Kursnamn: Grundläggande programmering med C++

Kurskod: IT120G

Program: Webbutveckling (WEBUG)

Namn: Oscar Gustavsson

Användarnamn: a21oscgu

Mail: a21oscgu@student.his.se

Inlämningsuppgift 2

”Kortduellen”

1. Problembeskrivning

Den här rapporten handlar om programmet ”Kortduellen” som har skapats med hjälp av LucidChart och Microsoft Visual Studio. C++ programmet som har skapats går i grund och botten ut på att två användare ska kunna spela ett kortspel mot varandra. Innan kortspelet börjar blandas kortleken och spelarna får därefter dra varsitt kort för att se vem som slutligen har dragit kortet med högst värde. Spelaren med högst valör på sitt kort vinner rundan och den som vinner bäst av tre rundor har vunnit en spelomgång. För att kunna spela spelet behöver användarna, enligt de redan fördefinierade kraven, kunna dra kort ur en kortlek, vilket leder till det första delproblemet.

* 1. Problem 1: kortleken

Programmets första delproblem är att programmet behöver någon form av array som representerar kortleken. Detta delproblemet bär med sig ett flertal mindre delproblem, vilka kan beskrivas med hjälp av ett antal frågeställningar. Vad är det för typ av kortlek? Hur kan korten representeras i en array på ett effektivt sätt och finns det jokrar i kortleken?

* 1. Problem 2: att blanda kortleken

Det andra större delproblemet i C++ programmet är den mest väsentliga delen inom programmet, att blanda kortleken. Det bakomliggande problemet innan användarna drar sina första kort är därför hur kortleken kan blandas på ett realistiskt sätt.

* 1. Övriga problem och funktioner i programmet

Det sista övergripande delproblemet i programmet ”Kortduellen” är spelets alla övriga funktioner och detaljer. Det första delproblemet är att kortspelet behöver interagera med användarna genom hela spelets gång, därför att användarna ska kunna förstå vad programmet exekverar i realtid. Det andra delproblemet är hur användarna ska dra kort. Med andra ord, hur ska den här processen simuleras? Det tredje delproblemet i programmet är ifall användarna ska kunna avsluta spelet när som helst under spelets gång. De två sista delproblemen är för det första ifall användaren ska kunna spela mer än en spelomgång utan att behöva starta om spelet och för det andra vad som sker ifall användaren matar in fel tecken vid inmatning.

1. Antaganden och krav
   1. Problem 1: kortleken

De krav som redan har definierats för programmets kortlek är för det första att korten i kortleken ska ha olika valörer och olika färger och för det andra att en array används för att representera kortleken.

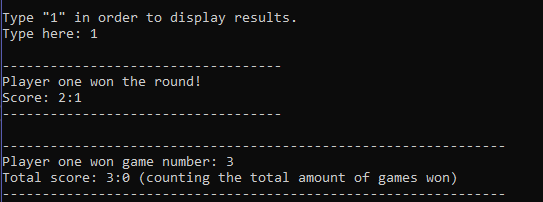
Ett eget antagande som har ställts på kortleken är att den inte ska innehålla några jokrar samt att valören ess i denna kortleken representerar talet fjorton och inte talet ett. Det tredje antaget är att kortleken är en klassisk kortlek som innehåller femtiotvå kort, från den lägsta valören: två till den högsta valören: ess. Ett krav som har ställts på programmet är att det ska skriva ut valörerna elva, tolv, tretton och fjorton med dess särskilda namn, nämligen knekt (jack), drottning (queen), kung (king) och ess (ace).

* 1. Problem 2: att blanda kortleken

Hela kortleken ska blandas innan spelarna drar sina kort enligt de fördefinierade kraven. På grund av det har ett antagande ställts på programmet, vilket är att korten i kortleken, som representeras i form av en array, ska byta plats i array:en. Programmet ska alltså, med andra ord, inte slumpa ut kort till spelarna ur en sorterad kortlek, utan kortleken ska blandas som en vanlig kortlek och därefter ska spelarna enkelt kunna dra de två översta korten (se rad 164–181 i källkoden).

* 1. Övriga problem och funktioner i programmet

De övriga fördefinierade kraven är bl.a. att spelaren med högst valör på sitt kort vinner rundan. Ifall spelarna drar samma valör kommer färgen på kortet istället att avgöra vinsten. Spader har högst värde, hjärter näst högst, ruter lägre och klöver den lägsta färgen. Det innebär t.ex. att en spelare med kortet spader 2, vinner över en spelare med kortet klöver 2. Fler krav som redan har definierats för spelet är att programmet tydligt ska mata ut vilka kort som spelarna har dragit innan vinnaren avgörs samt att programmet tydligt ska visa spelarna vem som leder en spelomgång samt vem det är som vinner spelomgången. Dessutom ska ett poängsystem finnas, också ska programmet vara uppdelat i tre stycken funktioner. Ett antagande som har ställts på programmets poängsystem är att det endast ska räkna spelarnas vinster, ej spelarnas antal förluster, gentemot varandra. Anledningen till att detta beslutet har tagits är för att det enkelt går att räkna ut hur många förluster som exempelvis spelare ett har, genom att istället endast räkna antalet vinster som spelare två har (se bild 1).



*Bild 1 – Bilden visar ett skärmklipp där poängssystemet berättar att spelare ett har vunnit tre spelomgångar. Spelare två har alltså förlorat tre spelomgångar.*

1. Lösningsdesign i C++

För att lösa de frågeställningar och krav som tidigare ställts har ett flertal tekniker använts i det skapade programmet ”Kortduellen” och något viktigt att notera är att programmet matar ut meddelanden till användaren vid varje delmoment som sker under spelets gång.

* 1. Problem 1: Kortleken

För att representera kortleken som en array har en endimensionell array använts (se bild 2). Kortleken består av 52 element eftersom kortleken består av 52 kort.

// Array för kortleken

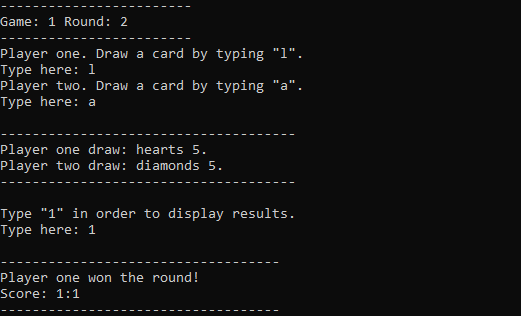
int deck[52] = { 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,

33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59 };

*Bild 2 – Bilden visar källkoden på den endimensionella array:en ”deck”, som representerar kortleken (rad 160–161 i källkoden).*

Datatypen för heltals-array:en är av typen heltal (integer). Det finns flera matematiska fördelar med att representera kortleken i form av heltal. En fördel med att representera kortleken i form av heltal är att det blir mycket enklare att jämföra tal som exempelvis 2 och 7 när dem är i form av heltal och inte i form av två enskilda strängar. Programmet kollar vilken spelares kort som har högst valör och då är denna fördelen mycket användbar.

En annan fördel med att använda heltal är för att det går att representera varje enskilt kort i form av ett heltal var. Första kortet i array:en bär numret 8, andra kortet bär numret 9 och så vidare ända tills det sista kortet, som bär numret 59 (se bild 2). För att räkna ut vilken färg som kortet bär används modulo (%) 4, eftersom kortleken består av fyra färger. Kortets representerande nummer modulo 4, leder till en ny siffra mellan 0 och 3. Siffran 0 representerar färgen spader, siffran 1 representerar färgen hjärter, siffran 2 representerar färgen ruter och siffran 3 representerar färgen klöver. Med hjälp av denna beräkningen är det enkelt att jämföra vilken spelare som vinner när spelarna har samma valör på kortet (bild 3).



*Bild 3 – Bilden visar ett skärmklipp där två spelare har dragit kort med samma valör, 5. Spelare ett vinner rundan eftersom kortfärgen hjärter (1) har lägre värde än ruter (2, se rad 200–212 i källkoden).*

För att räkna ut vilken valör som kortet bär används division (/) 4. Kortets representerade nummer delat med 4 (fyra används eftersom det finns fyra färger i kortleken) kommer att bli lika med kortets valör. T.ex. representeras det sista elementet i kortleken av talet 59. Talet 59 modulo 4 blir lika med 3, elementet bär alltså färgen klöver. Talet 59 delat med 4 blir 14,75, vilket i C++ avrundas till talet 14, som står för valören ess. Talet 59 i array:en representerar alltså kortet klöver ess.

Den tredje fördelen med att representera korten som enskilda heltal är att det inte blir komplicerat att blanda korten. Ifall det första kortet ska byta plats med det sista kortet, kommer talen 8 och 59 att byta plats. Elementet 59 kommer fortfarande att representera kortet klöver ess och kortet 8 kommer fortfarande att representera kortet spader två även om placeringen på dessa två element har ändrats (se bild 2).

* 1. Problem 2: Att blanda kortleken

För att blanda kortleken används funktionen ”shuffleDeck” (se bild 4).

void shuffleDeck(int deck[], int lengthOfDeck)

{

int temporary = 0, randomIndex = 0;

for (int card = 0; card < lengthOfDeck; card++)

{

randomIndex = rand() % lengthOfDeck;

temporary = deck[card];

deck[card] = deck[randomIndex];

deck[randomIndex] = temporary;

}

}

*Bild 4 – Bilden visar källkoden till funktionen ”shuffleDeck” (se rad 25–35 i källkoden).*

Den här funktionen tar emot två parametrar, den ena är en array med datatypen integer och den andra är en variabel med datatypen integer. Variabeln ”lengthOfDeck” representerar antalet element som ska skickas in i funktionen från array:en. På rad 160–161 initieras kortleken ”deck” och på rad 163 anropas funktionen ”shuffleDeck”, där parametrarna ”deck” och ”52” skickas in i funktionen. Funktionen ”shuffleDeck” består endast av en for-loop som fortsätter så länge det finns fler kort att blanda, i det här fallet går loopen igenom och blandar 52 gånger. Växlingen mellan två element i kortleken är inte komplicerad. For-loopen kommer att börja med att byta ut det första elementet i array:en, vilket är element 0, med ett slumpmässigt element. Detta sker genom att element 0:s värde först kopieras in i en temporär variabel för att sedan skrivas över med det slumpmässigt valda elementets värde. Därefter skrivs det slumpmässigt utvalda elementets värde över med den temporära variabelns värde. Efter det kommer for-loopen i programmet att fortsätta växla värden, alltså kort, ifall det finns fler kort att blanda. Efter att kortleken har blandats kommer spelarna få dra de två översta korten i kortleken (vilket kommar ett bli de två första korten, element 0 och 1). Funktionen ”shuffleDeck” är en funktion som inte returnerar något värde.

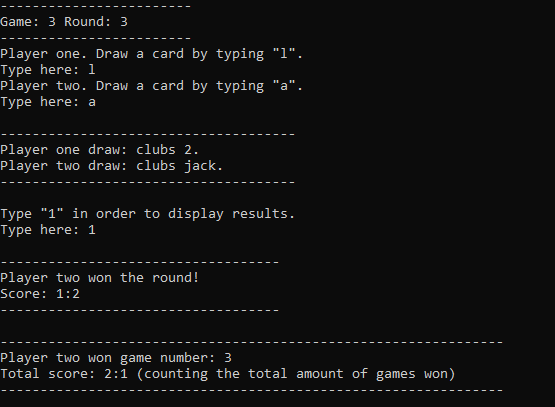
* 1. Bekräftelse

En funktion i programmet som används två gånger är funktionen ”confirmation” (se rad 4–23 i källkoden). Denna funktionen tar inte emot några parametrar men returnerar ett heltal i form av en variabel. Funktionens syfte är att användas ifall spelarna vill stänga av spelet i både spelets start och i spelets slut. Dessutom är syftet med funktionen också att korta ner programmets källkod eftersom funktionen används mer än en gång i programmet. Ifall spelarna anger att de vill stänga av spelet i början eller slutet av spelet kommer funktionen ”confirmation” att anropas och frågan ”är du säker?” (are you sure, på engelska) kommer att ställas till spelarna. Ifall spelarna anger att de är säkra kommer spelet att stängas av eftersom att heltalet 1 returneras. Ifall spelarna istället anger att de är osäkra kommer funktionen att returnera heltalet 0, också kommer programmet att gå tillbaka till det tidigare steget i spelet.

* 1. Övrigt

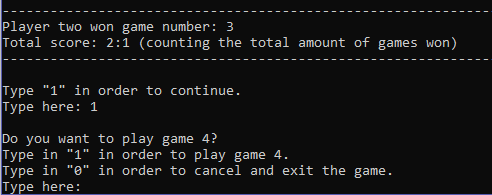
Kortleken kommer alltid att vara sorterad när spelet startas. Dock kommer kortleken alltid att blandas innan spelarna drar sina kort. I spelet kommer spelare 1 att dra sitt kort genom att slå in ett ”l” på tangentbordet när det efterfrågas. Spelare 2 kommer därefter att dra sitt kort genom att slå in ett ”a” på tangentbordet när programmet ber spelaren att dra sitt kort. Ifall en utav spelarna istället slår in en nolla kommer spelet att avslutas direkt. Tangenterna ”l” och ”a” används eftersom spelarna skall kunna sitta bredvid varandra när dem spelar kortspelet ”kortduellen”.

När spelarna spelar en runda kommer de först att få dra ett kort var, därefter kommer korten att presenteras och till sist kommer spelet att berätta vem som vann spelrundan. Det går att beräkna vem som vann rundan innan spelet berättar vem som vann rundan genom att endast jämföra korten själv. När en runda är avslutad kommer spelet även berätta för spelarna vem det är som leder spelomgången samt att de kan gå vidare med nästa runda (i de fall då en spelare inte redan har vunnit hela spelomgången). Ifall en spelare vinner en spelomgång, alltså två av tre rundor, kommer spelet att berätta vem som vann den sista rundan, vem som vann spelomgången samt vem som har vunnit flest spelomgångar (se bild 5).



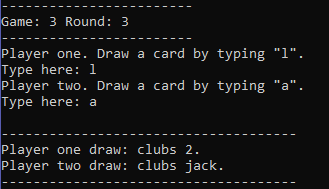
*Bild 5 – Bilden visar ett skärmklipp på den sista rundan i en spelomgång samt poängsystemet.*

Ifall spelarna vill spela mer än en spelomgång, är detta möjligt. Detta kommer att synas eftersom programmet har kapaciteten att beräkna antalet spelomgångar samt eftersom poängsystemet kommer att räkna antalet vinster mellan spelarna (se bild 6).



*Bild 6 – Bilden visar ett skärmklipp på en spelare som vann spelomgång nummer 3 samt ett skärmklipp på två spelare som får förfrågan om de vill spela en till spelomgång.*

Den tredje och sista funktionen i programmet är en funktion med namnet ”printCardColor”. Syftet med denna funktionen är att förbättra utseendet på spelet. Efter att spelarna har dragit sina kort ska spelarnas kort presenteras och då anropas funktionen ”printCardColor”. Funktionen returnerar inga värden men den tar emot en del parametrar för att sedan, beroende på parametrarnas värden, skriva ut olika svar i fönstret. Exempelvis skriver funktionen ut orden ”clubs jack” ifall den ena spelarens kort är en klöver knekt. Utan funktionen ”printCardColor” skulle programmet ha presenterat spelarens kort som ”3 11” vilket inte är lika enkelt att förstå som ”clubs jack” (se bild 7). Detsamma gäller för de andra speciella korten. Ifall valören 12 dras skrivs ”queen” ut istället, ifall valör 13 dras skrivs ”king” ut istället och ifall valör 14 dras kommer programmet istället att skriva ut ordet ”ace”.



*Bild 7 – Bilden visar ett skärmklipp där två spelare har dragit korten klöver två och klöver knekt. Orden ”clubs jack” skrivs ut eftersom kortet har färgen med talet 3 och valören med talet 11 samt för att funktionen ”printCardColor” istället väljer att skriva ut ”clubs jack”.*

1. Diskussion

Array:en blev enligt mig väldigt effektiv och simpel eftersom den inte är multidimensionell, visst kan det vara svårt att förstå den vid första ögonkastet men dokumentationen i denna rapporten är meningen att läsas. Dessutom blev det enligt mig en väldigt bra kombination med ”deck” och ”shuffleDeck” eftersom programmet på det sättet faktiskt speglar en realistisk kortlek som går att blandas.

Alla variabler i programmet som inte behöver vara globala har placerats så lokalt det går eftersom det är onödigt att ha variabler på global nivå.

Utav tre funktioner är det endast en funktion som återanvänds i programmet. Funktionen som används mer än en gång är funktionen ”confirmation”. Funktionerna ”shuffleDeck” och ”printCardColor” används endast en gång var i programmet och hade likagärna kunnat skrivits ut i main.

Ifall spelarna matar in ett fel värde vid ett givet steg, kan spelet i vissa fall hamna i en oändlig loop. Min utgångspunkt i denna uppgift har varit att skapa ett program för spelare som inte matar in fel sorts datatyper.