

C-ohjelmointi, kevät 2006

Taulukot

- Yksiuotteiset taulukot
- Moniuotteiset taulukot
- Dynaamiset taulukot
- Binääritiedostot

Luento 8
21.3.2006

Luennon sisältö

- Taulukoiden käsittelyä
 - Yksiuotteiset taulukot
 - Määrittely
 - Vertailu
 - Alustus
 - Vakiotaulukko
 - Taulukko parametrina
 - Moniuotteiset taulukot
 - Dynaamiset taulukot
- Binääritiedostot

Taulukot (arrays) (Müldnerin kirjan luku 10)

Yksiuotteiset taulukot

- C:ssä taulukot ovat staattisia (koko tiedettävä ennen kääntämistä), mutta muuten samankaltaisia kuin Javassa.
- Alkaa aina nollassa
- määrittely: `type arrayName[size];`

```
int luvut[20];  
char *nimef[5*10+1];
```

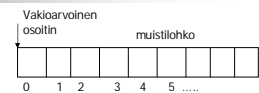
```
#define SIZE 10  
int taulu1[SIZE];
```

```
const int koko = 20; /* vain funktioissa */  
/* int taulu2[koko]; tässä laitton */  
int foo(){  
    int taulu2[koko];
```

Siirrettävyys:
warning:
ISO C90 forbids variable-size
array 'taulu'
(ISO C99 sallii käytön, kuten
myös gcc)

Mikä taulukko on?

- Taulukko on osoitin
 - Vakioarvoinen eli osoittaa aina samaan paikkaan
 - Osoittaa muistilohkoon, josta on varattu tilaa taulukon alkiolle



Muistilohkossa on tilaa taulukon koon ilmoittamalle määrälle taulukon tyyppin kokoisia olioita

```
int luvut[20];
```

Tilaa 20 kokonaisluvulle
↑
luvut

```
char nimet[5*10];
```

Tilaa 51 char-tyyppiselle osoittimelle
↑
nimet

Viittaukset eri alkioidiin: luvut[0], luvut[1], luvut[2] ... luvut[19]

HUOM! Ajoaikana järjestelmä ei tarkista indeksien oikeellisuutta. Viittaus luvut[20] ei välttämättä aiheuta suoritusvirhettä, vaan ohjelma vain toimii, miten sattuu toimimaan!

Yleisiä virheitä määrittelyssä ja käytössä

- Taulukon koko ilmoitettava vakiona
 - `int lkm=5; int taulu[lkm]; /*virheellinen*/`
- Taulukko on vakioarvoinen osoitin, jonka arvoa ei saa muuttaa
 - `int luvut[20];`
 - `char *p;`
 -
 - `luvut = p; /*ei näin*/`
 - `p=luvut; /*ihan OK! Nyt p:kin osoittaa samaan*/`
- Varo sivuvaikutuksia
 - `a[i] = i++; /*toiminta riippuu toteutuksesta!*/`
 - Kumpi tehdään ensin, kasvatusta vai käyttö?
ensin `a[i]` ja sitten vasta `i++`?
vai ensin `i++` ja sitten `a[i]`?



luvut → []
p ↑
Luvut osoittaa aina tähän muistilohkoon!

```
4 5  
4 5  
Kumpi??
```

Taulukko-osoitin ≠ tavallinen osoitin

- `int luvut[20]`
 - Vakio
 - Osoittaa aina samaan tietyn kokoiseen muistialueeseen
 - `sizeof(luvut)` on 20 kokonaisluvun tarvitsema tavunäärä eli koko taulukon koko

- `int *osoitin`
 - Muuttuja, joka voi osoittaa eri paikkoihin
 - Mitään muistialuetta ei ole varattu
 - `sizeof(osoitin)` on osoittimen tarvitsema tavunäärä

Kokojen tulostaminen

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv) {
    int *taulu[20];
    int *s;
    printf ("Taulukon 'taulu' koko: %d\n", sizeof(taulu));
    printf ("Osoittimen 's' koko: %d\n", sizeof(s));
    printf ("Taulukon alkion koko: %d\n", sizeof(taulu[0]));
    printf ("Taulukon alkoiden lukumäärä: %d\n",
           sizeof(taulu)/sizeof(taulu[0]));
    return 0;
}
```

```
int *taulu[20]; int *s;
printf ("Taulukon 'taulu' koko: %d\n", sizeof(taulu)); /* => 80 */
printf ("Osoittimen 's' koko: %d\n", sizeof(s)); /* => 4 */
printf ("Taulukon alkion koko: %d\n", sizeof(taulu[0])); /* => 4 */
printf ("Taulukon alkoiden lukumäärä: %d\n", sizeof(taulu)/sizeof(taulu[0])); /* => 20 */
```

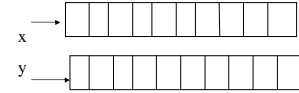
C-ohjelmointi
Kevät 2006

Liisa Marttinen

7

Taulukkojen vertailu: sama sisältö?

```
#define SIZE 10
int x[SIZE];
int y[SIZE];
```



```
int *px,*py;
for (px=x, py = y; px<x+SIZE; px++, py++)
    if (*px!=*py) .....
```

Toimii vain jos
saman kokoisia!
Entä jos eivät ole?

```
for (px=x; px<x+SIZE; px++)
    if (*px!=*y[px-x]) .....
```

```
for (i=0; i<SIZE; i++)
    if (x[i] != y[i]) ....
```

Entä vertailu X==Y??
Mitä tässä verrataan?

C-ohjelmointi
Kevät 2006

Liisa Marttinen

8

Taulukon alustaminen

n Taulukko alustetaan antamalla sen alkoiden arvot muodossa

```
{a1, a2, ..., an}
```

Esim.

```
int t[] = {1, 2, 3};
```



Taulukon nollaus:
int taulu[10000]={0};

```
int t[3] = {1, 2, 3};
```



Alkoiden lukumäärä määrää
taulukon koon!

```
int t[3] = {1, 2};
```



Täytetään järjestyksessä.
Loput nolilla.

```
int t[3] = {1, 2, 3, 4};
```



Kolmen alkion taulukossa
ei ole tilaa arvolle '4' =>
virhet!

C-ohjelmointi
Kevät 2006

Liisa Marttinen

9

Vakioksi määritelty taulukko

```
n const int days[] = {1, 2, 3, 4, 5, 6, 7}
```

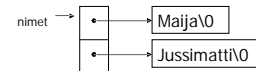
osoittimen days tyyppi: const int * const

n Merkkijonovakiot:

```
char nimi[] = "Pekka";
```



```
char *nimet[] = {"Maija", "Jussimatti"};
```



C-ohjelmointi
Kevät 2006

Liisa Marttinen

10

Taulukon kopiointi

```
n for (i=0; i<SIZE; i++) x[i] = y[i];
```

Oltava saman kokoisia!
Entä jos eivät ole?

```
kokox=sizeof(x);
```

```
kokoy = sizeof(y);
```

```
if (kokox < kokoy) koko = kokoy;
else koko = kokox;
```

```
for (i=0; i<koko; i++) x[i] = y[i];
```

koko = kokox < kokoy ? kokox : kokoy;

C-ohjelmointi
Kevät 2006

Liisa Marttinen

11

Taulukko parametrina

n funktion määrittelyssä voi käyttää:

```
int miniT(double arr[], int size);
```

```
int miniP(double *arr, int size);
```

n Funktion sisällä taulukkoparametreja käsitellään aina osoittimena

=> Taulukon kokoa ei voi kysyä näin!

```
sizeof( arr ) (= osoitinmuuttujan koko)
```

C-ohjelmointi
Kevät 2006

Liisa Marttinen

12

Esimerkki:

maxmin palauttaa taulukon maksimi ja minimiarvon

```
int maxmin (double arr[], int size, double *max,
double *min) {
double *p
if (arr==NULL || size <= 0) return 0;
for (*max=*min = arr[0], p = arr+1; p<arr + size; p++) {
if(*max < *p) *max = *p;
if (*min > *p) *min = *p;
}
return 1;
}
```

Kutsu: maxmin(x, SIZE, &max, &min);

Mitä tekee seuraava kutsu?
maxmin(x+3, 5, &max, &min);

C-ohjelmointi
Kevät 2006

Lisa Marttinen

13

Paikalliset static-aulukot

```
static char* opnimi(int n) {
static char* operaattori [] =
{"lvalue", "rvalue", "push", "+", "-"};
return operaattori[n];
}
```

static => funktion
yksityinen ja
pysyvä taulukko

```
char *setName (int i){
char name1[] = "Maija";
char name2[] = "Jussi";
if (i=0) return name1;
return name2;
}
```

Paikalliselle muuttujalle varataan
tilaa pinosta joka kutsukerralla
erikseen.
char *p = setName(1);
Osoitin p jää osoittamaan siihen
kohtaan pinoa, josta tällä kertaa
varattiin tilat, mutta joka
funktion suorituksen jälkeen
vapautettiin.

Varo
roikkumaan
jaavia
osoittimia!

C-ohjelmointi
Kevät 2006

Lisa Marttinen

14

Mielivaltaisen pitkän rivin lukeminen

- Varataan tarpeeksi suuri muistilohko ja toivotaan sen riittävän.
- Varataan esim. 80 merkin muistilohko ja aina tarvittaessa kasvatetaan riville varatun muistilohkon pituutta (dynaaminen taulukko)
- Käytetään rekursiivista funktiota: aina kun varattu (esim. 80 merkin) muistialue on täynnä ja rivi yhä jatkuu, niin funktio kutsuu itse itseään. Edellisen kutsun muistialue jää talteen pinoon ja uusi suorituskertaa varaa uuden muistialueen pinosta.

C-ohjelmointi
Kevät 2006

Lisa Marttinen

15

Esimerkki: mielivaltaisen pitkän rivin lukeminen rekursiivisesti

```
#define KOKO 80
int luerivi(File *in, char **tulos){
char puskur[KOKO];
int c, i, base; /*luettu merkki, indeksimuuttuja, apumuuttuja */
static int luettu = 0; /* tämän rivipatkan koko */
for (i=0; i< KOKO; i++) {
c = fgetc(in);
if (c==EOF) { ... } /*virhetilanne */
if (c== '\n') break; /*rivi loppui*/
puskuri[i] = c;
luettu += i;
}
if (c!=EOF && c!='\n'){ /*viela luettavaa */
if (luerivi(in, tulos) == 0) { luettu=0; return 0; }
} else { /* kaikki luettu */
if ((*tulos = malloc((luettu+1) * sizeof(char))) == NULL) { luettu = 0; return 0; }
(*tulos)[koko]='\0'; /* NULL-merkki viimeiseksi */
base = luettu -i;
memcpy(*tulos +base, puskuri, i)
luettu -=i;
return 1;
}
}
```

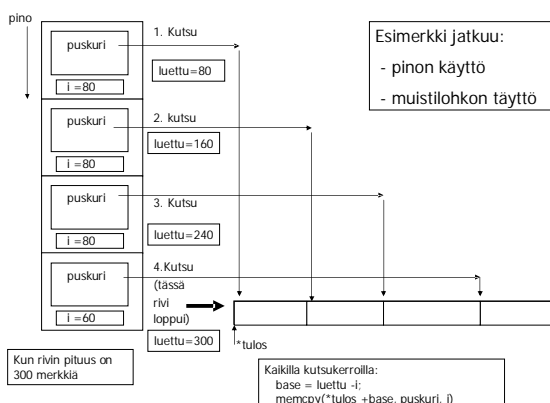
Luetaan korkeintaan 80 merkkiä puskurin. Joka kutsukerralla eri puskur, joka jää talteen pinoon.

Rekursiivinen kutsu

Kukin kutsukerta kopioi lukemansa puskurin oikeaan paikkaan yhteistä muistitilaa.

Tämä suoritetaan vain kerran: kun viimeinen patka riviä on luettu!

Vertaa rekursiivinen kertoman laskeminen lito-kursilla!



Moniulotteinen taulukko

C:n moniulotteiset taulukot ovat yksiulotteisia taulukoita, joiden alkiot ovat taulukoita

int t[3][2] = { {1,2}, {11,12}, {21,22}};

	0	1
0	1	2
1	11	12
2	21	22

```
for (i=0; i<3; i++) {
for (j = 0; j<2; j++)
printf ("%d[%d][%d] = %d\n", i, j, t[i][j]);
putchar('\n');
}
```

t[0][0] = 1 t[0][1] = 2
t[1][0] = 11 t[1][1] = 12
t[2][0] = 21 t[2][1] = 22

C-ohjelmointi
Kevät 2006

Lisa Marttinen

18

```
static char paivat [2][13] = {
    {0, 31, 28, 31, 30, 31,30,31, 30, 31,30, 31},
    {0, 31, 29, 31, 30, 31,30,31, 30, 31,30, 31}
};
```

```
lkm = paivat[karkaus][2];
```

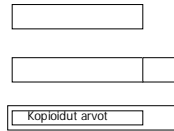
Kun karkaus ==0, niin lkm = 28,
kun karkaus ==1, niin lkm = 29

Dynaaminen taulukko

```
void* malloc (size_t koko);
void* calloc (size_t koko);
void free( void* pt);
void* realloc(void* pt, size_t koko);
```

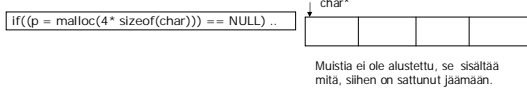
- Käytetään dynaamista muistin varaamista (malloc, calloc)
- Kun taulukolle varattu muistilohko on täynnä eikä siihen enää mahdu alkioita,
 - Yritetään kasvattaa jo varattua muistilohkoa varaamalla lisää muistia heti sen sen perästä.
 - Jos tämä ei onnistu, niin varataan jostain muualta muistista suurempi muistilohko, jonne kopioidaan pieneksi käyneen muistilohkon tiedot, ja vapautetaan tämän varaama muistitila.
 - Voidaan käyttää funktiota realloc tai laatia itse funktio, joka varaa tarvittaessa muistia (malloc, calloc), suorittaa kopiointin ja vapauttaa (free) turhaksi käyneen muistilohkon.

void on geneerinen osoitintyyppi
"a pointer to something, but we don't know yet to what kind of thing"

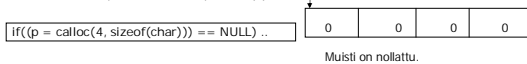


malloc-funktio (void* malloc (size_t sz);)
calloc-funktio (void* calloc(size_t n, size_t sz);)

```
n (char*) malloc(4*sizeof(char));
```



```
n calloc(4, sizeof(char));
```

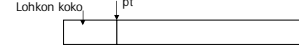


Muistin vapauttaminen: free (void free(void* pt);)

Vapauttaa osoittimen osoittaman muistilohkon

- Mistä se tietää lohkon koon?
- Lohkoa varattaessa (malloc, calloc) sen koko on talletettu muistiin juuri ennen lohkon alkua

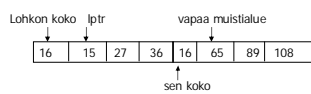
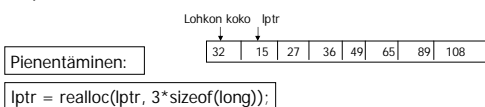
'dangling pointer':
Jää osoittamaan poistettua muistialueeseen.



- Vain calloc:lla ja malloc:lla varattujen alueiden vapauttamisen
- Vapauta kukin alue vain yhden kerran!

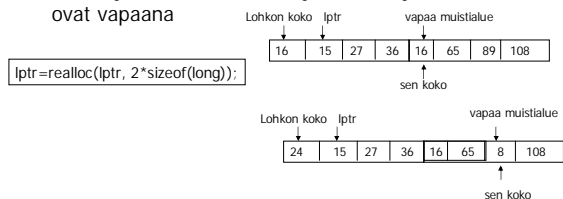
Taulukon koon muuttaminen: realloc (void* realloc(void* pt, size_t sz);)

malloc- tai calloc-funktiolla varatun muistilohkon, esim. taulukon, kokoa voidaan sekä suurentaa että pienentää.



Varatun muistilohkon koon kasvattaminen realloc-funktiolla (1)

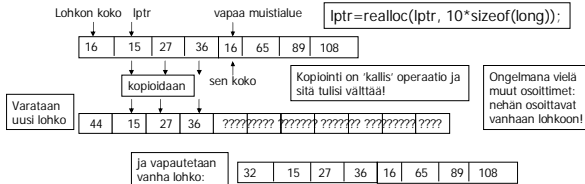
Käyttöjärjestelmä pitää kirjaa malloc:lla ja calloc:lla varattujen alueiden koosta ja tietää myös, mitkä alueet ovat vapaana



Varatun muistilohkon koon kasvattaminen realloc-funktiolla (1)

realloc onnistuu aina, kun vain jossain on vapaata muistia tarvittava määrä!

- Jos kasvatettavan lohkon perässä ei ole riittävästi vapaata tilaa, niin varataan muistista riittävän suuri vapaa tila, kopioidaan sinne taulukon alkiot ja vapautetaan taulukolle alun perin varattu tila.



C-ohjelmointi
Kevät 2006

Liisa Marttinen

25

Dynaamisen taulukon käyttö

Lajittelussa

```
printf("Montako alkioita lajitellaan?\n");
scanf("%i", &n);
if ((alkiot = malloc(n * sizeof(float))) ==
    NULL) mallocerror();
read_file(alkiot, n); /*luetaan alkiot*/
sort_data(alkiot, n);
```

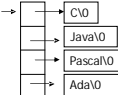
- Eri pituisten merkkijonojen (sanojen tai nimien) tallettaminen taulukkoon

```
scanf("%s20", buf); /*sanan lukeminen*/
len = strlen(buf); /*sanan pituus*/
if ((sanat[i] = malloc((len+1) *
    sizeof(char))) == NULL) mallocerror();
strcpy(sanat[i], buf);
```

alkiot



sanat



C-ohjelmointi
Kevät 2006

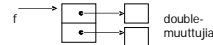
Liisa Marttinen

26

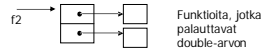
Monimutkaisia määritelmiä?

- [] -merkeillä korkeampi presedenssi kuin *-merkillä

double *f[2];



double (*f2[2])()



double (*f3())[]

f3 on funktio, joka palauttaa osoittimen double-taulukkoon.

double *(f4[])()

VIRHE! Ei voi olla funktiotaulukkoa! Vain osoitteita funktioihin.

C-ohjelmointi
Kevät 2006

Liisa Marttinen

27

Binääritiedostot

- Tiedon tiiviiseen tallettamiseen
- Ei rivirakennetta => ei voida käsitellä standardeilla välineillä
- Eivät suoraan ihmisen luettavissa
- Usein eivät ole siirrettäviä koneesta toiseen

```
fopen("tied1", "wb");
fopen("tied2", "rb");
```

C-ohjelmointi
Kevät 2006

Liisa Marttinen

28

Operaatioita binääritiedostoille

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

- hajasaantitiedostoja (random access)
- Kun tiedosto on avattu, kahva osoittaa sen hetkiseen käsittelykohtaan
- Operaatioita
 - long ftell(FILE *f) palauttaa käsittelykohdan
 - int fseek(FILE *f, long offset, int mode) siirtää käsittelykohtaa siirtymän (offset) verran mode kertoo mistä kohtaa siirto alkaa:
 - SEEK_SET tiedoston alusta
 - SEEK_CUR nykykohdasta
 - SEEK_END lopusta
 - rewind(FILE *f) kelaat tiedoston alkuun

C-ohjelmointi
Kevät 2006

Liisa Marttinen

29

Esimerkki:

Annetun tiedoston koon selvittäminen

```
long fileSize(const char *filename) {
    FILE *f;
    long size;
    if ((f = fopen(filename, "rb")) == NULL) return -1L;
    if (fseek(f, 0L, SEEK_END) == 0) { /* OK! */
        size = ftell(f);
        if (flose(f) == EOF) return -1L;
        return size;
    }
    flose(f);
    return -1L;
}
```

C-ohjelmointi
Kevät 2006

Liisa Marttinen

30

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

Binäärinen lukeminen ja kirjoittaminen

= oliolohkojen kirjoittaminen ja lukeminen

- n `size_t fread (void *buf, size_t elsize, size_t count, FILE *in);`
lukee tiedostosta in count:n ilmoittaman määrän oliolohkoja muistilohkoon, johon buf osoittaa. Oliolohkon koon ilmoittaa elsize.
Palauttaa luettujen objektien määrän. Virhetilanteessa palauttaa nollan.
- n `size_t fwrite (void *buf, size_t elsize, size_t count, FILE *out);`
Kirjoittaa tiedostoon out count:n ilmoittaman määrän oliolohkoja buf:n osoittamasta muistilohkosta. Oliolohkon koon ilmoittaa elsize.
- n Olettavat, että tiedosto on jo avattu oikeaa toimintaa (lukemista tai kirjoittamista) varten.

Tekstitiedostosta binääritiedostoon ja binääritiedostosta tekstitiedostoon

- n Teksti => binääri
 - n `while (fscanf(in, "%f", &d) == 1) if (fwrite (&d, sizeof(double), 1, out) != 1)`
- n Binääri => teksti
 - n `while(fread(&d, sizeof(double), 1, in) == 1) {`
 `i++;`
 `if (i== Max) {`
 `putchar('\n');`
 `i = 0;`
 `}`
 `fprintf(out, "%f\t", d);`
 `.....`
 `}`

Ensi kerralla

- n Modulaarinen ohjelmointi
- n Kirjastofunktioiden käyttö
- n Kertausta
 - n Mitä asioita tarpeen kerrata?
 - n osoittimia