

C-kurssi syksy 2007

Luento 2: tyypit, rakenteet, makrot
5.9.2008

Luennon sisältö

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

Yksinkertaiset tyypit

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

Lisämääreet: int ja char

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

Merkit

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

Kokonaisluvut

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

```
sizeof(short) <= sizeof(int) <= sizeof(long)
```

Otsikkotiedosto limits.h

- n Siirrettävyyden vuoksi tyyppien maksimikoot määritelty erillisessä tiedostossa – voivat vaihtua
- n Laitoksen ympäristössä otsikkotiedostot ovat hakemistossa /usr/include/
- n Otsikkotiedostot liitetään omaan ohjelmaan esikäntäjän ohjauskomennolla:
`#include <limits.h>`

Reaaliluvut

- n float
- n double
- n long double
- n 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 tyypimuunnos

- n C:ssä voi muuttujan arvon tyyppiä vaihtaa lausekkeessa määreellä **(uusityyppi)mtja**
 - Tämä voi olla jopa välttämätöntä
 - Vakioarvojen tyypitys 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);
```

Ylivuoto

Errors



Kahden positiivisen kokonaisluvun ylivuodon testaamista ei saa tehdä vertailulla

`i + j > INT_MAX`

Miksi?

vaan vertailulla

```
i > INT_MAX - j
```

Luennon sisältö

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

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

Vakioiden määrittely
- n Vakiot määritellään kuten muuttujat, mutta niiden määrittely alkaa sanalla <code>const</code> - n Vakioiden nimet kirjoitetaan ISOILLA KIRJAIMILLA (vakiintunut tapa) <pre>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";</pre>

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

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

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

Ehtolausekkeet
- n Looginen AND - n Looginen OR - n Ehdollinen lauseke - n Pilkkulauseke - n Käytä sulkuja selkeyttämään lausekkeitä <pre>e1 && e2 e1 e2 e1 ? e2 : e3 e1, e2</pre>

Errors

Assosiativisuus



n lauseke

$$a < b < c$$

n tulkitaan

$$(a < b) < c$$

n 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

```

0 [] . ->
! ~ - + - - * (tyyppi) sizeof
* / %
+ -
<< >>
< <= > >=
== !=
&
^
|
&&
||
?:
= *= /= %= += -= <<= >>= &= != ^=

```

Errors

Vältä virheitä



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

u Tarkista rajat (vältä 'off by one')

u Huomaa että nämä eivät ole loogisia vertailuja!!!

```

e1 & e2
e1 | e2
if(x = 1) ...

```

programming Guidelines

Tyylistä



n Älä laita välilyöntiä seuraaviin :

```

-> . [] ! ~ ++ -- -(etum.)
* (osoitin) &

```

n Yleensä erota seuraavat välilyönneillä:

```

= += ?: + < &&
+ (yhteenl.) ja vastaavat

```

```

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

```

Lauserakenteet

n Ehto

n Toistot

```

if (ehto)
    lause;
else
    lause;

```

```

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

```

```

for (;)
    lause

```

```

while (1) {
    lauseet
}

```

```

do {
    lauseet
} while (ehto);

```

n 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 tyyppisiä c:n piirteitä

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

Poistuminen syvästä rakenteesta

- n 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:
```

- n 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"

programming Guidelines

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;
}
```

\dioms


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

- n Merkki kerrallaan

```

int getchar()
int putchar(int)

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

    if ((c = getchar()) == EOF)
        return EXIT_FAILURE;
    putchar(c);
    putchar('\n');

    return EXIT_SUCCESS;
}

```

HUOM: Nämä otsikkotiedostot
tarvitaan funktioiden käyttöä varten

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%0x", 17, 18, 19);

```

Reaalilukujen muotoilukomennot

Reaaliluvuille käytetään seuraavia ohjauksia (oletustarkkuus on 6):

- f [-] ddd.ddd
- e [-] d.ddddde{sign}dd
- E [-] d.ddddde{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 alkoiden määrän ja EOF, jos yhtään alkia 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.

\dioms



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

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

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