Programmeerimine C-keeles

http://cs.ut.ee/~isotamm
http://nasm.ateh10.net

1. Loeng

Ain Isotamm
Jorma Rebane

Eesmärk

- Aine edukalt läbinud tudeng:
  - Teab C-keele võimalusi
  - Teab milliste ülesannete jaoks on C-keel otstarbekohene
  - Oskab C programme lugeda ja kirjutada
  - Oskab C programme kompileerida, testida ja siluda
- Mis tuleb tudengile kasuks:
  - Tunneb tugevalt C standard library funktsioone
  - Tunneb Makefile süntaksit
  - Eristab kiiret koodi aeglasest
- Lõppeesmärk:
  - Tunneme otseseid seoseid C ja masinkoodi vahel

Toimumised

- Loengud
  - 4 C-keele loengut, T 12:00
    - 08. sep, 22. sep, 06. okt ja 20. okt
  - Suuresti teoreetiline osa
  - Loengus räägitu on eeldus praktikumideks

- Praktikumid
  - 8 praktikumi, R 14:00 ja 16:00, alates 4. sep
  - Loengus õpitu detailsem lahtiseletus
  - Palju programmeerimist

C Arvestus

- Kirjutada programm Makroassembleris:
  1. Keskmise keerukusega C programm – nt: **graafiline kaardimäng**
  2. Mitu lihtsat funktsiooni – nt: **tuntuimate sort fn-ide võrldus**

- Tulemused:
  - E-mail isotamm@ut.ee : „C arvestus – Nimi Nimi“
  - Kokkupakituna – nt: **"John Smith – sorting algos.zip"**
    - Lähtekood – **peab** olema kommenteeritud
    - Makefile
    - Pilt programmist töös
    - Seletav README

ASM Arvestus

- Kirjutada programm Makroassembleris:
  1. Keskmise keerukusega ASM funktsioon – nt: **myprintf**
  2. Mitu lihtsat ASM funktsiooni – nt: **strcmp + memchr + strstr**
  3. ASM funktsioon C programmis – nt: **tekstitöötlus + ASM memchr**

- Tulemused:
  - E-mail jorma.rebane@gmail.com : „Makroassembler – Nimi Nimi“
  - Kokkupakituna – nt: **"John Smith – sorting algos.zip"**
    - Lähtekood – **peab** olema kommenteeritud
    - Makefile
    - Pilt programmist töös
    - Seletav README

Lugemismaterjal

- Iseseisev õpe suur osa ainest:
  - The C Programming Language, 2nd Edition (Kernighan & Ritchie)
  - C Programming: A Modern Approach, 2nd Edition
  - 21st Century C: C Tips from the New School
  - Expert C reference: Deep C Secrets

  - Programmeerimine C-keeles (Ain Isotamm)
  - http://cs.ut.ee/~isotamm

- C Standard Library:

  - C Library Reference: http://www.cplusplus.com/reference/clibrary/

Hello, World

```c
#include <stdio.h>


int main()
{
    printf("hello, world\n");

    getchar(); // press any key...
    return 0;
}
```

Hello, World

- Kompileerimine käsurealt:

```
gcc  hello.c  -o hello.exe
```

Süntaks

```
// reakommentaar
/* mitme rea
   kommentaar */
#include <stdio.h> // süsteemse teegi lisamine
#include "hello.h" // lokaalse teegi lisamine

// globaalsed avalikud muutujad:
int PublicStaticVar = 5;

// mooduli sisene peidetud muutuja:
static int PrivateStaticVar = 5;
```

Süntaks

```
// programmi alguspunkt (minimaalne)
int main()
{
}
/* argc – argv parameetrite arv
   argv – käsurea argumendid
   envp – süsteemimuutujad */
int main(int argc, char* argv[], char* envp[])
{
    return 0; // tagastuskood süsteemile
}
```

Süntaks

```
// funktsioonid
[return-type] function-name( [parameters,...] )

// deklaratsioon (päisfailis "hello.h"):
void hello();

// implementatsioon (programmfailis "hello.c"):
#include "hello.h"
void hello()
{
    printf("hello!\n");
}
```

Süntaks

```
void standard_types()
{
    char   c  = 'A';         // 1 bait/tähemärk
    short  s  = 32767;       // 16-bit täisarv
    int    i  = 2147483647;// masina täisarv (16/32 bit)
    long   l  = 2147483647;// 32-bit täisarv
    float  f  = 0.5f;        // ujukomaarv
    double d  = 0.5;         // topeltäpsusega ujukomaarv

    unsigned int u = 4294967295;         // märgita täisarv
    long long ll = 9223372036854775807;// 64-bit täisarv
    unsigned long long ull;   // 64-bit unsigned int
    short float sf;           // 16-bit float
}
```

Süntaks

```c
#include <stdint.h>

void standard_integer_types()
{
    int8_t i8;     // 8-bit  signed int (char)
    int16_t i16;   // 16-bit signed int
    int32_t i32;   // 32-bit signed int
    int64_t i64;   // 64-bit signed int

    uint8_t  u8;   // unsigned
    uint16_t u16;
    uint32_t u32;
    uint64_t u64;
}
```

Süntaks

```
#define ARR_SIZE 30
static int global_arr[30];
static int global_arr2[ARR_SIZE];

static int multidimensional[10][20][30];

void arrays()
{
    int arr[4] = { 0, 1, 2, 3 };
    int i = 2;
    printf("arr[%d] = %d\n", i, arr[i]);

    const int sizeInBytes = sizeof(arr);           // 16 bytes
    const int countOf = sizeof(arr) / sizeof(int); // 4
}
```

Süntaks

```
// Variable-Length Arrays (-std=C99)
void varlen_arrays(int length)
{
    int arr[length];

    // sizeof arvutatakse käivitamise ajal:

    int sizeInBytes = sizeof(arr); // length * sizeof(int)
}
```

Süntaks

```
// Struktuurid aitavad muutujaid mugavalt grupeerida

struct Vector2
{
    float x;
    float y;
};

void structs()
{
    struct Vector2 v = { 0.5f, -1.5f };
    v.x += 1.0f;
    v.y += 1.0f;

    printf("{ %f, %f }\n", v.x, v.y);
}
```

Süntaks

```
// Typedef abil saab defineerida endale mugavdatud tüübidefinitsioone

typedef unsigned int  uint;      // kasutamine:   uint i = 0;

typedef struct Vector2  Vector2; // kasutamine:   Vector2 v;

typedef char InputArray[80];     // kasutamine:   InputArray input;

typedef void (*FuncPointerType)(char* argStr); // kasutamine:

void hello(char* str) {
    printf("hello, %s\n", str);
}
int main() {
    FuncPointerType fn = &hello; // grab pointer to function hello
    fn("world!");
}
```

Süntaks

```
// Typedef lihtsustab struktuuride kasutamist
typedef struct {
    float x, y;
} Vector2;

void vec2_increase(Vector2* v)
{
    v->x += 1.0f;
    v->y += 1.0f;
}
void structs_typedef()
{
    Vector2 v = { 0.5f, -1.5f };
    vec2_increase(&v);

    printf("{ %f, %f }\n", v.x, v.y);
}
```

Süntaks

```
// funktsiooni tüübid tuleks alati Typedef abil deklareerida!

double measure(int* array, int length,
               void (*setupArray)(int* array, int length),
               void (*sortArray)(int* array, int length))
{
    setupArray(array, length);

    double t = mytimer_get_time();

    sortArray(array, length);

    return mytimer_get_time() - t; // elapsed
}
```

Süntaks

```
// funktsiooni tüübid tuleks alati Typedef abil deklareerida!

typedef void (*ArrayFunc)(int* array, int length);

double measure(int* array, int length,
               ArrayFunc setupArray, ArrayFunc sortArray)
{
    setupArray(array, length);

    double t = mytimer_get_time();

    sortArray(array, length);

    return mytimer_get_time() - t; // elapsed
}
```

Kohtumiseni Praktikumis!

• • •

R 14:00 – 16:00

R 16:00 – 18:00