Les 1 – 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

Zorg dat je ontwikkelomgeving probleemloos werkt. (Her)installeer dus

- Visual Studio met de C/C++ workload (zodat je C++ en C programma's kunt bouwen); OF
- Visual Studio Code plus de C/C++ toolchain van Microsoft of de mingw (minimal Gnu for Windows) toolchain

Installeer eventueel beide zodat je kunt kiezen.

Visual Studio Code gaan we in de komende weken vaker gebruiken maar Visual Studio is voor deze eerste lessen comfortabeler omdat je er al eerder mee gewerkt hebt. Kies voor nu wat je zelf prettig vindt.

Maak een source file (b.v. main.c) met onderstaande C source code:

```c
/* Hello Avans in C */

#include <stdio.h>

int main() {
    printf("Hello Avans");
    return 0;
}
```

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

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>
#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));
}
```

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
```

Compileer en run main.c. Controleer de output. Zijn er warnings of errors? Kun je die verklaren?

Stap 2

Breid main.c uit met deze code:

```
/* Opdracht 1.2 - Functie aanroepen in C */
. . .

int main() {
    . . .

    unsigned int d = difference(b, a);
    printf("The difference is %u\n", d);
    d = difference(a, b);
    printf("The difference is %u\n", d);
}

. . .

unsigned int difference(int x, int y) {
    return x - y;
}
```

Compileer de code. Zijn er warnings of errors? (Kijk in VS Code in de tab PROBLEMS)
Verklaar voor jezelf de oorzaak van de warning. Hoe zou je de warning kunnen voorkomen?

Voer de code uit en bekijk de output. Is het resultaat wat je verwacht?

Stap 3

Verander de '%u' in de printf() call in '%d' en run opnieuw. Wat voor output krijg je nu? Welke conclusie trek je daaruit?

Verander de '%d' of '%u' in een '%x' en run opnieuw. Welke getalrepresentatie is dit? Zoek dit op in de online documentatie van printf().

Stap 4

Breid main.c uit met deze code:

```
. . .
int main() {
    . . .

    printf("The square of %d is %d\n", 8, square(8));
}

. . .

square(int n) {
    return n * n;
}
```

Mis je iets in deze code? Verwacht je dat deze code compileert?
Compileer de code. Zijn er warnings of errors?
Run de code. Klopt de output?

Dit is natuurlijk geen nette functiedefinitie. Pas de code aan zodat hij er wel netjes uitziet (hint: return type).

Opdracht 1.3

Stap 1

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

```c
/* Opdracht 1.3 - if statements */
#include <stdio.h>

void do_if_tests();

int main() {
    do_if_tests();
}

void do_if_tests() {
    int value = 0;
    if (value == 0) {
        printf("value == 0\n");
    }
    if (value = 1) {
        printf("value = 1\n");
    }
    // value = 0; // Add this line in step 3
    // value = 17; // Add this line in step 4
    // value = -1; // Add this line in step 5
    /* Add these lines below in step 2
    if (value) {
        printf("value appears to be true\n");
    } else {
        printf("value appears to be false\n");
    }
    */
}
```

Waar dient deze regel code voor?

Wat klopt hier niet?

Compileer en run deze code. Welke output verwacht je en klopt dat met wat je krijgt?
Verklaar waarom je ook de output "value = 1" te zien krijgt. Gebruik de debugger om je hypothese te controleren. Waar gaat het mis? Corrigeer de source code zoals jij denkt dat goed is (hint: er ontbreekt 1 karakter).
Waarom krijg je geen warning of error? Dat is dus iets om goed op te letten in het vervolg…!

Stap 2

Haal het commentaar weg bij de regels die gemarkeerd zijn met 'step 2'.
Verwacht je een error of warning van de compiler?
Compileer en run de code en verklaar de output die je krijgt. Wat betekent dat voor de wijze waarop de taal C omgaat met boolean expressies?
Dit is kenmerkend voor de taal C: getallen kunnen als boolean expressies dienen.

Stap 3

Herhaal voor de regel gemarkeerd met 'step 3'. Is het resultaat true of false?

Stap 4

Herhaal voor de regel gemarkeerd met 'step 4'. Is het resultaat true of false?