

Sänka skepp

Ett program skrivet i C++

Inlämningsuppgift 3, Procedurell Programmering HT 2011

Emma Jonsson (a11emmjo@student.his.se), Webbutvecklare - Programmering

Sammanfattning

I denna rapport får man följa skapandet av ett spel där användaren skall lista ut var datorn har slumpat fram 2 skepp på en spelplan, även är känt som det gamla hederliga "Sänka skepp". Spelet skrivs i C++kod och man får följa med ända från kraven som uppdragsgivaren ställer, via självaste programmeringsfasen till slutprodukten.

1 Introduktion

Uppgiften var att skapa ett spel med C++ som sedan skulle spelas i terminalmiljö. Spelet skulle vara det gamla hederliga "Sänka skepp", men det skulle bara fungera åt ett håll. När användaren startade spelet skulle två skepp slumpas ut på en spelplan och sedan skulle användaren försöka sänka skeppen genom att ange en ruta som denne ville skjuta på. Om ett skepp fanns på vald ruta skulle skeppet träffas och detta skulle markeras med en ring, men om det däremot inte fanns något skepp skulle detta markeras med ett kryss. Oavsett resultat skulle feedback ges och om användaren lyckades sänka bägge skeppen skulle spelet avslutas. Användaren skulle alltså inte själv slumpa ut några skepp som datorn i sin tur skulle sänka. Målet med spelet skulle vara att sänka skeppen på så få antal skott som möjligt.

2 Krav

Vid skapandet av spelet fanns vissa krav man var tvungen att följa. Några har redan tagits upp i introduktionen men jag väljer ändå att återupprepa dessa och alltså lista alla krav här nedan.

- Spelet ska skapas med hjälp av C++.
- Spelfältet skall representeras av ett rutfält på 5*5 rutor.(se Bild 1)

	A	B	C	D
1				
2				
3				
4				

Bild 1 Spelfältet

- Spelfältet inkluderar alltid 2 skepp.
- Vardera skepp är i storleken 1 ruta.
- Skeppens placering skall slumpas fram innan varje spelomgång startar.
- Spelfältet skall alltid skrivas ut på skärmen när användaren anger ett skott.
- En träff skall representeras med symbolen O.
- En miss skall representeras med symbolen X.
- Varje skott skall räknas för att kunna presenteras för användaren när skeppen är sänkta.

- Spelomgången är slut då användaren sänkt bägge skeppen.
- Användarens highscore, dvs. minst antal skott för att sänka bägge skeppen, ska kunna visas.
- Användaren skall kunna få valet att visa highscoren innan varje spelomgång börjar.
- Highscoren är alltid 0 när spelet startas för första gången.
- Användaren får spela hur många gånger som helst.
- Programmet måste vara uppdelat i minst 3 st. egendefinierade funktioner.
- En tvådimensionell array måste användas för att representera spelfältet.

3 Egna antaganden

Då man försökte att lösa problemet med hjälp de krav man fått så uppstod många luckor. Där fick man alltså göra egna antaganden för att kunna skapa ett komplett fungerande spel. Nedan listas de antaganden som har ansetts varit nödvändiga för att uppgiften skall bli komplett.

- Användaren behöver kunna läsa reglerna om denne ej har spelat detta eller liknande spel tidigare.
- Eftersom möjligheten att kunna se sin highscore och kunna välja när man vill avsluta skulle finnas, så behöver någon sorts meny skapas.
- 4 val bör finnas i menyn och för att välja ska man skriva in korrekt siffra. 1 för att läsa regler, 2 för att starta en spelomgång, 3 för att se sin highscore och till sist 4 för att avsluta programmet.
- Skeppen som slumpas ut ska inte kunna hamna på samma ruta.
- Innan skeppen slumpas ut inför varje ny spelomgång måste deras placering nollställas för att de inte ska kunna existera fler än 2 skepp.
- När man uppger ruta att skjuta på ska detta göras i koordinater. Valfri bokstav i x-led (A-D) och valfri siffra i y-led (1-4). Ex. C2.
- Det skall vara möjligt att uppge koordinaterna med både små och stora bokstäver.
- Koordinaten ska antingen kunna uppges med bokstaven först eller siffran först.
- Om man ber om att få se highscoren innan man spelat en omgång ska man få som feedback att detta inte är möjligt.
- Om man väljer att avsluta skall det kontrolleras om man verkligen vill avsluta. Detta för att undvika ev. feltryckning.
- Det är lätt att råka trycka fel när man exempelvis skall knappa in svaret på en fråga, därav behövs det ett säkerhetsnät som fångar upp ev. felinmatningar.

4 Problemhantering

För att kunna hantera det stora grundproblemet, som var att skapa spelet, så var man tvungen att bryta ner det i mindre delproblem. Nedan listas de problem som dök upp och hur de löstes under arbetets gång.

4.1 Menyn

Problemet var att programmet behövde ha en meny som skulle finnas tillgänglig hela tiden utom under själva spelomgången. Menyn skulle ha 4 olika val som skulle kunna väljas genom att användaren skrev in en valfri siffra. Om användaren råkade skriva en ogiltig input skulle detta inte påverka programmet utan användaren skulle bara bli informerad om att denne skrivit fel och få försöka igen.

Lösningen blev att skapa en funktion kallad **meny** av typen `int`. Valet av typen `int` var för att resultatet skulle bli en siffra. När funktionen kördes ombads användaren att göra ett val utifrån de fyra förslag som sedan följde. Om användaren skrev en giltig input, alltså 1, 2, 3 eller 4, så sparades valet i en variabel som sedan skulle bli funktionens output-värde. Om användaren däremot skrev in en ogiltig input så ignorerades inputen med hjälp av `cin.ignore` och feltillståndet, som skulle orsaka en evighetsloop, upphävdes med hjälp av `cin.clear`. Allt detta omgavs av en `while`-loop som endast blev falsk och gjorde att funktionen avslutades om användaren skrivit in 1, 2, 3 eller 4.

I **main()** löstes detta sedan med att outputen från funktionen sparades i en variabel. Detta skrevs enligt följande: **`val=meny();`** Precis som i matematik så måste först värdet i parentesen räknas ut, i detta fall måste funktionen **`meny`** köras för att ett värde skall tas fram, och sedan sparas värdet i variabeln **`val`**. Variabelns värde kontrollerades sedan med hjälp av en `if`-sats och det som användaren hade valt utfördes.

4.2 Regler

Problemet var att om användaren inte hade spelat spelet tidigare så skulle denne få möjligheten att läsa spelets regler.

Lösningen blev att skapa en funktion av typen `void`. Detta för att funktionen inte skulle returnera ett resultat utan enbart skulle köras och sedan avslutas. När funktionen kördes skrevs reglerna ut och sedan avslutades funktionen.

I **main()** löstes detta sedan med att lägga in anropningen av funktionen **`regler`** i `if`-satsen som kontrollerade användarens val i menyn. Om valet hade varit 1 så kördes alltså regel-funktionen.

4.3 Spela

För att kunna spela spelet så behövde flera mindre delproblem lösas och dessa tas upp i tur och ordning, så som det utförs i programmet, nedan. Grundproblemet var dock att spelplanen skulle kunna ändra utseende beroende på vad som hände i spelet.

Lösningen till detta blev att skapa två stycken tvådimensionella arrayer (en tvådimensionell array kan liknas vid en tabell där du dels måste uppge antal rader, dels antal kolumner). En med typen `int` där man alltså kunde spara olika värden i arrayens element (rutor i tabellen) som styrde om rutan var tom, innehöll ett skepp eller hade blivit skjuten på och missat alt. träffat ett skepp. En annan med typen `string` där man istället kunde skriva in olika tecken. Dessa två skapades som globala variabler eftersom flera funktioner behövde ha tillgång till och kunna förändra deras värden.

4.3.1 Nollställa spelplanen

Problemet var att om man skulle kunna spela spelet flera gånger så var man tvungen att nollställa spelplanen.

Detta löstes genom att man först nollställde variabeln som sparade antal skott användaren skjutit, därefter variabeln som sparade antal träffar användaren fått, och till sist så anropades en funktion kallad **nolla** av typen void. I funktionen **nolla** så skapades en for-loop som gick igenom båda arrayernas element och satte dem alla till antingen 0 (i int-arrayen) eller mellanslag (i string-arrayen). Se förklaringen till detta i 4.3.3

Allt detta utfördes sedan först av allt om valet i menyn hade varit 2, dvs. att användaren ville spela.

4.3.2 Slumpa ut skeppen

Problemet var att två skepp skulle slumpas ut på spelplanen.

Lösningen blev att skapa en funktion kallad **slumpa** av typen void. I funktionen sattes en variabel för vardera skepp som sedan gavs ett slumpat värde mellan 1-16 med hjälp av en slump-funktion (detta för att det fanns 16 spelbara rutor på spelplanen). Detta placerades sedan i en while-loop med en if-sats som kontrollerade om **skepp1** och **skepp2** hade fått samma värde. Endast om **skepp1** och **skepp2** hade olika värden blev loopen falsk. Detta var för att förhindra att skeppen skulle kunna placeras på samma ruta.

Därefter så kontrollerades skeppen framslumpade värden med en if-sats. Beroende på vilka siffror som slumpats fram så förändrades två av int-arrayens element.

4.3.3 Spelplanen

Problemet var att skapa en spelplan bestående av 5*5 rutor (se Bild 1) där 4*4 rutor skulle kunna ha olika värden och därmed kunde förändra utseende. Spelplanen skulle dyka upp efter varje gång som användaren hade lagt en gissning för att se resultatet.

Lösningen blev att skapa en funktion av typen void som kallades **spelplan**. I funktionen skapades en for-loop som gick igenom alla int-arrayens element (rad för rad och kolumn för kolumn) och beroende på innehållet så förändrades elementens innehåll i string-arrayen. Värdet 0 betydde en tom ruta på spelplanen och blev i string-arrayen ett mellanslag. Värdet 1 betydde dolt skepp och blev även det ett mellanslag. Om värdet däremot var 2 eller 3 betydde det att användaren redan hade skjutit här och beroende på om det var miss eller träff blev det ett kryss alt. en ring. Anledningen till att en del rutor fick ett mellanslag som innehåll var för att spelplanen skulle hålla sin form.

Till sist skrevs spelplanen ut med hjälp av cout och i vardera ruta skrevs string-arrayens respektive element ut. För att spelplanen skulle skrivas ut på ett snyggt sätt användes även olika specialtecken som sparades i char-variabler.

4.3.4 Skjuta

Problemet var att man skulle kunna skjuta på valfri ruta på spelplanen och sedan skulle man få feedback på resultatet. Man skulle kunna skjuta flera gånger ända tills man hade skjutit ner skeppen och sedan skulle antal skott det tog sparas i en highscore.

Lösningen blev att skapa en while-loop och i den först av allt anropa funktionen **spelplan** (se 4.3.3), detta för att man ju hela tiden skulle kunna se sitt resultat på spelplanen när man skjutit ett skott.

Därefter anropades en funktion kallad **skott** av typen int. I skott-funktionen bads användaren att uppgge valfri koordinat och sedan kontrollerades svaret med en if-sats. Om inputen var längre än två tecken eller var en felaktig kombination (korrekt kombination finns att läsa i *Egna antaganden*) gick man direkt vidare till en cin.ignore och en cin.clear precis som i funktionen för menyn. Detta för att förhindra ev. ogiltig input. Allt omgavs av en while-loop som endast blev falsk och avslutade loopen och funktionen om man skrivit in en giltig input. Till sist returnerades antingen en etta eller nolla beroende på skottets resultat.

När funktionen hade körts kontrollerades det med hjälp av en if-sats om returen från den hade varit 1. Om så var fallet så adderades 1 till en variabel som sparade antal träffar.

Därefter adderades 1 även till en variabel som sparade de antal skott användaren hade skjutit.

Med hjälp av en if-sats kontrollerades sedan värdet på träff-variabeln. Om värdet var 1 så fick användaren feedback på att det nu bara fanns ett skepp kvar, men om värdet istället var 2 så utfördes följande kommandon. Funktionen **spelplan** anropades igen, detta för att man skulle se resultatet av vinsten, och sedan blev användaren informerad om att denne vunnit och antal skott det tagit att sänka bägge skeppen uppgavs. Därefter jämfördes med hjälp av en if-sats skott-variabeln med en variabel kallad **highscore** (se 4.4). Om skott-variabelns värde var mindre än **highscore** så skrevs highscoren över, men först kontrollerades det med hjälp av en if-sats om **highscore** hade värdet 17. Var det inte det så informerades användaren om att denne hade fått ett nytt highscore. Till sist gjordes while-loopen (se ovan) som sträckte sig över hela skjut-sekvensen falsk och användaren kom åter igen ut till menyn.

4.4 Highscore

Problemet var att användarens highscore skulle kunna sparas och visas när användaren bad om det.

Lösningen blev att skapa en variabel av typen int som kallades **highscore**. Värdet sattes sedan till 17 eftersom det var ett tal som var omöjligt för användaren att själv få då det bara fanns 16 rutor på spelplanen. Om användaren valde 3 i menyn (för att se sin highscore) så kontrollerades highscorens värde. Om värdet var 17 så informerades användaren om att det inte fanns någon highscore eftersom denne omöjligt kunde ha spelat än. Om värdet däremot var lägre än 17 så fick användaren se sitt highscore.

4.5 Avsluta

Problemet var att användaren själv skulle kunna välja när denne ville avsluta programmet.

Lösningen blev att om valet var 4 i menyn så ombads användaren bekräfta att denne verkligen ville avsluta. Svaret kontrollerades av en if-sats omgiven av en while-loop som gjorde att om svaret var en ogiltig input användes cin.ignore och cin.clear och användaren fick återigen svara. Om svaret var nej så avslutades loopen och användaren kom tillbaka till menyn. Om svaret däremot var ja så avslutades både denna loopen och den som omgav hela spelet (se 4.6) och programmet avslutades.

4.6 Spela igen

Problemet var att användaren skulle kunna spela spelet hur många gånger som helst.

Lösningen blev att omge anropningen av funktionen **meny** och dess efterföljande if-satser med en while-loop som endast blev falsk och avslutades om användaren valde via menyn att faktiskt avsluta.

5 Buggar

När man programmerar ett spel så är det lätt att missa felaktiga alternativ som användaren kan tänkas utföra och därför svårt att skapa "skyddsnät" som verkligen skyddar mot alla tänkbara fel och buggar. De buggar som har dykt upp och åtgärdats under utvecklingen av detta spel är följande:

- Vad händer om användaren svarar med en ogiltig input i menyn?

Om användaren svarar med en annan typ av tecken än int, eller om svaret inte är 1, 2, 3, eller 4 så anropas cin.clear som upphäver feltillståndet som skulle orsakat en evighetsloop och cin.ignore som ignorerar en sträng på upp till 1000 tecken och ett ENTER. Om användaren däremot svarar 1- 4 följt av ett komma, en punkt eller något annat tecken och därefter fler tecken så utförs trots detta det kommando som den första siffran står för. Detta scenario är alltså inte åtgärdat.

- Vad händer om användaren svarar med en ogiltig input när denne skall uppge koordinat?

Om användaren svarar med fler än 2 tecken så tolkas det som en ogiltig input och förhindras med en if-sats som anropar `cin.clear` (se ovan) och `cin.ignore` (se ovan).

Om användaren skriver med stor eller liten bokstav och om siffran eller bokstaven kommer först spelar ingen roll. Alla dessa alternativ kontrolleras med en if-sats och är alla giltiga inputs.

- Vad händer om användaren svarar med en ogiltig input i avsluta-sekvensen?

Om användaren svarar med en annan typ av tecken än `int`, eller om svaret inte är 1 eller 2 så anropas `cin.clear` (se ovan) och `cin.ignore` (se ovan). Om användaren däremot svarar 1 eller 2 följt av ett komma, en punkt eller något annat tecken och därefter fler tecken så utförs trots detta det kommando som den första siffran står för. Detta scenario är alltså inte åtgärdat.

- Vad händer om användaren skjuter på samma ruta som denne redan skjutit på?

Att användaren skulle kunna skjuta på samma ruta som tidigare förhindras av att `int`-arrayen inte bara kan ha värdena 0 och 1 för vatten och dolt skepp utan även 3 och 4 för missat och träffat skepp. Om elementet som skottet skjutit på har värdet 3 eller 4 så får användaren veta att denne redan skjutit där och får ett nytt försök på sig.

6 Reflektioner







Detta spel har inte varit lika svårt som det förra (Gissa talet) att skapa men det har ändå tagit sin tid. Precis som med det förra spelet har man lärt sig mycket under tidens gång, och precis som du säger så skulle man inte ha gjort alls likadant om man hade tagit sig an uppgiften på nytt. Den stora utmaningen tycker jag dock har varit att skriva ihop rapporten. Detta på grund av att det har varit mycket svårare att behålla den röda tråden i texten i och med att spelet har hoppat så mycket fram och tillbaka pga. alla funktioner.

På tal om funktioner så skulle jag vilja säga att det är underbart när man förstår sig på hur de fungerar och kan använda sig av dem och på så sätt slipper upprepa den text eller de funktioner som används flera gånger. Jag vill även säga att jag absolut föredrar att skriva funktionsdeklarationen överst i koden och funktionsdefinitionerna efter **main()**. Detta gör koden mycket mer överskådlig. Att jag valde att använda mig utav så många funktioner (6 st.) var dels för att alla dessa innehöll funktioner som skulle upprepas många gånger i spelet, och dels för att göra **main()** ännu mer lättöverskådligt.

Tvådimensionella arrayer får mig att tänka på en tabell där den ena siffran i arrayen står för rader och den andra för kolumner, men enligt Skansholm (2011) handlar det egentligen om element i element. Detta stämmer ju ändå ihop med en tabell då varje rad innehåller flera kolumner, alltså flera rutor. Att jag valde att använda 2 st. arrayer var för att jag ville ha 2 olika typer av värden, dels string, dels `int`. Jag har dock nu i efterhand förstått att om jag använt mig utav en struct istället så hade det räckt med en array, men jag är inte så insatt i detta än att jag tänker argumentera för det. Det får mig dock att tänka på att den andra inlämningsuppgiften nog borde ha utförts parallellt med denna. Det hade nog gett mig fler möjligheter till att utföra denna på ett så bra sätt som möjligt.

Precis som med mina 2 tvådimensionella arrayer som jag kunde gjort annorlunda, så borde jag även kunna städa bort många utav mina if-satser. Jag har använt mig av förskräckligt många tycker jag, men huvudsaken är väl dock att programmet fungerar, och fördelen är väl att det ger mig större kontroll över vad programmet faktiskt gör i olika scenarion. Problemet är ju naturligtvis dock risken för den mänskliga faktorn, som jag fått lida för ett par gånger. Förvånansvärt få dock under uppkomsten av detta spel.

En annan sak som jag kunde ha ändrat på och därmed sparat på antalet rader (även om det inte var målet med denna uppgift) var att jag i min if-sats för träff eller bom i funktionen **skott** skrev responsen. Detta kunde jag ju istället ha lagt i **main()** och med andra ord bara ha behövt skriva en gång för vardera träff och bom, men det är lätt att vara efterklok.

Spelplanen var nog det första jag satte igång med och också det som har bråkat mest. Jag har haft stora problem när jag försökt få den att se snygg ut och jag har fortfarande inte fått helt ordning på den. Största anledningen till det är väl dock att jag använder mig av Dev C++ på en Windows-dator hemma och av Unix i skolan. Överföringen av koden mellan dessa program är tyvärr inte helt smärfri. De tecken som jag har haft för avsikt att använda är i alla fall följande: 218 = , 191 = , 195 = , 180 = , 192 =  och 217 = . Siffrorna är deras värden i en teckentabell och de kallas för ALT-koder.

Avslutningsvis vill jag bara ta upp samma sak som jag skrev i min förra rapport. Det är väldigt roligt att se när programmets funktioner växer fram och det är en stor njutning när man får dem att fungera. Jag hoppas och tror verkligen att jag har valt rätt utbildning. Det känns så i alla fall.

Referenser

Skansholm, J. (2011) *C++ direkt*. Lund: Studentlitteratur AB.

```
# include <iostream>
# include <iomanip> //Detta bibliotek använder jag för att kunna göra
indrag i texten.

using namespace std;

// - Globala variabler -

int array[4][4]; //Dold spelplan
string plan[4][4]; //Synlig spelplan

// - Funktionsdeklarationer -

int meny(); //Min meny-funktion.

void regler(); //Min regel-funktion.

void nolla(); //Denna funktion nollställer alla värden på min spelplan.

void slumpa(); //Denna funktion slumpar fram och placerar ut skeppen.

void spelplan(); //Denna funktion ritar ut spelplanen.

int skott(); //Denna funktion kontrollerar skottet.

main()
{
    int val; //Här sparas valet från menyfunktionen.

    int highscore=17;

    int antalskott=0; //Skotträknare.

    int traff=0; //Träffräknare.

    int avslutasvar; //Här sparas valet om man verkligen vill avsluta.

    bool programloop=true;

    bool spelloop=true;

    bool avslutaloop=true;
```

```
cout <<endl <<endl <<setw(64) <<"- - * Välkommen till spelet \"Sänka
skepp\"! * - -" <<endl <<endl;

cout
<<"
_____ " <<endl;

while(programloop) //Loopar mitt program tills man väljer att avsluta.
{
    val=meny(); //Denna kör min meny och sparar sedan svaret i variabeln
val.

    if(val==1)

        regler(); //Visar reglerna.

else if(val==2)
{
    antalskott=0; //Skotträknaren nollställs.

    traff=0; //Antal träffar nollställs.

    nolla(); //Tömmer spelplanen.

    slumpa(); //Slumpar ut skepp.

    spelloop=true;

    while(spelloop) //Loopar spelomgången tills man hittat alla skepp.
    {

        spelplan(); //Ritar ut spelplanen.

        if (skott()==1) //Först körs min skottfunktion och sedan
kontrolleras det om man fått träff.

            traff++; //Om träff så ökar träffräknaren med 1.

        antalskott++; //Skotträknaren ökar med 1.

        if(traff>0)
```

```
{  
    if(traff==1)  
        cout <<"Det finns bara ett skepp kvar. Kom igen nu!" <<endl;  
  
    else if(traff==2)  
    {  
        spelplan(); //Ritar ut spelplanen.  
  
        cout <<endl <<"Det krävdes " <<antalskott <<" skott, och nu  
har du äntligen sänkt bägge";  
  
        cout <<" skeppen. Du kan se din " <<endl <<"highscore om du  
trycker 3 i menyn. Spelet är slut.";  
  
        cout <<endl <<endl;  
  
        if(antalskott<highscore)  
        {  
            if(highscore!=17) //Om highscoren har ändrats tidigare.  
            {  
                cout <<"Grattis, du har fått ett nytt highscore! Du kan  
se ditt nya highscore om du";  
  
                cout <<endl <<"trycker 3 i menyn." <<endl <<endl;  
            }  
  
            highscore=antalskott;  
        }  
  
        spelloop=false;  
    }  
}  
}  
  
else if(val==3) //Visar highscoren.
```

```
{

    if(highscore==17)

        cout <<endl <<"Du har inte spelat än och kan därför inte se någon
highscore." <<endl <<endl;

    else

        cout <<endl <<"Din highscore är " <<highscore <<" antal skott."
<<endl <<endl;

}

else if(val==4)

{

    cout <<endl <<"Är du verkligen säker på att du vill avsluta? Din
highscore kommer inte sparas " <<endl;

    cout <<"till nästa gång. Tryck valfri siffra (1=Ja, 2=Nej): " <<endl
<<endl;

    avslutaloop=true;

    while(avslutaloop) //En loop används för att kontrollera ev felaktig
input.

    {

        if((cin >>avslutasvar)&&(avslutasvar==1||avslutasvar==2)) //Först
skrivs Cin in, sedan kontrolleras om det är

        {

            //rätt typ (rätt typ gör "cin>>val" till sann) och till

            if(avslutasvar==1)
                //sist kontrolleras om det är ett giltigt värde.

            {

                cout <<endl <<"Tack för att du spelat!" <<endl <<endl;

                avslutaloop=false;

                programloop=false;

            }

        }

    }

}
```

```
        else //Om man inte är säker på att avsluta loopas man om till
menyn.

            avslutaloop=false;

        }

        else

        {

            cout <<endl <<"Ogiltigt val, var god försök igen." <<endl
<<endl;

            cin.clear();

            cin.ignore(1000, '\n'); //Ignorerar upp till 1000 tecken och
användarens ENTER

        }

    }

}

}

}

int meny() //Definition av menyfunktion.

{

    int val; //Här sparas valet som görs.

    bool loop=true;

    while(loop) //En loop används för att kontrollera ev felaktig input.

    {

        cout <<endl <<setw(60) <<"Vad vill du göra? Tryck valfri siffra: "
<<endl;

        cout <<setw(69) <<"1. Läs reglerna      2. Spela      3. Se HighScore
4. Avsluta" <<endl <<endl;

        if((cin >>val)&&(val>=1)&&(val<=4)) //Först skrivs Cin in, sedan
kontrolleras om det är
```

```
{ //rätt typ (rätt typ gör
"cin>>val" till sann) och till

    loop=false; //sist kontrolleras
om det är ett giltigt värde.

    return val;

}

else

{

    cout <<endl <<"Ogiltigt val, var god försök igen." <<endl <<endl;

    cin.clear();

    cin.ignore(1000, '\n'); //Ignorerar upp till 1000 tecken och
användarens ENTER

}

}

}

void regler() //Definition av regel-funktion.

{

    cout <<endl <<endl <<setw(41) <<"Regler" <<endl <<endl;

    cout <<"Spelet går ut på att datorn slumpar ut 2 skepp på en spelplan som
är 4*4 rutor" <<endl;

    cout <<"stor. Din uppgift blir sedan att försöka hitta och sänka skeppen
genom att" <<endl;

    cout <<"skjuta på valfri koordinat. Dina skott räknas och när du har
hittat och sänkt" <<endl;

    cout <<"bägge skeppen, som vardera är en ruta stort, så har du vunnit och
dina antal" <<endl;

    cout <<"skott skrivs ut." <<endl;

    cout <<"Du kan spela hur många gånger du vill och ditt highscore sparas i
highscore-" <<endl;

    cout <<"listan, men stänger du av programmet så nollställs tyvärr din
highscore." <<endl <<endl;

}
```

```
void nolla() //Denna funktion nollställer alla värden på min spelplan.
```

```
{  
    for(int y=0; y<4; y++) //För varje rad...  
    {  
        for(int x=0; x<4; x++) //...kontrolleras varje kolumn...  
        {  
            array[y][x]=0; //...och nollställer arrayernas värden.  
            plan[y][x]=" ";  
        }  
    }  
}
```

```
void slumpa() //Denna funktion slumpar fram och placerar ut skeppen.
```

```
{  
    bool loop=true;  
    int skepp1=0;  
    int skepp2=0;  
  
    while(loop) //Loop för att undvika att bägge skeppen placeras på samma  
ruta.  
    {  
        srand(time(0)); //Randomfunktionen nollställs.  
        skepp1=rand()%16+1; //Ger skepp1 ett slumpat värde mellan 1 och 16.  
        skepp2=rand()%16+1; //Ger skepp2 ett slumpat värde mellan 1 och 16.  
  
        if(skepp1!=skepp2)  
            loop=false;  
    }  
}
```



```
//Beroende på skeppens tilldelade värde förändras arrayen enligt nedan.
```

```
if(skepp1==1)
```

```
    array[0][0]=1;
```

```
else if(skepp1==2)
```

```
    array[0][1]=1;
```

```
else if(skepp1==3)
```

```
    array[0][2]=1;
```

```
else if(skepp1==4)
```

```
    array[0][3]=1;
```

```
else if(skepp1==5)
```

```
    array[1][0]=1;
```

```
else if(skepp1==6)
```

```
    array[1][1]=1;
```

```
else if(skepp1==7)
```

```
    array[1][2]=1;
```

```
else if(skepp1==8)
```

```
    array[1][3]=1;
```

```
else if(skepp1==9)
```

```
    array[2][0]=1;
```

```
else if(skepp1==10)
```

```
    array[2][1]=1;
```

```
else if(skepp1==11)
```

```
    array[2][2]=1;
```

```
else if(skepp1==12)
```

```
    array[2][3]=1;
```

```
else if(skepp1==13)
```

```
    array[3][0]=1;
```

```
else if(skepp1==14)
```

```
    array[3][1]=1;
```

```
else if(skepp1==15)
```

```
    array[3][2]=1;
```

```
else if(skepp1==16)
```

```
    array[3][3]=1;
```

```
if(skepp2==1)
```

```
    array[0][0]=1;
```

```
else if(skepp2==2)
```

```
    array[0][1]=1;
```

```
else if(skepp2==3)
```

```
    array[0][2]=1;
```

```
else if(skepp2==4)
    array[0][3]=1;

else if(skepp2==5)
    array[1][0]=1;

else if(skepp2==6)
    array[1][1]=1;

else if(skepp2==7)
    array[1][2]=1;

else if(skepp2==8)
    array[1][3]=1;

else if(skepp2==9)
    array[2][0]=1;

else if(skepp2==10)
    array[2][1]=1;

else if(skepp2==11)
    array[2][2]=1;

else if(skepp2==12)
    array[2][3]=1;

else if(skepp2==13)
    array[3][0]=1;
```

```
else if(skepp2==14)
    array[3][1]=1;

else if(skepp2==15)
    array[3][2]=1;

else if(skepp2==16)
    array[3][3]=1;
}

void spelplan() //Denna funktion ritar ut spelplanen.
{

    for(int y=0; y<4; y++) //För varje rad...
    {
        for(int x=0; x<4; x++) //...kontrolleras varje kolumn...
        {
            if(array[y][x]==0) //...och ritar ut korrekt tecken i spelplanen.
                plan[y][x]=" ";

            else if(array[y][x]==1)
                plan[y][x]=" ";

            else if(array[y][x]==2)
                plan[y][x]="X";

            else if(array[y][x]==3)
                plan[y][x]="O";
        }
    }
```

```

}

//För att arrayen ska funka i windows
char vohorn=218; //Vänster övre hörn på spelplanen
char hohorn=191; //Höger övre hörn
char vsida=195; //Vänster sida
char hsida=180; //Höger sida
char vnhorn=192; //Vänster nedre hörn
char hnhorn=217; //Höger nedre hörn

/*//För att arrayen ska funka i Unix
string vohorn="+"; //Vänster övre hörn på spelplanen
string hohorn="+"; //Höger övre hörn
char vsida=21; //Vänster sida
char hsida=22; //Höger sida
string vnhorn="+"; //Vänster nedre hörn
string hnhorn="+"; //Höger nedre hörn*/

cout <<endl <<vohorn <<"-----" <<hohorn <<endl;
cout <<"|   | A | B | C | D |" <<endl;
cout <<vsida <<"-----" <<hsida <<endl;
cout <<"| 1 | " <<plan[0][0] <<" | " <<plan[0][1] <<" | " <<plan[0][2]
<<" | " <<plan[0][3] <<" |" <<endl;
cout <<vsida <<"-----" <<hsida <<endl;
cout <<"| 2 | " <<plan[1][0] <<" | " <<plan[1][1] <<" | " <<plan[1][2]
<<" | " <<plan[1][3] <<" |" <<endl;
cout <<vsida <<"-----" <<hsida <<endl;
cout <<"| 3 | " <<plan[2][0] <<" | " <<plan[2][1] <<" | " <<plan[2][2]
<<" | " <<plan[2][3] <<" |" <<endl;
cout <<vsida <<"-----" <<hsida <<endl;
cout <<"| 4 | " <<plan[3][0] <<" | " <<plan[3][1] <<" | " <<plan[3][2]
<<" | " <<plan[3][3] <<" |" <<endl;

```

```
    cout <<vnhorn <<"-----" <<hnhorn <<endl <<endl;
}

int skott() //Denna funktion kontrollerar skottet.
{
    string skott;

    bool tmloop=true; //Kontroll av träff eller miss.

    while(tmloop) //En loop används för att kontrollera ev felaktig input.
    {
        cout <<"Ange skottets koordinat: ";

        cin    >>skott;

        if(!skott[2]) //Kontrollerar om det är rätt antal tecken. Tecken1=0,
        Tecken2=1 o ENTER=2.
        {
            if(skott=="a1"||skott=="A1"||skott=="1a"||skott=="1A")
            {
                if(array[0][0]==0)
                {
                    array[0][0]=2;

                    cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;

                    return 0;
                }

                else if(array[0][0]==1)
                {
                    array[0][0]=3;

                    cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;

                    return 1;
                }
            }
        }
    }
}
```

```
    }

    else

        cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;

    }

    if(skott=="b1" || skott=="B1" || skott=="1b" || skott=="1B")
    {
        if(array[0][1]==0)
        {
            array[0][1]=2;

            cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;

            return 0;

        }

        else if(array[0][1]==1)
        {
            array[0][1]=3;

            cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;

            return 1;

        }

        else

            cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;

    }

    if(skott=="c1" || skott=="C1" || skott=="1c" || skott=="1C")
    {

        if(array[0][2]==0)
```

```
{  
    array[0][2]=2;  
    cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;  
    return 0;  
}  
  
else if(array[0][2]==1)  
{  
    array[0][2]=3;  
    cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;  
    return 1;  
}  
  
else  
    cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl  
<<endl;  
}  
  
if(skott=="d1" || skott=="D1" || skott=="1d" || skott=="1D")  
{  
    if(array[0][3]==0)  
    {  
        array[0][3]=2;  
        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;  
        return 0;  
    }  
  
    else if(array[0][3]==1)  
    {  
        array[0][3]=3;  
        cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;  
    }  
}
```



```
        return 1;
    }

    else

        cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;
    }

    if(skott=="a2" || skott=="A2" || skott=="2a" || skott=="2A")
    {
        if(array[1][0]==0)
        {
            array[1][0]=2;

            cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;

            return 0;
        }

        else if(array[1][0]==1)
        {
            array[1][0]=3;

            cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;

            return 1;
        }

        else

            cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;
    }

    if(skott=="b2" || skott=="B2" || skott=="2b" || skott=="2B")
    {
```

```
    if(array[1][1]==0)
    {
        array[1][1]=2;

        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;

        return 0;
    }

    else if(array[1][1]==1)
    {
        array[1][1]=3;

        cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;

        return 1;
    }

    else
        cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;
}

if(skott=="c2" || skott=="C2" || skott=="2c" || skott=="2C")
{
    if(array[1][2]==0)
    {
        array[1][2]=2;

        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;

        return 0;
    }

    else if(array[1][2]==1)
    {
        array[1][2]=3;
```

```
        cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;

        return 1;

    }

    else

        cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;

    }

    if(skott=="d2" || skott=="D2" || skott=="2d" || skott=="2D")

    {

        if(array[1][3]==0)

        {

            array[1][3]=2;

            cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;

            return 0;

        }

        else if(array[1][3]==1)

        {

            array[1][3]=3;

            cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;

            return 1;

        }

        else

            cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;

    }

    if(skott=="a3" || skott=="A3" || skott=="3a" || skott=="3A")
```

```
{  
  
    if(array[2][0]==0)  
    {  
  
        array[2][0]=2;  
  
        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;  
  
        return 0;  
    }  
  
    else if(array[2][0]==1)  
    {  
  
        array[2][0]=3;  
  
        cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;  
  
        return 1;  
    }  
  
    else  
  
        cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl  
<<endl;  
}  
  
  
if(skott=="b3" || skott=="B3" || skott=="3b" || skott=="3B")  
{  
  
    if(array[2][1]==0)  
    {  
  
        array[2][1]=2;  
  
        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;  
  
        return 0;  
    }  
  
    else if(array[2][1]==1)  
    {  
  

```

```
        array[2][1]=3;

        cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;

        return 1;

    }

    else

        cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;

    }

    if(skott=="c3" || skott=="C3" || skott=="3c" || skott=="3C")

    {

        if(array[2][2]==0)

        {

            array[2][2]=2;

            cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;

            return 0;

        }

        else if(array[2][2]==1)

        {

            array[2][2]=3;

            cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;

            return 1;

        }

        else

            cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;

    }
```

```
if(skott=="d3" || skott=="D3" || skott=="3d" || skott=="3D")
{
    if(array[2][3]==0)
    {
        array[2][3]=2;
        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;
        return 0;
    }

    else if(array[2][3]==1)
    {
        array[2][3]=3;
        cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;
        return 1;
    }

    else
        cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;
}

if(skott=="a4" || skott=="A4" || skott=="4a" || skott=="4A")
{
    if(array[3][0]==0)
    {
        array[3][0]=2;
        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;
        return 0;
    }

    else if(array[3][0]==1)
```

```
{  
    array[3][0]=3;  
    cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;  
    return 1;  
}  
  
else  
    cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl  
<<endl;  
}  
  
if(skott=="b4" || skott=="B4" || skott=="4b" || skott=="4B")  
{  
    if(array[3][1]==0)  
    {  
        array[3][1]=2;  
        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;  
        return 0;  
    }  
  
    else if(array[3][1]==1)  
    {  
        array[3][1]=3;  
        cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;  
        return 1;  
    }  
  
    else  
        cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl  
<<endl;  
}
```

```
if(skott=="c4" || skott=="C4" || skott=="4c" || skott=="4C")
{
    if(array[3][2]==0)
    {
        array[3][2]=2;
        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;
        return 0;
    }

    else if(array[3][2]==1)
    {
        array[3][2]=3;
        cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;
        return 1;
    }

    else
        cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;
}

if(skott=="d4" || skott=="D4" || skott=="4d" || skott=="4D")
{
    if(array[3][3]==0)
    {
        array[3][3]=2;
        cout <<endl <<"Plask!!! Tyvärr, du missade!" <<endl;
        return 0;
    }
}
```



```
        else if(array[3][3]==1)
        {
            array[3][3]=3;

            cout <<endl <<"Kaboom!!! Grattis, du träffade!" <<endl;

            return 1;
        }

        else
            cout <<endl <<"Du har ju redan skjutit här. Försök igen!" <<endl
<<endl;
    }
}

cout <<endl <<"Ogiltigt val, var god försök igen." <<endl <<endl;

cin.clear();

cin.ignore(1000, '\n'); //Ignorerar upp till 1000 tecken och
användarens ENTER.

}

}
```