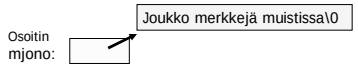
C-ohjelmointi, syksy 2008

Merkkijonot
Komentoriviparametrit

Luento 7
26.9.2008

Merkkijonot (strings) (Müldnerin kirjan luku 9)

- C:ssä merkkijono ei ole ennaltamääritelty datatyyppi (kuten Javassa)
- Merkkijono on osoitin merkkeihin



- Merkkijonojen käsittelyä varten standardikirjasto string.h

Luennon sisältö

- Kirjastofunktiot merkkien käsittelyyn
- Merkkijonon määrittely ja alustaminen ja käyttö
- Merkkijonojen lukeminen ja kirjoittaminen
  - formatoitu I/O merkkijonoille
    - sscanf ja sprintf
  - Riveittäin lukeminen ja tulostaminen
    - fgets ja fputs; gets ja puts
- Kirjastofunktiot merkkijonojen käsittelyyn
- Komentoriviparametrit

Merkeistä

Escape-merkit (samat kuin Javassa)
'\n' = rivinvaihtomerkki
'\t' = tabulaattori
'\v' = pystysuora tabulointi
'\b' = peruutusmerkki
'\r' = rivinalkuunpalautusmerkki
'\f' = sivunvaihtomerkki
'\a' = hälytysmerkki, yleensä äänimerkki
'\\' = kenoviiva
'\'' = heittomerkki
'\"' = lainausmerkki

'\0' = merkkijonon lopetusmerkki

- 'c' on merkki ja "c" on merkkijono
- datatyyppi int
  - signed char, unsigned char
    - L'a' long int
      esim. japanin tai kiinan merkkeihin
- Syöttö ja tulostus
  - getchar, putchar
  - scanf("format", &var), printf("format", exp)
  - fgets, fputs, fscanf, fprintf

```
if((c=getchar()) == EOF) ...
if(scanf("%d%d", &i,&j) !=2) ...
if((c=fgetc(tkahva)) == EOF) ..
while((c= getchar()) != '\n') ...
while ((c=getchar()) !=EOF) ...
while(getchar() != '\n' ) ;
```


#include<ctype.h>

Kirjastofunktioita merkkien käsittelyyn

- Standardikirjaston ctype.h funktioita
  - Merkkien luokitteluun
    - islower(int c)
    - isdigit(int c)
      palauttaa nollan, jos merkki ei ole kysyttyä tyyppiä, muuten nollasta eroavan arvon
  - Merkkimuutoksiin
    - tolower(int c)
    - toupper(int c)
      jos muutos onnistuu, palauttaa muutetun arvon, muuten palauttaa EOF:n

Merkkien luokittelufunktiot

if (c>='a' && c<= 'z') ...

Kirjastofunktioilla siirrettävämpää koodia!

Alfanumeeriset
- int isalnum(int c) is c an alphanumeric
- int isalpha(int c) is c an alphabetic letter
- int islower(int c) is c a lower case letter
- int isupper(int c) is c an upper case letter
- int isdigit(int c) is c a digit
- int isxdigit(int c) is c a hexadecimal digit
- int isodigit(int c) is c an octal digit

Muut merkit
- int isprint(int c) is c printable (not a control character)
- int isgraph(int c) is c printable (not a space)
- int ispunct(int c) is c printable (not space or alphanumeric)
- int isspace(int c) is c whitespace

Merkkijonon määrittely ja muistinvaraaminen merkkijonolle

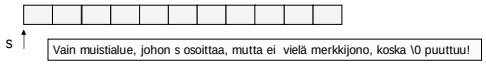
- Merkkijono määritellään osoittimena, joka osoittaa merkkeihin: char *s;
- Merkkijonolle pitää myös varata tilaa muistista.
  - Myös \0-merkille on varattava tilaa
  - Kirjastofunktiot osaavat käsitellä merkkijonona vain sellaista merkkien jonoa, joka päättyy \0-merkkiin!
- Esim. 10 merkin jonolle varattava 11 merkin kokoista muistialuetta:

"muistinvaraus"-fraasi

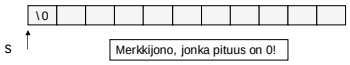
```
#define SIZE 10
if((s = malloc((SIZE+1)*sizeof(char))) == NULL) ....
```

Mitä tehdään varauksen epäonnistuessa?

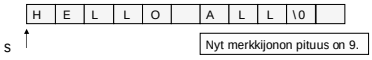
Merkkijonon alustaminen ja merkkijonon pituus



Alustettu, tyhjä merkkijono: s[0] = '\0';



Merkkijonoon voidaan tallettaa lisää merkkejä:



Merkkijonon muistinvaraus ja i:nnen merkin osoittaminen

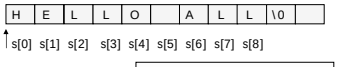
idioms

- Merkkijonon muistinvaraus aina calloc-funktiolla, koska calloc nollaa muistialueen => merkkijono on alustettu

```
if((s = calloc(n+1, sizeof(char))) == NULL) ..
```

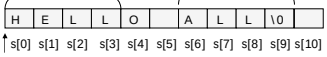
- viittaus merkkijonon s i:nteen merkkiin s[i] :llä

(0<= i < merkkijonon pituus)



S[3] = 'L' ja s[6]='A'

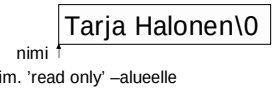
Merkkijonon prefiksi (alkupää) ja suffiksi (loppupää)



- Merkkijonon loppupäähän pääsee helposti käsiksi
  - Esimerkiksi
    - s+6 osoittaa merkkijonoa "ALL", jonka pituus on 3 merkkiä
    - s+8 osoittaa merkkijonoa "L", jonka pituus on 1 merkki
    - s+9 osoittaa tyhjää merkkijonoa, jonka pituus on 0
    - s+10 ei osoita mihinkään merkkijonoon!
- Alkupäähän esim. merkkijonoon "HELL" on taas vaikeampi päästä käsiksi, koska se ei pääty \0-merkkiin.

Merkkijonovakio

- char *nimi = "Tarja Halonen"
  - Vakion sisältöä ei saa muuttaa



    - On voitu tallettaa esim. 'read only' –alueelle
    - Ei saa välittää parametrina sellaiselle funktiolle, joka muuttaa parametrinaan saamaa merkkijonoa
- Huomaa ero: 'T' ja "T"
  - 'T' on merkki T
  - "T" on merkkijono eli merkki T ja sitä seuraava null-merkki T\0

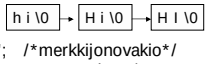
Merkkijono parametrina ja paluuarvona

- Koska merkkijono on itseasiassa osoitin, sitä voidaan käyttää samalla tavoin kuin osoittimia yleensä
  - Merkkijonon tulee olla alustettu (muuten ei ole merkkijono!)
  - Merkkijonovakioita ei saa muuttaa

```
# funktio muuttaa
# merkkijonon 1. merkin
# isoksi kirjaimeksi

void modify(char *s) {
  s[0] = toupper(s[0]);
}
```

```
char *p;  /* EI VIELÄ!  modify(p); */
if((p = calloc(10, sizeof(char))) == NULL)
  errorf();
p[0] = 'h'; p[1] =  'i'; /* p[2] == '\0' */
modify(p);
modify(p+1);
char *q = "hello";   /*merkkijonovakio*/
modify(q);  /* EI, EI; EEII  näin! */
```



Merkkijono parametrina ja paluuarvona (2)

(vanha merkkijono ei muutu, nyt muutos tehdään sen kopioon)

```
char *modify(const char *s) {
   char *news;  /* uusi merkkijono */
   char *ps;    /* selaa vanhaa */
   char *pn;    /* selaa uutta */
   if ((news = calloc (length(s)+1), sizeof(char))) == NULL) return NULL;
   for (ps = s, pn = news; *ps; ps++, pn++)
       *pn = *ps;
  *pn = *ps;   /* kopioi  vielä \0-merkin /*
   news[0] = toupper(news[0]);
   return news;
}
```

Muista vapauttaa muistilohko, kun sitä ei enää tarvita:
free (news);

p → tarja H.\0

```
char *p = "tarja H.";
char *q = modify(p);
```

tai

```
char *q =modify("tarja H.");
```

Tarja H.\0 ← q

q = modify(p+3);

p → tarja H.\0 q → Ja H.\0

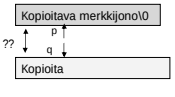
Erilaisia kopiointitapoja

- while((q[i]= p[i]) != '\0') i++;
- while((*q=*p)!='\0') {q++; p++;}
- while ((*q++=*p++) !='\0');
- while (*q++ = *p++);

Tekevät saman asian: kopioivat merkit merkkijonosta p merkkijonoon q.

Muista varata muistista tilaa merkkijonolle q ennen kopioimista.

Älä hukkaa merkkijonojen alkuja!



Muutetun merkkijonon palautus parametrina

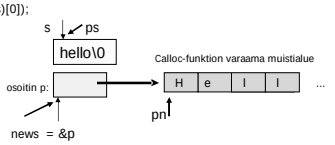
```
void modify1(const char *s, char **news) {
/* return through parameter a copy of s modified*/
   char *ps;    /* selaa vanhaa */
   char *pn;    /* selaa uutta */
   if (s == NULL)  return -1;
   if ((*news = calloc (length(s)+1), sizeof(char))) == NULL)
      return NULL;
   for (ps = s, pn = *news; *ps; ps++, pn++)
       *pn = *ps;
  *pn = *ps;   /* kopioi  vielä \0-merkin /*
  (*news)[0] = toupper((*news)[0]);
}
```

osoitin → char

news

osoitin, joka osoittaa osoittimeen, joka osoittaa char –tyyppiseen muuttujaan

```
char *p;
modify1("hello", &p);
```



Esimerkki: funktio tarkistaa, onko annettu merkkijono kelvollinen kokonaisluku joko desimaalina tai heksadesimaalina esitettynä.

```
int isNumber(const char *s) {
   if (s == NULL || s[0] == '\0')  /* tyhjä merkkijono */
         return 0;
   # onko heksaluku eli tyyppiä "0x2A68"?
   if (s[0] == '0') { /*nolla ensimmäisenä*/
      if (s[1] == '\0') return 1;  /*pelkkä nolla kelpaa*/
      if(s[1] =='x' || s[1] == 'X')  { /*heksaluku?*/
         if (s[2] == '\0') return 0; /**"0x" ei riitä*/
         for (s +=2; *s; s++) if (!isxdigit (*s)) return 0;
         return 1;  /* kelvollinen heksaluku */
      }
   }
   # onko desimaaliluku
   for (;*s; s++)
       if (!isdigit (*s)) return 0;
   return 1;
}
```

Fraasi merkkijonon läpikäyntiin:

for (p = s; *p; p++)
    käytä *p:tä;

Merkkijono ja formatoitu I/O

- "%s" sekä syötössä että tulostuksessa
  - Syötössä
    - merkkijonon alussa olevat tyhjät ohitetaan ja lukeminen aloitetaan ensimmäisestä ei-tyhjästä merkistä
    - Luetaan yksi sana eli seuraavaan tyhjään merkkiin asti
    - => scanf lukee vain yhden sanan

```
const int SIZE = 7;
char *s;
if ((s = malloc((SIZE + 1) * sizeof(char))) == NULL)
    virhetilanne();
scanf("%s", s);
```

scanf ("%7s",s);

Mitä luetaan, jos syöte on "Java language" ?

Entä, jos syöte on "language Java"?

Entä scanf ("%SIZEs",s); ?

Ei käy, sillä tulkitsee %S:n formaattiksi

Merkkijonon lukeminen

- scanf ("%s", s) koska s on osoitin
- Ennen lukemista merkkijonolle on oltava muistia varattuna
- Muistia on oltava tarpeeksi! Varmista rajoittamalla pituus.

```
if (scanf("%10s", s) !=1)
   error
```

Lukee yhden korkeintaan 10 merkin mittaisen sanan.

Varaa tilaa myös \0-merkille.

Merkkijonon tulostus

- printf("%s", str)
  - tulostaa osoittimen str osoittaman muistilohkon kaikki merkit \0-merkkiin saakka

```
char *s = "C ja Java ovat kieliä."
printf ("%s\n", s);
printf ("%s\n", s+5);
```

C ja Java ovat kieliä.\0

s s+5

Tulostus:

```
C ja Java ovat kieliä.
Java ovat kieliä.
```


Esimerkki

Ohjelma lukee yhden korkeintaan 10 merkin mittaisen sanan ja tulostaa siinä olleiden pienten kirjainten lukumäärän.

```
int lower(char *s) { /* return number of l.c. letters */
  int i;
  char *q;
  for(i = 0, q = s, *q, q++)
        if(islower(*q))
          i++;
  return i;
}
```

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

```
int main() {
  const int M = 10;
  char *p;
  if((p = calloc(M + 1, sizeof(char)) == NULL)
     return EXIT_FAILURE;
  if(scanf("%10s", p) != 1)  return EXIT_FAILURE;
  printf("%d lower case letters in %s\n", lower(p), p);
  return EXIT_SUCCESS;
}
```

```
for(i = 0, q = s; *q; q++)
        if(islower(*q))
          i++;
```

```
p = s; i=0;
while (*p++)
if (islower(*p))  i++;
```


sscanf ja sprintf

merkkijonosta lukeminen ja merkkijonoon kirjoittaminen

```
int sscanf (s, "format", arguments)
```

- merkkijonot luvuiksi

```
if (sscanf (p,"%lf", &sd) !=1) ...
if(sscanf(s+6, "%d%f", &i, &d) != 2) ...
```

```
int sprintf (s, "format", arguments)
```

- merkkijonon kokoamiseen 'osista'

```
sprintf(p, "Syötetty luku oli %d", j);
sprintf(s, "%s %d %f", "test", 1, 1.5);
```

5678.993456

Luvut 34 46.998\0

test 1 1.5\0

Tehokkaampiakin tapoja näihin toimintoihin on!

p:n ja s:n oltava alustettuja merkkijonoja, joille on varattu riittävästi tilaa muistista calloc-funktiolla.

fgets ja fputs

tiedosto **ó** merkkijono (myös stdin ja stdout)

Rivi kerrallaan lukeminen ja kirjoittaminen

```
char* fgets(char *buf, int n, FILE *in);
```

Lukee yhden rivin, mutta enintään n-1 merkkiä tiedostosta in ja tallettaa sen muistilohkoon buf. Tallettaa myös rivinlopetusmerkit. fgets onnistuessaan aina kirjoittaa muistilohkoon viimeiseksi \0-merkin.

```
int fputs (const char *s, FILE *out);
```

Kirjoittaa merkkijonon s (ilman \0-merkkiä) tiedostoon out.

gets ja puts

stdin-syöttö ja stdout-tulostus

```
char * gets(char *buf);
```

Lukee aina koko rivin (ei siis korkeintaan tiettyä määrää) eikä talleta rivin lopetusmerkkiä muistilohkoon. ÄLÄ KÄYTÄ !

```
int puts(const char *buf);
```

Kirjoittaa merkkijonon ja päättää sen aina rivinvaihdolla.

Rivi kerrallaan lukemisen yleinen ongelma:
aina oletettava jokin maksimipituus jolle varataan tilaa!

<string.h>

Merkkijono-operaatiot:

- Runsaasti merkkijonoja käsitteleviä funktioita
  - Merkkijonon pituus

```
size_t strlen (const char *string);
```

Huom! merkkijonojen pituuksien vertailu

```
if (strlen(x) >= strlen(y)) ...
if ((int)strlen(x) – SIZE  >= 0) ...
```

  - Merkkijonon kopiointi

```
char *strcpy(char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t n);
```

  - Merkkijonon liittäminen toiseen

```
char *strcat(char *dest, const char *src);
char *strncat(char *dest, const char *src, size_t n)
```

Varmista, että

- kopio on alustettu merkkijono
- kopiolle on varattu tarpeeksi muistitilaa
- kopioon tulee \0-merkki.

<string.h>

Lisää merkkijono-operaatioita

- Merkkijonojen vertailu (merkki merkiltä)

```
int strcmp(const char *s1, const char *s2);
int strcnmp(const char *s1, const char *s2, size_t n);
```

Palauttaa
<0 jos s1 < s2
0 jos s1 == s2
>0 jos s1 > s2

- Merkin tai merkkijonon etsiminen

```
char *strchr(const char *str, int c);
char *strrchr(const char *str, int c);
char *strstr(const char *str, const char *substr);
```

ensimmäinen c:n esiintymä
viimeinen c:n esiintymä
etsii merkkijonoa

```
size_t strspn(const char *str, const char *set);
size_t strcspn(const char *str, const char *set);
char *strpbrk(const char *str, const char *set);
```

set:in kuuluvia alussa
set:iin kuulumatomia alussa
1. set:iin kuuluvaan viite

merkkijoukon merkkien esiintyminen merkkijonossa, palauttaa esiintymispaikan (ohitettujen merkkien lukumääränä tai osoittimena)

<string.h>

Yhä merkkijono-operaatiota

- Merkkijonon jako 'sanoiksi' (token) erotusmerkein

/home/avirta/kurssi/C/2006/testi.c\0
str
char *sep "\";

```
char *strtok(char *str, const char *sep);
```

strtok(NULL, sep);

ensimmäisellä kerralla annettava merkkijono; seuraavilla kerroilla voi antaa sen tilalla NULL:in

- Tällöin jatkaa seuraavan sanan etsimistä samasta merkkijonosta.

Tee kopio ja käytä sitä!

Muuttaa merkkijonoa: \0 jokaisen havaitun sanan perään. Palauttaa str:n osia.

- Merkkijono numeroiksi <stdlib.h>

p s

```
double strtod(const char *s, char **p);
long strtol(const char *s, char **p, int base);
unsigned long  strtoul(const char *s, char **p, int base);
```

fail: s:n alkuun
success: eka muuttamaton merkki

ANSI C –versioita, jotka korvaavat vanhat: atof, atoi ja atol

Esimerkki: Merkkijonon riisuminen turhista edessä ja perässä olevista tyhjämerkeistä tai muista turhista merkeistä

Heippa vaan | \0 → Heippa vaan\0

Mäkinerin kirjan esimerkki 9-13

```
/* strip from s leading and trailing characters from
 * set. For example:
 * char *p = strip(" ,hi, how are you,", " ,");
 */
char *strip(const char *s, const char *set) {
 int start = strspn(s, set); /* leading character*/
 int end;                    /* trailing characters */
 char *kopy;
 int length = strlen(s); /*merkkijonon pituus*/

 if(length != start) { /* there are characters not in set */
  for(end = length; end > 1; end--) /* trailing */
    if(strchr(set, s[end]) == NULL)  /* onko poistettava merkki */
      break;
  length = end - start + 1; /* left after strip */
```


Esimerkki jatkuu:

```
/*char *strip() continued */
 if((kopy = calloc(length + 1, sizeof(char)))==NULL)
    return NULL;
 memcpy(kopy, s + start, length);
 kopy[length] = '\0';
 } /* length != start */

 else { /* here, no characters in s */
  if((kopy = calloc(length + 1, sizeof(char)))==NULL)
     return NULL;
  strcpy(kopy, s);
  }

 return kopy;
}
```

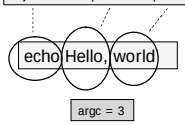

Komentoriviparametrit

```
int main (int argc, char **argv);
int main (int argc, char *argv[]);
```

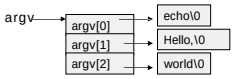
argc merkkijonojen lukumäärä

argv osoitin osoitinlohkoon

ohjelmannimi parametri1 parametri 2 ...



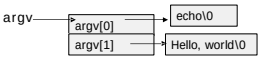
argc = 3



echo "Hello, world"

argc = 2

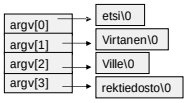
" " -merkit toimivat joissakin järjestelmissä!



Osoittaminen komentoriviargumentteihin

etsi Virtanen Ville rektiedosto

argc: 4 argv:



Standardi vaatii myös, että argv[argc] on NULL.

argv[0] tai *argv osoittaa 1. parametriin eli ohjelman nimeen ("etsi"),
argv[1] tai *(argv+1) osoittaa 2. parametriin ("Virtanen")
argv[2] tai *(argv+2) osoittaa 3. parametriin ("Ville")
argv[3] tai *(argv+3) osoittaa 4. parametriin ("rektiedosto")

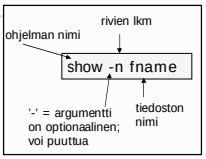
argv[0][0] tai (*argv)[0] tai **argv osoittaa 1. parametrin 1. merkkiin
argv[2][4] tai (*(argv+2))[4] tai *(*(argv+2)+4) osoittaa 3. parametrin 5. merkkiin.

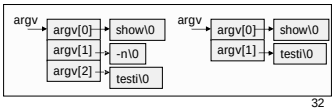
Komentoparametrien lukumäärän tarkistus

```
/* Tarkista komentoriviparametrien lukumäärä! */
int main(int argc, char **argv) {
 ....
   switch(argc) {
   case  4: ...  /* kaikki  tiedot annettu komentorivillä*/
   case 3:  ...  /*OK!  käytetään oletusarvoa*/
   default: fprintf(stderr, "Väärä käyttötapa: %s .. \n",
              argv[0]);  /*Voisi myös kertoa oikean käytötavan!*/
              return EXIT_FAILURE;
}
```

Komentoriviparametrien käyttö:
Tiedoston rivien tulostus näytölle

```
#define DEFAULT 10
#define MAX     80
/*tulostaa  näytölle tiedoston n ensimmäistä riviä */
int display(const char *fname, int n,  int Max);
int main(int argc, char **argv) {
   int lines = DEFAULT;
   switch(argc) {
   case 3: /* selvitä rivien lukumäärä argumentti */
      if(argv[1][0] != '-' ||  sscanf(argv[1] + 1, "%d", &lines)!=1 || lines <= 0)
      return EXIT_FAILURE;
      argv++;          /* no break: retrieve filename */
   case 2:  if(display(argv[1], lines, MAX) == 0)  return EXIT_FAILURE;
            break;
   default:  return EXIT_FAILURE;
 }
 return EXIT_SUCCESS;
}
```





Ohjelma laskee ja tulostaa parameteina annettujen lukujen summan

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv) {
   int i;
   double luku,  summa=0.0;
   char **p;
   char *s;
   if (argc==1)
    {
      printf ("Parametreja voi olla vaihteleva");
      printf (" määrä ja ne voivat olla");
      printf (" kokonaislukuja tai liukulukuja");
      printf ("\nKäyttö: SUMMA arg1 arg2 ...
            argn\n");
      exit(0); /* lopetetaan ohjelman toiminta
     */
       }
```

```
if ((s = calloc(80, sizeof(char))) ==
     NULL) return 1;
p = &s;
for (i=1; i<argc; i++)
   {
      luku = strtod(argv[i], p);
       summa=summa+luku;
        }
        printf ("Lukujen summa on
    %.2lf\n",summa);
    return 0;
   }
```


Mitä opittiin?

- Merkkijonojen alustaminen ja käsittely
- Merkkijonojen syöttö ja tulostus
- Standardikirjaston funktioita merkkijonojen käsittelyyn
- Komentoriviparametrien käyttö

Ensi kerralla

- Taulukoiden käsittelyä
  - Yksiulotteiset taulukot
    - Määrittely, kopiointi, vertailu
    - Taulukko parametrina
    - Alustus ja talletus
  - Moniulotteiset taulukot
  - Dynaamiset taulukot