

C-kurssi

Viikko 1: tyypit, rakenteet, makrot

Luennon sisältö

- Tyypit
 - int, char, float, double
 - signed, unsigned
 - short, long
- Vakiot
 - const
- Rakenteet
 - if, for, while, switch, do-while
 - Syöttö ja tulostus
- Makrot
 - #define

Yksinkertaiset tyypit

- int – kokonaisluku
- char – yksi merkki (c:ssä oikeastaan kokonaisluku)
- Float, double – reaaliluku (liukuluku)
- HUOM: ei boolean tyyppiä
 - Käytetään kokonaislukuja
 - 0 on epätosi, FALSE ja
 - kaikki muut arvot tosia, TRUE
- Tyyppien kokoa (tavuina tai bitteinä) ei taata ympäristöstä toiseen (standardi määrittelee hyväksyttävät minimikoot)

Lisämääreet: int ja char

- Etumerkki: signed, unsigned
 - unsigned int
 - signed char
- Kokomääreitä: short, long
 - long char
 - short int
- Yhdistelmät:
 - signed short int
 - unsigned long int

Merkit

- Merkit ovat oikeastaan kokonaislukuja
- Merkistöstä EI saa olettaa mitään
- Jos haluaa oikeasti tehdä siirrettävää ja kaikissa järjestelmissä toimivaa koodia pitää eksplisiittisesti käyttää tyyppiä
signed char tai unsigned char

Kokonaisluvut

- Rajat otsikkotiedostossa limits.h
- Yläraja INT_MAX aina ≥ 32767
- Laitoksen ympäristössä 2147483647
- "-" - SHRT_MAX on 32767 (signed short int)
- Kaikenkokoisille kokonaisluvuille on omat maksimi- ja minimiarvot
- Kokonaislukuja käytettäessä voi kertoa tyyppin kirjaimella arvon jälkeen (U, L)

sizeof(short) \leq sizeof(int) \leq sizeof(long)

Otsikkotiedosto limits.h

- Siirrettävyyden vuoksi tyyppien maksimikoot määritelty erillisessä tiedostossa – voivat vaihtua
- Laitoksen ympäristössä otsikkotiedostot ovat hakemistossa /usr/include/
- Otsikkotiedostot liitetään omaan ohjelmaan esikäntäjän ohjauskomennolla:

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

Reaaliluvut


- float
- double
- long double

- Otsikkotiedostossa float.h on koot ja rajat

sizeof(float) <= sizeof(double) <= sizeof(long double)

Tyypimuunnoksia

Jos lausekkeen operandit ovat erityyppisiä, niin c:ssä tehdään automaattinen tyypimuunnos laskutoimitusta varten aritmeettisten tyyppien välillä. Aina pienemmän tarkkuuden omaava tyyppi muunnetaan tarkemmaksi tyyppiä seuraavasti:

- `int` ja `char`
 - `unsigned`
 - `long`
 - `unsigned long`
 - `float`
 - `double`
 - `long double`
- 

Eksplisiittinen tyyppimuunnos

- C:ssä voi muuttujan arvon tyyppiä vaihtaa lausekkeessa määreellä **(uusityyppi)mt ja**
 - Tämä voi olla jopa välttämätöntä
 - Vakioarvojen tyyitys U ja L määreillä

```
int x; char merkki = 'a';  
float c, f;  
x = merkki + 17;  
c = (5/9)*(f-32); /* arvo on 0 !! */  
c = ( (double)5/9 ) * (f-32);
```

Errors



Ylivuoto

Kahden positiivisen kokonaisluvun ylivuodon testaamista ei saa tehdä vertailulla

$$i + j > \text{INT_MAX}$$

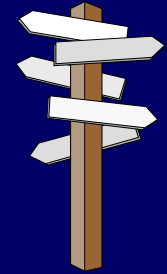
Miksi?

vaan vertailulla

$$i > \text{INT_MAX} - j$$

Luennon sisältö

- Tyypit
 - int, char, float, double
 - signed, unsigned
 - short, long
- Vakiot
 - const
- Rakenteet
 - if, for, while, switch, do-while
- Makrot
 - #define



Tunnuksista

- Käytä *yhtenäistä* tyyliä nimissä.
- Nimien pitää olla kuvaavia.
- Selvyyden vuoksi voit sekoittaa isoja ja pieniä kirjaimia. HUOM: ne ovat merkitseviä:
longIdentifier, longest
- Vain 31 ensimmäistä merkkiä käytössä
- (Aikoinaan vain 6 merkkiä erotti)

Vakioiden määrittely

- Vakiot määritellään kuten muuttujat, mutta niiden määrittely alkaa sanalla `const`
- Vakioiden nimet kirjoitetaan ISOILLA KIRJAIMILLA (vakiintunut tapa)

```
const float PI = 3.1412;  
const int ISO_LUKU = 0xFF7D;  
const int TRUE = 1;  
const int FALSE = 0;  
const char A_KIRJAIN = 'a';  
const char [] MJONO = "merkkijonossa on lainausmerkit";
```

Kokonaislukuvakiot

- Luvun saa syöttää etumerkillä tai ilman
 - Kymmenkantaisena: 1234567789
 - Oktaalina (kahdeksankantaisena): 034 01234567
 - Heksadesimaalina: 0x12ABCDEF
- Luvun tyyppi määräytyy sen arvon mukaan:
 - Jos mahtuu int tyyppiin
 - int
 - Muuten jos mahtuu long tyyppiin
 - long
 - Muuten jos mahtuu unsigned long
 - unsigned long
 - Muuten määrittelemätön (liian suuri)
- Tyypin voi myös määrätä kirjaimella U tai L
 - 12U on unsigned int ja 7L on long int

Merkkivakiot ja merkkijonovakiot

- Merkkivakio 'a' '\065' '\xA6'
- Merkkijonovakio "absdefkkjhaöj"
- Merkkivakio talletetaan yhden merkin tilaan (ja se siis on oikeasti numero)
- Merkkijono talletetaan sen tarvitsemaan tilaan (peräkkäisiä muistipaikkoja) ja loppuun vielä merkkijonon päättävä arvo '\0', vakiona toimii osoitin tähän muistialueeseen!!!

Luennon sisältö

- Tyypit
 - int, char, float, double
 - signed, unsigned
 - short, long
- Vakiot
 - const
- Rakenteet
 - if, for, while, switch, do-while
 - Syöttö ja tulostus
- Makrot
 - #define

Ehtolausekkeet

- Looginen AND
- Looginen OR
- Ehdollinen lauseke
- Pilkkulauseke

- Käytä sulkuja selkeyttämään lausekkeita

```
e1 && e2
```

```
e1 || e2
```

```
e1 ? e2 : e3
```

```
e1, e2
```

Errors



Assosiatiivisuus

- lauseke

$$a < b < c$$

- tulkitaan

$$(a < b) < c$$

- ja sen merkitys on eri kuin lausekkeen

$$a < b \ \&\& \ b < c$$

Evaluointi- järjestys

Samalla rivillä samanarvoisia

Aritm. operaatiot
Bittisiirrot vasen ja oikea

suuruusvertailut

Bittivertailut

and

or

Ehdollinen vertailu



sijoitukset

() [] . ->

! ~ - ++ -- & * (tyyppi) sizeof

* / %

+ -

<< >>

< <= > >=

== !=

&

^

|

&&

||

?:

= *= /= %= += -= <<= >>= &= != ^=

,

Errors



Vältä virheitä

◆ `i = 8` on ihan eri asia kuin `i == 8`

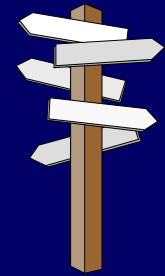
◆ Tarkista rajat (vältä 'off by one')

◆ Huomaa että nämä eivät ole loogisia vertailuja!!!

`e1 & e2`

`e1 | e2`

`if(x = 1) ...`



Tyylistä

- Älä laita välilyöntiä seuraaviin :

-> . [] ! ~ ++ -- -(etum.)

* (osoitin) &

- Yleensä erota seuraavat välilyönneillä:

= += ?: + < &&

+ (yhteenl.) ja vastaavat

```
a->b      a[i]      *c
          a = a + 2;
          a= b+ 1;
          a = a+b * 2;
```

Lauserakenteet

- Ehto
- Toistot

```
if (ehto)
    lause;
else
    lause;
```

```
If (ehto) {
    useita lauseita
} else {
    useita lauseita
}
```

```
for (;;)
    lause
```

```
while (1) {
    lauseet
}
```

```
do {
    lauseet
} while (ehto);
```

- Toistojen keskeytys
 - Break - hyppää toistoa seuraavaan lauseeseen
 - Continue – hyppää seuraavalle kierrokselle
 - Näissä ei saa olla nimeä!! (vrt. Java)

Break - käyttö

```
while (1) {  
    printf("anna kaksi lukua a ja b, a < b:");  
    if (scanf("%d%d", &a, &b) != 2)  
        break;  
    if (a < b)  
        break;  
    ...  
}  
/* break jatkaa suoritusta tästä */
```

Tässä on tyypillisiä c:n piirteitä

- ikuinen toisto while(1)
- virheentarkistus !!
- tulostus ja syöttö käyttäen standardifunktioita

Poistuminen syvästä rakenteesta

- Poistuminen useamman tason yli silmukassa on tehtävä goto -lauseella

```
for(i = 0; i < length; i++)  
    for(j = 0; j < length1; j++)  
        if(f(i, j) == 0)  
            goto done;  
  
done:
```

- Break jatkaisi ulomman silmukan seuraavaa kierrosta!

Valinta: switch esimerkkiohjelma

```
.. /* ohjelman alkuosa ja muuttujien esittelyt */
Printf("Syötä korkeintaan %d merkkiä\n", LIMIT);
For (i = 1; i <= LIMIT; i++) {
    if ( (c=getchar()) == EOF)
        break; /* tiedosto loppui CTRL-D -merkki */
    switch (c) {
    case ' ' : valil++;
                break;
    case '\t': tabul++;
                break;
    case '*' : tahti++;
                break;
    default  : if (c>='a' && c<='z')
                    pienia++;
    }
}
... /* täällä voidaan sitten vaikka tulostaa */
```

```
/* Program that reads two integer values, and  
 * outputs the maximum of these values.  
 */
```

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

```
int main() {
```

```
    int i, j;
```

```
    printf("Enter two integers:");
```

```
    if(scanf("%d%d", &i, &j) != 2) {
```

```
        fprintf(stderr, "wrong input\n");
```

```
        return EXIT_FAILURE;
```

```
    }
```

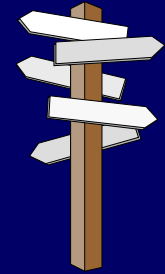
```
    printf("Maximum of %d and %d is %d\n",
```

```
           i, j, i > j ? i : j);
```

```
    return EXIT_SUCCESS;
```

```
}
```

“Lue
kaksi
lukua”



Control Statements

Tämä toisto

```
while(expr != 0)  
    statement;
```

on identtinen tämän kanssa

```
while(expr)  
    statement;
```

Miksi?

Kirjan esimerkki 4.4

```
/* Example 4.4
 * Read characters until "." or EOF and
 * output
 * the ASCII value of the largest input
 * character.
 */
#include <stdio.h>
int main() {
    const char SENTINEL = '.';
    int aux;
    int maxi = 0;
```

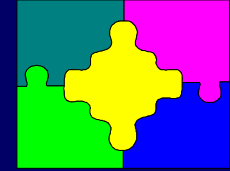
```
printf("Enter characters,. to terminate\n");
```

```
while(1) {  
    if((aux = getchar())== EOF || aux == SENTINEL)  
        break;  
  
    if(aux > maxi)  
        maxi = aux;  
}
```

*Idiom ?
(sanonta)*

```
printf("The largest value: %d\n", maxi);  
return EXIT_SUCCESS;  
}
```

Idioms



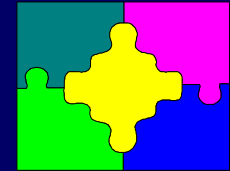
Lue merkkejä loppumerkkiin asti yksi kerrallaan

```
while(1) {  
    if((aux = getchar()) == EOF || aux == SENTINEL)  
        break;  
    ...  
}
```

or:

```
while(1) {  
    if((aux = getchar()) == EOF)  
        break;  
    if(aux == SENTINEL)  
        break;  
    ...  
}
```

\dioms



Lue kokonaislukuja

```
while(1) {  
    if (scanf("%d", &i) != 1 ||  
        i == SENTINEL)  
        break;  
  
    ...  
}
```


Syöttö ja tulostus lyhyesti

- Merkki kerrallaan

```
int getchar()
```

```
int putchar(int)
```

- Muotoiltuna

```
int scanf("format", &var)
```

```
int printf("format", exp)
```

```
/* File: ex1.c
 * Program that reads a single character and
 * outputs it, followed by end-of-line
 */
```

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

HUOM: Nämä otsikkotiedostot
tarvitaan funktioiden käyttöä varten

```
int main() {
    int c;    /* chars must be read as ints */

    if ((c = getchar()) == EOF)
        return EXIT_FAILURE;

    putchar(c);
    putchar('\n');

    return EXIT_SUCCESS;
}
```

Kokonaislukujen muotoilukomennot

Kokonaislukujen muotoilussa käytetään seuraavia määreitä:

d etumerkillinen kokonaisluku

ld long decimal

u etumerkitön kokonaisluku

o oktaali (kahdeksankantainen)

x, X heksadesimaali (kuusitoistakantainen)

```
printf("%d%o%x", 17, 18, 19);
```

Reaalilukujen muotoilukomennot

Reaaliluvuille käytetään seuraavia ohjauksia
(oletustarkkuus on 6):

f	[-] ddd.ddd
e	[-] d. ddddde{sign}dd
E	[-] d. ddddE{sign}dd
g	fe (vain lyhyempi)
G	FE

```
printf("%5.3f\n", 123.3456789);
```

```
printf("%5.3e\n", 123.3456789);
```

```
123.346
```

```
1.233e+02
```

Merkkien ja merkkijonojen muotoilu

Merkkejä ja merkkijonoja muotoillaan seuraavasti:

c merkki

s merkkijono

```
printf("%c", 'a');
```

```
printf("%d", 'a');
```

```
printf("This %s test", "is");
```

`scanf()` - paluuarvot

`scanf()` palauttaa arvonaan luettujen alkioiden määrän ja EOF, jos yhtään alkiota ei saatu luettua ennen tiedoston loppumista.

Esimerkiksi `scanf("%d%d", &i, &j)`

voi palauttaa seuraavat arvot:

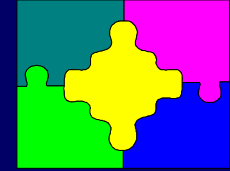
2 Jos molemmat luvut onnistuivat

1 Jos vain i saatiin luettua

0 jos luku epäonnistui kokonaan

EOF jos tiedosto päättyi.

Idioms



Pyydä ja lue yksi merkki

```
printf("Enter integer: ");  
if(scanf("%d", &i) != 1 ) ...  
    /* error, else OK */
```

Lue kaksi kokonaislukua

```
if(scanf("%d%d", &i, &j) != 2 ) ...  
    /* error, else OK */
```

Makro

- Esikäntäjän ohjauksennoilla voi määritellä makroja
- Makro on tunnus, joka korvataan sisällöllään ohjelmakoodiin ennen varsinaista käännöstä
- HUOM: Koko loppurivi on korvaava arvo!

```
#define MAKSIMI 30  
#define NIMI "Oiva Ohjelmoija"  
#define TRUE 1  
#define FALSE 0
```