

T-106.3101 Ohjelmoinnin jatkokurssi T2 (6 op)

Luento 1: Kurssin järjestelyt ja johdanto C-kieleen

Sami Liedes

17. tammikuuta 2008
(päivitetty 22. tammikuuta 2008)

Goals of the course

The goal of the course is to give the student

- ▶ An understanding of the differences between procedural (*lausekielinen*) and object oriented (*oliopohjainen*) programming
- ▶ Good practical C language programming skills
- ▶ An understanding on what happens internally when a C program is executed
- ▶ The skills needed to use common programming tools in the Unix environment

Goals of the course, continued

You will also learn skills needed when working on a larger software project and software development in a team:

- ▶ Version control
- ▶ Testing
- ▶ Division of work when subtasks in some way overlap

The lectures will be given in Finnish (after this introductory part). All course assistants will be able to serve you in English.

Course personnel

Lecturer:	Sami Liedes
Head assistant:	Anssi Kolehmainen
Assistants:	Tatu Kilappa
	Vesa Linja-aho
	Visa Putkinen
Responsible professor:	Lauri Malmi

Information channels 1/3

Official announcements:

- ▶ Course web page
 - ▶ <http://www.cs.hut.fi/Opinnot/T-106.3101/K2008/English/>
 - ▶ <http://www.cs.hut.fi/Opinnot/T-106.3101/K2008/>
- ▶ News group opinnot.tik.t2 (both for announcements and questions)
- ▶ All announcements will be posted *both* to the web page and the news group.

Information channels 2/3

Electronic channels for your questions:

- ▶ News group
- ▶ Course e-mail address: t1063101@cs.hut.fi
- ▶ For generic questions, ask in the news group
- ▶ If your question is such that no-one else would learn anything from the answer, use the course e-mail address
- ▶ Generally, expect a faster reply to posts in the news group

Information channels 3/3

IRC channel: #jatko2 (IRCNet)

- ▶ Unofficial channel, however in practice usually quick help is available there
- ▶ IRC = real time textual chat
- ▶ On a Computing Centre or Niksula computer, type `irssi`
- ▶ In `irssi`, type
`/connect irc.cs.hut.fi`
`/join #jatko2`
- ▶ Type `/quit` to end

Passing the course

To pass the course, you will have to pass

- ▶ Programming exercises (total 3000 points) in the 3rd period
 - ▶ Enrol in [WebTopi](#) to get a user account to the [Goblin](#) home exercise system.
 - ▶ You also need a Niksula user account
- ▶ A 3 (max 4) person project (total 5000 points) in the 4th period
 - ▶ Consists of testing and extending a small WWW game
 - ▶ In the project, everyone needs to do their own share
 - ▶ The project cannot be done in smaller groups

Passing the course (2)

- ▶ No exam

- ▶ Final grade:

if $E < 1500$ or $P < 2500$ then 0

else $\min(5, \text{trunc}((E+P-4000) / 750) + 1)$

where

E = points from exercises and

P = points from project

Passing the course (3)

Additionally, it is possible to take the course T-106.3105
Programming Exercise Project (2 cr) during this course

- ▶ Voluntary
- ▶ Small exercises in the Perl language
- ▶ Goal: To understand when using a scripting language is a better solution than C programming, and to learn the basics of Perl scripting
- ▶ Will probably only available in Finnish
- ▶ More information will be given later

Prerequisites & Corresponding courses

- ▶ Prerequisites:
 - ▶ T-106.1200/1203/1206 Basics of Programming T/L/Y
 - ▶ T-106.1240/1243 Intermediate course in Programming T1/L1
 - ▶ T-106.1220 Data Structures and Algorithms T
 - ▶ You *will* find this course difficult if you don't know about linked lists and trees
- ▶ Corresponding courses:
 - ▶ AS-0.1101 C Programming (4 cr) – close enough that you cannot include both in your degree
 - ▶ This course can be replaced by taking *both* AS-0.1101 and AS-0.1102 C++ Programming (4 cr)

Course literature

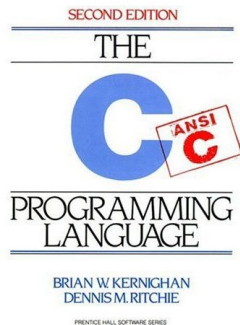
- ▶ Probably possible to do without a book if you take the lectures
- ▶ A book for self-study and a good reference/handbook:

Kernighan, Ritchie

The C Programming Language (2nd Edition)

Prentice Hall, 1988

ISBN 0-13-110362-8 (paperback)



Proseduraalinen ohjelmointi

Ohjelman rakenne jäsennetään toiminnoista käsin:

- ▶ Keskeistä "mitä tehdään"
- ▶ Kuin Java, jossa käytetään vain staattisia metodeja?

Proseduraalisissa kielissä (C) ei ole olioita:

- ▶ *tietue* = kuin luokka ilman metodeita ja private-kenttiä

Oliopohjainen ohjelmointi

Ohjelman rakenne jäsennetään datasta käsin:

- ▶ Keskeistä "mille tehdään"
- ▶ Käsiteltävä tieto jäsennetään luokkina
- ▶ Tieto tallennetaan olioihin
- ▶ Metodit kertovat, miten tietoa käsitellään
- ▶ Perintä mahdollistaa analogisten rakenteiden luontevan kuvauksen

Proseduraalisen ohjelman suunnittelu

Keskeisenä asiana, miten ohjelma jaetaan toiminnallisiin osiin

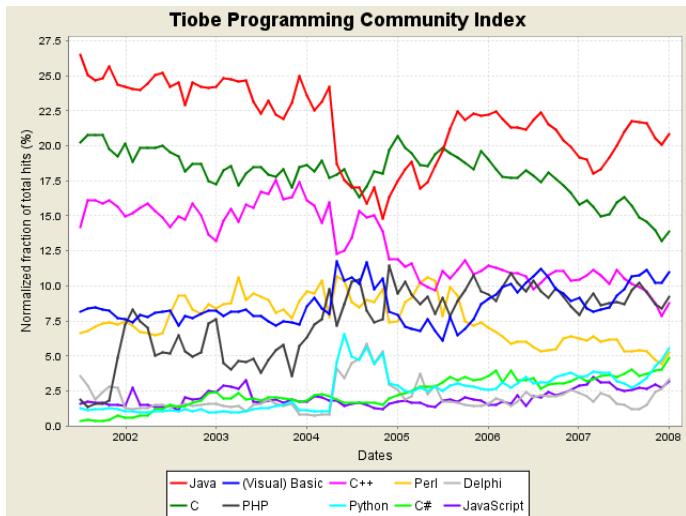
- ▶ Top-down-suunnittelussa toiminta jaetaan osiin, joita edelleen tarkennetaan pienempinä osina: Auton kori, moottori, jarrut.
- ▶ Bottom-up-suunnittelussa pohditaan aluksi sitä, millaisia ”osasia” ohjelman toteuttamisessa tarvitaan: Mutterit, pultit, laakerit, akselit.
- ▶ Käytännössä molempia tarvitaan

C-kielen historia

- ▶ Bell Labs, 1972
 - ▶ Ken Thompson (B)
 - ▶ Dennis Ritchie ja Brian Kernighan (K&R C)
- ▶ Ennen C:tä Unix oli kirjoitettu assemblyllä
 - ▶ 1973 suurin osa Unixin ytimestä oli uudelleenkirjoitettu C:llä
- ▶ Suosittu erityisesti systeemiohjelmoinnissa
 - ▶ Laitteistoläheinen koodi: Käyttöjärjestelmät, laiteajurit, sulautetut ohjelmistot, ns. firmware
- ▶ K&R C → ANSI C (C89) \approx ISO C (C90) → ISO C99



Kielen suosio



Lähde: <http://www.tiobe.com/>

C-kielen ominaispiirteitä

- ▶ Mahdollisuus kirjoittaa kompaktia koodia
- ▶ Laiteriippumaton
 - ▶ mutta mahdollista tehdä hyvin laiteriippuvaista koodia
- ▶ Modulaarinen ohjelmointi vaatii hyvää kuria
- ▶ Melko matalan tason kieli
 - ▶ Vain vähän rajoitteita ohjelmoijan vapaudelle
 - ▶ Kokenut ohjelmoija voi käyttää hyväkseen kielen suomia vapauksia
 - ▶ Toisaalta enemmän mahdollisuuksia tehdä virheitä kuin vaikkapa Javalla

Esimerkkiohjelman

```
1  /* standardikirjaston otsikoita, funktioiden prototyypit */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define SIZE 10 /* makro */
6
7  int main(void)
8  {
9      int i, count = 0, array[SIZE] = {0};
10     printf("Anna %d kokonaislukua\n", SIZE);
11     for (i = 0; i < SIZE; i++)
12         scanf("%d", &array[i]);
13     printf("Positiivisia lukuja ovat\n");
14     for (i = 0; i < SIZE; i++)
15         if (array[i] > 0) {
16             printf("%d ", array[i]);
17             count++;
18         }
19     printf("\nPositiivisia lukuja oli %d kpl\n", count);
20     return EXIT_SUCCESS;
21 }
```

Esimerkkiohjelma 2

```

1 #include <stdio.h>
2 main(t,_,a) char *a;
3 {return!0<t?t<3?main(-79,-13,a+main(-87,1-_,
4 main(-86,0,a+1)+a)):1,t<_?main(t+1,_,a):3,main(-94,-27+t,a
5 )&&t==2?_<13?main(2,_,+1,"%s %d %d\n"):9:16:t<0?t<-72?main(
6 t,"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{**+,/w{%+,/w#q#n+,/#{L,+,/n{n+\\
7 ,/+#n+,/#;#q#n+,/+k#;*,/'r : 'd*'3,}{w+k w'K:'+}e#';dq#'l q#'+d'K#!/\
8 +k#;q#'r}eKK#}w'r}eKK{nL}'/#;#q#n')}{#}w')}{nL}'/+#n';d}rw' i;# )n\
9 l)!/n{n#'; r#{w'r nc{nL}'/#{L,+ 'K {rw' iK{;[{nL}'/w#q#\
10 n'wk nw' iwk{KK{nL}!/w{% 'l##w# ' i; :{nL}'/*{q#'ld;r'}{nLwb!/*de}'c \
11 ;;{nL}'-{}rw}'/+,}##*}#nc,' ,#nw}'/+kd'+e}+;\
12 #'rdq#w! nr'/ ') }+}{rL#'{n' ')# }'+}##(!!/")
13 :t<-50?_==*a?putchar(a[31]):main(-65,_,a+1):main((*a=='')+t,_,a\
14 +1):0<t?main(2,2,"%s"):a=='/'||main(0,main(-61,*a,"!ek;dc \
15 i@bK'(q)-[w]*%n+r3#l,{ }:\nuwloca-0;m .vpbks,fxntdCeghiry"),a+1);}

```

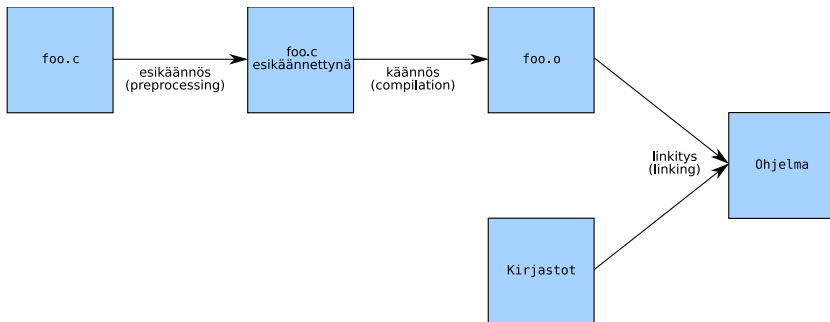
Lähde: <http://www.cs.cf.ac.uk/Dave/C/node4.html>

Ohjelman kääntäminen

- ▶ `gcc -g -Wall -ansi ohjelma.c`
- ▶ Mikäli lähdekooditiedostoja on useampia, ne käännetään ensin objektitiedostoiksi ja linkitetään tämän jälkeen
 - ▶ `gcc -g -Wall -ansi -c main.c`
`gcc -g -Wall -ansi -c stack.c`
`gcc -o my_program main.o stack.o`
 - ▶ Kohtuullisen kokoisilla ohjelmilla nämä voidaan yhdistää yhdeksi komennoksi:
`gcc -g -Wall -ansi main.c stack.c -o my_program`
- ▶ Vain `.c`-tiedostot tarvitsee kääntää, `.h`-tiedostoja ei koskaan
- ▶ Tämän jälkeen, kun vain yhtä tiedostoa on muutettu, riittää `gcc -g -Wall -ansi stack.c`
`gcc -o my_program main.o stack.o`

Mitä käännöksessä tapahtuu?

- ▶ Esikäntäjä laventaa mm. `#include`-direktiivit ja makrot C-koodiksi (tämä ei ole välikieltä kuten Javassa)



Muistin varaaminen C:ssä, idea

- ▶ Automaattisesti pinosta:
 - ▶ Väliaikaisille muuttujille
 - ▶ Kuten Javassa
- ▶ Keosta:
 - ▶ Dynaamisille tietorakenteille
 - ▶ Ohjelmoijan on varattava (`malloc()`) ja vapautettava (`free()`) muisti
 - ▶ Jos muistin unohtaa vapauttaa, sitä "vuotaa" ja se vapautuu vasta ohjelman suorituksen päättyessä (kriittistä esim. palvelinohjelmistoissa)
- ▶ Lisäksi staattinen muistinvaraus tiedolle, joka on olemassa koko ohjelman eliniän

C-ohjelman rakenne

- ▶ Ohjelma sisältää joukon määrittelyjä ja lauseita
 - ▶ Peruskontrollirakenteet kuten Javassa (1.4)
 - ▶ Muuttujien tyypit: Perustyypit, osoittimet, rakenteiset tyypit
- ▶ Isompi ohjelma jaetaan yleensä eri tiedostoihin
 - ▶ Otsikkotiedostoissa (.h) tietoja, joita tarvitaan ohjelman muissa osissa (rajapinta)
 - ▶ Toteutus muilta moduuleilta piilossa (.c) - ei .h-tiedostossa
- ▶ "Omien rajapintojen" käyttöönotto tapahtuu direktiivillä `#include "stack.h"`
 - ▶ `#include`-direktiivissä lainausmerkit ("`stack.h`") omille .h-tiedostoille, `<>`-merkit (`<stdio.h>`) kääntäjän ja järjestelmän otsikoille