Les 2 – Opdrachten

Introductie

In de workshop heb je kennis gemaakt met de basisbeginselen van de programmeertaal C. Om die basiskennis goed te laten bezinken en te kunnen toepassen is het belangrijk om daarmee te oefenen. Daar zijn deze opdrachten voor.

Je zult hopelijk merken dat de taal C meer kennis en begrip van de interne werking van computers vereist dan talen zoals C# en Java waar je eerder in de opleiding TI mee hebt gewerkt. Deze opdrachten zijn bedoeld om je te helpen daarmee vertrouwd te raken.

Instructie

Je krijgt telkens een stukje C source code als startpunt en een beschrijving van wat er geprogrammeerd moet worden. Neem die code over in je programmeeromgeving. In deze les is dat nog Visual Studio of Visual Studio Code + C compiler, later ook ESP-IDF + ESP32 board. Het is waarschijnlijk handig als je voor elke opdracht een aparte map (en evt. een apart project in Visual Studio) maakt.

Schrijf de gevraagde uitbreidingen en/of aanpassingen aan de source code. Compileer en run je code en kijk goed naar de output van zowel de compiler als jouw code.
C compilers zijn (om historische redenen) soms erg tolerant en accepteren code die eigenlijk fout is, maar geven dan wel warnings. Een warning is C is dus soms gewoon een error, **dus neem de warnings serieus**, ook als je ze in Java of C# vaak negeerde!

Doorloop de code ook eens met de debugger en check tussentijds of er gebeurt wat je verwacht. Zet breakpoints op de belangrijke plekken en laat de debugger daar dus stoppen om de waarde van variabelen te kunnen bekijken.

Het is al vaker gezegd maar we herhalen het opnieuw: het belangrijkste van de opdrachten is **niet** de juiste output maar dat wat nodig is om tot die output te komen. Wees dus kritisch op je eigen begrip van wat er gebeurt. Vraag de docent(en) om uitleg als je twijfelt of als je iets niet begrijpt. Een docent helpt je graag verder in je leerproces.

Opdracht 1.1

1. Maak een map aan voor het project waar je in gaat werken.
2. Open Visual Studio Code en open de folder via **File** >> **Open Folder…**
3. Maak een source file (b.v. main.c) met bijvoorbeeld onderstaande C source code:

```c
#include <stdio.h>
#include "main.h"

int main() {
    int a = 3;
    int b = 5;

    int c = sum(a, b);
    printf("The sum of %d and %d is %d\n", a, b, c);
    print_sum(a, b);
}

int sum(int x, int y) {
    return x + y;
}

void print_sum(int x, int y) {
    printf("Printing the sum: %d + %d = %d\n\n", x, y, sum(x, y));
}
```

4. Maak een header file voor de main.

Maak een file main.h met de volgende inhoud:

```c
#ifndef MAIN_H
#define MAIN_H

int sum(int x, int y);
void print_sum(int x, int y);

#endif
```

5. Compileer en run de code. Check dat dit foutloos compileert en dat je de verwachte output in je terminalvenster te zien krijgt.

5. Maak zelf een tweede source bestand met bijbehorend header file.
6. Bijvoorbeeld math.c en math.h
7. Include math.h in main.h
8. Verplaats de sum functie naar de math files.

Voorbeelcode van de header files zijn:
De main header file:

```
#ifndef MAIN_H
#define MAIN_H

#include ”math.h”

void print_sum(int x, int y);

#endif
```

De math header file:

```
#ifndef MATH_H
#define MATH_H

int sum(int x, int y);

#endif
```

De main en math implementatie zul je op basis van bovenstaande informatie zelf moeten kunnen maken.

Opdracht 1.2

Stap 1

Maak in een nieuw project een file main.c met de volgende inhoud:

```c
/* Opdracht 1.2 - Functie aanroepen in C */
#include <stdio.h>

struct Person
{
    char name[50];
    int bsn;
    float budget;
};


int main() {
    int bsn = 3;
    float budget = 5;

    struct Person rvr;
    rvr.bsn = bsn;

    printf("The bsn of %s is %d \n", rvr.name, rvr.bsn);

}
```

Compileer en run main.c. Controleer de output. Zorg dat deze printf de gegevens van de struct Person goed uitprint.

Opdracht 1.3

Stap 1

Maak in een nieuw project een nieuwe main.c file met de volgende inhoud:

```
/* Opdracht 1.3 - memory classes  */
#include <stdio.h>

extern int var;

int main() {
      //var = 10 ;
      printf("%d ", changeMe());
      printf("%d ", changeMe());
      return 0;
}

int changeMe() {
    static int value = 0;
    value++;

    return value;
}
```

Compileer en run deze code. Welke output verwacht je en klopt dat met wat je krijgt?

Verklaar waarom je bij de output "1 2 " te zien krijgt. Gebruik de debugger om je hypothese te controleren.

Stap 2

Haal het commentaar weg bij de regel //var = 10;.
Verwacht je een error of warning van de compiler?
Compileer en run de code en verklaar de output die je krijgt.

Stap 3

Maak een header file aan die de extern int var toewijst.
Verwacht je een error of warning van de compiler?
Compileer en run de code en verklaar de output die je krijgt.