Velkommen til INF2100

Jeg er *Dag Langmyhr* (dag@ifi.uio.no).

Dagens tema:

- Hva går kurset ut på?
  - Bakgrunn for kurset
  - Hvordan gjennomføres kurset?
  - Hvordan får man det godkjent?
- Pause (med registrering av fremmøte)
- Programmeringsspråket RusC
  - En kort oversikt
  - Syntaks
  - Fire eksempler

Bakgrunnen for INF2100

I INF1000–10 har dere lært å programmere, men bare små programmer (< 1000 linjer).

Hensikten med INF2100 er

- å gi mer programmeringstrening
- forstå mekanismene man trenger i større programmer (som objektorientering og moduler)

Prosjektet

Prosjektet er valgt fordi det har en nytteverdi i seg selv:

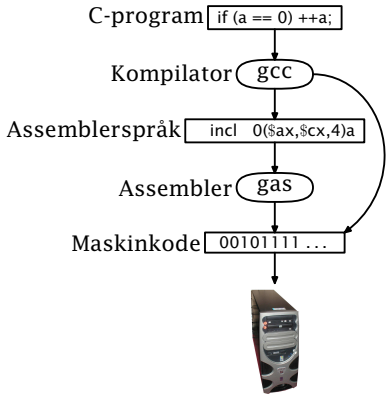
Skriv en kompilator for programmeringsspråket RusC.

Dette gir

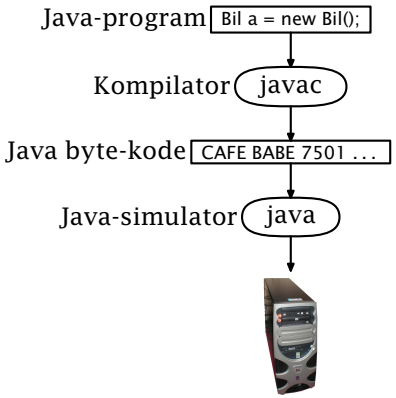
- forståelse for hvordan en kompilator fungerer
- kjennskap til maskin- og assemblerspråk
- hvordan et programmeringsspråk er definert og bygget opp

Hva gjør en kompilator?

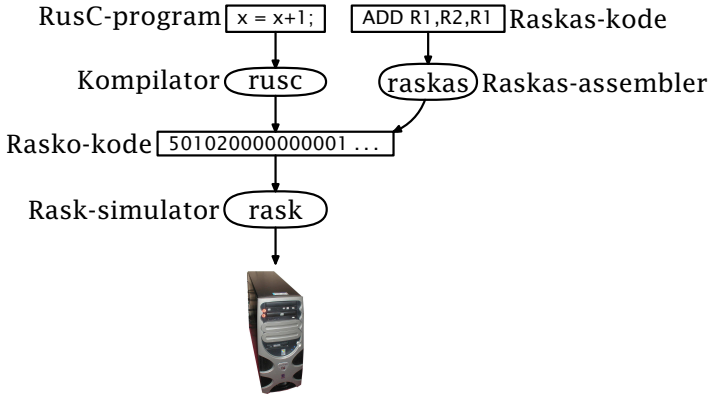
En kompilator oversetter et program til en annen kode, oftest maskin- eller assemblerkode.



Noen kompilatorer oversetter til en intern kode som *tolkes* av en interpret. (Det finnes også Java-kompilatorer som lager maskinkode, men de er ikke så vanlige.)



Opplegget for RusC ligner mye på Java-systemet.



Oppgaven *deres* er å skrive denne kompilatoren.

Ulike programmeringsspråk

I dette kurset skal vi innom flere språk:

- **Java** benyttes til all implementasjon.
- **RusC** er målet for prosjektet.
- **Rasko** er maskinkoden til datamaskinen vår *Rask*.
- **Raskas** er assemblerspråket til Rasko.

Oppbyggingen av kurset

Kurset har fem hovedkomponenter:

Forelesningnen holdes i utgangspunktet nesten hver uke.

Gruppeøvelsene er 2 timer hver uke.

Gruppearbeid for å løse oppgavene. Man jobber to og to (eller alene om man vil). Man kan velge partner selv, eller la gruppelæreren plukke ut par.

Kompendiet gir en grundig innføring i Minila og prosjektet.

Nettsidene er også en viktig informasjonskanal.

Godkjenning av kurset

Kurset har ikke karakterer, bare bestått/ikke bestått.

Det er tre obligatoriske oppgaver. Når de er godkjent, er kurset godkjent.

Men . . .

Mot slutten av semesteret vil noen bli plukket ut til en samtale om programmet de har laget. Dette kan man stryke på.

Siden man normalt jobber i lag, forventes at

- begge har kjennskap til hovedstrukturen i programmet
- begge kan identifisere sin del av programmet (som skal være rundt halvparten) der de kan forklare nøye hvorfor koden er blitt slik den er.

Samarbeid og fusk

Samarbeid og utveksling av ideer er bra!

Kopiering og fusk er ikke bra!

Gode råd for samarbeid

- Snakk gjerne med andre om ideer.
- Kopier aldri andres kode.

Programmeringsspråket RusC

Dette språket («Rudimentary simple C») er en meget forenklet utgave av C spesiallaget til dette kurset.

p1.rusc

```
func main ()
{
   putchar('!');   putchar(10);
}
```

p1.rusc

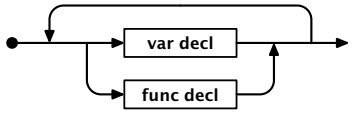
Programmet kompileres og kjøres med kommandoene

```
> rusc p1.rusc
> rask p1.rask
!
```


Grammatikk

Vi kan beskrive grammatikken til RusC med *jernbanediagrammer*:

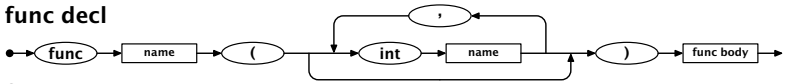
program



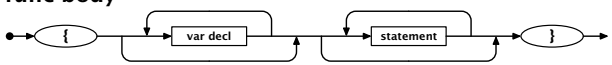
Et program er bare (som i C) en samling deklarasjoner av variable og funksjoner.

Funksjoner deklareres som i C og Java (men kan ha maksimalt fire parametre):

func decl



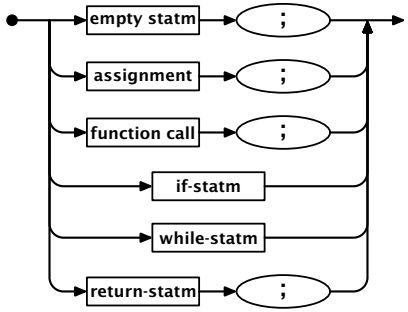
func body



Funksjonen main er hovedprogrammet.

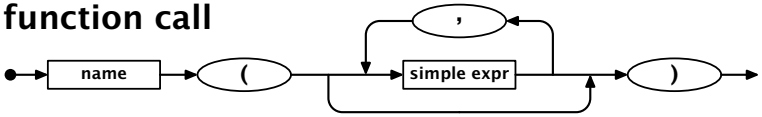
RusC har færre setninger enn C og Java:

statement



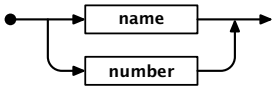
Funksjonskall er som forventet

function call



men parametrene kan bare være ganske enkle:

simple expr



Grammatikken forteller om et program er korrekt når det gjelder oppsettet (men det kan allikevel ha logiske feil).

Dette programmet er ikke korrekt:

```
func main ()
{
   putchar('?');
   putint(getint() + 1);   putchar(10);
}
```

Kjøring gir

```
> rusc p1b.rusc
RusC error in line 4: Calling function getint is illegal here!
```


p2.rusc

```
func main ()
{
   int c, kind;

   putchar('?');  c = getchar();  /* Read a letter. */
   if (c >= 'a') {
      /* Convert to uppercase (if required) */
      c = c - 32;
   }

   kind = 'C';  /* Initially, assume a consonant. */
   if (c == 'A') { kind = 'V'; }
   else { if (c == 'E') { kind = 'V'; }
      else { if (c == 'I') { kind = 'V'; }
         else { if (c == 'O') { kind = 'V'; }
              else { if (c == 'U') { kind = 'V'; }}}}}

   putchar(kind);   putchar(10);   exit(0);
}
```



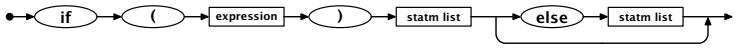
```
> rusc p2.rusc
> rask p2.rask
?f
C
```

Programmet leser en bokstav og avgjør om det en vokal eller konsonant.

If-setninger

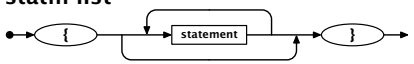
If-setninger tester på 0 (*usant*) og ikke-0 (*sant*).

if-statm

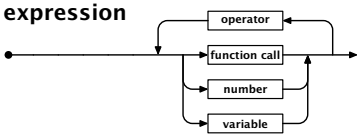


Legg merke til at det alltid skal stå krøller rundt alternativene:

statm list



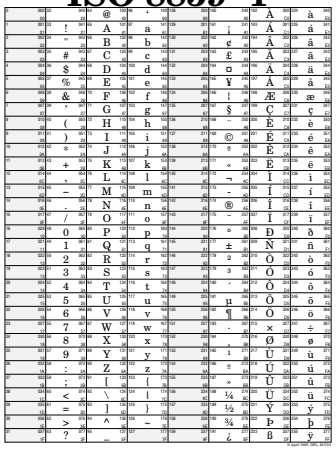
expression



- Lovlige operatorer er de fire regneartene og seks sammenligningene.
- De har samme *presedens*.
- Vi har ikke parenteser i uttrykkene.

Derfor må vi bruke flere variable enn vi er vant til.

ISO 8859–1



p3.rusc

```
int LF;

func gcd (int a, int b)
{
   while (a != b) {
      if (a < b) {
         b = b-a;
      } else {
         a = a-b;
      }
   }
   return a;
}
```

```
func main ()
{
   int v1, v2, res;

   LF = 10;
   putchar('?');
   v1 = getint();   v2 = getint();
   res = gcd(v1,v2);
   putint(res);  putchar(LF);
}
```

Programmet finner *største felles divisor*:

```
> rusc p3.rusc
> rask p2.rask
52  221
13
```

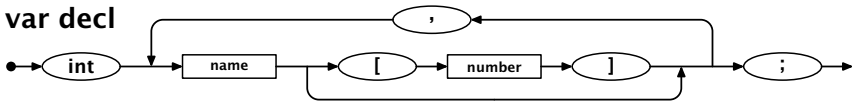

Biblioteket

Rusc kjenner til disse fem funksjonene:

exit(*status*)	Avslutter programmet
getchar()	Leser et tegn fra tastaturet
getint()	Leser et heltall fra tastaturet
putchar(*n*)	Skriver ut et tegn på skjermen
putint(*n*)	Skriver ut et heltall på skjermen

Vektorer

var decl




```
int a [3];
```

deklarerer a med elementene a[0], a[1] og a[2]. Det er ingen sjekk på grensene.

primes.rusc, del 1

```
/* Program ‘primes´
   ----------------
   Finds all prime numbers up to 1000 (using the technique called
   "the sieve of Eratosthenes") and prints them nicely formatted.
*/

int prime[1001];  /* The sieve */
int LF;           /* LF */
```

primes.rusc, del 2

```
func find_primes ()
{
   /* Remove all non-primes from the sieve: */

   int i1, i2;

   i1 = 2;
   while (i1 <= 1000) {
      i2 = 2*i1;
      while (i2 <= 1000) {
         prime[i2] = 0;  i2 = i2+i1;
      }
      i1 = i1+1;
   }
}
```

primes.rusc, del 3

```
func mod (int a, int b)
{
   /* Computes a%b. */

   int ax;

   ax = a/b;  ax = ax*b;
   return a - ax;
}
```

primes.rusc, del 4

```
func p3 (int v)
{
   /* Does a ‘printf("%3d", v)´;
      assumes 0<=v<=999. */

   if (v <= 9) {
      putchar(’ ’);   putchar(’ ’);
   } else {
      if (v <= 99) {
         putchar(’ ’);
      };
   }
   putint(v);
}
```

primes.rusc, del 5

```
func print_primes ()
{
   /* Print the primes, 10 on each line. */

   int n_printed, i;

   n_printed = 0;  i = 1;
   while (i <= 1000) {
      if (prime[i]) {
         if (mod(n_printed,10) == 0 * n_printed) {
            putchar(LF);
         }
         putchar(’ ’);   p3(i);   n_printed = n_printed+1;
      }
      i = i+1;
   }
   putchar(LF);
}
```

primes.rusc, del 6

```
func main ()
{
   int i;

   LF = 10;
   /* Initialize the sieve by assuming all numbers >1 to be primes: */
   prime[1] = 0;
   i = 2;
   while (i <= 1000) {
      prime[i] = 1;  i = i+1;
   }

   /* Find and print the primes: */
   find_primes();  print_primes();
}
```



```
> rusc primes.rusc
> rask primes.rask
   2   3   5   7  11  13  17  19  23  29
  31  37  41  43  47  53  59  61  67  71
  73  79  83  89  97 101 103 107 109 113
 127 131 137 139 149 151 157 163 167 173
 179 181 191 193 197 199 211 223 227 229
 233 239 241 251 257 263 269 271 277 281
 283 293 307 311 313 317 331 337 347 349
 353 359 367 373 379 383 389 397 401 409
 419 421 431 433 439 443 449 457 461 463
 467 479 487 491 499 503 509 521 523 541
 547 557 563 569 571 577 587 593 599 601
 607 613 617 619 631 641 643 647 653 659
 661 673 677 683 691 701 709 719 727 733
 739 743 751 757 761 769 773 787 797 809
 811 821 823 827 829 839 853 857 859 863
 877 881 883 887 907 911 919 929 937 941
 947 953 967 971 977 983 991 997
```


Oppsummering

RusC er et meget enkelt programmeringsspråk.

- Det bør være enkelt å lære.
- Ikke alt man forventer, er lov. Sjekk grammatikken!