Algoritmid ja andmestruktuurid
IFI6228.DT
6 EAP

Kursuse ülesehitusest

Kuus tundi nädalas:

- 4x45 min praktikumi eraldi rühmades (E&K);
- 2x45 min loengut või harjutust kõigile korraga (K);

Lõppeb kirjaliku eksamiga:

Eksami eelduseks on kontrolltöö semestri teises pooles, millega demonstreerite oma arusaamist C-keelest.

Kogu hinne tuleb eksamist.

Kursuse sisust

Tutvus mitmete huvitavate andmestruktuuride ja algoritmidega:

- pinu, järjekord, erinevad puud, graaf, paisktabel;
- sorteerimine, otsimine, puu- ja graafialgoritmid;
- algoritmimise strateegiad ja algoritmide keerukus.

Tutvus C-keelega, sh viitmuutuja mõiste ja kasutamisega, mis võiks natuke rohkem panna mõtlema sellele, mis arvuti sees toimub.

Materjalid

Kursuseprogrammist leiate kaks raamatut:

V. Leppikson, *Programmeerimine C keeles.* 1997

J. Kiho, *Algoritmid ja andmestruktuurid.* 2003.

Ka järgnevad on huvitavad:

R. Sedgewick, *Algorithms in C* (Parts 1-5). 2002.

A. Isotamm, *Programmeerimine C-keeles ("Algoritmide ja andmestruktuuride" näiteil).* TÜ, 2009

Klassikalised raamatud

Donald Knuth, *The Art of Computer Programming* Volumes 1-3 (1968 – 1973), Volume 4: 2008

T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms*, Third Ed 2009 MIT õpik (viidatakse kui CLRS) (First Ed 1989)

Ja internetist leiab lõpmata palju materjali.

Soovitused ellujäämiseks

Tutvu enne loengut loengumaterjaliga.

Peale loengut loe üle loengumaterjal (mitte ainult slaidid), mõtle läbi ja püüa mõisteid meelde jätta ning nendest aru saada.

Peale harjutustundi vaata üle tunnis tehtud ülesanded ja lõpeta see, mis pooleli jäi. **NB!** Enamus harjutustunnis uuritust läheb hiljem programmeerimisele!!

Peale praktikatundi lõpeta tunnis pooleli jäänud ülesanded.

Proovi saada sõbraks C keelega.

Ülevaade C-keelest ehk selgitusi keelekonstruktsioonidest võrdluses Pythoniga

Interpreteeritav vs kompileeritav programmeerimiskeel

Python on interpreteeritav keel.

C on kompileeritav keel.

Mida tähendab üks ja mida teine mõiste??

Kompileerimisprotsess

Leksiline (leksikaalne) analüüs – analüsaator töötleb koodi sümbol-haaval, tuvastab koodis lekseemid (*lexical token*) ja asendab need märkidega.

Süntaksi analüüs – (ka parsimine) ehitatakse süntaksipuu, tuvastatakse laused ja uuritakse programmi vastavust programmeerimiskeele grammatikale.

Semantiline analüüs – uuritakse, kas on järgitud keelenõudeid (sõltub keelest!): näiteks kas omistuslaues on andmetüübid omavahel vastavuses või kas muutujad on enne kasutamist deklareeritud.

Optimeerimine - sõltuvalt kompilaatori "osavusest" võib järgneda koodi optimeerimine, muutes näiteks lausete järjekorda mälukasutuse vähendamiseks.

Kompileerimisprotsess

Koodi genereerimine – võetakse vahekoodis programm ja tõlgitakse konkreetse arvuti masinkoodi

Selle käigus võidake koodile lisada "koodijupikesi" teistest masinkoodis programmidest (nt erinevad teegid).

Linkimine – nii võidakse ka nimetada eelnevat tegevust ja sõltub kompilaatorist, kas seda osa kuidagi ersitatakse.

Lisa saad lugeda siit:
https://www.tutorialspoint.com/compiler_design/compiler_design_phases_of_compiler.htm

Eelnevast jutust on oluline meeles pidada, et kompileerimisel ilmnevad vead võivad tekkida erinevates faasides ning ühe vea likvideerimise järel võib ilmneda viis uut viga.

Kompilaator ja IDE

C-keelseid programme saab kirjutada ja töödelda erinevates keskkondades. Olen kasutanud DevCpp (5.11) IDEt. Sobib ka CodeBlocks (GNU GPL)

Toetab GCC kompilaatoreid (*GNU Compiler Collection*) – erinevate keelte kompilaatorite kogum, mis on valminud GNU projekti käigus.

C-keele kompilaator oli esimene sellest seeriast (1987).

Kasutame C-keelt (mitte C++ ega C#).

Katsetame koodi kirjutamist ja töötlemist ka Linuxis.

Erinevate IDEdega peab olema ettevaatlik, sest C-keelel on mitmeid dialekte, st väikeseid erinevusi süntaksis, funktsioonide kasutuses jms.

C keele ajaloost

Loodud Dennis Ritchie poolt 1969-72 a

Kasutatud OS-i Unix ümberkirjutamiseks (eelmine versioon oli Assembleris).

1978. a. C keelt kirjeldav raamat:

"The C Programming Language", autorid Brian Kernighan ja Dennis Ritchie

https://en.wikipedia.org/wiki/C_(programming_language)#/media/File:The_C_Programming_Language,_First_Edition_Cover.svg

Raamatus olev C kirjeldus ei ole üheselt mõistetav

Standardiseeritud 1989 ANSI poolt (nn ANSI C, C89), 1990 ISO poolt (C90). Viimane standard ISO/IEC 9899:2018 (nn C17). Ettevalmistamisel on järgmine.

C keel

C keel:

- on üldotstarbeline *(general-purpose)*
- on imperatiivne *(imperative)*
- toetab struktuurprogrammeerimist *(structured programming)*
- muutuja skoop e kehtivuspirkond on staatiline, seotud koodiosaga *(lexical variable scope)*
- võimaldab rekursiooni *(recursion)*
- kasutab staatilist andmetüübi kontrolli *(static type system)*

hello world

Sellisel kujul ilmus maailma kuulsaim(?) programm esimest korda Brian Kernighan'i ja Dennis Ritchie (kutsutakse K&R) C-keele raamatus "The C Programming Language".

```
#include <stdio.h>

main()

{

    printf("hello, world\n");

}
```

hello world

Järgnevat varianti võib lugeda standardile vastavaks:

```
#include <stdio.h>
int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

Teine näide K&R raamatust

```
/* print Fahrenheit-Celsius table
      for f = 0, 20,..., 300 */
main()
{
    int lower, upper, step;
    float fahr, celsius;
    lower = 0;    /* lower limit of table */
    upper = 300; /* upper limit */
    step = 20;    /* step size */

    fahr = lower;
    while (fahr <= upper) {
        celsius = (5.0/9.0) * (fahr-32.0);
        printf ("%4.0f %6.1f\n", fahr, celsius);
        fahr=fahr+step;
    }
}
```

Andmetüübid ja muutujad

C-s tuleb igale muutujale määrata andmetüüp (muutuja deklareeritakse / defineeritakse)

Andmetüübid on (osaline valik):

- Täisarvud: `int` (4 baiti); formaat `%d`
- Ujukomaarvud: `float` (4 baiti), `double` (8 baiti); formaadid `%f, %lf`
- Sümbol: `char` (1 bait, sõltub kooditabelist); formaat `%c`

Deklareerime kaks 4-baidist täisarv-muutujat:

```
int arv1, arv2;
```

Andmetüübi täpsustused

Tavaliste andmetüüpide ees võime kohata veel järgmisi keelesõnu: `long`, `long long`, `short`, `signed`, `unsigned`.

Täpsed suurused sõltuvad arvutisüsteemist.

`long long int` on tavaliselt 2x suurem kui `int`, `short int` aga 2x väiksem (vastavalt 32, 64 või 16 bitti)

Ainult postiivsete (märgita) arvude kasutamiseks `unsigned int`

Erineval andmetüübil erinev formaat (sisestamisel, trükkimisel).

Sisend ja väljund

Andmete sisestsamise ja trükkimise funktsioonid paiknevad `stdio` **teegis** *(library).*

```
#include <stdio.h>
```

Trüki küsimus `printf()`-ga ja vastust loe `scanf()`-ga (sisendis arvud tühikutega eraldatult)

```
int a1, a2;
printf("Sisesta kaks täisarvu ");
scanf("%d %d",&a1,&a2);
printf("Arvud on %d ja %d.\n",a1,a2);
```

Sisend ja väljund

```
scanf("%d %d",&k1,&k2);
```

Ootab sisendiks kahte tühikuga eraldatud täisarvu.

`&` on muutuja muutmälu aadressi tähis ja tema tähendusest räägime täpsemalt edaspidi. Seda võib mõtestada nii, et scanf()-ile öeldakse, kuhu mälus väärtused kirjutada.

Tuleta meelde, et üks muutuja omadustest on tema mäluaadress, mille järgi arvuti / programm teda muutmälust (RAM) üles leiab.

Näide ...

... formaadi kasutamisest `scanf()`-ga:

```
int pikk_m,pikk_cm;
printf("Sisesta pikkus ");
scanf("%d m %d cm",&pikk_m,&pikk_cm);
```

Ootab sisendit:

```
12 m 50 cm
```

NB! Ebakorrektne formaadikasutus põhjustab valede andmete lugemise. Sealjuures ei pruugi tekkida täitmisaegset viga, vaid muutujatesse loetakse valed väärtused.

Aritmeetikatehted ja omistamine

Toimub üldiselt Pythonist tuttaval viisil.

Puudub astendamistehe, seda asendab funktsioon `pow(arv,aste)`.

Jagamistehte tulemus ja andmetüüp sõltub operandide tüüpidest (`int/int` `->` täisarvuline jagamine).

Tehete järjekord ja sulgude kasutus on tavapärane.

`i++` on nn inkrement ja `i--` nn dekrement (suurendamine ja vähendamine 1 võrra)

Tüübiteisendused

Kasutatakse tüübiteisenduse *(type cast)* operaatorit.

```
f1 = (float)i1 / i2;
```

Tehteks muudab `(float) i1` väärtuse ujukomaarvuks, toimub tavaline jagamine.

NB! `i1` sisaldab jätkuvalt täisarvu!!

Mis on järgmise tehte tulemuseks?

```
(int)29.55 + (int)21.99
```

See on unaarne operaator, nagu märk -, tehete järjekorras ees.

Võrdlus- ja loogikatehted

C-s on andmetüüp _Bool, nn tõeväärtustüüp. Sinna salvestatakse 0 (väär) või 1 (tõene)

Võrdlustehted: `==, !=, >=, <=, >, <`

Nt töötab omistamine: `a = 5 > 4` (a saab vrt 1)

Loogikatehted: `!` (*not*), `||` (*or*), `&&` (*and*)

Tõeseks loetakse mistahes mittenulliline operand, **vääraks** nulliga võrduv operand.

Mõtle loogikatehteid kasutades väärtustele *true* ja *false* ning kirjuta võrdlustehted korralikult välja!

Liitlause

C-s kasutatakse mõistet **liitlause** *(compound statement)* tähistamaks mitmest lausest koosnevat plokki.

Liitlaused paiknevad tavaliselt juhtlausete sees (`if, while, for, ...`)

Liitlaused ümbritsetakse looksulgudega `{}`

Iga lause liitlauses lõppeb semikooloniga `;`

Mis on Pythoni analoog?

Tingimuslause ehk if-lause

`if`-lause töötab täiesti harjumuspärasel viisil.

Süntaks:

```
if (tingimus)
   {laused, kui tingimus tõene}
else
   {laused, kui tingimus vale}
```

`if` ja `else` järel on liitlause.

`else`-osa võib puududa.

Näide

```
...
if (leitud==1) {
   printf("On olemas\n");
   }
else {
   printf("Ei ole olemas\n");
   }
...
```

Selles näites võiksid looksulud ka puududa, sest liitlauseks on üks lause.

Korduslaused

Korduslauseid *(iteration statement)* ehk silmuslauseid on C-keeles kolm:

- `while`-lause (eelkontrolliga tsükkel);
- `do`-lause (järelkontrolliga tsükkel);
- `for`-lause (määratud korduste arvuga tsükkel);

`while`- ja `do`-laused

`while`-lause sarnaneb oma tööpõhimõtetelt Pythoni vastavale lausele. Süntaks:

```
while (jätkamise tingimus)
    {korduses täidetavad laused}
```

`do`-lauses on tingimus tsükli lõpus. Süntaks:

```
do
    {korduses täidetavad laused}
while (jätkamise tingimus)
```

for-lause

`For`-lause tööpõhimõte erineb oluliselt Pythoni `for`-lause omast. Süntaks:

```
for (avaldis1;avaldis2;avaldis3)
    {tsüklis täidetavad laused}
```

`avaldis1` täidetakse üks kord enne tsükli algust

`avaldis2` on tsükli jätkamise (loogika)tingimus, kontrollitakse enne iga kordust

`avaldis3` täidetakse iga korduse lõpus

`For`-lause näide

Ehkki `for`-lausel võib olla mitmeid kummalisi ja esmapilgul raskesti mõistetavaid variante, on kõige tavapärasem järgmine:

```
int i,n;
n=5;
for (i=0; i<n; i++)
   printf("%d",i);
```

Selgitus: enne tsüklit tehe `i=0;` kontroll, kas `i<n`. Kui on, siis trükitakse `i`, suurendatakse `i` 1 võrra (`i++`). Tegevus kordub.

Proovi sama kirja panna `while`-lausega.

Lõpetuseks

Sellega on olulised laused üle vaadatud. Katsetame lihtsamate (ja võib olla ka tuttavate) ülesannete lahendamisega, harjumaks C-keele süntaksiga.

Enim probleeme tekib arvatavasti andmete formaadiga sisestamisel.

Tegelikult on lisaks `scanf()`-le veel sisestuse funktsioone, mis on andmetüübi-põhised ja mida on soovitav kasutada stringe / märke sisestades.