Enkla datatyper minne

143.56

”Sonja”

sant

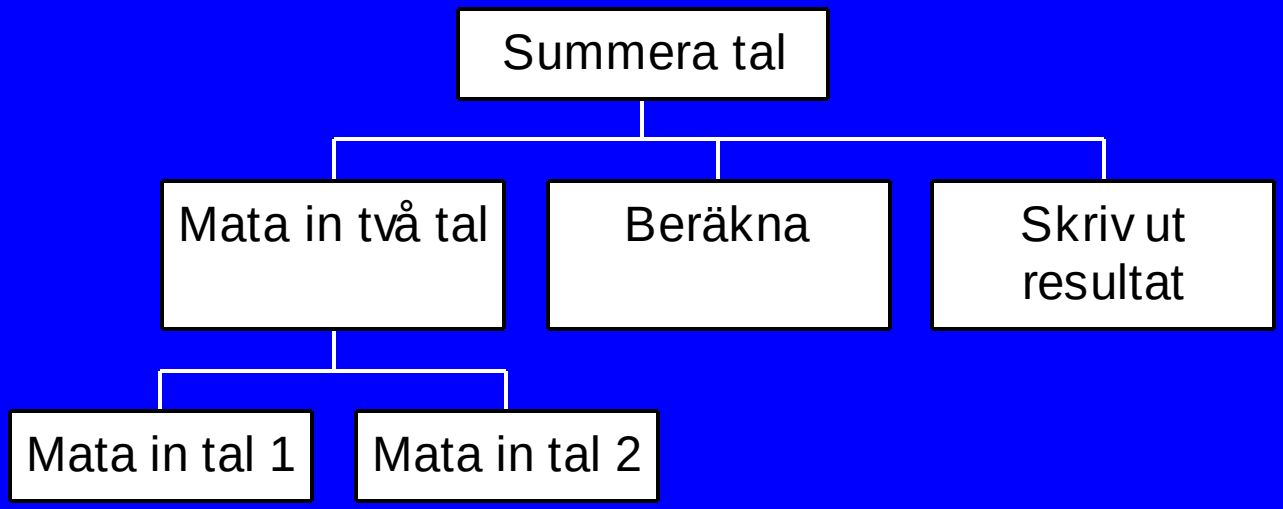
’A’

falskt

18

Addera två tal

Algoritmen



Addera två tal

Pseudokod

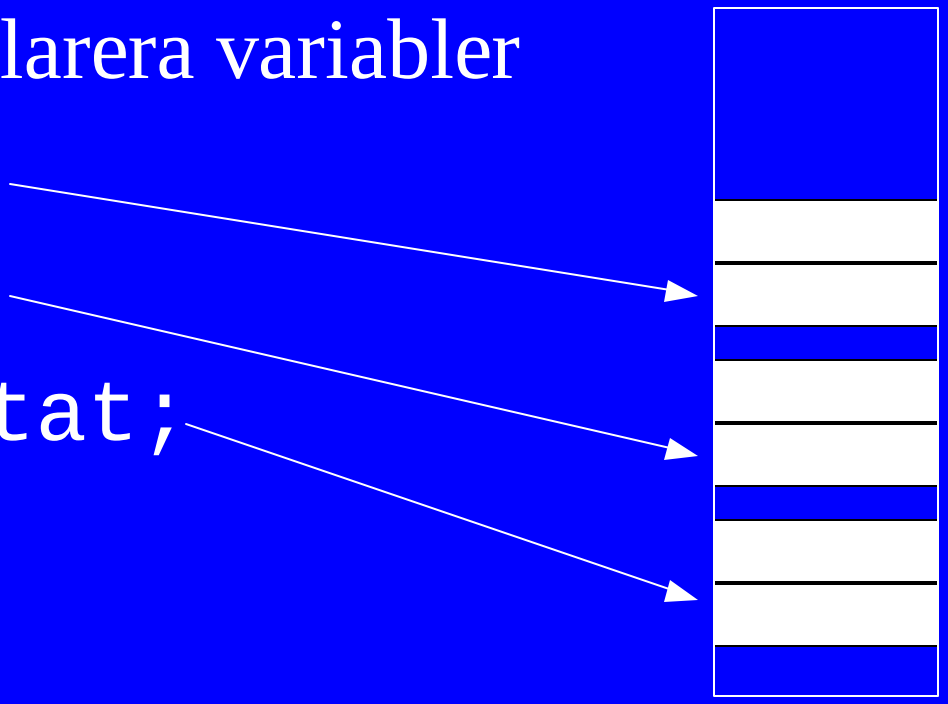
```
starta
   mata in tal
        mata in tal 1
        mata in tal 2
   beräkna summa
   skriv ut resultat
slut
```

Addera två tal - data

Reservera minnesutrymme

genom att deklarera variabler

```
int tal1;
int tal2;
int resultat;
```



Addera två tal

Kodning i C (++)

```
#include <iostream.h>
void main(void){
  // Deklaration av variabler
  int tal1, tal2, resultat;
  // Mata in tal
  // Beräkna summa
  // Skriv ut resultat
}
```

Headerfil

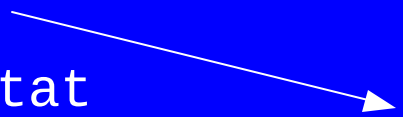
Kommentar

Addera två tal

```
#include <iostream.h>
void main(void){
   // Deklaration av variabler
   int tal1, tal2, resultat;
   // Mata in tal
   cin >> tal1;
   cin >> tal2;
   // Beräkna summa
   resultat = tal1 + tal2;
   // Skriv ut resultat
   cout << ”Resultat: ” << resultat;
}
```

Alternativ

```
#include <iostream.h>
void main(void){
   // Deklaration av variabler
   int tal1, tal2;
   // Mata in tal
   cout << ”Mata in två heltal -->”;
   cin >> tal1 >> tal2;
   // Beräkna summa
   // Skriv ut resultat
   cout << ”Resultat :” << ( tal1 + tal2 );
}
```



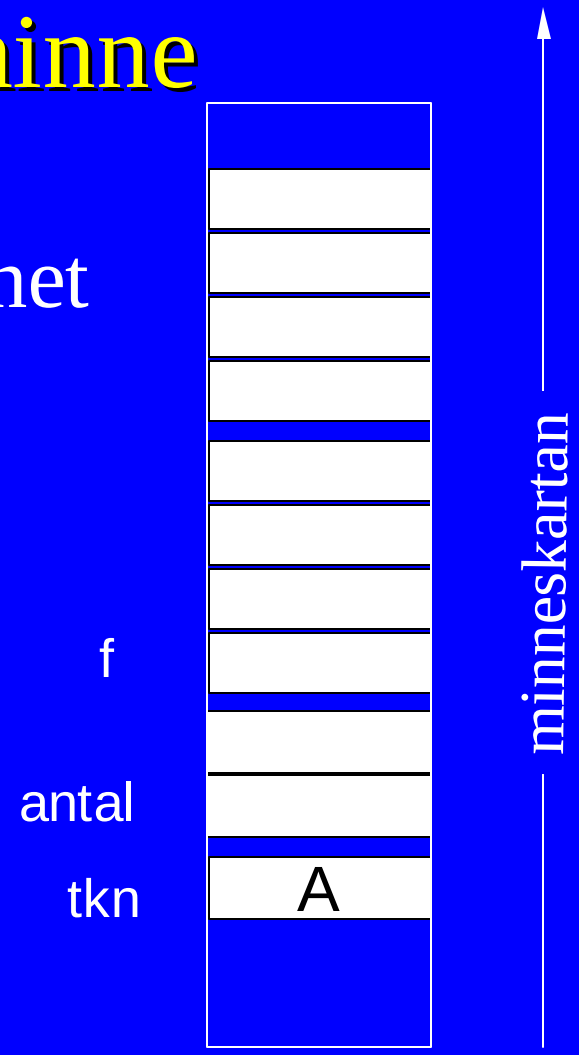
Datatyper - minne

Variabler = *dataobjekt* i minnet

```
char tkn  = 'A';
int antal = 20;
float f   = 19,7;
```

Olika datatyper

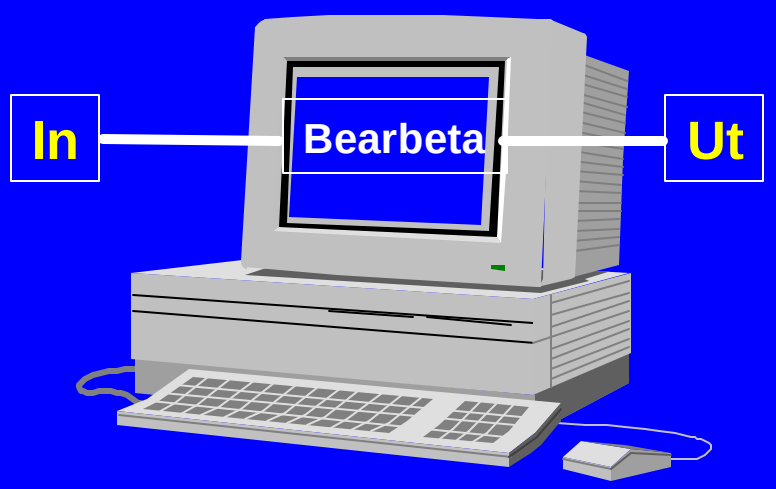
tar olika stor plats i minnet



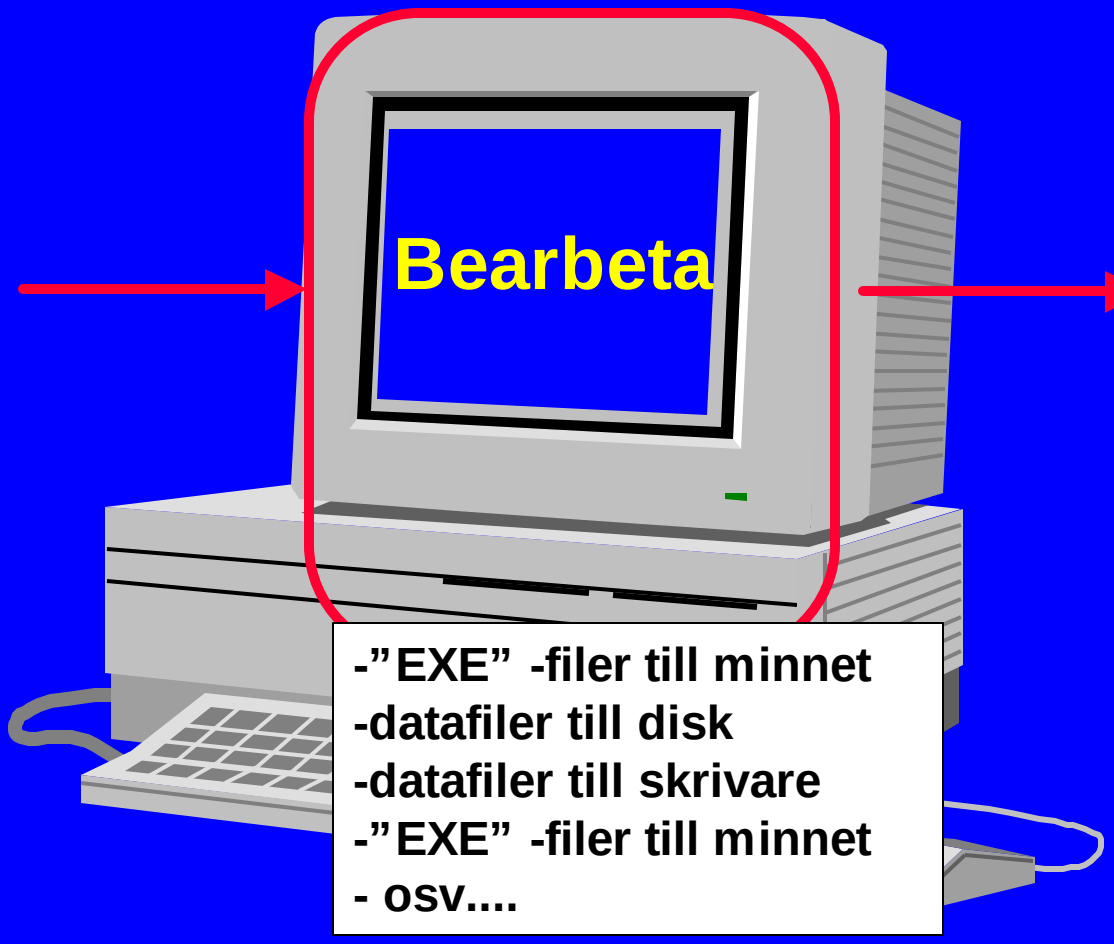
Datalagring i ANSI-C

Datorns funktion

- Data in
- Bearbeta data
- Data ut



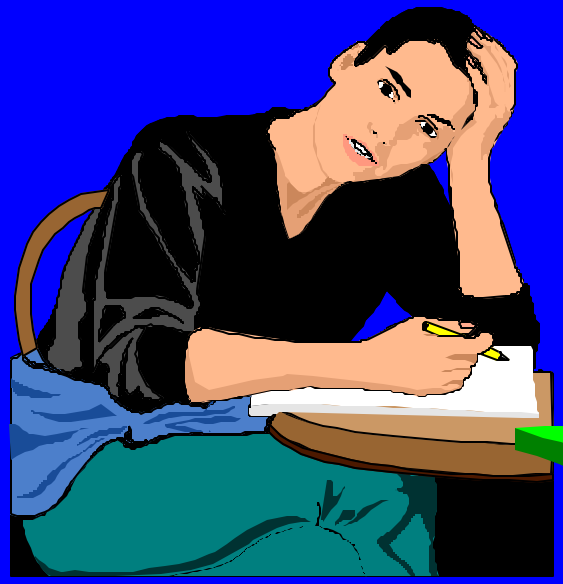
Operativsystemet arbetar med ”filer ”



Operativsystemet arbetar med ”filer ”

- Operativsystemet laddar en ”EXE-fil” som skall bearbeta data
- Denna ”EXE-fil” kan ha sitt ursprung i en C-källkodsfil
- Operativsystemet tillhandahåller en *infil* varifrån programmet *läser* data
- Operativsystemet tillhandahåller en *utfil* till vilken programmet *skriver* data

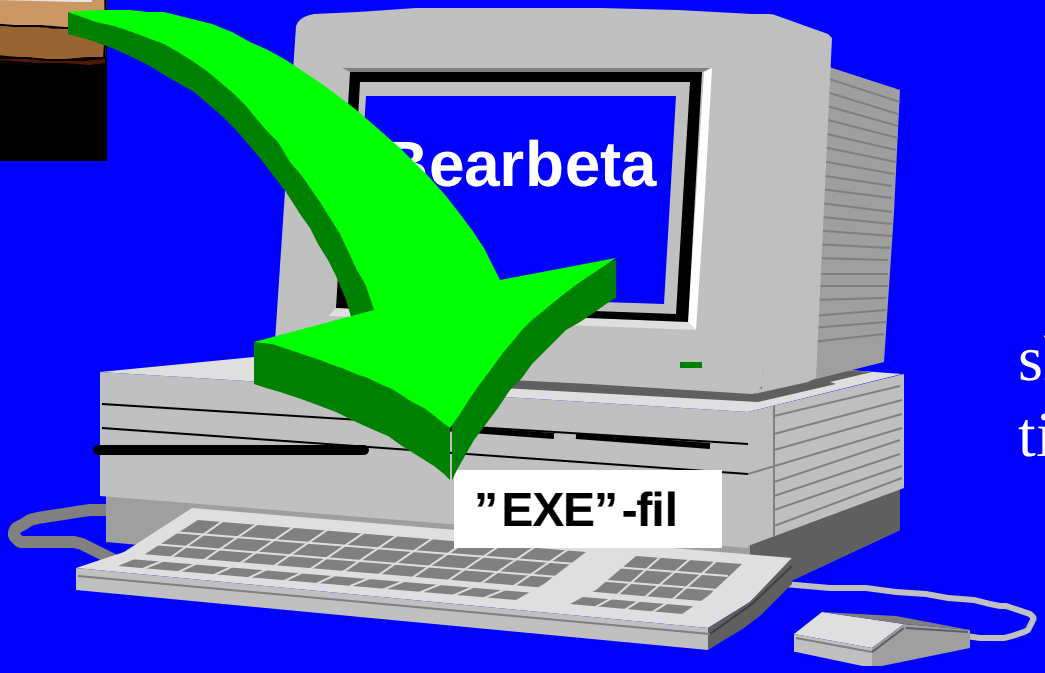
Hur fångar man upp data i ett C-program ?



?

källkod

läsa data
från infil



skriva data
till utfil

Skapa minne, *variabel*

Första steget blir att skapa, *deklarera* en *variabler*

```
/* Beräknar kapitaltillväxt på 10 år framåt eller bakåt */
#include <iostream.h>
void TabellPaSkarmen( int , float );
int main ( void  )
{
        float  kapital ;
        const float RANTA = 3.5;
        int    antalAr;
        cout << "Insatt kapital och antal år ?(-->1000 10)--> ";
        cin >> kapital, antalAr;
        TabellPaSkarmen( antalAr, kapital );

        return 0;
}
```

identifierare, namnet på variabeln

minnestyp

Namn på variabler - *identifierare*

Siffror, _ , bokstäver (gemena, versaler) går bra
Ej å ä ö ü..

Variabelnamn är signifikanta för
åtminstone de första 31 tecknen

C skiljer på versaler och gemena,
`Kapital` och **`kapital`** är inte samma variabel

Reserverade ord (nyckelord) och får inte användas

Reserverade ord

Reserverade ord kan inte användas som identifierare!

`auto`	`break`	`case`	`char`	`const`
`continue`	`default`	`do`	`double`	`else`
`enum`	`extern`	`float`	`for`	`goto`
`if`	`int`	`long`	`register`	`return`
`short`	`signed`	`sizeof`	`static`	`struct`
`switch`	`typedef`	`union`	`unsigned`	`void`
`volatile`	`while`			

Variabeltyper

- beror på vad man tänker lagra

heltalstyper

- teckenlösa heltal (2 bytes lagrar talen [0, 65535])
- heltal med tecken (2 bytes lagrar talen [-32768, 32767])
- tecken i texter (ASCII-koden i 1 byte: $1000001_B = 65_D$ = 'A')
- logiska värden, ***boolean*** (= 0 falskt , allt annat är sant)
- bitmönster (10011101 varje bit betyder något t ex tänd eller släckt lampa)

det lagrade värdet är exakt

flyttalstyper

- reella talen (t.ex. 3.56E-10 = $3.56 \cdot 10^{-10}$)

det lagrade värdet är oftast en approximation

Variabeltyper

heltalstyper i ANSI-C

[...] betyder, ”kan utlämnas”

– `signed char`	minst 1 byte
– `unsigned char`	minst 1byte
– `[signed] short [int]`	minst 2 byte
– `unsigned short [int]`	minst 2 byte
– `[signed] int`	minst 2 byte
– `unsigned [int]`	minst 2 byte
– `[signed] long [int]`	minst 4 byte
– `unsigned long [int]`	minst 4 byte

unsigned ger teckenlöst heltal som är bra för att lagra ”stora” positiva heltal och bitmönster


Variabeltyper

heltalstyper i ANSI-C

Hur lagrar man ett tecken, t. ex. A ?

- det man lagrar är tecknets ASCII-kod som stoppas in i en variabel, minne av *heltalstypen* char (charcter = tecken).
- ASCII, *American Standard Code for Information Interchange*

definierar variabel asciiTecken

```
char   asciiTecken = 'A';
```

asciiTecken

1	0	0	0	0	0	0	1

1 byte i minnet

Teckenlagring - ASCII

ASCII-Tabell hexadecimal kod, styrtecken ej utskrivna

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	•
80	€	•	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	•	Ž	•
90	•	‘	’	“	”	•	–	—	˜	™	š	›	œ	•	ž	Ÿ
A0		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B0	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

å Å ä Ä ö Ö

OBS, ordningen!

Det finns flera standards för dessa (ej standard?)

Vad händer?

```
char tecken;
tecken = 'Q';
tecken = tecken + 2;
cout << tecken;
```

Vad skrivs ut?

Variabeltyper

Flyttalstyper i ANSI-C, fördefinierade

- **`float`** ? storleken är implementationsberoende, testa programmet nedan
- **`double`** minst lika stor som float
- **`long double`** minst lika stor som double

används för att lagra de reella talen, noggrannheten är ej standard men skall motsvara minst 6 siffror i området [10^{-37}, 10^{+37}]

```
TabellPaSkarmen( int , float );

int main ( void )
{
        float       kapital ;
        int         antalAr;
        printf("Insatt kapital och
        scanf("%f%d", &kapital,
        TabellPaSkarmen( antalA
```


FLOATSIZ.EXE


FLOAT.EXE


DOUBLE.EXE


L_DOUBLE.EXE

Fånga data

från tangentbordet

Tangentbordet (CON) är en textfil som genererar ASCII-koder
Det behövs funktioner

- som *konverterar (tolkar)* den text som läses in
- och som *lagrar* det tolkade datat i en *variabel* (minne).

Funktioner som klarar detta ligger i **`cin`**

Fånga data

från tangentbordet


```
cin >> ettHeltal;
```

konvertera, tolka

lagra, fånga

texten: 0XFF
eller
talet: 255 ?

variabel

?	?	?	?	?	?	?	?
?	?	?	?	?	?	?	?
?	?	?	?	?	?	?	?

primärminne

Skriva data

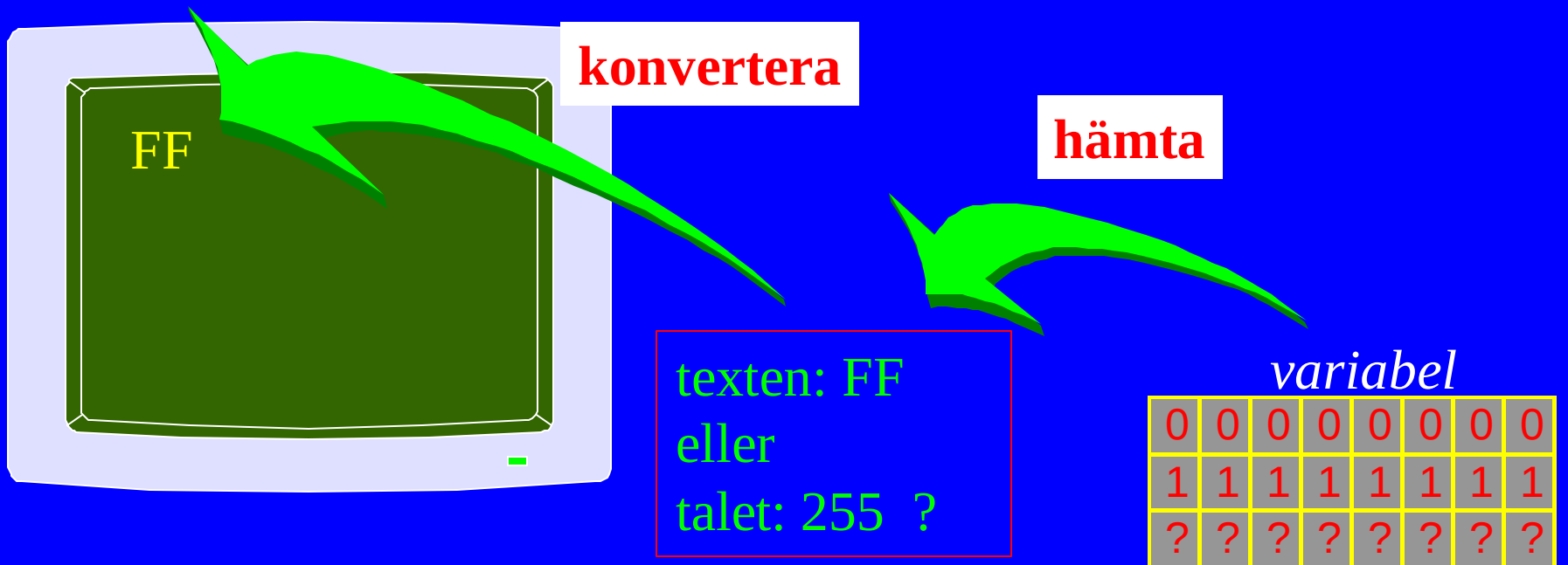
till bildskärm

Eftersom bildskärmen (CON) är en textfil (i detta fall) som tar ASCII-koder så behöves funktioner

som *konverterar* binärdata

och som hämtar datat från rätt variabel (minne).

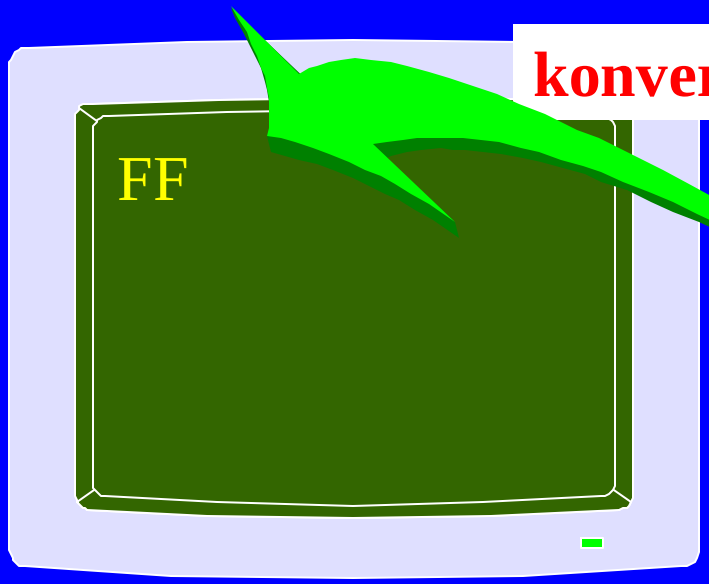
Funktioner som gör detta ligger i **`cout`**



Skriva data

till bildskärm

```
cout << ettHeltal;
```



konvertera

hämta

texten: FF
eller
talet: 255 ?

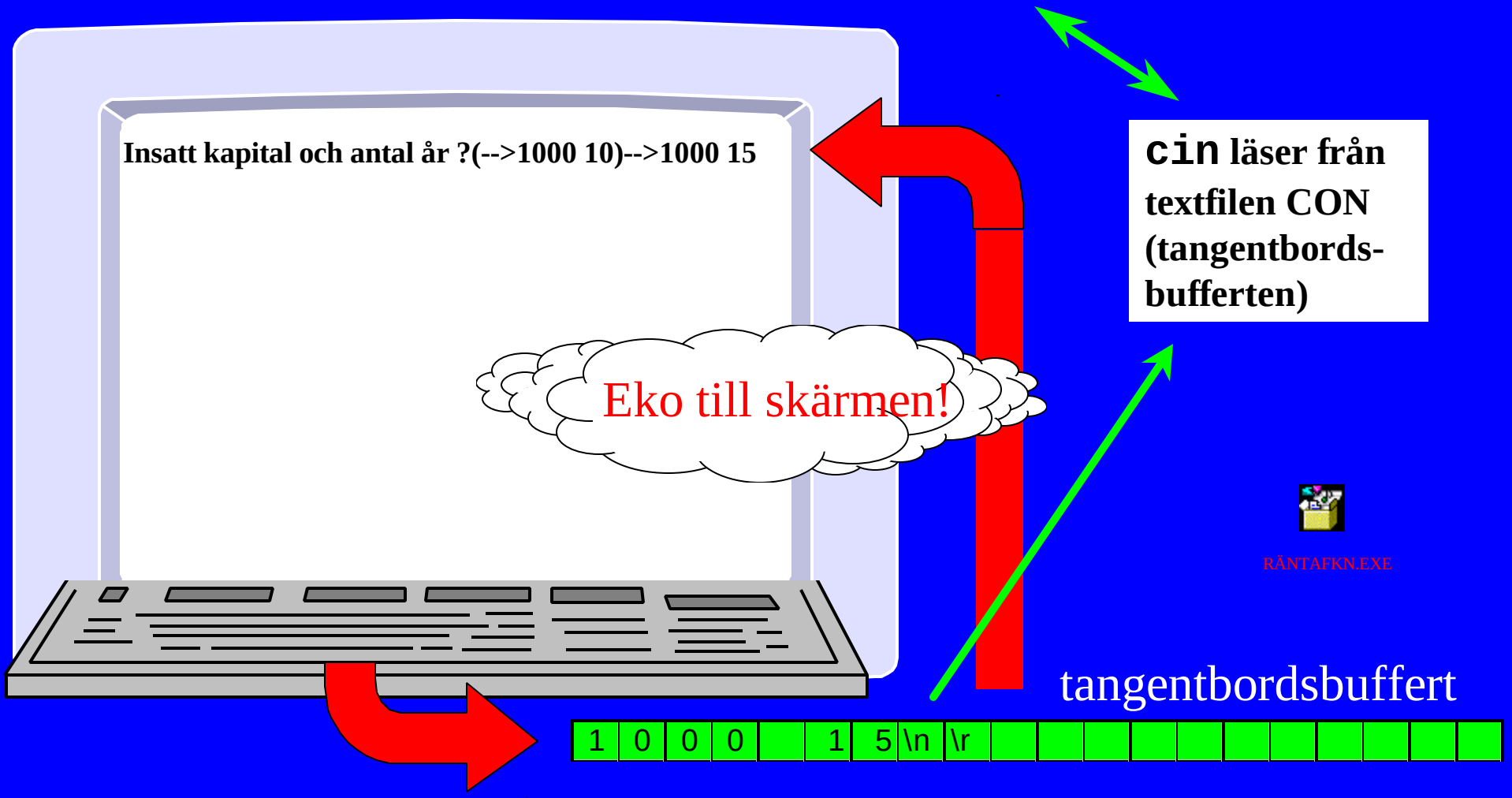
variabel

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
?	?	?	?	?	?	?	?

primärminne

`cin` *vad händer i detalj..*

```
cin >> kapital >> antalAr;
```



Sammanfattning

Variabler måste deklareras

- är namn (identifierare) på dataobjekt i minnet
- variabeltyp väljs efter vad som skall lagras
- enkla typer för heltal, tecken och flyttal

Dataobjekt måste *tolkas* som datatyp

- i minnet
- vid läsning och skrivning från/till filer

Sammanfattning

Heltalstyper

- finns för olika stora tal
- kan tolkas med och utan tecken

Flyttalstyper

- finns för olika precision och storlek
- olika i olika operativsystem - testa!

Sammanfattning

Teckentyper

- är heltal om en byte
- tolkas normalt som tecken
- 7 bitars ASCII är ’normal’ standard
- många standards finns för 8 bit
- Nytt är UNICODE 16 bit character set

Det finns *många* standards för teckenkodning

character encoding

Nästa gång

- Mer om enkla datatyper
- Operationer på data
- Typomvandling - hur farligt är det ?
- Automatisk typomvandling
- Läsbar och självdokumenterande kod

Öva - Testa!

Guruns bästa tips:

Om du undrar hur något fungerar i C

Skriv ett miniprogram!