PROGRAMMERING A

C++

DEL 1 av 2.

Kompendium med lektionsanteckningar och övningsuppgifter

Stefan Sundin

Torsbergsgymnasiet Bollnäs

Innehållsförteckning

Inledning

Datorer styrs av program. Genom att byta program kan du få datorn att utföra olika saker. Den kan fungera som ordbehandlare eller kalkylator, visa film, spela upp musik eller fungera som hemstudio för inspelning av musik. Detta skiljer den från andra maskiner som är skapade för att enbart utföra vissa förutbestämda saker.

Datorns ”hjärna” är processorn eller **CPU**’n (Central Processing Unit) som styr och reglerar datorns alla övriga enheter. Processorn hämtar instruktioner och data från primärminnet (Random Access Memory – **RAM**).

För att ett program skall kunna köras måste det alltså först placeras i primärminnet. Detta minne kan man tänka sig som en följd av numrerade minnesfack. Vanligtvis består varje minnesfack av **8 bitar** (en **byte**), där varje bit kan anta två lägen 1 eller 0.
Ett program, som laddats in i primärminnet, består av ett antal sammanhängande minnesfack. Ett eller flera fack innehåller instruktioner. En instruktion talar om för datorn att den skall utföra en bestämd sak, t.ex. skriva ut HALLÅ till skärmen eller addera två heltal.
Processorn läser dessa instruktioner och utför dem i tur och ordning.

Programmering är ett sätt att få en maskin att utföra en uppgift. När du programmerar skriver du en mängd kod som, om allt går rätt, mynnar ut i ett program.

När vi skriver program i exempelvis C++ måste vi, när vi vill köra programmet, konvertera koden till maskinkod (ettor och nollor). Den här processen kallas att **kompilera**. Kompilering görs med hjälp av en **kompilator**. En kompilator är förstås också ett program, skrivet för att konvertera den skrivna koden (**källkoden**) till **maskinkod**. Varje kompilator är konstruerad för att översätta ett visst programspråk, så för att kompilera C++-kod behöver du en C++ -kompilator, vilken antingen kan vara ett fristående program eller vara integrerad i ett programutvecklingsverktyg som Visual Studio.Net eller Bloodshed Dev C++.

Kompilatorn läser programmet från den textfil (källkod) man skapat och kontrollerar först att programmet följer språkreglerna för C++.
Om fel upptäcks, avbryts kompileringen och programmeraren uppmanas att rätta till dessa. Sedan felen rättats får man göra ett nytt försök att kompilera programmet. Vanligtvis får man upprepa denna process att antal gånger innan kompilatorn anser att källkoden är felfri.
Kompilatorn går då vidare och översätter programmet till maskinkod. När kompilatorn är klar med detta mellanlagras koden normalt i en s.k. objekt-fil. (med filändelsen .o eller .obj .) Dessa objektfiler är inte färdiga program utan det är länkarens uppgift att sammanfoga - "länka ihop” - dessa till ett färdigt program. Normalt görs detta automatiskt och du märker inte att det är skillnad på kompilator och länkare.

När kompilering och länkning är avklarat har du förhoppningsvis en exe-fil, en exekverbar- (körbar) fil, eller helt enkelt ett program.

För att ditt program skall kunna köras måste det, som sagt, först placeras i primärminnet . När du kör ditt program begär du alltså att programmet skall laddas in i primärminnet och sedan exekveras instruktion för instruktion. Operativsystemet letar då reda på ditt program och kopierar in det i primärminnet. Kontrollen överlåts sedan till programmet som får köras till det är färdigt eller avbryts.

C++ Introduktion

Uppgift 1:1
Starta kompilatorn Dev C++
Gå in på verktyg (tools) – gränssnitt(interface)
Välj språk Svenska
Utseende Gnome.

Välj Arkiv – nytt – källfil

Skriv in följande kod

```
#include <iostream>                    //inkludera biblioteket iostream

using namespace std;                   //skriv alltid så – förklaras senare

int main( )                            //C++ program startar I funktionen main
{                                      //{ Här börjar funktionen main
  cout<<"HALLO WORLD"<<endl;           // utskriftsmetod cout<<
  cout<<"HALLO WORLD"<<endl;           //<<endl  ger ny rad
  cout<<"HALLO WORLD"<<endl;
  cout<<"HALLO WORLD"<<endl;

  system("PAUSE");                 // för att inte programmet skall blixtra förbi
  return 0;                        //avslut – allt OK – returnera 0
}
//avslutar funktionen main()
```

Välj arkiv spara som.
I din katalog gör du en underkatalog som du döper till C++
I denna katalog vill jag sedan att du **gör en ny katalog för varje program du skapar**.
Skapa alltså en ny katalog och döp den till HalloWorld och spara ditt program där.

Välj sedan exekvera – kompilera.
Om allt nu fungerar skall en ruta visa sig – där får du information om hur kompileringen framskrider.
Om du skrivit in koden rätt bör allt gå bra och texten DONE visar sig. Det betyder att en körbar fil har skapats och ditt program nu är klart att exekveras (köras).

Välj Exekvera – kör, luta dig sedan tillbaka och se ditt första c++-program exekveras.

Uppgift 1:2
Gör sedan 2 nya program som skriver ut ditt namn och din adress.
Först på detta vis:
Kalle. Persson. Krukstigen 14. Bollnas.

Sedan så här:
Kalle. Persson.
Krukstigen 14.
Bollnas.

Vill du kan du göra fler liknande program – Det bästa sättet att lära sig syntaxen i ett programmeringsspråk är att skriva kod, ofta och länge.

Kommentarer i kod
Kommentarer på en rad föregås av //
```
//En kommentar
```
Kommentarer över flera rader föregås av /* och avslutas med */
```
/* En kom-
mentar till*/
```

Kom ihåg att börja all kod med

```
/**************************
Programnamn
Programmerarens namn
Datum
Ev. Förklaring "vad gör programmet"
**************************
```

Variabler och tilldelning

En variabel kan sägas vara en behållare som kan innehålla olika data.
En variabel kan dock endast innehålla en typ av data, T.ex:

```
int         heltal (integer)
float       decimaltal(float)
char        tecken (character)
```

Deklaration eller definiering av en variabel
Deklarationssats: `int plats;` `//obs avsluta med semikolon.`

Deklarationssatsen anger lagringstyp (int) och namnger lagringsplatsen (plats). Systemet tar hand om detaljerna (allokering av minne o.s.v.).

Programmet kommer nu att kunna använda namnet plats för att identifiera det värde som lagras på det avsatta utrymmet i minnet.

•Variabelns namn kan vara i stort sett vad som helst men det får inte börja med en siffra, innehålla mellanslag eller de svenska tecknen å, ä, ö.
•Namnet skall ha anknytning till variabelns funktion.

•Kompilatorn skiljer på versaler och gemener, Tal1 och tal1är alltså inte samma namn.
•Vill du namnge en variabel med flera ord så kan du använda någon av dessa varianter: total_pris eller TotalPrice.

Tilldelningssats
Under förutsättning att variabeln plats har definierats tidigare kan du ge den ett värde med hjälp av tilldelningsoperatorn =

```
plats = 234;
```

Observera skillnaden mot likhetstecken inom matematiken!

Variabeln kan också tilldelas ett värde direkt när den skapas

```
Int plats = 234;
```

Detta kallas **initiering**.

Beskriv, med egna ord eller i bilder, hur det går till att deklarera, tilldela samt använda en variabel. Använd utrymmet nedan!

Mer om variabler

Variablerna kan man utföra diverse operationer på som + - * / .
Satsen: `plats=plats+1;`
Ökar variabelns värde med 1.

Exempel:

```
/************************************************
Litet matteprogram
SSN

Programmet visar diverse operationer på variabler
************************************************

#include <iostream>                    //inkluderat Bibliotek
using namespace std;                   //vilket namespace skall användas

int main()
{

int a=10;                                    //initiering
int b=5;
int c;                                       //deklaration

c=a*b;                                       //tilldelning
```

```
cout<<"the value of int c is "<<c<<endl;

system("PAUSE");
return 0;
}
```

Uppgifter:
3:1. Utöka programmet ovan genom att utföra en subtraktion på variabeln c så att den får värdet 40.
Variabelns värde skall sedan skrivas ut på skärmen.

3:2 Utför lämplig operation på c så att variabeln återfår sitt ursprungliga värde (det går att göra på olika sätt).
Variabelns värde ska återigen skrivas ut på skärmen.

Om du vill kan du lägga in system(”PAUSE”); kommandot mellan varje del i programmet så att programmet exekveras ”bit för bit” på skärmen.

Inmatning

Du vet redan att funktionen cout<< sköter utskrift från program till skärm.
Det finns en motsvarande funktion som sköter inmatning från tangentbord till ditt program. Denna funktion heter cin>>. För att använda den måste du inkludera headerfilen iostream (det har du också gjort tidigare när du använt cout<<). cin>> skickar ett dataflöde från den s.k. inbufferten till programmet.

```
#include <iostream>        //inkluderat Bibliotek
using namespace std;       //vilket namespace skall användas

int main()
{
int a;                     //deklaration av variabel a (heltal)

cout<<"Ange varde för variabel a "<<endl; //skrivs på skärmen

cout<<"tryck sedan enter "<<endl;

cin>>a;                    //inmatat värde läses in till variabel a

cout<<"variabeln a har nu varde "<<a<<endl;

system("PAUSE");
return 0;
}
```

Uppgifter:

4:1. Gör ett program som läser in värden till tre variabler, summerar dessa samt skriver ut resultatet till skärmen.

4:2. Gör ett program som frågar efter ett antal km och sedan omvandlar dessa till meter och skriver ut detta till skärmen.

4:3. Gör ett program som frågar efter din ålder och sedan talar om för dig hur många år du har kvar till din 50 årsdag samt till din pensionsdag.

4:4.Gör ett program som anger medelvärdet av tal1 och tal 2.

4:5 Gör ett program som konverterar SEK till euro enligt följande (kurs 1€ = 8.95SEK).
Välj själv lämplig variabeltyp.
(utskrift till skärm) Ange antal SEK.
>>inmatning av valt belopp från användare
(utskrift till skärm) Du får(antal)Euro.

4:6 Utveckla sedan ditt program så att användaren också får ange gällande växlingskurs.

4:7 Skriv ett program som omvandlar från Celsiusgrader (C) till Fahrenheit (F) med formeln F=1,8C+32. Programmet skall ha följande utskrifter:
Ange en temperatur i Celsiusgrader: >>20
Temperaturen motsvarar 68 Fahrenheit

Vägval, en snabbtitt på if – else .

Jämförelseoperatorer:

<	(mindre än)
>	(större än)
<=	(mindre eller lika med)
>=	(större eller lika med)
==	(är lika med) //skilj på detta och tilldelning
!=	(skilt från)

Villkorsstyrda vägval:

If- else satsen
Den enklaste formen:

```
if(uttryck)
sats;
```

Fler former:

```
if(uttryck)
{
en eller flera satser;
}
else if(uttryck)
{
en eller flera satser;
}
else
sats;
```

Block används så fort det är fler än en sats under en if- eller else-sats.

Ett litet exempel:

```
int a = 5, b = 9;

if (a<b)
cout<<”variabeln b är större än a”;

else if (a>b)
cout<<” variabeln a är större än b”;

else
cout<<” a och b är lika stora”;
```

Vi kommer att titta mer på if-else satser och andra villkorsstyrda vägval senare.

Uppgifter: If-satser och variabler.

1: Gör ett program som erbjuder användaren att välja ett resmål:
Ex:
Vart vill du resa?
Ange:
1 För Egypten
2 För England
3 För Irland
4 För USA
5 För Sverige
6 För Edsbyn

Utskriften som följer skall förstås vara beroende av vad användaren väljer:
Och skall styras av if-satser
Ex:
Planet till Kairo avgår den 27 januari 2005 klockan 14:00
(OBS, till edsbyn reser man med buss).

2: Gör en liten enkel räkneapparat:
Ex:
Ange tal 1:_____
Ange tal 2:_____

Ange räknesätt:
1 för division
2 för multiplikation
3 för addition
4 för subtraktion

Du valde subtraktion.
Differensen mellan 78 och 12 är 66.

3: Gör nu litet frågeprogram:

Där användaren får 5 frågor. Beroende på användarens svar skall det efter varje fråga skrivas ut RÄTT! eller FEL! (svaren bör kunna ges med ett tecken eller en siffra).
Användare får 1 poäng för rätt och 0 poäng för fel svar.
I slutet av programmet skall användarens totala poäng skrivas ut.

Överkurs!
Alldeles hysteriskt roligt blir det om datorn också piper vid varje felaktigt svar.
(ascii tecknet för pipande systemklocka har nummer 7).

4 : Gör ett program där användaren får välja mellan att räkna ut arean av en cirkel eller volymen på en kub (vill du kan du lägga med fler altarnativ).
Användaren skall beroende på val ange de uppgifter som behövs.

5:. Cirkus Flott har följande biljettpriser: Vuxen 75:- Barn 37,50.

Skriv ett program som frågar efter antalet vuxna och antalet barn och sedan skriver ut den totala kostnaden för biljetterna.

6. Gör ett program där man kan mata in en varas pris före moms. Momsen beräknas som 25% av varans pris före moms.

Anta att du matar in 100kr vid körningen. Programmet ska då redovisa utskriften:

Pris före moms 100:-

Moms 25:-

Pris med moms 125:-

Operatorer och typomvandling

Division

Vid division (/) ska man normalt använda flyttal.
Är variablerna deklarerade som heltal utförs annars heltalsdivision (20/3 = 6).
Detta har vi märkt av tidigare i kursen.
Vill vi ha reda på resten vid division mellan två heltal används operatorn % (20%3=2).

Uppgifter
1
Skriv ett program som beräknar resten vid division mellan två heltal. Programmet skall ha följande utskrift:
Ange två heltal: 15 6
Resten vid division är 3

2
Kopiera föregående program till en ny fil och modifiera det så att det istället skriver ut hur många hela gånger nämnaren går i täljaren. Programmet skall ha följande utskrift:
Ange två heltal: 15 6
6 går 2 hela gånger i 15

Typomvandling (typecast)

Antag att vi vill kunna dividera två flyttal (decimaltal) i övningen ovan, och ändå ange ”heltal och rest”. Då måste vi på något sätt göra om flyttal till heltal, eller varför inte vice versa? Det går att göra! Se exemplet nedan:

```
#include <iostream>
using namespace std;

int main()
{
float No1, No2;      //deklarerar två flyttal

cout<<"Mata in 2 heltal med mellanslag mellan talen "<<endl;

cin>>No1>>No2;              //här läser vi in två tal "på släng"

cout<<"Division av dessa blir "<<No1/No2<<endl;
```

```
cout<<"Heltalsdivision ger"<<(int)No1/(int)No2 <<endl; //till integer

cout<<"Resten blir "<<(int)No1%(int)No2<<endl;        //% ger resten!

system ("pause");

return 0;
}
```

Observera hur vi kan välja att läsa in två tal på samma rad!
Fråga: måste vi typomvandla både nämnare och täljare för att kvoten ska bli ett heltal? Testa själv!

Uppgift 3
Modifiera programmet ovan genom att deklarera och mata in heltal. Utför sedan divisionsoperationer på dessa.

Fler variabler och ASCII-Tabellen

Vi har tidigare använt variabler av typen integer (int) vilka kan hålla data i form av heltal.

Här presenteras några fler variabeltyper.

Short	**2 bytes**
int	**4 bytes**
long	**4 bytes**

kan alla hålla *heltal* – av olika storlek. Vi ska snart se efter hur stora!

Float	**4 bytes**
double	**8 bytes**
long double	**12-16 bytes**

håller alla *decimaltal* – av olika storlek.

Datatypen char

char används för att lagra tecken T.ex. 7, 0, @, b, B och #.

Dess storlek är endast en 1 byte, eftersom en byte är precis den storlek som behövs för att lagra ett tecken :-) .

char räknas som en heltalsvariabel, du ska strax se varför.

Datorn använder **ascii-tabellen** för att identifiera tecken och bokstäver. ASCII är en förkortning för American Standard Code for Information Interchange, en 7-bitars kod som representerar de mest grundläggande tecknen i det latinska alfabetet, siffror och andra tecken som används inom datavärlden. Med ASCII-tecken kan vi kommunicera med datorer, som har ett eget, binärt språk, uppbyggt på nollor (0) och ettor (1).

När vi skriver ASCII-tecken via tangentbordet gör datorn en binär tolkning av dem så att de kan läsas, bearbetas, lagras och hämtas.

En del tecken går inte att skriva i programmen på vanligt sätt. Lösningen är att använda s.k. escape-sekvenser istället för specialtecken och tangenter.

Escape-sekvenser inleds med tecknet \.

Några exempel:

\n	ny rad
\t	horisontell tab
\b	backsteg
\\	\

Med hjälp av escapesekvenser kan du också skriva de svenska tecknen **å, ä, ö** i programkoden.

Den ursprungliga ASCII-tabellen består av 128 tecken, våra svenska ”specialtecken återfinns i den utökade- (extended) tabellen.

utskrift av svenska specialtecken

\x86	ger tecknet	å	(86 är hexadecimal* representation för 134)
\x84	ger tecknet	ä	
\x94	ger tecknet	ö	
\x8F		Å	
\x8E		Ä	
\x99		Ö	

** Senare i kursen får du lära dig mer om bl.a. hexadecimal representation av tal.*

Här är ett annat sätt att komma åt våra svenska specialtecken:

Håll ned [Alt]-tangenten samtidigt som du skriver in något av följande värden på den numeriska delen av tangentbordet:

å = ALT + 0134

ä = ALT + 0132

ö = ALT + 0148

Å = ALT + 0143

Ä = ALT + 0142

Ö = ALT + 0153

Det kommer att se märkligt ut i din källkod, men när du kör programmet blir det bättre ☺.

Testa denna kod:

```
#include <iostream>
using namespace std;
int main()
{
cout<<"specialtecken \x86 \x84 \x94 \x8F \x8E \x99"<<endl;
system("PAUSE");
return 0;
}
```

Prova även den andra metoden [Alt] + nummer för att skriva ut motsvarande tecken.

Ett litet exempel på användning av char.

```
char svar= ' j ';                    //obs! Tecknet som tilldelas svar omges med ' '.
cout<<”Vill du fortsätta ange j annars n ”;
cin>>svar;
if (svar= = 'j')
 go on;
else
go home;
```

Övning : Testa att ”typecasta ” en integer till char vid utskrift!

PÅ nästa sida följer ASCII-tabellen, inklusive de första 32 s k icke-skrivbara tecknen.

ASCII_Tabellen

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	∑
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Några annorlunda operatorer

Vi har redan gått igenom fem olika operatorer:
+ - * / %

Exempel:

```
C =A/B                    // Dividerar A med B och lagrar i C
A =A*B                    //multiplicerar A med B och lagrar i A
```

Det finns dock några annorlunda ”korta” operatorer som tål att synas lite extra. Du behöver inte använda dessa skrivsätt, men de kan vara bra att känna till om du skulle ”springa på dem”.
Vi tittar på några exempel och vad de gör:

```
A + = B          ger A=A+B
A - = B          ger A=A-B
A * = B          ger A=A*B
A / = B          ger A=A/B
A% =B            ger A=A%B
```

Förändringsoperatorer
Vi har tidigare ökat eller minskat en variabels värde enligt följande exempel:

```
Tal1=Tal1+1;        eller        Tal1=Tal1-1;
```

Kortare kan vi skriva:

```
A++        ++A        ger A=A+1
A--        --A        ger A=A-1
```

(Om vi skriver variabeln före eller efter ++ ger viss skillnad. Testa gärna det i en utskrift)

Logiska operatorer

Vi börjar med några operatorer som kallas jämförelseoperatorer. Olika typer av jämförelseoperatorer används bl a vid urval med villkorssatser (förklaras senare).
Antag att A tilldelas värdet 4: int A=4;

<	Mindre än	A<5	true
>	Större än	A>5	false
<=	Mindre eller lika med	A<=4.5	true
>=	Större eller lika med	A>=3.9	true
==	Lika med	A==3	false
!=	Inte lika med	A!=3	true

Vad menar vi med värde i denna tabell? För att förstå det introducerar vi nu en datatyp vi inte tidigare använt: bool. Denna datatyp kan anta två värden, sant (true) eller falskt (false).
Det kanske går att visa ännu bättre med ett exempel:

```
bool kalle=(5>4);
cout<<kalle;                    //ger utskriften 1 (true)
kalle = (5!=5);                 //eftersom 5 inte alls är skilt från 5 ger detta kalle värdet 0
(false).
cout<<kalle;
```

(en meningslöst programmeringsexempel men jag tror att ni förstod iallafall?)
Lite mer valmöjligheter att jobba med får vi om vi tar till följande operatorer:

&&	Och
||	Eller *
!	Icke

*(ni hittar detta || tecken på samma tangent som <>tecknen på tangentbordet)

Dessa operatorer öppnar dörren till en helt ny värld för oss. **Den booleska logikens underbara värld!** I den kan man grotta ned sig riktigt (och det gör många). Vi hinner dock bara glänta lite på dörren här, men vi traskar på i ullstrumporna och ger oss raskt i kast med några exempel:

10<11 || 9>12	Sant eftersom 10 är mindre än 11
10>11 || 9<12	Sant eftersom 9 är mindre än 12
10>9 || 9<12	Sant
10<9 || 9>12	Falskt

Det räcker alltså med att ett av påståendena är korrekt för att utsagan skall vara sann.

10<12 && 12>10 Falskt (båda uttrycken måste gälla).

Vägval: Repetition if -else

I programmering uppkommer ofta olika valsituationer, t ex att välja ut det största eller minsta talet. Detta åstadkommer man enkelt med en if-sats.
Ett programexempel som läser in två tal och skriver ut det största kan se ut på följande sätt:

```
#include <iostream.h>
using namespace std;
main()

{
int a,b;                                        //Deklaration av integer a och b.

cout << "Ange två heltal: " ;

cin >> a >> b;                                  //läser in  a och b från tangentbordet.

if (a>b)
              cout << a << " Är störst";
else
             cout << b << " Är störst";
}
```

Övning 1
Modifiera (och förbättra) koden så att om a är ”större eller lika med b” så gäller första if-satsen, annars gäller else-villkoret.

Övning 2
Modifiera koden så att den omfattar tre olika stora variabler a, b och c.
Om a är mindre än b och samtidigt större än c så skriv ut ”Bra, du vann”. Annars skriv ut ”Sorry, du förlorade”

Strukturdiagram

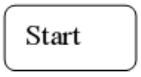
Nu börjar koden bli lite marig att hålla reda på – med if-satser kan det bli riktigt stökigt i programmerarens huvud! Nu kommer vi till en viktig del av programmeringskursen.
Fram till nu har vi mest sysslat med algoritmer, i form av källkod. När vi börjar skriva ett program så är det viktigt att ha klart för sig vad som ska göras i en uppgift, och hur vi ska lösa den.

Pseudokod

Första steget är pseudokoden. Då skriver vi ned med vanlig svenska vad som programmet ska utföra. Gör detta strukturerat, rad för rad!
Nästa steg är att göra ett strukturdiagram. Dessa är standardiserade och kan förstås av inte bara olika nationaliteter utan också olika programspråk. Låt oss titta på några olika symboler i ett strukturdiagram:

Start- och stoppruta

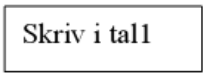
Ex:



Som det hörs på namnet så startar och slutar programmet med en sådan.

Sats

Ex:



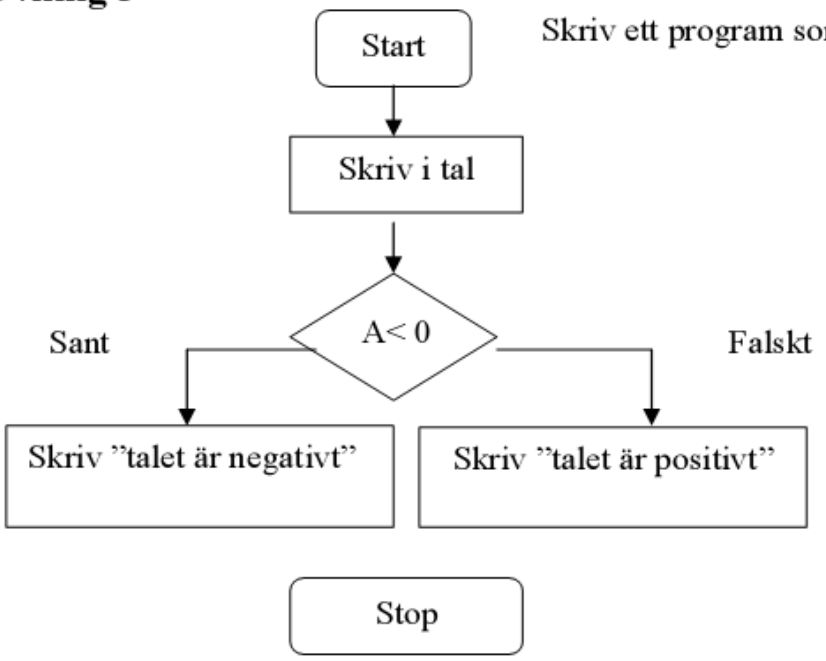
Här utförs alltså något.

Selektion

Ex:

Här väljer programmet mellan två vägar. En if-sats!

Övning 3



Övningar på strukturdiagram

Gör pseudokod och strukturdiagram till övning 1 och övning 2 här nedan. Det går att rita för hand!

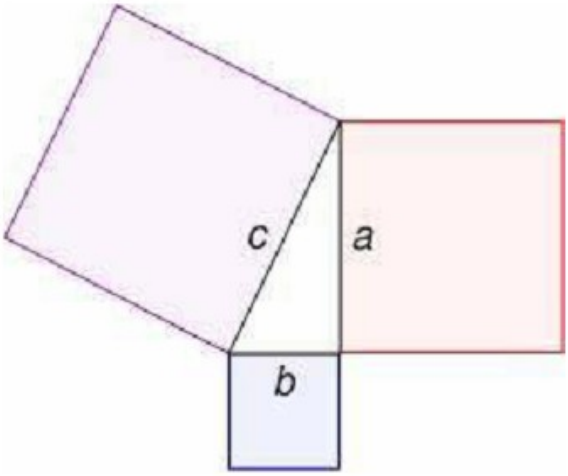
Uppgifter i pseudokod, strukturdiagram och källkod.

1.
Skriv ett program som låter användaren mata in ett uppmätt avstånd i enheten cm på en karta samt kartans skala. Därefter ska programmet presentera det verkliga avståndet i enheten meter.

2.
Skriv ett program som låter användaren mata in ett uppmätt avstånd i mm, cm eller m på en karta samt kartans skala. Därefter ska programmet presentera det verkliga avståndet i enheten meter.

3.
Skriv ett program som löser ekvationen a * x - b = c. Användaren ska mata in värden på koefficienterna a, b och c under körningen, varvid programmet ger värdet på x.

4.
Skriv ett program -=PYTHAGORAS=- som med hjälp av t.ex. a och b ger c. Låt användaren välja mellan om det är hypotenusan eller kateter som ska beräknas.



I biblioteket **cmath** finns två användbara funktioner:

double sqrt(double tal);
// funktion som returnerar kvadratroten av talet tal

double pow(double t1, double t2);
// funktion som returnerar tal t1 upphöjt till tal t2

Funktionerna används så här:

```
#include <iostream>
#include <stdlib>
#include <cmath>

int main()
{
double x,y,z=0; //alla variablerna initieras till 0
  x=2;
  y=16;
  z=4;

cout<<"roten ur 16 = "<<sqrt(y)<<endl; //anrop av sqrt med parametern y

cout<<"4 upphojt till 2 = "<<pow(z,x)<<endl<<endl;;

  system("PAUSE");
```

```
  return 0;
}
```

Konstanter

När man i förväg vet att en variabel inte ska eller inte får ändra värde kan och bör man deklarera den som en **konstant**. Det är vanligt att man namnger sina konstanter med VERSALER.
Exempel:
En cirkels omkrets dividerat med dess diameter ger alltid samma tal (nämligen Pi), alltså gör vi en konstant av detta tal.

```
const float PI=3.14;          //nu går PI inte att ändra på

float radie=4;

float omkrets =2*radie*PI;

cout<< ”omkretsen är ” <<omkrets<<endl;
```

LOOPAR

while

Med kommandot while ( ) testas ett villkor och om det är sant utförs satsen eller satsblocket som följer. När varje sats utförts går programmet tillbaka till while ( ) och upprepar testet. Om villkoret i första testet är falskt utförs inte satsen och om villkoret är sant måste någonting i den följande satsen/satserna göra så att villkoret blir falskt, annars kommer repetitionen aldrig att upphöra. Programmet har då hamnat i en icke terminerande loop – en evighetsmaskin!

Den allmänna formen för while-satsen är:

```
while(villkor)
{
sats1
sats2
sats 3
och så vidare.
}
```

Följande exempel skriver ut värdet för variabeln siffra t.o.m 200:

```
#include<iostream.h>
```

```
using namespace std;
int main()
{
int siffra;
cout << "skriv in ett tal: ";
cin >> siffra;

while(siffra<201)
{
 cout << siffra << "\n";
 siffra++;
 }
 return 0;
}
```

do-while

Denna loop är i princip samma sak som ovanstående. Skillnaden ligger i att man alltid går minst ett varv i loopen, eftersom villkoret testas först i slutet.

```
int main()
{
 int x;
 cout << "skriv in ett tal: ";
cin >> x;

 do                    //do : utför följande satser
 {
 cout << x << "\n";
 x++;
 }
while(x<201);         // så länge villkoret gäller

return 0;
}
```

Du kan använda while när du vill att en programkörning skall upprepas till dess användaren väljer att avsluta. Detta åstadkommer du genom att lägga hela programslingan i en while-loop.

```
int main()
{
int siffra;
 int svar=1;                    //villkoret för while

 while (svar==1)              //Villkoret för while
 {
cout<<"skriv in ett tal,programmet fyller på upp till 200:"<<endl;
cin >> siffra;
while(siffra<200)
 {
 cout << siffra << "\n";
siffra++;
 }                              // forts nästa sida

cout<< " vill du fortsatta? "<<endl;
 cout<< "skriv 1 för ja - 2 för nej" <<endl;
 cin>>svar;                     //användaren styr iterationen
 }
 return 0;
}
```

Uppgifter:

10:1 Gör ett program där användaren får mata in ett heltal (n). Programmet skall därefter skriva ut ”multiplikationstabellen” för talet. Om det inlästa talet är 5 skall alltså 5, 10, 15, 20, 25,. . . . . 50 skrivas ut. Använd while( ) satsen.

10:1 b Förändra programmet så att användaren får välja om han/hon vill kan köra programmet en eller flera gånger.

10:2 Tänk dig följande scenario. Antalet råttor är för närvarande 100 st i Kattbo kommun. Antalet råttor fördubblas varje månad. Hur många månader tar det innan antalet råttor uppnått en miljon?

Tips: låt inledningen på varje varv i loopen undersöka om antalet råttor uppnått en miljon.

10:2b Förändra sedan programmet så att du ser ökningen månad för månad.

10:3** Snabbköpskassörskan i kvarteret där jag bor söker ett nytt program till sitt kassaregister. Hon vill ha möjlighet att skriva in ett valfritt antal varor – vid inmatning av 0 (noll) skall varornas värde summeras och summan skrivas ut.

10:3b Utöka programmet så att även momsen 25% specificeras.

Användaren skall även ha möjlighet att köra programmet igen – när en ny kund dyker upp

(**Uppgift 10:3**)

Exempel på en körning av programmet ovan:

Starta inmatningen - avsluta med 0

Pris:12.67_____
Pris:23.56_____

Pris:23.56_____

Pris:234.00____

Pris:0_____

Totalsumma:293.79

Varav moms:58.758

Ny kund? [j] / [n]:___

For-satsen

Man brukar ofta utnyttja en for-sats när man har en repetition där det finns en räknare som skall räknas upp eller ned för varje varv.

Den allmänna formen för for-loopen är följande:

```
for(initialisering; upprepningsvillkor; ändringsuttryck)
{
sats 1;
sats 2;
sats 3;
……
}
```

I programmet här nedan kan du se att for-loopen deklarerar och initialiserar en variabel **`int i=10;`**. Denna variabel kallas för räknare eller index (det är mer eller mindre standard att man ger räknaren namnet i). Om upprepningsvillkoret är uppfyllt utförs satserna mellan { och }. Därefter utförs ändringsuttrycket. Om du vill att ändringsuttrycket ska utföras före satserna kan du skriva uppräkningen som prefix, så här: ++i.

For-satsen här nedan skriver ut talen 10, 9, 8, 7, …..0.

```
#include <iostream.h>
int main()
{
  for(int i=10; i>=0; i--)
     {
   cout<<i<<"\n";
   }
system("PAUSE");
return 0;
}
```

Uppgifter:

11:1. Gör en forloop där användaren får ange start och stopp -värde samt steglängd.

Programmet skall skriva ut största värdet först och det lägsta sist med önskad steglängd mellan varje värde.

11:2 Gör ett program som med hjälp av en forloop skriver ut ascii-tabellens 128 tecken.

(tänk typecast).

11:3 Ändra ascii-programmet så att användaren anger slutvärde d.v.s om användaren skriver in 70 skall programmet skriva ut asciteckned 0 - 70.

11:4 Gör ett program som skriver ut multiplikationstabellen (t.o.m n*10) för ett av användaren angivet tal.

11:5 Gör ett program som finner alla heltalslosningar

till ekvationen **`3x - 7y = 1,  med -50<=x=>50 och -20<=y=>20`**.

Lite fler if-satser

Den enklaste satsen har vi redan testat: If – else (om villkoret uppfylls så gör man detta, annars något annat). Man kan också strunta i else-satsen om inte programmet ska utföra något för false. Om vi måste utföra många satser (för t.ex true) så samlar vi dessa satser mellan två ”spetsparenteser”:

Ex.

```
#include <iostream>
#include <conio.h> //för getch();
using namespace std;

int main()
{
  int age=0;
  cout<<”How old are you?”<<endl;
  cin>>age;

        if (age>=18)
        {
              cout<<”Hmmm, maybe you drive a car?”<<endl;
              cout<<”Bye bye”;
        }
```

```
                else
                  cout<<"I guess you have a bicycle?";
              getch();//istället för system("PAUSE")

    return 0;
}
```

*Lägg märke till hur koden är **indenterad** (indrag) på satser som hör samman.*

If-else if

Möjligheten att hantera flera val än två finns naturligtvis. Vi skulle kunna bygga upp detta med enkla if-else, men det blir snabbt mycket otympligt att hålla reda på. Så här gör man:

```
if ( uttryck 1)
          Sats1
else if (uttryck2)
          Sats2
else if (uttryck3)
          Sats3
else
          Sats4
```

Uppgift 13:1 Tillverka ett program som tillåter andra kompisar komma in på din fest. De får inte vara mindre än ”2 år yngre” än du, men heller inte äldre än ”2 år äldre”. Gör först psuedokod och strukturdiagram! Kom ihåg att öva indentering (indrag).
Du skall kunna köra programmet om och om igen ”smidigt om det kommer fler än en besökare till festen”. Använd While eller do-while till detta.

Uppgift 13:2 Modifiera din kod. Om användaren är mer än 30 år äldre så uppmana vederbörande att gå hem till Hamrelund eller Ängslunda.

Slumptal

Slumptalsfunktioner finns i inkluderingsfilen cstdlib

Funktionen rand() ger ett heltal som är större än eller lika med 0.

För att inte samma slumptalserie ska genereras varje gång programmet körs måste du mata funktionen med olika startvärden, s.k. ”frön” vid olika körningar. Det gör man med srand(startvärde). Ett sätt att få olika startvärden är att anropa systemets klocka som (*håll i dej nu*) visar olika tider vid olika tillfällen. Klockan får du tillgång till om du inkluderar filen time.h.

Anropa srand(time(0)) innan rand() används första gången.

För att generera tal från t.ex. 0 - 9 heltalsdividerar du med 10 och sparar resten (modulodivision har vi ju gått igenom några ggr). Resten kan ju då bara bli tal från 0 till 9 (10 – 1).

```
int main()
{
srand(time(0));
int resultat;

for(int i=0; i<5; i++)
{
resultat=1+rand()%10;   //modulodivision (slumptal+1%10)
cout<<resultat<<endl;
}
      system("PAUSE");
      return 0;
}
```

Uppgift 14:1 Skapa ett program som slumpar fram en eller flera lottorader
(7 nummer mellan 1 och 35).

Uppgift 14:2 Gör ett program som skapar ett lösenord. Ordet skall slumpas fram genom användning av gemenerna a - z.

Uppgift 14:2 Här kommer nu en lite större uppgift. Du skall tillverka ett spel där användaren får gissa på ett tal, mellan 1 och 100, som slumpats fram av datorn. Beroende på hur användaren gissar skal ditt program reagera på olika sätt.

- Fel- din gissning är för låg. Försök igen!
- Fel- din gissning är för låg. Försök igen!
- Rätt! Talet var (slumpat tal).

Användaren får 7 försök efter dessa skall programmet avbrytas.

Du kan förresten lägga till en liten stilfull fanfar när användaren lyckas gissa rätt. Till detta kan du använda koden nedan..

```
#include <cstdlib>
#include <iostream>
#include <windows.h>

using namespace std;

int main(int argc, char *argv[])
{
    Beep(131, 200);
    Beep(131, 200);
    Beep(131, 400);
    Beep(165, 400);
    Beep(196, 400);
    Beep(262, 900);

    system("PAUSE");
```

```
    return EXIT_SUCCESS;
}
```

Storlek på datatyper

Du vet nu att en variabel av typen **int** kan hålla ett *heltal*.
Satsen: `int minEgenVariabel;`
Skapar plats i minnet för ett heltal och namnger denna plats minEgenVariabel.

Hur stor plats?
Detta kan variera beroende på system men det vanligaste är, som du sett tidigare i kursen, 4 bytes.
Detta kan man kontrollera genom funktionen **sizeof( )**. Du kan prova att skriva följande utskriftsats:

```
cout<<sizeof(int);
```

Systemet svarar med att skriva ut storleken på en int uttryckt i bytes (en byte = 8bitar).
sizeof ( ) kan naturligtvis användas med andra typer av variabler t.ex. double och float.

En int-variabel är automatiskt ”signed” (ett heltalsvärde med ett + eller minustecken framför) och delar då sitt omfång mellan negativa och positiva tal.

Med prefixet unsigned (unsigned int) kan variabeln endast hålla positiva tal och då dubbelt så stora som en signed variabel.

Vet vi att vårt program endast kommer att arbeta med positiva tal kan vi ange vår heltalsvariabel som en unsigned int. Denna kan då hålla värden mellan 0 och 4294967295.
En signed int kan hålla heltalsvärden mellan -2147483648 och 2147483647.

Det stämmer alltså med vad som skrevs lite tidigare i kapitel 5:
En integers storlek är 4 bytes
En byte är 8 bitar.
En byte = 2^8 = 256
Och 2^{32}= 4294967296

Headerfilen **limits.h** innehåller information om gränserna för heltalstyperna genom att definiera symboliska namn för olika gränsvärden. Den definierar t.ex INT_MAX som systemets största int-värde.

Overflow

Försöker du ge en variabel ett för stort värde inträffar något som kallas overflow,

variabeln ”går över gränsen”. Som exempel kan vi tänka oss en variabel short X (2 bytes) som initieras till maximalt positivt värde, vilket är 32767. Om följande operation utförs på variabeln `:X=X+1`;

Då inträffar ett overflow och variabeln får lägsta möjliga negativa värde -32768.

Jämför:

1. Föreställ dig att klockan är 24.00.
2. Lägg till en minut.
3. Du kanske ville att klockan skulle bli 24.01? Istället blev den ju 00.01.

Här har faktiskt skett ett overflow. Klockan kunde inte ta emot mer, utan började om på sitt lägsta värde, noll...

Övning:
Skriv in, kompilera och betrakta resultatet av följande programsnutt:

```
#include <iostream>
#include <limits.h>

using namespace std;

int main()
{
int olle;  //deklaration av variabel av typen integer
unsigned int gun;     //dekl. Av en unsigned integer

olle=INT_MAX;                    //tilldelning ”olle får ett värde”

cout<<"maxvarde for signedInt\n"<<endl;
cout<<olle<<endl;
cout<<"minvarde for signedInt\n"<<endl;

olle=INT_MIN;          //tilldelning ”olle får ett nytt värde”
cout<<olle<<endl;

gun=UINT_MAX;
cout<<"minvarde for unsigned int"<<endl;
cout<<gun<<endl<<endl;

system("PAUSE");
return 0;
}
/**********************************/
```

Nu till en programmeringsuppgift där du får klara dig helt på egen hand.
Uppg 1:
Tänk dig följande scenario, och åskådliggör detta med ett program:

Olle och Greta har var sitt bankkonto
Olles konto är av typen int (int olle;)
Gretas konto är av typen unsigned int (unsigned int greta;)
Olle initieras till maxvärdet för integer (INT_MAX)
Samma summa sätts in på gretas konto (greta=olle;).

Både Greta och Olle vinner nu varsin krona på lotteri: Eftersom ingen av dem lider någon akut brist på pengar sätter båda in hela vinsten på sina konton.
Hur mycket har nu Olle på sitt konto?
Hur mycket har Greta?
Genom ett datorhaveri på banken nollställs både Gretas och Olles konton.
Dagen efter tar Olle ut 1 krona.
Det gör också Greta.
Vad är nu saldot på de båda spararnas respektive konton? Vem är gladast?

Array – en lista eller vektor

Man kan göra listor, arrays eller vektorer (olika namn på samma sak) av alla de typer av variabler vi tittat på. Skilj mellan vektor i programmering och i matematiken.
– Vi kommer att försöka använda termen array i kursen.

Istället för att deklarera en hel hög variabler - så här:
int Honken;
int Lill_Strimma;
int Virus;
int Stisse;
int Ulf_Sterner

Kan man istället skriva detta som:
```
int TreKr [21];
```

Nu skapas plats i minnet för 21 variabler av typen integer. Dessa minnesplatser kommer att benämnas
TreKr[0], TreKr[1], TreKr[2], ..., TreKr[19], TreKr[20]
OBS: numreringen börjar på 0!

Initiering av array

En array kan initieras på lite olika sätt:

```
int TreKr [21];  //deklaration

TreKr [0]=3;
TreKr [1]=7;
TreKr [2]=9;
TreKr [3]=4;
.
.
```

Man kan också lämna hakparentesen öppen och bestämma storleken vid initieringen:

```
int torsdag [ ] ={5,12,19,26};
```
Arrayen torsdag får nu fyra minnesplatser: torsdag[0], torsdag[1], torsdag[2], torsdag[3].

Satsen: `cout<<torsdag[3];`
Kommer att ge utskriften **26.**

Låt oss titta på ett kodexempel, och vi introducerar getchar( ); istället för system(”pause”):

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main()
{
int testArray[4];

   cout<<"ge mig fyra heltal!"<<endl;

   cin>>testArray[0]>> testArray[1]>>testArray[2]>>testArray[3];

cout<<"Talen du angav var: "<<endl;

   cout<<testArray[0]<<endl;
   cout<<testArray[1]<<endl;
   cout<<testArray[2]<<endl;
   cout<<testArray[3]<<endl;

   cout<<"Adios!";

getchar( );       //i stället för system(“PAUSE”) väntar på tang-tryck.
return 0;
}
```

ÖVNING 8:1

Skapa ett program som låter användaren läsa in sin födelsedag (ex 850231) till en array. Skriv sedan ut när födelsedagen inträffar. OBS: varje siffra ska ta en position i arrayen!

Textsträngar

En textsträng (array av tecken) är en följd av tecken, exempelvis är *Anders* en textsträng som består av 6 tecken. Vill vi läsa in en textsträng innehållande högst 8 tecken till variabeln *namn* – som är en array av typen char - gör vi följande variabeldeklaration:

```
char namn[8];
```

Variabeln kan även *tilldelas* ett värde vid deklarationen:

```
char namn[8] = ”anders”;
```

Jämför: int TreKr[ ]={1,2,3,4};

```
cout << ”Ange ett namn: ”;
cin >> namn;
```

Vill du *skriva ut* tredje tecknet i textsträngen gör du följande utmatning:

```
cout<<namn[2];
```

Tecknet *d* skrivs då ut om variabeln *namn* innehåller strängen *anders*. I strängen har första tecknet ordningsnumret 0, andra tecknet ordningsnumret 1 o s v, enligt figuren intill.
I position 6, d v s i positionen efter det sist inlästa tecknet, hamnar ett radslut s.k. null-tecken (\n).

0	1	2	3	4	5	6	7
a	n	d	e	r	s		

Det kan fungera att läsa in fler än 8 tecken till en strängvariabel som är deklarerad för 8 tecken, med detta är *inte att rekommendera*.

ÖVNING 8:2
Skriv ett program som läser in ditt förnamn och som sedan skriver ut det igen på följande sätt:

```
Ange ditt förnamn:
>>Christopher

Du heter Christopher, som börjar på bokstaven C.
```

Hur avgör cin>> när en sträng är klar för inmatning?
Du *kan inte* skriva nulltecknet från tangentbordet, så cin måste använda ett annat sätt för att avgöra när en sträng tar slut.
Den teknik cin använder är att låta blanktecken: **mellanslag, tabbar, och radslut markera slutet på en sträng.**
Det betyder att cin bara kan läsa in *ett* ord till en textsträng (array av tecken). När ordet har skiljts lägger cin automatiskt in ett avslutande nulltecken.

ÖVNING 8:3

Utöka programmet på följande sätt:

Ange ditt efternamn:

>>Walker

Du heter Walker, som börjar på bokstaven W

Vad heter din hund?

>>Devil

Aha, din hund heter **Devil**, då har du en häst som heter Hero, eller hur?

Hur läser vi in flera ord?

istream, (inkluderad i iostream) som cin är ett exempel på, har några medlemsfunktioner som klarar att läsa in en hel rad tecken.
getline( ) läser in en helrad och låter radslutstecknet (enter-tryckning) markera slutet på inmatningen.
Den aktiveras med funktionsanropet **cin.getline()** som tar två argument, namnet på den array som ska ta emot inmatningen och antalet tecken som kan läsas in.

Ex: **cin.getline(namn, 8);**

Här kommer ibland problem att uppstå: När cin>> gör sin första inläsning kommer radslutstecknet att lämnas kvar-

nästa inläsning kommer direkt att stöta på ett radslutstecken och därför uppfatta att inläsningen är slut.
Ingenting kommer alltså att läsas in.

Det finns en lösning på detta problem – en variant av cin.get, med tom parameterlista läser endast in ett tecken.

cin.get( ) kan alltså användas för att ta bort radslutstecknet från inmatningskön.

En annan metod är **cin.ignore(1000, '\n');**
Som plockar bort alla tecken till och med första nyradstecken (max 1000 tecken).

ÖVNING 8:4

Utöka programmet från föregående övning så att utskriften blir följande

Ange ditt förnamn:

>>Christopher

Du heter Christopher, som börjar på bokstaven C

Ange ditt för och efternamn:

<<Christopher Walker

Du heter **Christopher Walker** men är mer känd under namnet Fantomen.

ÖVNING 8:5

Lägg till att användaren (Fantomen?) även får läsa in sitt telefonnummer (du kan separera rikt- och telefonnummer med t.ex. mellanslag).

Hur du fyller en lista med hjälp av en forloop:

```
int storlek=8;

int hastighet[storlek]; // hastighet har 8 platser 0 -7

for( int i =0; i<=storlek; i++)
{
hastighet[i]=5*(i+1)   //får värden:5,10,15...,35,40.
}
```

Hur du skriver ut en lista med hjälp av en forloop:

```
for( int i =0; i<=storlek; i++)
{
cout<<hastighet[i]<<endl; //Skriver ut:5,10,15...,35,40.
}
```

OBS: <u>Den variabel du initierar i loopen (int i=0) finns/syns endast i loopen</u>. Den skapas när loopen startar och dör (raderas) när loopen terminerar(avslutas).

11:6 Gör ett program som fyller en heltalslista med värden och sedan skriver ut dem
Använd for-loop.

11:7* Skapa ett program i vilket användaren får ange ett valfritt antal värden som skall jämföras **`(cin>>a;  int lista[a];).`**
När användaren angett dessa värden skall ditt program jämföra dem och skriva ut det lägsta (Använd en lista (array) samt en eller flera valfria repetitionssatser):

En körning av programmet kan se ut på detta vis:

```
Ange det antal värden du vill jämföra:6__
Skriv in det första värdet: 12_
Nästa värde:77_
Nästa värde:11_
```

```
Nästa värde:3__
Nästa värde:2__
Nästa värde:4__

Inlästa värden
12 77 11 3 2 4

Det lägsta värdet är 4 och finns på position 6 i listan

Tryck på en valfri tangent för att fortsätta...
```

Klassen <string>

En liten tjuvtitt på klassen string – fördjupning i stringklassen kommer i kurs B.

String är en klass som ersätter/kompletterar C++:s strängar (char-array).
Jämförelse av strängar kan göras med hjälp av de vanliga = = != > < >= <= operatorerna och kopiering kan ske med tilldelningsoperatorn.
Du behöver inte, som med char-arrayer ange storlek på strängen.
Exempel på stränghantering med Sträng-klassen:

```
    #include <string>
using namespace std;

int main()
{
    string Text1 = "Hej!";
```

```
    string Text2;
    Text2 = Text1;

    if(Text1 == Text2)
      cout << "Text 1 & 2 var lika!" << endl;

    Text1[1] = 'o';
    cout << Text1 << endl;
    cout << Text2 << endl;

    if(Text1 != Text2)
      cout << "Text 1 & 2 var olika!" << endl;
  else
   cout << "Text 1 & 2 var lika!" << endl;
}
```

Inläsning till ett objekt ur klassen string sker enklast med getline(cin, variabelnamn).
Metoden getline() läser in en hel rad tecken och avslutas genom [Enter] / ny rad.

```
    #include <string>
using namespace std;
int main();
{
    string Text2;
    cout<<”Skriv någonting!”;
    getline(cin, Text2);
    cout << "Du skrev" <<Text2<< endl;

}
```

Övning: Testa klassen String med ett enkelt program där du läser in namn och adress till strängarna namn och adress.

Nästlade loopar

Du kan lägga repetitionssatser i repetitionssatser (nästlade loopar).
En for-sats i en for sats eller i en while-sats eller tvärtom.
Här kommer ett exempel på en nästlad for-sats:

```
int main()
{
    for( int i =1; i<5; i++)
    {
         cout<<"varv "<<i<<endl;
         for(int x=0; x<i; x++)
         {
                 cout<<"hej svejs! ";
```

```
        }
        cout<<endl;
    }
   getchar();
    return EXIT_SUCCESS;
}
```

Programkoden ovan ger denna trevliga utskrift

```
varv 1
hej svejs!
varv 2
hej svejs! hej svejs!
varv 3
hej svejs! hej svejs! hej svejs!
varv 4
hej svejs! hej svejs! hej svejs! hej svejs!
```

Uppgift 12:1 Din uppgift blir nu att skapa ett program som med hjälp av en nästlad repetitionssats skapar följande utskrift

```
*
**
***
****
*****
******
*******
********
*********
**********
***********
```

Switch-case

Ibland kan *switch-case* vara händigt. Du kanske har ringt till en telefonsvarare på en myndighet eller liknande? Där möts man ofta av något slags meny-val, där man ombeds trycka 1 för det ena, 2 för något annat etc. *switch-case* är som klippt och skuret för detta!

Ex:

```
#include <iostream>
#include <conio.h> //för getch();
using namespace std;

int main()
{
   int Tal;
                cout<<"For Saldo - tryck 1. For transaktioner - tryck 2. "<<endl;
```

```
        cout<< "For telefonist – tryck nat annat"<<endl;

cin>>Tal;
switch (Tal)
        {
        case 1:
                cout<<"Du far nu veta ditt saldo"<<endl;
                break;                                  //om inte alla case-satser ska utföras!


        case 2:
                cout<<"Du far flytta pengar"<<endl;
                break;

       default:                                         //tar hand om alla feltryckningar
                cout<<"Du blir kopplad till telefonist"<<endl;
        }

   getch();
   return(0);
}
```

Uppgift 15:1 Gör en snygg meny med en mängd val. För varje val ska olika utskrifter presenteras!

En enkel kalkylator

Att bygga en kalkylator där användaren får fylla i sitt tal är faktiskt inte helt lätt! Anta att man skriver 7+2 i en cin>>. Plustecknet kommer inte att läsas in som ett plustecken, utan bara som ett vanligt tecken (!@? - öh).
Här gäller det att var klurig. Titta på exemplet, det är i inläsningssatsen det händer!
Inläsningen till en integer avslutas när heltalet avslutas – en charvariabel innehåller endast ett tecken (+ eller-) som läses in till raknesatt – vad som finns kvar att läsa in är då ett nytt heltal som läses in till tal2.

```
#include <iostream>
#include <conio.h> //för getch();
using namespace std;

int main()
{
   double tal1, tal2;           //operander
   char raknesatt;              //operator plus eller minus
   cout<<"Skriv in uttrycket som ska beräknas (plus eller minus): "<<endl;
   cin>>tal1>>raknesatt>>tal2;
   switch (raknesatt)
     {
     case '+':
          cout<<"Svar: "<<tal1+tal2<<endl;
          break;
     case '-':
```

```
            cout<<"Svar: "<<tal1-tal2<<endl;
            break;
        default:
            cout<<"Felinmatning"<<endl;
        }
    getch();
    return(0);
}
```

Observera hur vi gör i case-satsen för ’+’ eller’-’.

Uppgift 15:2 Modifiera koden i exemplet så att vi även kan räkna division och multiplikation. En mycket bättre räknare alltså!

Uppgift 15:3 Programmering och matte hör ju intimt samman. Duktiga programmerare är också ofta rätt slängda i matte! Speciellt rutinoperationer kan ju datorer enkelt räkna – då slipper ju vi! Nu kommer en uppgift som du som elever i Ma B har stor nytta av:
2:a gradens ekvationer löser man enklast med den populära **pq-formeln**:

Om vi har skrivit en ekvation på formen $\mathbf{x^2 + px + q = 0}$
Så vet vi (genom kvadratkomplettering) att denna har lösningarna:

$$x = -\frac{p}{2} \pm \sqrt{(\frac{p^2}{4} - q)}$$

Skapa ett program som låter användaren ange värdena på p och q, varefter programmet beräknar rötterna. Det ska vara användarvänligt!
Modifiera koden så att programmet inte slutar att fungera om reella rötter saknas! Du får gärna berätta för användaren att den har ”komplexa rötter” (det kommer i MaE).
Låt användaren mata in en valfri 2:a grads ekvation, tex $4x^2 – 3{,}5\ x – 23{,}22$.

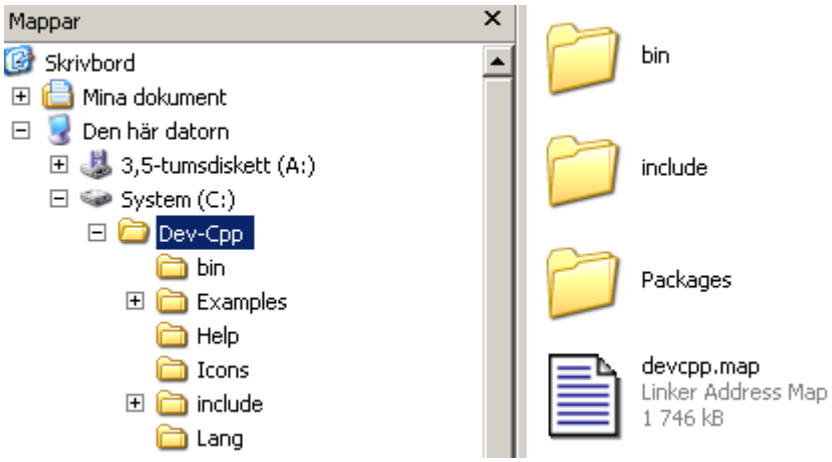
Använd gärna utrymmet härunder till strukturdiagram!

APPENDIX

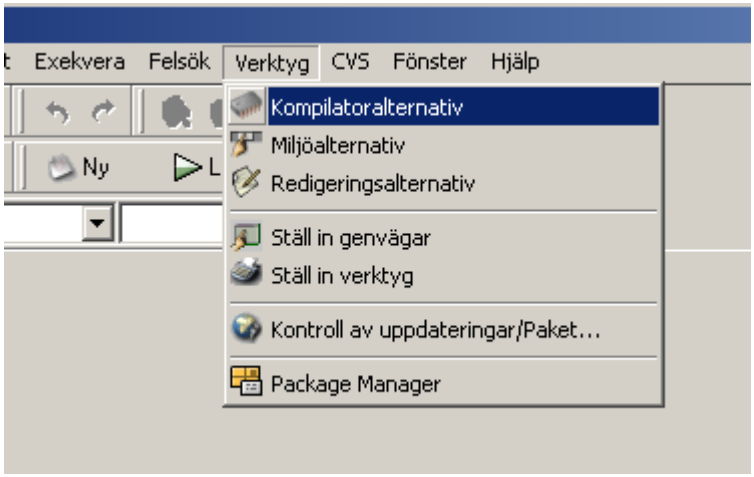
conio.h

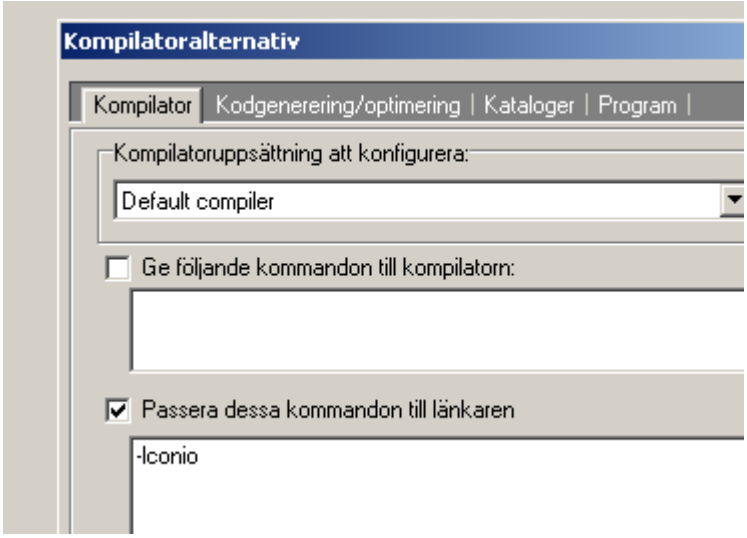
conio.h är ett Borland tillägg, ingen standard header. Det är ett mycket reducerat conio-bibliotek som medföljer utvecklingsmiljön Bloodshed Dev C++. Vill du använda fler funktioner finns windows egna **console functions** .

Det finns en implementation av conio för Dev-C++ och kompilatorn MinGW. Den är skriven av: Hongli Lai, T Korrovi, Andrew Westcott och Adrian Sandor. Den ger dig möjlighet att använda t.ex gotoxy(x, y) och delay(milliSek). På skolan finns detta bibliotek, förhoppningsvis, redan (VT-05) installerat men på din hemdator måste du sköta installationen själv.
Gör på följande sätt:
Se först till att du har den senaste versionen av DevC++ . Ladda hem den från följande adress: http://prdownloads.sourceforge.net/dev-cpp/devcpp4991setup.exe.
Följ sedan instruktionerna för att avinstallera och nyinstallera DevC++ på rätt sätt.
Ladda hem console_20041107, och spara den på lämpligt ställe. I katalogen hittar du två underkataloger include och lib. Kopiera de tre filerna i include-katalogen. Öppna sedan utforskaren och din Dev-Cpp-katalog



Öppna sedan include-katalogen och klistra in filerna där (välj skriv över). Gör sedan samma sak med innehållet i lib-katalogen. För att använda conio måste du sedan inkludera conio.h i din kälkodsfil #include <conio.h> och länka med –lconio (-litet L).





Sen ska det bara vara att åka.

Hur avinstallerar jag Dev-C++ korrekt?

Dev-C++ 4.0.x:
avinstallera från kontrollpanelen – lägg till och ta bort program, klart.
Dev-C++ 4.9.x.x:
Följ dessa instruktioner noggrant.
Kör först uninstall.exe som finns i katalogen Dev-cpp. Ta sedan bort denna katalog. Använd sedan sökfunktionen: (Start – Sök) och ta bort alla devcpp.* -filer som du finner. Du måste ange att sökningen skall inkludera även dolda filer. Viktigt är att du tar bort devcpp.cfg och devcpp.ini.

Hur installerar jag Dev-C++ korrekt?

Enklast är att låta installationsprogrammet ta hand om hela proceduren. Vill du specificera katalog för installationen så se till att katalogen inte har några mellanslag i sökvägen.
Exempel på **korrekta** sökvägar är : **C:\Dev-Cpp, D:\Dev-Cpp och** C:\progs\devcpp.
Exempel på **felaktig** sökväg: **C:\Program Files\Dev-Cpp, C:\Dev Cpp**

Kortkommandon i DEV C++

These commands control how the cursor behaves.

Command	Description
Left Arrow	Left one character
Right Arrow	Right one character

Up Arrow	Up one line
Down Arrow	Down one line
Control + Left	Left one word
Control + Right	Right one word
Home	Start of line
End	End of line
Page Up	Up one page
Page Down	Down one page
Left one page	
Right one page	
Control + Page Up	Top of page
Control + Page Down	Bottom of page
Control + Home	Abs begin
Control + End	Abs end

These commands control how the currently highlighted text behaves.

Command	Description
Shift + Left	Select left
Shift + Right	Select right
Shift + Up	Select up
Shift + Down	Select down
Control + Shift + Left	Select word left
Control + Shift + Right	Select word right
Shift + Home	Select line start
Shift + End	Select line end
Shift + Page Up	Select page up
Shift + Page Down	Select page down
	Select page left
	Select page right
Control + Shift + Page Up	Select page top
Control + Shift + Page Down	Select page bottom
Control + Shift + Home	Select editor top
Control + Shift + End	Select gotoxy
Control + A	Select All
Control + Insert	Copy select to clip
Control + C	

These commands control everything to do with scrolling.

Command	Description
Scroll Up + Control	Go up one line
Scroll Down + Control	Go down one line
Scroll Left	Left one character
Scroll Right	Right one character

These commands control modes

Command	Description
Set insert mode	
Set overwrite mode	

Insert	Toggle insert/overwrite
Control + Shift + N	Selection type is normal
Control + Shift + C	Selection type is column
Control + Shift + L	Selection type is line

Actions:

These commands perform various actions.

Command	Description
Control + Shift + B	Go to matching bracket
Control + (number)	Move to marker (number)
Control + Shift + (number)	Set marker (number)
F1	Context sensitive help on word

All the commands having to do with deleting.

Command	Description
Backspace	Character to left
Shift + Backspace	
Delete	Character to right
Control + T	Word to right
Control + Backspace	Word to left
From cursor to start of line	
Control + Shift + Y	From cursor to end of line
Control + Y	Current line

Everything in editor

Enter	Line break at current position, move caret
Shift + Enter	
Control + M	
Control + N	Line break at current position, no move caret

Insert character at curent position

Alt + Backspace	Perform undo if available
Control + Z	Perform undo if available
Alt + Shift + Back	Perform redo if available
Control + Shift + Z	Perform redo if available
Shift + Delete	Remove selection place on clipboard
Control + X	Remove selection place on clipboard
Shift + Insert	Move clipboard contents to current position
Control + V	Move clipboard contents to current position
Control + Shift + I	Move selection to right
Control + Shift + U	Move selection to left
Tab	Tab key
Shift + Tab	Tab to left

PREPROCESSORN

Alla kodrader som börjar med # är preprocessordirektiv.
Preprocessorn läser källkod, letar efter direktiv och makron, tar bort programkommentarer, blanksteg och tomma rader, tolkar koden och skriver ny källkod som sedan läses av kompilatorn.

#	Nulldirektiv – ingen åtgärd
# include	Infogar en källkodsfil vid direktivet
# define	Definierar ett makro
# undef	Tar bort makrodefinitionen
# if	Kompilerar koden om direktivet är sant
# ifdef	Om makrot har definierats kompileras koden
# ifndef	Om makrot inte har definierats kompileras koden
# endif	Avslutar block med #if och #else villkor.
# error	Avslutar kompileringen och visar ett felmeddelande,.

include

Inkluderingfiler kan infogas på två sätt.
 # include <iostream>
#include ”minFil.h”

Det första sättet innebär att preprocessorn söker efter inkluderingsfilen bland de inkluderingsfiler som ingår i kompilatorn.
Det andra sättet innebär att preprocessorn söker inkluderingsfilen i programmet som kompileras (i samma katalog som källkoden ligger).

Du kan undvika att inkluderingsfiler infogas flera ggr med hjälp av villkorskontroller i en inkluderingsfil.

```
//minFil.h
# ifndef    minFil.h          // Om makrot har definierats
#define     minFil.h          // Definierar ett makro
# endif                       // Avslutar block med #if och #else villkor.
```

#define

Direktivet ovan anger ett makro. Vanligtvis deklarerar direktivet en identifierare och lägger till kod som ersätter identifieraren när den förekommer i källkoden.

```
# define MAXVARDE  20;
int lista [MAXVARDE];
```

Här är MAXVARDE antalet element i en lista (array). Värdet kan användas på fler ställen i programmet.

Fler exempel:
#define MONITORHEIGHT 600
#define MONITORWIDTH 800

ANSI-C++

the way to include header files from the standard library has changed.
The standard specifies the following modification from the C way of including standard header files:

- Header file names no longer maintain the **.h** extension typical of the C language and of pre-standard C++ compilers, as in the case of **stdio.h**, **stdlib.h**, **iostream.h**, etc. This extension **h** simply disappears and files previously known as **iostream.h** become **iostream** (without **.h**).
- Header files that come from the C language now have to be preceded by a **c** character in order to distinguish them from the new C++ exclusive header files that have the same name. For example **stdio.h** becomes **cstdio** .
- All classes and functions defined in standard libraries are under the **std** *namespace* instead of being global. This not applies to C macros that remain as C macros.

Here you have a list of the standard C++ header files:
<algorithm> <bitset> <deque> <exception> <fstream> <functional> <iomanip> <ios> <iosfwd> <iostream> <istream> <iterator> <limits> <list> <locale> <map> <memory> <new> <numeric> <ostream> <queue> <set> <sstream> <stack> <stdexcept> <streambuf> <string> <typeinfo> <utility> <valarray> <vector>
And here is a list of the C header files included in ANSI-C++ with their new name and their equivalents in ANSI-C:

ANSI-C++	ANSI-C
<cassert>	<assert.h>
<cctype>	<ctype.h>
<cerrno>	<errno.h>
<cfloat>	<float.h>
<ciso646>	<iso646.h>
<climits>	<limits.h>
<clocale>	<locale.h>
<cmath>	<math.h>
<csetjmp>	<setjmp.h>
<csignal>	<signal.h>

<cstdarg>	<stdarg.h>
<cstddef>	<stddef.h>
<cstdio>	<stdio.h>
<cstdlib>	<stdlib.h>
<cstring>	<string.h>
<ctime>	<time.h>
<cwchar>	<wchar.h>
<cwtype>	<wtype.h>

Since now classes and functions of the standard libraries are located within the **std namespac**e we must use the C++ **using** directive for that these become usable in the same way they were in pre-standard code. For example, to be able to use all the standard classes of **iostream** we would have to include something similar to this:

```
#include <iostream>
using namespace std;
```

that would be equivalent to the old expression

```
#include <iostream.h>
```

previous to the standard.

Nevertheless for compatibility with ANSI-C, the use of **name.h** way to include C header files is allowed. Therefore, both following examples are valid and equivalent in a compiler which fulfills **ANSI-C++** specifications: