

T-106.3100, Ohjelmoinnin jatkokurssi T2

Luento 4:
Osoittimia ja taulukoita (taas kerran)
Standardikirjastoista (IO)

Tänään

- Osoittimia ja taulukoita (taas)
- Standardikirjastoja
- Osoittimia ja taulukoita (edelleen)

Taulukoista ja osoittimista

- `int taulu[10][10]; /* staattinen */`
 - kaksiulotteinen kokonaislukutaulukko
 - yhtenäinen muistialue (oikeastaan yksiulotteinen taulukko)
- `int *taulu[10]; /* puoliksi dynaaminen */`
 - yksiulotteinen taulukko int-osoittimia
 - Osoittimille ei ole vielä varattu tilaa
 - esimerkiksi: `malloc(10*sizeof(int))`

Taulukoista ja osoittimista

- Kaksiulotteinen dynaaminen taulukko

```
int **taulukko;  
taulukko = (int**)malloc(rows*sizeof(int*));  
for (int i=0; i<rows; i++)  
    taulukko[i] = (int*)malloc(cols*sizeof(int));  
...  
taulukko[x][y] = z;
```

- Miten taulukon varaama muisti tulisi vapauttaa?

Taulukoista ja osoittimista

- Taulukot funktioiden argumenttina
 - Välitettäessä taulukko funktiolle argumenttina välitetään todellisuudessa ensimmäisen alkion osoite
 - Taulukkoa ei kopioida kokonaisuudessaan aktivaatietueeseen
 - Taulukon pituus tulee tarvittaessa välittää erikseen

Taulukoista ja osoittimista

- Taulukot funktioiden argumenttina
 - Välitettäessä taulukko funktiolle argumenttina välitetään todellisuudessa ensimmäisen alkion osoite
 - Taulukkoa ei kopioida kokonaisuudessaan aktivaatietueeseen
 - Taulukon pituus tulee tarvittaessa välittää erikseen
- Kaksiulotteiset taulukot argumentteina
 - `void foo(int taulu[][COLS]) { ...`

Taulukoista ja osoittimista

- Taulukoida voidaan indeksoida helposti hakasulkujen avulla
- `void foo(int taulu[][10]) {...`
 - `taulu[i][j] == (*(taulu + i)+j)`
 - mihin taulu siis osoittaa?

Taulukoista ja osoittimista

- Taulukoida voidaan indeksoida helposti hakasulkujen avulla
- `void foo(int taulu[][MAX_COL]) {...`
 - `taulu[i][j] == (*(taulu + i)+j)`
 - mihin `taulu` siis osoittaa?
 - `int (*taulu)[MAX_COL]`
 - `taulu == *taulu`, mutta mikä on niiden ero?

Merkkijonotaulukot, komentoriviargumentit

- `main(int argc, char** argv)`

- `a.out 1 + 2`

- `argc = 4`

- `argv[0] = "a.out"`

- `argv[1] = "1"`

- `argv[2] = "+"`

- `argv[3] = "2"`

- `a.out 1+2`

- `argc = 2`

- `argv[1] = "1+2"`

Ansi-standardikirjastot

Ansi-standardikirjastot

<assert.h>	Contains the assert macro [...]
<ctype.h>	[...] used to classify characters by their types or to convert between upper and lower case [...]
<errno.h>	For testing error codes reported by library functions.
<float.h>	Contains defined constants specifying the implementation-specific properties of the floating-point library [...]
<limits.h>	
<locale.h>	For setlocale() and related constants. [...]
<math.h>	For computing common mathematical functions
<setjmp.h>	Declares the macros setjmp and longjmp, which are used for non-local exits
<signal.h>	For controlling various exceptional conditions
<stdarg.h>	For accessing a varying number of arguments passed to functions.
<stddef.h>	For defining several useful types and macros.
<stdio.h>	Provides the core input and output capabilities of the C language. This file includes the venerable printf function.
<stdlib.h>	For performing a variety of operations, including conversion, pseudo-random numbers, memory allocation, process control, environment, signalling, searching, and sorting.
<string.h>	For manipulating several kinds of strings.
<time.h>	For converting between various time and date formats.

Auttava käsi

- man-sivut
 - tiedettävä hieman, mitä etsii
- Jokin käsikirja
 - K&R
- WWW
 - <http://www.cppreference.com/>

Ansi-standardikirjastot

- <assert.h> Contains the assert macro [...]
- <ctype.h> [...] used to classify characters by their types or to convert between upper and lower case [...]
- <errno.h> For testing error codes reported by library functions.
- <float.h> Contains defined constants specifying the implementation-specific properties of the floating-point library [...]
- <limits.h>
- <locale.h> For setlocale() and related constants. [...]
- <math.h> For computing common mathematical functions
- <setjmp.h> Declares the macros setjmp and longjmp, which are used for non-local exits
- <signal.h> For controlling various exceptional conditions
- <stdarg.h> For accessing a varying number of arguments passed to functions.
- <stddef.h> For defining several useful types and macros.
- <stdio.h> Provides the core input and output capabilities of the C language. This file includes the venerable printf function.
- <stdlib.h> For performing a variety of operations, including conversion, pseudo-random numbers, memory allocation, process control, environment, signalling, searching, and sorting.
- <string.h> For manipulating several kinds of strings.
- <time.h> For converting between various time and date formats.

stdio.h

- Makroja
 - EOF
- Tyypinmäärittelyjä
 - FILE
- Tiedostonkäsittelyfunktioita
 - FILE *fopen(const char *path, const char *mode);
- Muita IO-funktioita
 - int printf(const char *format, ...)
- Perusvirtoja kuvaavat muuttujat
 - FILE *stdin, *stdout, *stderr

tulostaminen

- `fputc`, `putc`, `putchar`
 - yksi merkki
- `fputs`, `puts`
 - merkkijono
 - `puts` lisää rivinvaihdon
- `fprintf`, `printf`
 - merkkijonojen muotoiltu tulostus

printf

- `printf(const char *format, ...)`
 - ...
 - `stdout`
 - Muotoilukoodit:
 - minkä tyyppisiä muuttujia
 - varsinaista muotoilua
 - Esim:
 - ```
double pi = 3.14;
char hello[] = "Hello World";
printf("%s. and by the way Pi=%f",hello, pi);
```

# printf - muotolukoodit

- Esimerkkejä muuttujien tyypeistä:
  - %c merkki
  - %d kokonaisluku
  - %f liukuluku
  - %s merkkijono
  - %u etumerkitön kokonaisluku
  - ...

# printf - muotolukoodit

```
int a = 123;
printf("%d\n", a); "123"
printf("%5d\n", a); " 123"
printf("%-5d\n", a); "123 "

float x = 33.3456789;
printf("%f\n", x); "33.345679"
printf("%e\n", x); "3.33457e+01"
printf("%8.3f\n", x); " 33.346"
```

# fprintf

- `fprintf(FILE *stream, const char *format, ...)`
  - Kuten `printf`, mutta tiedostoon
  - Muista myös:
    - `stdin`
    - `stderr`
- `sprintf` tulostaa merkkitaulukkoon

# Syötteen lukeminen

- `fgetc`, `getc`, `getchar`
  - yksi merkki
- `fgets`, `gets`
  - merkkijono
  - käyttäjän pitää varata muisti ennen näiden käyttämistä
    - `gets` ja puskurin ylivuoto, älä käytä `gets`-funktiota
- `fscanf`, `scanf`
  - perustietotyyppien lukeminen
- `ungetc(int c, FILE *fp)`
  - Palauttaa virtaan yhden merkin

# scanf

- Muotoillun syötteen lukemiseksi
- `int scanf (char *format, ...);`
  - Palauttaa luettujen alkioiden määrän tai EOF
  - Esim:
    - ```
#include <stdio.h>
int main(void) {
    int n;
    while (scanf("%d", &n) > 0) /* entä scanf("%d:%d",...) */
        printf ("%d\n", n);
    return 0;
}
```
 - `fscanf, sscanf`

Miksi scanf ja fscanf?

- tai printf ja fprintf?
- scanf ja printf ovat oikeasti vain makroja, jotka käyttävät stdin, stdout virtoja

Tiedostojen käsittely

- Tiedosto-osoittimen avulla
 - FILE *f;
- Avaaminen
 - fopen
 - avattava ennen käyttämistä
- Sulkeminen
 - fclose
 - lopuksi varatut resurssit on vapautettava

avaaminen ja sulkeminen

- `FILE *fopen(const char *path, const char *mode);`
 - Esimerkkejä moodista: "r", "rb", "w", "rw", "a", ...
- Avauksen onnistuminen on syytä tarkistaa
 - `if ((myfile = fopen(file_name, "r")) == NULL)`
 - Goblinissa ei kannata pyrkiä kirjoittamaan tiedostoon
- `int fclose(FILE *file_pointer)`
- Käyttöjärjestelmä toki vapauttaa ohjelman varaamat resurssit aikanaan, mutta...

Esimerkki

```
int main(void)
{
    int ch;
    FILE *fp;

    if ( (fp = fopen("foo.txt", "r")) == NULL) {
        fprintf(stderr, "Error: Cannot open file!\n");
        return EXIT_FAILURE;
    }

    while((ch = fgetc(fp)) != EOF) {
        putc(ch, stdout);
    }

    fclose(fp);
    return EXIT_SUCCESS;
}
```

Merkkijonojen käsittelyä

```
#include <string.h>
```

```
strcat() /* katenointi */  
strchr() /* merkin etsiminen merkkijonosta */  
strcmp() /* merkkijonojen vertailu */  
strcpy() /* merkkijonojen kopiointi */  
strlen() /* merkkijonon pituus ilman '\0'-merkkiä */  
strtok() /* jaa merkkijono tokeneihin */
```

```
/* Lue lisää man sivuilta: man strcat  
/* Käyttäjä vastaa muistin varaamisesta ja sen riittämisestä */  
/* Katso myös vastaavat mem alkuiset fktiot */
```