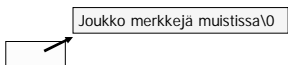


Merkkijonot Komentoriviparametrit

Luento 7
29.9.2006

Merkkijonot (strings) (Müldnerin kirjan luku 9)

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

Osoitin miono: 

- n Merkkijonojen käsittelyä varten standardikirjasto string.h

Luennon sisältö

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

Merkeistä

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

Escape-merkit (samat kuin Javassa)
"n" = rivinvaihtomerkki
"t" = tabulaattori
"v" = pystysuora tabulointi
"b" = peruutusmerkki
"r" = rivinalkuunpalautusmerkki
"f" = sivunvaihtomerkki
"a" = hälytysmerkki, yleensä äänimerkki
"v" = kenoviiva
"v" = heittomerkki
"v" = lainausmerkki

"\0" = merkkijonon lopetusmerkki

```
if((c=getchar()) == EOF) ...  
if((c=getchar()) != '\n') ...  
if((c=getchar()) == EOF) ...  
while((c = getchar()) != '\n') ...  
while ((c=getchar()) != EOF) ...  
while(getchar() != '\n') :
```

Kirjastofunktioita merkkien käsittelyyn

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

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

Merkkien luokittelufunktiot

Alfanumeeriset
n int isalnum(int c)
n int isalpha(int c)
n int islower(int c)
n int isupper(int c)
n int isdigit(int c)
n int isxdigit(int c)
n int isodigit(int c)

is c an alphanumeric
is c an alphabetic letter
is c a lower case letter
is c an upper case letter
is c a digit
is c a hexadecimal digit
is c an octal digit

Muut merkit
n int isprint(int c)
n int isgraph(int c)
n int ispunct(int c)
n int isspace(int c)

is c printable (not a control character)
is c printable (not a space)
is c printable (not space or alphanumeric)
is c whitespace

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

Kirjastofunktioilla siirrettävämpää koodia!

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:


```

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

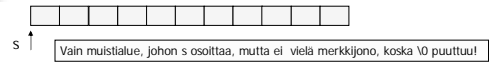
muistinvaraus-fraasi

Mitä tehdään varauksen epäonnistuessa?

C-ohjelmointi
Syksy 2006

7

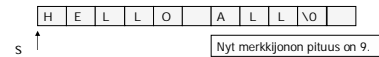
Merkkijonon alustaminen ja merkkijonon pituus



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



Merkkijonoon voidaan tallettaa lisää merkkejä:



C-ohjelmointi
Syksy 2006

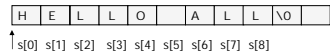
8

Merkkijonon muistinvaraus ja i:nnen merkin osoittaminen

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

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

- viittaus merkkijonon `s`:n i:nneen merkkiin `s[i]`:llä
($0 \leq i < \text{merkkijonon pituus}$)

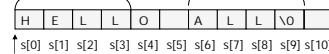


`S[3] = 'L' ja s[6]='A'`

C-ohjelmointi
Syksy 2006

9

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 päätty `\0`-merkkiin.

C-ohjelmointi
Syksy 2006

10

Merkkijonovakio

- `char *nimi = "Tarja Halonen"`

- Vakion sisältöä

ei saa muuttaa

Tarja Halonen\0

- On voitu tallettaa esim. 'read only' -alueelle
- Ei saa välittää parametrinaan 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`

C-ohjelmointi
Syksy 2006

11

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, EII näin! */
            
```



C-ohjelmointi
Syksy 2006

12

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; /* kopioi vielä \0-merkin */
    news[0] = toupper(news[0]);
    return news;
}
```

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

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

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

```
q = modify(p+3);
```

p ↓
tarja H.\0

Tarja H.\0

q ↑

C-ohjelmointi
Syksy 2006

13

Erilaisia kopiointitapoja

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

Tekevät saman asian: kopioivat merkit merkkijonosta p merkkijonoon q.
Muista varata muistia tilaa merkkijonolle q ennen kopiointia.
Ala hukkaa merkkijonojen alkujat

Kopioitava merkkijono\0

??

Kopioita

C-ohjelmointi
Syksy 2006

14

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; /* kopioi vielä \0-merkin */
    (*news)[0] = toupper((*news)[0]);
}
```

osoitin → i char

news osoitin, joka osoittaa osoittimeen, joka osoittaa char -tyyppiseen muuttujaan

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

s ↓ ps
hello\0

osoitin p: →

news = &p

Calloc-funktion varaama muistialue

H e l l o

pn ↑

C-ohjelmointi
Syksy 2006

15

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;

C-ohjelmointi
Syksy 2006

16

Merkkijono ja formatoitu I/O

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

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

Entä scanf ("%SZs", s);

Mitä luetaan, jos syöte on "Java language"?

Entä, jos syöte on "language Java"?

Ei käy, sillä lainausmerkkien sisällä ei käsitellä makroja!

C-ohjelmointi
Syksy 2006

17

Merkkijonon lukeminen



- n scanf ("%s", s) koska s on osoitin
- n Ennen lukemista merkkijonolle on oltava muistia varattuna
- n 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.

C-ohjelmointi
Syksy 2006

18

Merkkijonon tulostus

`n printf("%s", str)`

`n` 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

```
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;
}
```

```
int main() {
const int M = 10;
char *p;
if ((p = calloc(M + 1, sizeof(char))) == NULL)
return EXIT_FAILURE;
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;
}
```

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

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

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

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

scanf ja sprintf

merkkijonosta lukeminen ja merkkijonoon kirjoittaminen

```
int sscanf (s, "format", argument)
n merkkijonon kokoamiseen 'osista'
n merkkijonot luvuiksi
if (sscanf (p, "%lf", &sd) != 1) ...
if (sscanf(s+6, "%d%e", &i, &d) != 2) ...
```

```
int sprintf (s, "format", arguments)
sprintf(p, "Syötetty luku oli %d", j);
sprintf(s, "%s %d %e", "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`-funktioilla.

fgets ja fputs

tiedosto **f** 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 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.

`int puts(const char *buf);`

Kirjoittaa merkkijonon ja päättää sen aina rivinvaihdolla.

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

Merkkijono-operaatiot:

<string.h>

`n` Runsaasti merkkijonoja käsitteleviä funktioita

`n` Merkkijonon pituus

`size_t strlen (const char *string);`

Huom! merkkijonojen pituuksien vertailu

```
if (strlen(x) >= strlen(y)) ...
```

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

`n` Merkkijonon kopiointi

```
char *strcpy(char *dest, const char *src);
```

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

`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
- kopiointiin tulee `\0`-merkki.

<string.h>

Lisää merkkijono-operaatioita

Merkkijonon vertailu (merkki merkillä)

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

Merkin tai merkkijonon etsiminen

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

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

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 aluksia
 set:in kuulumattomia aluksia
 !_set:in kuuluvaan viite

merkkijonon merkin esiintyminen merkkijonossa, palauttaa esiintymäpaikan (ohitettujen merkkien lukumääränä tai osoittimena)

C-ohjelmointi
 Syksy 2006 25

<string.h>

Yhä merkkijono-operaatioita

Merkkijonon jako 'sanoiksi' (token)
 erotusmerkein

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

ensimmäisellä kerralla annettava merkkijono: seuraavilla kerralla voi antaa sen tilalla NULL:in
 Talloin jatkaa seuraavan sanan etsimistä samasta merkkijonosta.

Muuttaa merkkijonoa: \0 jokaisen havaitun sanan perään.

Merkkijono numeroiksi

```
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);
```

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

fail: s:n alkuun success: eikä muuttamaton merkki

C-ohjelmointi
 Syksy 2006 26

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

Heippa vaan | \0 → Heippa vaan\0 Muldnerin 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 s */
    for(end = length; end > 1; end--) /* trailing */
      if(strchr(set, s[end]) == NULL) /* onko poistettava merkki */
        break;
    length = end - start + 1; /* left after strip */
  }
}
```

C-ohjelmointi
 Syksy 2006 27

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;
}
```

C-ohjelmointi
 Syksy 2006 28

Komentoriviparametrit

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

argc merkkijonon lukumäärä
argv osoitin osoitinlohkoon

ohjelmannimi parametri1 parametri 2 ...

echo Hello world
 argc = 3

echo "Hello, world"
 argc = 2

**-merkit toimivat jossakin järjestyksessä

argv → argv[0] → echo\0
 argv → argv[1] → Hello, \0
 argv → argv[2] → world\0

argv → argv[0] → echo\0
 argv → argv[1] → Hello, world\0

C-ohjelmointi
 Syksy 2006 29

Osoittaminen komentoriviparamentteihin

etsi Virtanen Ville rek tiedosto

argc: 4 argv: → argv[0] → etsi\0
 argv[1] → Virtanen\0
 argv[2] → Ville\0
 argv[3] → rek tiedosto\0

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 ("rek tiedosto")

argv[0][0] tai (*argv)[0] tai **argv osoittaa 1. parametriin 1. merkkiin
 argv[2][4] tai (*(argv+2))[4] tai *((argv+2)+4) osoittaa 3. parametriin 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 käytötapa: %s ..\n",
        argv[0]); /*Voisi myös kertoa oikean käytötavan!*/
        return EXIT_FAILURE;
    }
}

```

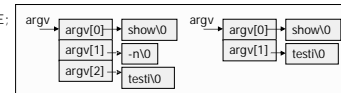
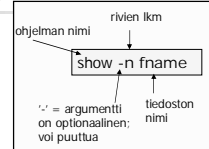


Komentoriviparametrien käyttö: Tiedoston rivien tulostus näytölle

```

#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 summa

```

#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
%.2f\n",summa);
return 0;
}

```

Mitä opittiin?

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

Ensi kerralla

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