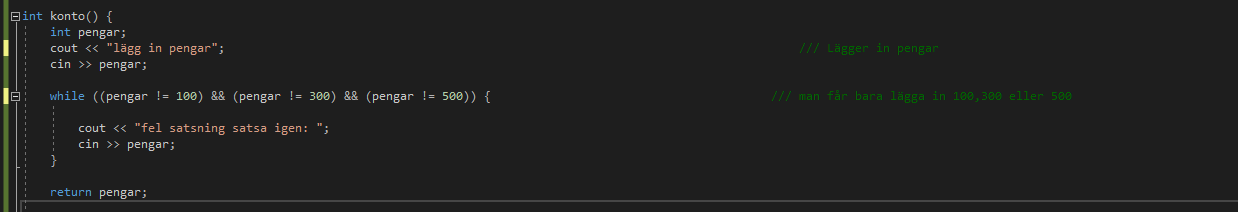
# Antaganden och krav

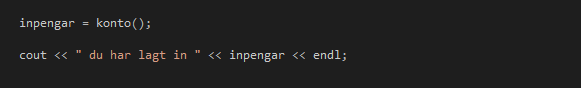
Ett dolt krav som måste besvaras är vad som händer om spelaren får 6 rader. Ett annat dolt krav är att konto saldot ska tydligt synas under hela programmet. Detta kommer att hjälpa spelaren att komma ihåg hur mycket den kan satsa. Det tredje och sista dolda kravet är att spelaren inte kan satsa pengar om den har inga pengar på kontot.

# Lösningsdesign

Beskriv hur du tänkt och gått tillväga för att lösa uppgiften. För uppgift 1 anger du även ditt flödesdiagram: en modell som löser uppgiftens problem.

**Funktioner**

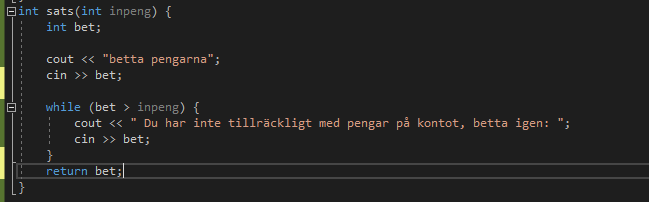
Det första problemet var att göra självständiga funktioner. Den första funktionen som behövdes var att lägga in pengar. 

Denna funktion är en parameter då den returnerar ett värde. Funktionen måste vara utanför main för att det ska vara en funktion. I denna funktion lägger jag till pengar med hjälp av att göra en int för pengarna och sedan lägga in pengar i min variabel med hjälp av cin. Sedan så behövdes det en while loop för att bestämma att det inte får läggas något annat än 100, 300 eller 500. En while loop kommer att förhindra spelet kan gå vidare i oändlighet tills spelaren sätter in rätt värde. Efter det så returneras värdet till huvudprogrammet i en variabel. 

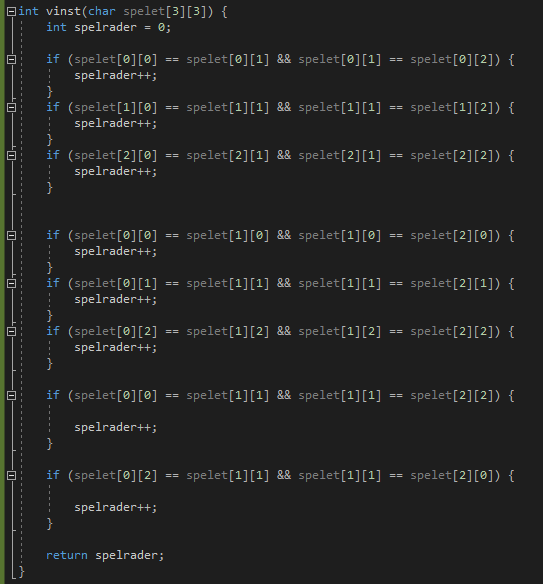
Det viktiga är att skriva ut konto för att funktionen ska köras i programmet. Det är dock också viktigt att lägga över funktionens resultat i en ny variabel som man redan kan skriva ut genom en cout. Nu kommer det att synas hur mycket du har lagt in så fort du har lagt in pengar. Efter det ska det skapas en funktion för satsning. Den funktionen ska innehålla indata. Först måste pengarna som användaren har lagt in skickas in till satsnings delen. Detta gör man med att skriva den data man vill överföra till funktionen i en parantes bredvid den funktionen den ska skickas till.



Efter det behöver man skriva en ny variabel bredvid funktionen för att det ska överföra variabeln.



Nu har pengarna spelaren har lagt in skickats till den funktionen. Det som sker i den funktionen är att man först skriver in sin satsning och sedan bestämmer att satsningen inte kan vara större än pengarna spelaren har lagt in. Sedan returnerar vi funktionen till huvudprogrammet för att göra en variabel för satsningen. Anledningen till varför man gör en variabel för pengar och satsningen i huvudprogrammet är för att om man skulle skriva ut funktionens namn varje gång man vill tillkalla den så skulle funktionen köras om i programmet. Nu innehåller programmet både en indata och en parameter. Det ända som saknas nu är en tredje själv stående funktion för att färdigställa första kravet. Den tredje funktionen känner av hur många rader spelaren har vunnit.

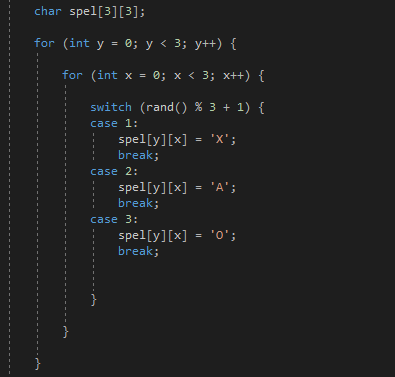


Först tillkallas det data ifrån spelet som ligger i huvudprogrammet. Nu vet funktionen vilka slumpmässiga symboler som skrevs ut på programmet. Det den gör nu är att den kollar om varje rad lodrätt, vågrätt och diagonal har samma symboler. Om de har samma symboler så ger den de en poäng. Sedan skickar den informationen till huvudmenyn igen då det skrivs ut hur många rader spelaren vann.

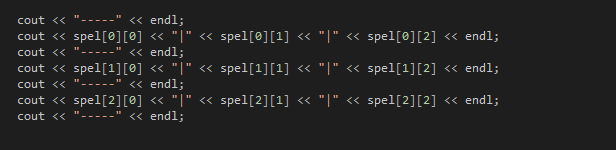


**Spelfält**

Ett av kraven för programmet är att spelfältet ska vara en tvådimensionell array som får ut slumpmässiga siffror.



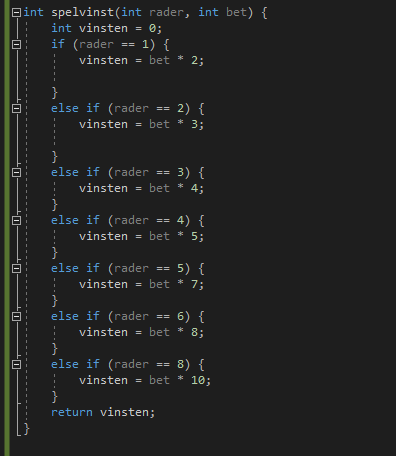
Spelfältet gjordes med hjälp av en char för att det går att skriva ut bokstäver med det. Efter att den tvådimensionella arrayen är skapad så måste vi nu få in slumpmässiga symboler i arrayen. Det gör skapade med att sätta in två for loopar. En är för y som är lodrätt och x som är vågrätt. Det börjar på att båda x och y är 0. Sedan plussar man på 1 hela tiden till 3 som är utanför arrayen då den börjar på 0. Samtidigt som for loopen börjar på 0.0 så har vi en switch sats. En switch sats funkar som en if sats. Den ger olika värden till våra boxar i arrayen med hjälp av ran() %3+1. När vi har fått en slumpmässig siffra så går den till en av de casen som står längre ner. Om det får till exempelvis en etta så skrivs det ut som ett x. Sedan börjar den om igen och då plussar man på 1 både på x och y tills hela arrayen är utritade med slumpmässiga symboler. Nu behövs det att rita ut programmet. Det görs med en cout.



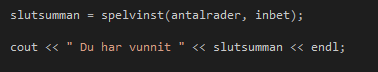
Hela spelets koordination måste skrivas ut för att få med hela spelet. Spelet måste även innehålla sträck i mellan så att siffrorna inte sitter ihop. Det är även viktigt att avsluta raden så spelet inte sitter i en lång rad.

**Vinstsummor**

Ett till krav var att spelaren skulle vinna en viss summa beroende på vilken hur många rader som är lika.

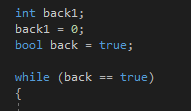


Först tillkallade vi både de vunna raderna och satsningen spelaren har lagt ifrån huvudprogrammet. Det funktionen gör sen är att kolla hur många rader spelaren vann. Vinstsummorna är utsätta ut efter kravet. När funktionen räknar ut hur många rader spelaren vann returnerar den värdet till huvudprogrammet som gör en ny variabel och skriver ut det till spelare.

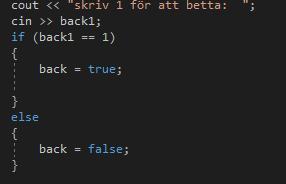


**Avsluta**

Ett av kraven är att spelaren ska kunna avsluta programmet när den vill. Det första som måste görs är att ge spelaren ett alternativ att lägga till pengar om den vill eller behöver det. Efter det så görs det en while loop med en bool.



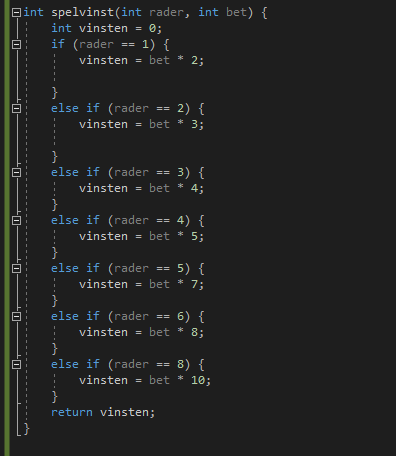
Först skapas det en ny variabel som är 0 och sedan en bool som är true. Om boolen är true så kommer programmet köra while loopen.



Här får spelaren bestämma om den vill köra vidare spelet när den trycker på 1. Då kommer bool vara true och då skickas spelaren till satsningen längre upp på huvudprogrammet. Annars kommer bool vara false och då kommer den avsluta programmet.

**Dolda krav**

Det första dolda kravet handlar om vad som spelaren vinner om hen får sex rader.



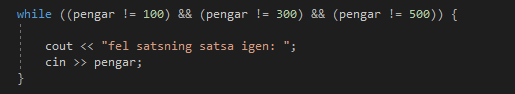
Problemet löstes med att spelaren får vinner gånger åtta om hen får sex rader. Det andra dolda kravet är att det ska tydligt synas hur mycket spelaren har på kontot. Detta ska vara med i början av programmet efter att spelaren har lagt in pengarna och efter att spelet är färdigt.



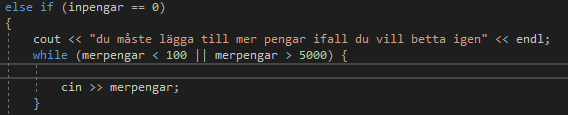


Sista dolda kravet är att spelaren inte ska kunna satsa pengar med 0 pengar på kontot. Det gjordes på två ställen på programmet. Ett i början när spelaren ska lägga in pengar och ett när den ska spela om.

Början:



Slutet:



# Diskussion

Diskutera din lösning genom att identifiera dess styrkor och brister.

Spelets styrkor är att det funkar precis som det skulle göra. Jag fick med alla krav och även de dolda för att spelet ska kunna funka. Det är även väldigt enkelt att förstå sig på både koden och själva spelet. Bristerna är docka att det är för mycket kodning i huvudprogrammet. Programmet kunde bestått bara utav funktioner. Jag försökte med det men det blev väldigt krångligt. En annan brist är att spelet buggar när man skriver bokstäver. Det har jag inga kunskaper om hur jag skulle kunna fixa det.

# Källkod

Bifoga din källkod i rapporten. Formatera den så att det går att följa koden. Ange kommentarer i koden som tydligt beskriver dess avsikt. Observera att koden måste överensstämma med flödesdiagrammet.

#include "pch.h"

#include <iostream>

#include <ctime>

using namespace std;

int konto() {

int pengar;

cout << "lägg in 100,300 eller 500: ";                                                                                  /// Lägger in pengar

cin >> pengar;

while ((pengar != 100) && (pengar != 300) && (pengar != 500)) {                                        /// man får bara lägga in 100,300 eller 500

 cout << "fel satsning satsa igen: ";

 cin >> pengar;

}

return pengar;

}

int sats(int inpeng) {                                                      /// satsa pengar

int bet;

cout << "betta pengarna";

cin >> bet;

while (bet > inpeng) {                                                     /// Får inte satsa mer än vad som finns på konto

 cout << " Du har inte tillräckligt med pengar på kontot, betta igen: ";

 cin >> bet;

}

return bet;

}

int vinst(char spelet[3][3]) {                                                            /// beräknar vunna rader

int spelrader = 0;

if (spelet[0][0] == spelet[0][1] && spelet[0][1] == spelet[0][2]) {                  /// beräknar alla lodrätta rader

 spelrader++;

}

if (spelet[1][0] == spelet[1][1] && spelet[1][1] == spelet[1][2]) {

 spelrader++;

}

if (spelet[2][0] == spelet[2][1] && spelet[2][1] == spelet[2][2]) {

 spelrader++;

}

if (spelet[0][0] == spelet[1][0] && spelet[1][0] == spelet[2][0]) {                   /// beräknar alla vågrätta rader

 spelrader++;

}

if (spelet[0][1] == spelet[1][1] && spelet[1][1] == spelet[2][1]) {

 spelrader++;

}

if (spelet[0][2] == spelet[1][2] && spelet[1][2] == spelet[2][2]) {

 spelrader++;

}

if (spelet[0][0] == spelet[1][1] && spelet[1][1] == spelet[2][2]) {                  /// beräknar båda diagonalerna

 spelrader++;

}

if (spelet[0][2] == spelet[1][1] && spelet[1][1] == spelet[2][0]) {

 spelrader++;

}

return spelrader;

}

int spelvinst(int rader, int bet) {            /// beräknar summan man har vunnit beroende

int vinsten = 0;

if (rader == 1) {

 vinsten = bet \* 2;

}

else if (rader == 2) {

 vinsten = bet \* 3;

}

else if (rader == 3) {

 vinsten = bet \* 4;

}

else if (rader == 4) {

 vinsten = bet \* 5;

}

else if (rader == 5) {

 vinsten = bet \* 7;

}

else if (rader == 6) {

 vinsten = bet \* 8;

}

else if (rader == 8) {

 vinsten = bet \* 10;

}

return vinsten;

}

int main()

{

setlocale(LC\_ALL, "swedish");

srand(time(0));

int inpengar;

int inbet;

int summa;

int y = 0;

int x = 0;

int antalrader = 0;

int vinstpengarna = 0;

int slutsumman = 0;

inpengar = konto();                                                /// Kör funktionen pengar och sedan överför all data till inpengar som är spelar kontot(return)

cout << " du har lagt in:  " << inpengar << "på kontot" << endl;

int back1;

back1 = 0;

bool back = true;

while (back == true)                                                                          /// kör en loop på spelet efter första omgången

{

 inbet = sats(inpengar);                                                    /// Kör satsningens funktionen, överför data till inbet som är variabel för sasningen i huvudprogrammet(return), Sen skickas det data ifrån spelar kontot upp till funktionen satsning(parameter)

 cout << " du har bettat:  " << inbet << endl;

 inpengar = inpengar - inbet;                                              /// tar bort satsningen ifrån pengarna på kontot

 char spel[3][3];                                                                 /// skapar spelet

 for (int y = 0; y < 3; y++) {                                              /// kört en loop som ger bokstäverna random siffror

  for (int x = 0; x < 3; x++) {

   switch (rand() % 3 + 1) {                                              /// därefter med hjälp av swistch omvandlar de till symboler

   case 1:

    spel[y][x] = 'X';

    break;

   case 2:

    spel[y][x] = 'A';

    break;

   case 3:

    spel[y][x] = 'O';

    break;

   }

  }

 }

 cout << "-----" << endl;                                                                /// skriver ut spelet med hjälp av en cout

 cout << spel[0][0] << "|" << spel[0][1] << "|" << spel[0][2] << endl;

 cout << "-----" << endl;

 cout << spel[1][0] << "|" << spel[1][1] << "|" << spel[1][2] << endl;

 cout << "-----" << endl;

 cout << spel[2][0] << "|" << spel[2][1] << "|" << spel[2][2] << endl;

 cout << "-----" << endl;

 antalrader = vinst(spel);                                                                   /// Kör funktionen som räknar hur många rader som spelaen vann, överför datan till en variabel i huvudprogrammet, skickar upp spelet till funktionen(parameter)

 cout << " du har fått:" << antalrader << "rader!" << endl;

 slutsumman = spelvinst(antalrader, inbet);                                                 /// kör funktionen som räknar ut spelvinsten, överför datan till variabel i huvudprogrammet, skickar data till funktionen om satsningen och de antal raderna som spelaren vann.

 cout << " Du har vunnit " << slutsumman << endl;

 inpengar = slutsumman + inpengar;                                                       /// räknar ut hur mycket spelaren har kvar på kontot efter omgången.

 int spelaigen;

 int merpengar;

 merpengar = 0;

 cout << "du har: " << inpengar << " på kontot" << endl;

 cout << "vill du spela igen? (1 för ja eller 2 för nej)";

 cin >> spelaigen;

 if (spelaigen == 1) {                          /// bestämmer om spelaren vill spela vidare eller inte

  if (inpengar > 0)                  /// om spelaren vill spela vidare så måste den ha mer än 0 på kontot, sen kan den välja om den vill lägga till mer

  {

   cout << "vill du lägga till mer pengar(skriv 0 om du inte vill): ";

   cin >> merpengar;

   while (merpengar < 0 || merpengar > 100000) {

    cout << "Du har gjort fel: " << endl;

    cin >> merpengar;

   }

  }

  else if (inpengar == 0)                        /// om den har 0 så måste den lägga in något.

  {

cout << "du måste lägga till mer pengar ifall du vill betta igen" << endl;

while (merpengar < 1 || merpengar > 100000) {

    cin >> merpengar;

   }

  }

  inpengar += merpengar;

  cout << "skriv 1 för att betta:  ";

  cin >> back1;

  if (back1 == 1)

  {

   back = true;

  }

  else

  {

   back = false;

  }

 }

 else {

  cout << "tack för att du spelade" << endl;

  return 0;

 }

}

}

# Eventuellt referenser

Föreläsningar 5 och 7 med [Christian Lennerholt](http://www.his.se/lenc/)

https://www.tutorialspoint.com/cplusplus/cpp\_switch\_statement.htm