Teil 5: Funktionen

Nutzen Sie bitte ausschließlich die Programmierumgebung VSC

Aufgabe 5.1: Verschiedene Funktionen

Erstellen Sie für folgende Funktionen eine passende Signatur

- `getCurrentDate(…)` liefert das aktuelle Datum als String in der Form "`TT.MM.JJJJ`" zurück.
- `getNumWhitespaces(…)` bekommt einen String übergeben, analysiert wie viele Leerzeichen (`whitespaces`) in dem String sind und gibt die Anzahl der `whitespaces` zurück.
- `quadraticFunc(…)` bekommt die Koeffizienten `p` und `q` einer quadratischen Gleichung (`f(x)=x²+px+q`) übergeben und das Argument `x`. Die Funktion liefert den Funktionswert an der Stelle `x` zurück.
- `getZeroPointsQuadraticFunc(…)` bekommt die Koeffizienten `p` und `q` einer quadratischen Gleichung (`f(x)=x²+px+q`) übergeben und soll die Nullstellen bestimmen. Die Funktion kann keine, eine oder zwei Nullstellen haben. Die Anzahl der Nullstellen soll als Rückgabewert zurück geliefert werden. Die Nullstellen sollen als Ausgabeparameter definiert werden.
- `swap(…)` soll zwei Argumente `x` und `y` bekommen, die vertauscht werden sollen.
- `polynom(…)` soll ein Array mit Koeffizienten vom Typ `float` übergeben bekommen, die Länge `n` des Koeffizienten Arrays und das Argument `x`. Die Funktion soll den Wert des Polynoms ($koeff_n \cdot x^n + koeff_{n-1} \cdot x^{n-1} + \cdots + koeff_1 \cdot x^1 + koeff_0$) an der Stelle `x` ausrechnen und als Rückgabewert zurück liefern.
- `sequence(…)` soll die ersten `n` Folgenglieder der Folge $a_n = 12 \cdot n \cdot (n-1)$ berechnen. Der Wert `n` wird übergeben, das Array mit den `n` Folgengliedern soll zurückgegeben werden.

Implementieren Sie nun die Funktionen und rufen Sie aus der `main()` Funktion heraus auf.

Aufgabe 5.2: Arbeiten mit Funktionen in der Projektstruktur

Erstellen Sie in VSC eine Projektstruktur und erzeugen neben der `main.c` auch eine `quadraticFunc.c` und eine `getZeroPoints.c`, in welchen Sie die beiden zugehörigen Funktionen (`quadraticFunc()` und `getZeroPointsQuadraticFunc()`) aus Aufgabe 5.1 implementieren. Erzeugen Sie dann eine Header Datei, in welcher Sie die beiden Funktionen deklarieren. Modifizieren Sie anschließend die `main` so, dass in dieser fortwährend (bis zu einem von Ihnen gewählten Abbruchkriterium) `p` und `q` vom User eingelesen werden, anschließend mittels `getZeroPointsQuadraticFunc` die Nullstellen bestimmt werden und dann mit Hilfe der Funktion `quadraticFunc` für die gefundene(n) Nullstelle(n) eine Probe gemacht wird, sofern es diese gibt.

Aufgabe 5.3: Speicherklassen

Erweitern Sie den Code aus Aufgabe 5.2 so, dass in der Funktion `quadraticFunc.c` eine Terminalausgabe stattfindet, wie viele Nullstellen bereits seit Programmstart gefunden wurden. Sie müssen also einen Zähler realisieren.

Achtung: Dies muss in der Funktion `quadraticFunc` passieren und darf nicht durch Übergabeparameter (das bedeutet: Signatur nicht ändern) oder globale Variablen (das bedeutet: Keine zusätzlichen Variablen oder Aufrufe in `main.c`) gelöst werden. Ändern Sie nur die Datei `quadraticFunc.c`.

Aufgabe 5.4: Structs in Funktionen

Legen Sie eine Struktur an für einen Punkt im 2-dimensionalen Koordinatensystem mit den Komponenten `x` und `y` für die Koordinaten.

Legen Sie eine weitere Struktur an für ein Rechteck mit einem Bezugspunkt (das kann der Mittelpunkt oder eine der Ecken des Rechtecks sein), und der Breite und Höhe als Komponenten.

Definieren Sie für beide Strukturen mit `typedef` zwei Datentypen `vec_t` und `rectangle_t`.

Erstellen Sie folgende Funktionen:

- `getArea(…)` liefert die Fläche des Rechtecks zurück
- `moveRect(…)` bekommt einen Offset übergeben für `x` und `y` Koordinate und versetzt den Bezugspunkt des Rechtecks entsprechend.
- `scaleRect(…)` bekommt eine neue Breite und Höhe des Rechtecks übergeben und skaliert das Rechteck entsprechend (hier ist die Lage des Bezugspunkts natürlich für das Ergebnis relevant).
- `intersect(…)` bekommt zwei Rechtecke übergeben und soll den Wert 1 zurückliefern, wenn die beiden Rechtecke sich überschneiden, und den Wert 0 zurückliefern, wenn die Rechtecke sich nicht überschneiden.

Aufgabe 5.5: Arrays in Funktionen

Sie sollen ein Programm entwickeln, das maximal zehn Widerstandswerte mit Hilfe einer von Ihnen zu entwickelnden Funktion `readResValue( )` einliest und in einem Array abspeichert.

Anschließend soll der äquivalente Ersatzwiderstand einer Parallelschaltung aller eingegebenen Widerstände berechnet werden mit Hilfe der Funktion `calcEquiRes( )`, welche als Parameter das Array der Einzelwiderstände übergeben bekommt.

Aufgabe 5.6: Strings in Funktionen

Geben Sie der folgenden Funktion einen selbsterklärenden Namen und verwenden diese in einer `main` für ein Array mit verschiedenen Vornamen.

```c
#include <string.h>

int explainingFuncNameWouldBeNice(char *name) {
    if (strcmp(name, "Paul") == 0) {
        return 1;
    }
    return 0;
}
```

Aufgabe 5.7: "Call by value" in "Call by reference" umwandeln

Bitte schreiben Sie den Code so um, dass aus der gezeigten "Call by value" Funktion eine "Call by reference" Funktion wird. Die Funktionalität muss identisch bleiben.

```c
#include <stdio.h>

int checkIfOdd(int checkVariable)
{
    return (checkVariable % 2);
}

int main(void){
    int oddOrEven = 42346;
    if(checkIfOdd(oddOrEven))
    {
        printf("Die Zahl ist ungerade\n");
    } else
    {
        printf("Die Zahl ist gerade\n");
    }
}
```

Aufgabe 5.8: "Call by reference" in "Call by value" umwandeln

Bitte schreiben Sie den Code so um, dass aus der gezeigten "Call by reference" Funktion eine "Call by value" Funktion wird. Die Funktionalität muss identisch bleiben.

```c
#include <stdio.h>
#define ISOK 0

int addFavoriteNumber(int* RefToAdd)
{
    *RefToAdd = *RefToAdd + 42;
    return ISOK;
}

int main(void){
    int somethingMissing = 12;
    int returnAddFunction = -1;

    returnAddFunction = addFavoriteNumber(&somethingMissing);
    if(returnAddFunction == ISOK)
    {
        printf("Jetzt ist sie vollstaendig, lautet: %d\n",somethingMissing);
    } else
    {
        printf("Da fehlt was, die Zahl lautet: %d\n",somethingMissing);
    }
}
```

Aufgabe 5.9: "Call by reference" oder "Call by value"

Sie bekommen von Ihrem Chef Funktionsbeschreibungen vorgelegt. Entscheiden Sie, ob Sie diese als "**Call by reference**" oder "**Call by value**" ausprogrammieren würden. Begründen Sie bitte Ihre Antwort und erstellen die Funktionssignatur.

Die Funktion soll:

1. den Mittelwert aller Inhalte eines `int` Arrays bestimmen.
2. mit einem gegebenen `int` Array `arrayOne` und einem weiterem `int` Array `arrayTwoTimes`, jedes Datenelement aus `arrayOne` mit zwei multiplizieren und in `arrayTwoTimes` an identischem Index ablegen (`arrayTwoTimes[i] := 2* arrayOne[i]`).
3. ein gegebenes Byte insofern manipulieren, dass das Byte um zwei Bits nach links verschoben wird und dann mit der Maske `0b01010101` das Bitweise-Und bestimmt wird.

Zur Übung: Versuchen Sie die Funktionen auch zu implementieren.