

# C-kurssi kevät 2006

Luento 2: tyypit, rakenteet, makrot  
24.1.2006

## 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

## Tasokokeen kohta 1: Taulukon järjestäminen

- Oletetaan taulukon indeksointi alkamaan nolasta

```
n = sizeof(t);
for (i=0; i<n; i++)
    for (j=i; j<n; j++)
        if (t[i] > t[j]) {
            temp = t[i];
            t[i] = t[j];
            t[j] = temp;
        }
```

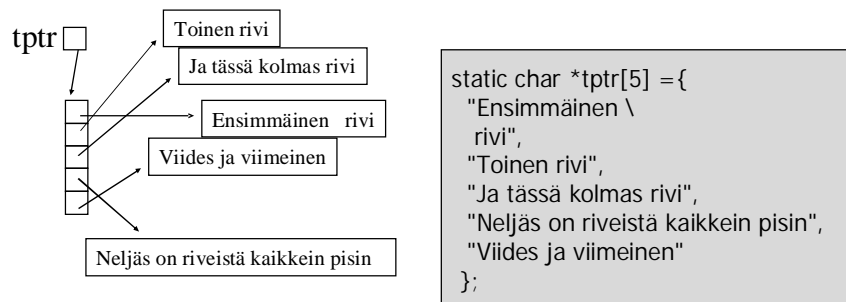
## Tasokokeen kohta 2

```
#include <stdio.h>
#include <string.h>
int longest(char **p, char *r) {
    char *s,*t;
    int max=0; len;
    while(*p++) {
        if ((len=strlen(*p)) > max ) {
            max==len;
            s=*p;
        }
        p++;
    }
    t=s;
    while(*r++ == *s++);
    r=t;
    return s - t;
}

int main () {
    int pit;
    char pisin_rivi[80];
    static char *tptr[5] = {
        "Ensimmäinen \
        rivi",
        "Toinen rivi",
        "Ja tässä kolmas rivi",
        "Neljäs on riveistä kaikkein pisin",
        "Viides ja viimeinen"
    };
    char *pp = pisin_rivi;
    pit= longest(tptr,pp);
    printf("%d merkkiä: %s\n",pit, pp);
    return 0;
}
```

## Tasokokeen kohta 2

- tptr on osoitintaulukko, jonka alkioina on osoittimia merkkijonovakioihin.
- Sen osoite välittää parametrina:  
int longest ( char \*\*p, char \*r)

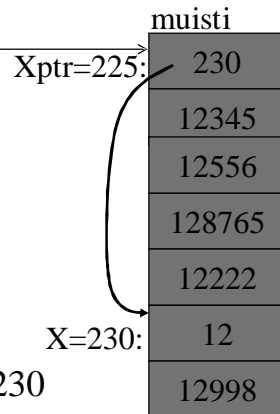


## Tasokokeen kohta 3

- while (\*q++ = \*p++);
  - Muistialueen kopiointi (esim. merkkijono)
- if ((c=fgetc(fileHandle)) == EOF)
  - Merkin luku tiedostosta, samalla tarkistetaan tiedoston päättyminen
- for (i=a, j=b; i<=j; i += 2, j += 2)
  - Silmukka, josta tulee ikuinen tai sitä ei suoriteta lainkaan

## Tieto ja sen osoite <sup>(3)</sup>

```
Xptr DC 0
X   DC 12
LOAD R1, =X   ; R1 ← 230
STORE R1, Xptr
LOAD R2, X     ; R2 ← 12
LOAD R3, @Xptr ; R3 ← 12
```



- Muuttujan X osoite on 230
- Muuttujan X arvo on 12
- Osoitinmuuttujan (pointterin) Xptr osoite on 225
- Osoitinmuuttujan Xptr arvo on 230
  - jonkun tiedon osoite (nyt X:n osoite)
- Osoitinmuuttujan Xptr osoittaman kokonaisluvun arvo on 12

```
C-kieli: Y = *ptrX /* ei ptrX:n arvo,
mutta ptrX:n osoittaman muuttujan arvo */
```

## 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

## 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  $\geq 32767$
- 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) <= sizeof(int) <= 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)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);
```

## Errors Ylivuoto



Ylivuodon testaamista ei saa tehdä vertailulla

```
i + j > INT_MAX
```

vaan vertailulla

```
i > INT_MAX - j
```

**Miksi?**



## Luennon sisältö

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

## programming Guidelines



### Tunnuksista

- Käytä *yhtenäistä* tyyliä nimissä.
- Nimien pitää olla kuvaavia.
- Selvyyden vuoksi voit sekoittaa isoja ja pieniä kirjaimia. HUOM: ne ovat merkitseviä:  
`longIdentifier, _k3`
- 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 0**1234567
  - Heksadesimaalina: **0x**12ABCDEF
- Luvun tyyppi määräytyy sen arvon mukaan:
  - Jos mahtuu int tyyppiin  
Ø int
  - Muuten jos mahtuu long tyyppiin  
Ø long
  - Muuten jos mahtuu unsigned long  
Ø 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

```
e1 && e2
e1 || e2
e1 ? e2 : e3
e1, e2
```

- Käytä **sulkuja** selkeyttämään lausekkeitä

## Errors

### Assosiatiivisuus



- lauseke  
 $a < b < c$
- tulkitaan  
 $(a < b) < c$
- ja sen merkitys on eli kuin lausekkeen  
 $a < b \ \&\& \ b < c$

# Evaluointi- järjestys

Samalla rivillä samanarvoisia

	{	() [] . ->
		! ~ - ++ -- & * (tyyppi) sizeof
Aritm. operaatiot	{	* / %
		+ -
Bittisiirrot vasen ja oikea		<< >>
	{	< <= > >=
suuruusvertailut	{	== !=
		&
Bittivertailut	{	^
and		&&
or		
Ehdollinen vertailu		?:
→ sijoitukset		= *= /= %= += -= <<= >>= &= != ^=
		,

# 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

- 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 */
```

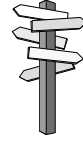
```
/* Program that reads two integer values, and
 * outputs the maximum of these values.
 */
#include <stdio.h>
int main() {
    int i, j;
    printf("Enter two integers:");
    if(scanf("%d%d", &i, &j) != 2) {
        fprintf(stderr, "wrong input\n");
        return EXIT_FAILURE;
    }
    printf("Maximum of %d and %d is %d\n",
           i, j, i > j ? i : j);
    return EXIT_SUCCESS;
}
```

“Lue  
kaksi  
lukua”



## 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

- 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>
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%o%x", 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 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 ohjauskomennoilla voi määritellä makroja
- Makro on tunnus, joka korvataan sisällöllään ohjelmakoodiin ennen varsinaista käännöstä
- HUOM: Koko loppurivi on korvaava arvo!

```
#define MAKSIMI 30  
#define NIMI "Tiina Niklander"  
#define TRUE 1  
#define FALSE 0
```