SPELMOTOR TXT

Program för att köra speläventyr från textfiler.

Eget projekt för kursen
Strukturerad Programmering C

Johan Kämpe
UIS16

BILD

2016-10-xx

Sammanfattning

Programmet SPELMOTOR TXT används för att spela textäventyr som är lagrade i externa textfiler. Programmet har stöd för val av spel, omstart av spel, och diverse extra-funktioner, så som färgbyte av spelets text.

Programmet navigerar i spelen med ”identifikationsnummer” som skrivs i textfilerna.

En egen inkluderingsfil används med diverse funktioner för text- och textfilshantering. I programmets huvudfil finns tre stycken funktioner samt main-funktion.

Innehåll

1 Inledning

1.1 Syfte

Syftet med projektet är att skapa ett eget program för inlämning i kursen *Strukturerad Programmering C*. Projektet ska ses som träning och en sorts utmaning.

Hädanefter kommer projektet ibland att hänvisas till som ”programmet”.

Syftet med programmet *Spelmotor TXT* är att kunna spela olika textäventyr som är lagrade i externa textfiler. Programmet ska automatiskt leta upp de val som spelaren kan göra i de olika spelmomenten, och sedan gå till nästa steg i textäventyret beroende av valinmatning.

I de externa textfilerna ska det finnas möjlighet välja när spelet ska avslutas, det ska finnas funktionalitet för radbrytning i texten. Det ska också finnas möjlighet att utföra enklare kommandon genom att använda sifferkoder i textfilen, till exempel färgbyte av kommandofönstrets text.

Inspirationen för att starta projektet var de gruppuppgifter som utförts under kursen *Strukturerad Programmering C*, där grupper om cirka fem personer samarbetade för att skapa textäventyr i programspråket C. I dessa övningar skrevs all spel-text direkt i källkoden.

1.2 Noteringar

Rapporten är tänkt att läsas av personer med en grundläggande förståelse inom programspråket C. Ingen ordlista finns i rapporten.

1.3 Länkar

GitHub
https://github.com/GoblinDynamiteer/spelmotor_txt

2 Genomförande och resultat

2.1 Använd programvara och litteratur

Programvaror

- Notepad++, texteditor
- Microsoft Word 2016
- Microsoft Notepad för Windows 10, texteditor
- Adobe Photoshop CC 2015, bildredigering
- GCC, GNU Compiler Collection, kompilator
- GIT, versionshanteringverktyg

Litteratur

- *C från början*, Jan Skansholm, 2016. – Boken kommer hänvisas till i texten som *”boken”*

2.2 Avgränsningar och krav

Enligt specifikation från utbildningen ska följande krav uppfyllas:

- Programmets källkod ska ha en omfattning om minst 50 rader kod, och vara fördelat på minst två filer.
- Om redan färdig kod används från Internet, ska detta markeras i källkoden och kodens upphovsrättsman ska krediteras.
- Enklare dokumentation ska skapas, som beskriver projektets syfte och programmets funktionalitet.
- Inlämning ska ske på Moodle. Källkod, dokumentation och körbar exe-fil ska lämnas in.
- Deadline för projektet är den 30 oktober 2016.

Andra/egna avgränsningar:

- Programmet ska skrivas i programspråket C.
- Programmet behöver enbart kunna användas med korrekt skrivna textfiler. Ingen kontroll ska utföras för att bekräfta att textfilen innehåller ett korrekt skrivet text-äventyr.

2.3 Metod

2.3.1 Skrivning av källkod och kompilering

Programmets källkod skrevs uteslutande i texteditorn Notepad++. Notepad++ har färgmallar för olika programspråk, vilket underlättar kodskrivandet och felsökning.

Notepad++ saknar inbyggt stöd för kompilering av kod, för att underlätta programmerandet används texteditorns ”run”-funktion med följande inmatning:

cmd -cmd /K d: & cd "$(CURRENT_DIRECTORY)" & gcc "$(FULL_CURRENT_PATH)" strings_text_v1.c -o "$(CURRENT_DIRECTORY)\$(NAME_PART)".exe & chcp 1252 & "$(CURRENT_DIRECTORY)\$(NAME_PART)".exe

Med detta kommando ges möjligheten att direkt kompilera den aktuella c-filen, tillsammans med strings_text_v1.c, som innehåller funktioner för text- och filhantering. Efter kompilering startas exe-filen automatiskt.

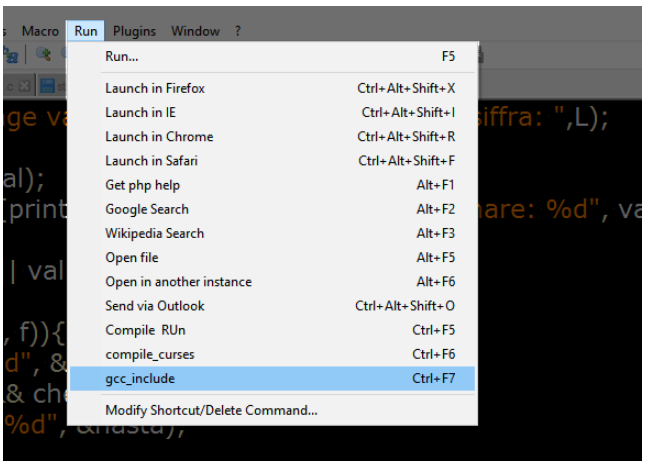

Bild 1 Run-menyn i programmet Notepad++

2.3.2 Utformning av textfiler för hållande av textäventyrspel

Innan arbetet med programmets källkod kunde påbörjas, skapades en första version av ett textäventyrsspel. Detta för att kunna anpassa programmet efter textfilens utformning.

Flera olika utföranden av textfilen testades parallellt med kodningen, innan det nuvarande utförandet fastställdes. Hur textfilerna med spel behöver utformas beskrivs senare i rapporten.

Windows-verktyget Notepad användes i första hand för att skriva textfilen.

2.4 Programmets funktion

Vid programmets start undersöks först om en textfil har givits som ett argument. Detta görs genom att skriva *spelmotortxt.exe textfil.txt* i kommandoprompten, där *spelmotortxt.exe* är den kompilerade programfilens namn och *textfil.txt* är den textfil som vill användas.

Om inget argument har givits uppmanas användaren att skriva in namnet på den textfil denne vill använda att spela med. Om denna fil inte kan hittas kommer programmet läsa in den fördefinierade textfilen *spel_default.txt*, som innehåller ”Björnspelet”. Om denna fil saknas eller inte kan läsas in avslutas programmet.

Vid användarinmatning av textfil kan användaren välja att avsluta programmet med kommandot EOF (End Of File) vilket skrivs in med tangentkombinationen *CTRL+Z* i en Windows-miljö. Om *spel_default.txt* vill spelas kan användaren välja att inte skriva in något filnamn, eller skriva ett medvetet inkorrekt filnamn.

Om programmet lyckas läsa in en korrekt utformad textfil kommer dess innehållande spel att startas.

Textäventyrspelen fungerar på följande sätt:

- Initialt visas spelets titel, och användaren uppmanas att trycka på en valfri tangent för att starta spelet.
- En text visas på skärmen som beskriver spelarens situation, två eller fler val listas upp för spelaren, för att symbolisera hur denne vill agera i den givna situationen.
- Varje val som listas representeras av en siffra, från 1 och uppåt. Spelaren uppmanas att skriva in en siffra för att välja hur spelet ska fortskrida.
- Om användaren slår in en siffra som inte har ett korresponderande val, uppmanar programmet användaren att försöka igen, tills denne skriver in en korrekt siffra.
- Spelet går sedan vidare till en annan situation/text, beroende på vilket val som har utförts. Nya val listas för användaren.
- Denna process repeteras tills det att spelaren har antingen vunnit eller förlorat äventyret, det kan finnas flera vinst- och förlustsituationer.
- När äventyret är färdigspelat ges spelaren möjlighet att spela om textäventyret från början genom att mata in tecknet *J* eller *j* på tangentbordet. Om något annat skrivs in avslutas programmet.

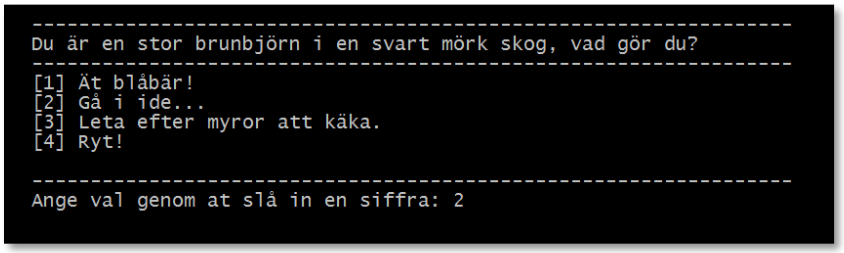

Bild 2 Textspel med val och användarinmatning, från spel_default.txt

2.5 Textfilernas utförande

För att programmet ska fungera korrekt behöver textfilerna som innehåller textäventyrspelen vara utformade på ett visst sätt. Programmet innehåller inga kontroller för att bestämma om textfilen är korrekt.

Nedan visas ett exempel, från *spel_default.txt*. Här visas de första raderna i textfilen.

Björnspelet (Default-äventyr)

!INITIAL TEXT
T1000|0000|Du är en stor brunbjörn i en svart mörk skog, vad gör du?
V1001|2100|Ät blåbär!
V1002|2200|Gå i ide...
V1003|2300|Leta efter myror att käka.
V1004|2400|Ryt!

Den första raden, här *Björnspelet (Default-äventyr)*, i textfilen ska vara namnet på spelet.

Den rad som börjar med *T1000* innehåller text som kommer visas i början av spelet. Bokstaven *T* innebär att det är en text som ska skrivas ut, och *1000* är dess identifikationsnummer. Just T1000 är den textrad om alltid visas först när spelet startas, förutom spelets titel.

Under T1000 listas de val som tillhör texten. Val ska börja med bokstaven *V* och sedan ha samma identifikationsnummer som texten de tillhör, fast ökat med 1 för varje val. Alltså innebär texten på rad *V1002* val nummer två för text T1000.

Efter de olika valens identifikationsnummer visas den text som ska visas ifall det valet väljs av spelaren. Så om spelaren väljer val 3, *Leta efter myror att käka*, så kommer programmet leta upp den rad i texten som börjar med *T2300*, och lista de eventuella val som tillhör denna text.

Exempel från *spel_default.txt*, om användaren skulle välja valet 3 ovan.

!MYROR
T2300|0000|Myror är inget vidare, smakar surt.|Men fungerar i krig.|Var vill du leta efter myror?
V2301|3100|Leta under det stora stenblocket du ser framför dig.
V2302|3200|I myrstacken, duh!|Det borde ju finnas en myriad av myror där.

I exemplen visas *!INITIAL TEXT* och *!MYROR*, dessa är kommentarer och kommer inte användas av programmet. Programmet behandlar enbart rader som börjar med ett T eller ett V, och textfilens första rad med spelets titel.

Tecknet | i texterna skrivs ut som radbrytningar. Detta kan användas i både text och för val. Dock inte för spelets titel på den första raden. Flera *(ex: |||)* tecken kan användas för att få flera radbrytningar efter varandra.

Den sifferkod som visas efter texternas identifikationsnummer används för att ge kommandon till programmet. För *0000* händer ingenting förutom att texten visas.

För sifferkoden 9999 som visas nedan i ett exempel från *spel_default.txt*, kommer äventyret att avslutas.

!GÅ I IDE – VINST
T2200|9999|Du går i ide: ZzzzzZzzzzz.|Bra björn, du vann!

Nedan listas de sifferkoder som är tillgängliga och vad de har för funktion i programmet:

- 0000 – Texten skrivs ut, ingen annan funktion.
- 9993 – Texten skrivs ut och byter färg till blå
- 9994 – Texten skrivs ut och byter färg till grön
- 9995 – Texten skrivs ut och byter färg till röd
- 9996 – Texten skrivs ut och byter färg till blå, spelet avslutas.
- 9997 – Texten skrivs ut och byter färg till grön, spelet avslutas.
- 9998 – Texten skrivs ut och byter färg till röd, spelet avslutas.
- 9999 – Texten skrivs ut och spelet avslutas.

2.6 Debug-läge

Programmet har stöd för ett enklare debug-läge. Läget aktiveras genom att initiera variabeln *_Bool debugMode = 1;*. För att avaktivera läget, sätts den till 0. Som standard är variabeln initierad till 0.

Variabeln är initierad i källkodens början, direkt efter funktionsdeklarationerna, utanför main-funktionen. Detta för att variabeln ska vara synlig för samtliga funktioner i filen.

Om debug-läget är aktiverat skrivs extra information ut till användaren, som kan användas för felsökning. Till exempel skriver vissa av funktionerna ut sina returvärden innan de ges. Debug-texter omges av hakparenteser.

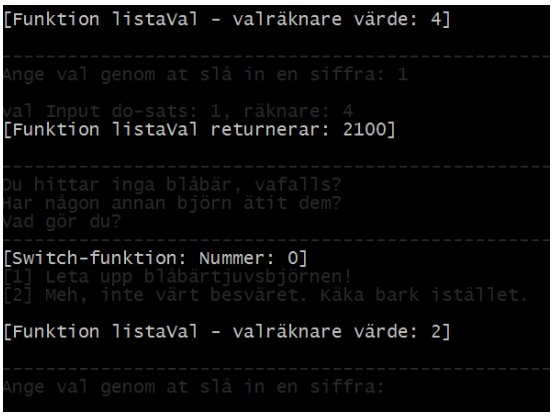

Bild 3 Extra debug-information från funktioner

2.7 Programmets kod

2.7.1 Macron

#define N 1000	Arraylängd för char-variabler, och för argument till vissa funktioner.
#define TEXTFIL "spel_default.txt"	Namn på den fördefinierade textfil som ska laddas, om inget argument ges till programmet vid start.
#define L "\n----------\n"	För utskrift av ”linjer” med printf. (här nedkortad längd).
#define LT 11	Antal tecken i textfilen som ska hoppas över på början av raden för att skriva ut text.
#define TEXTHASTIGHET 3	Väntetid i millisekunder mellan att varje enskilt tecken skrivs ut. Endast för visuell effekt, sätt till 0 för inte använda.

2.7.2 Variabler

FILE * textfil	Filvariabel för textäventyr i textform.
char s[N]	Håller rader från textfil. En rad i taget, eller N st. tecken.
char filnamn[N]	Inmatning av filnamn från användaren, om både argument saknas vid start av program, och spel_default.txt saknas.
int idNum	För uppletning av textrad att skriva ut från textfilen. Sätts till 1000 för varje gång ett spel startas, för att skriva ut spelets första text som ska ha just identifikationsnummer 1000.
int idCheck	Läser in den aktuella textradens identifikationsnummer, för att jämföra mot värdet i variabeln idNum.
int switchCheck	Läser in den aktuella textradens sifferkod, ex. 9999 för att avsluta spelet.
_Bool debugMode = 0	Variabel för att sätta debug-läge på eller av. Om den sätts till 1 skrivs diverse extra information ut från funktioner.
_Bool korrektVal = 1	Sätts till 0 så fort texten för uppmanande av användarinmatning visas första gången för varje val-serie. Om användaren skriver in ett inkorrekt val, så kommer en annan uppmanande text att visas.
int valRaknare = 1	Räknar hur många val som hittas för varje nytt valscenario. Används för att testa om spelaren slår in en korrekt valsiffra.
int nastaIdNum	Returneras från funktionen *listaVal* och sätts till variabeln *idNum* i main-funktionen, för att leta upp nästa textrad som ska skrivas ut.
char restart = ’j’	Variabel för omstart av spel, när spelet är avslutat ombeds användaren att mata in ett tecken. Om tecknet är ”j” eller ”J” startar spelet om från början.
int checkVal	BESKRIV

2.7.3 Inkluderade filer och bibliotek

Standardbibliotek

stdio.h

För diverse input/output-funktionalitet.

stdlib.h

För *system()*, som används för att köra systemkommandon. I detta program används följande systemkommandon:

- *cls* ”Clear Screen” - blankar skärmen.
- *pause* Pauserar programmet tills användaren trycker på en valfri tangent.
- *color* För färgbyte av kommandotolkens text och/eller bakgrund.
- *chcp* *”Change Code Page”*, används för att byta teckenkodning.

string.h

För *strlen()*, som returnerar teckenlängden på en sträng.

ctype.h

För funktionen *tolower()* som används för att kontrollera om användaren har skrivit in j/J om denne vill spela om ett avslutat spel. Inmatningen konverteras till en gemen och jämförs med ”j”.

windows.h

För funktionen *Sleep()*. Funktionen används i programmet för att få en kort fördröjning mellan varje teckenutskrift i funktionen *skrivUtText* som beskrivs i nästa kapitel.

Enligt Wikipedia innehåller *windows.h* egna inkluderingar av *string.h* och *ctype.h* och dessa hade eventuellt inte behövt inkluderats i källkodens text. Dock behålls inkluderingarna för garanterad funktionalitet.

Egna filer

strings_text_v1.c

Innehåller funktioner för text och textfil-hantering. Funktionerna beskrivs i källkoden och i nästa kapitel i rapporten. Funktionerna är lika de som finns i boken, fast med annan namngivning.

strings_text_v1.h

Innehåller deklarationer av funktioner i strings_text_v1.c, samt inkluderingar av standardbibliotek.

2.7.4 Funktioner

Funktioner som ligger i huvud-filen FILNAMN.C:

int listaVal(int a, FILE *f);
Funktionen skriver ut de val som finns för det aktuella scenariot i spelet. Som parametrar får funktionen textens identifikationsnummer som en int-variabel, samt den inlästa textfilen.

Funktionen kontrollerar att användaren matar in en korrekt siffra för val, och returnerar sedan det identifikationsnummer som tillhör nästa text som ska skrivas ut.

void skrivUtText(char *string, int n, _Bool linjer);
Funktionen skriver ut text till skärmen, ett tecken i taget. Mellan varje teckenutskrift sker en fördröjning på lika många millisekunder som macrot *TEXTHASTIGHET* är definierat till. Som parametrar får funktionen textsträngen som ska skrivas ut, en int-variabel med antal tecken som ska skrivas ut. _Bool-variabeln bestämmer om funktionen ska skriva ut ”rader” i början och slutet av textblocken.

_Bool textSwitchar(int s);
Funktionen används för att köra vissa kommandon som kan användas i spelet. Som t.ex. att avsluta spelet eller att byta färg på texten. Som parametrar får funktionen den sifferkod som finns efter den aktuella textradens identifikationsnummer. Funktionen returnerar en etta eller nolla beroende på om spelet ska avslutas eller inte.

Funktioner som finns i strings_text_v1.c:

Utförligare kommentering finns i källkodsfilen

_Bool removeNewLine(char[]);
Funktionen kontrollerar om det sista tecknet I en textsträng är ett nyradstecken, och raderar den om så är fallet. Denna funktion är identisk med bokens funktion *remove_nl* på sida 199.

void clearBuffer(void);
Tömmer textbufferten, denna funktion är identisk med bokens funktion *skip_line* på sida 190.

void clearBufferFil(FILE *f);
Tömmer textfilinläsningsbufferten, denna funktion är identisk med bokens funktion *fskip_line* på sida 226.

_Bool radInput(char[], int);
Skriver textinmatning från användare till char-variabel, använder sig av funktionerna *clearBuffer* och *removeNewLine.* Funktionen är identisk med bokens funktion *read_line* på sida 200.

_Bool textfilTillString(char[], int, FILE *f);
Funktion som läser in rader från textfil till char-variabel. Funktionen returnerar 0 när inläsning misslyckas eller är klar. Vid korrekt inläsning returneras 1. Funktionen är identisk med bokens funktion *fread_line* på sida 226

_Bool TTS(char *a, int n, FILE *f);
Kan användas i stället för funktionen *textfilTillString*. TTS anropar funktionen *textfilTillString* och returnerar dess värde.

2.7.4 Main-funktionen

I detta kapitel beskrivs grundläggande vad som händer i programmets kod, för ytterligare kommentering hänvisas läsaren till källkodsfilerna.

- Initialt deklareras och initieras variabler och funktioner som finns i main-filen. Med funktionen *system()* sätts kommandotolkens teckenkodning till 1252 och skärmens innehåll blankas.
  Sedan skrivs programmets namn ut med den egna funktionen *skrivUtText()*.
  Med bool-argumentet 1 för funktionen skrivs ”linjer” ut före och efter texten.

- Pekarvariabeln *textfil* sätts till det argument som givits till programmet vid start, med funktionen *fopen()* och argument **”r”** som innebär endast läsning av textfil. *textfil* kommer ge returvärdet **NULL** om textfilen inte kan öppnas. Detta nyttjas i tre **if**-satser, som triggar om en textfil inte kunnat öppnas. Om *textfil* är **NULL** kommer programmet först att be användaren om att skriva in namnet på den textfil som denne vill använda. Inmatning från användaren görs med den egna funktionen *radInput()* och sätts till variabeln *filnamn*.

  Om *textfil* inte kan öppna den användarinmatade filen i *filnamn*, så triggar nästa **if**-villkor där programmets standardfil för spel *spel_default.txt*, läses in. Om denna fil inte kan läsas avslutas programmet.

- En **while**-loop kapslar in resterande programkod i main-funktionen, villkoret för att loopen ska köra är om det gemena tecknet i variabeln *restart* är lika med ”j”. Variabeln initieras till ”j” innan **while**-loopen, för att den ska köra första gången. För att testa det gemena tecknet används funktionen *tolower().* Varje gång ett spel är färdigt uppmanas användaren att välja om denne vill spela om det aktuella spelet från början. Om användaren skriver in något annat än J eller j kommer programmet att avslutas.

- Variabeln *idNum* sätts till **1000**, *idNum* innehåller identifikationsnumret till den textrad som ska skrivas ut på skärmen. Just **1000** är spelets första text att skriva ut.

- Den egna funktionen *TTS()* anropas för att läsa in textfilens första rad (eller *N* st. tecken, det som sker först) till variabeln *s. N* är ett macro definierat till **1000.** Om raden i textfilen innehåller fler än 1000 tecken blir inte inläsningen korrekt. Sedan skrivs spelets titel ut med *printf()*, som ska vara textfilens första rad. Programmet pauserar med *system("pause")*, tills det att användaren trycker på en valfri tangent.

- En **while**-loop körs med villkoret ”*TTS(s, N, textfil)”.*
  TTS() ger ett sanningsvärde i retur så länge inläsning sker korrekt. När textfilen är slut kommer sanningsvärdet att bli 0. Detta innebär att för varje varv i **while**-loopen kommer en textrad i textfilen att behandlas, så länge den inte har fler än *N* st. tecken.

- Identifikationsnumret i textraden i variabeln *s* läses in till variabeln *idCheck*. Funktionen *sscanf()* används. *sscanf()* fungerar likt *scanf()*, men får input från en teckenström (*s*) istället för från tangentbordet. Då identifikationsnumret börjar på radens andra tecken, används *s+1* som argument. *s* pekar på strängens första värde, *s+1* på dess andra osv.

- En **if**-sats triggar när rätt rad att skriva ut hittas i textfilen. Villkoret är att värdet i *idCheck*, som sätts till ett nytt värde för varje rad som läses från textfilen, är lika med värdet i *idNum*. Samt att radens första tecken är ”T”. När denna rad hittas skrivs texten ut med funktionen *skrivUtText().*

- Den funna radens sifferkod läses in till variabeln *switchCheck*. Funktionen textSwitchar anropas med värdet i switchCheck som argument. I funktionen används en **switch**-sats för att utföra olika moment beroende på vad värdet i *switchCheck* är. Om spelet ska avslutas ger funktionen sanningsvärdet **0**, annars **1**. Då funktionen är relativt enkel kommer den inte beskrivas utförligare i denna rapport.

  Om värdet är **0** triggar en **if**-sats i main-funktionen, som skriver ut att spelet är avslutat, med funktionen *skrivUtText()*. Sedan används break för att bryta den inre **while**-loopen.
  När spelet är avslutat anropas *rewind()* för att ”spola tillbaka” textfilen, textbufferten töms med den egna funktionen *clearBuffer().* Därefter får användaren möjlighet att spela om det aktuella spelet genom att skriva in ett tecken, som sätts till variabeln *restart* med funktionen *getchar().*

- Om spelet inte ska avslutas anropas funktionen *listaVal()* med värdet i *idNum* som argument, samt textfil-variabeln *textfil*. Funktionen beskrivs i nästa kapitel. Funktionen listar de val användaren kan göra och ger i returvärde ett identifikationsnummer som skrivs till *idNum*, beroende på vilket val användaren gjorde.
  Textfilen spolas tillbaka med funktionen *rewind()*, och den inre while-loopen börjar om, för att leta upp nästa rad att skriva ut.

2.7.5 Funktionen listaVal()

BILAGA

Ändringslogg

Reflektioner diskussion

Onödigt söka hela textfilen, om långt spel tar det ev tid

Namngivning variabler