**02 Berechnung des Folgetages**

**Zusammenfassung der Problemstellung**

Das Programm soll von einem eingegebenen gültigen Datum das Datum des Folgetages berechnen und ausgeben.

Die Gültigkeit des eingegebenen Datums wird geprüft und der Folgetag wird für alle Daten ab dem 1. Januar 1583 berechnet. Schaltjahre werden mit berücksichtigt.

**Lösungsvarianten**

Für die Berechnung haben wir ein struct für das Datum erstellt und ein enum für jeden Monat. Wir haben uns für zwei zusätzliche Funktionen nebst der main-Funktion entschieden, eine Funktion welche die Gültigkeit prüft und eine Funktion, welche das den Folgetag berechnet. Wir haben uns für zusätzliche Funktionen entschieden, um verschiedene Aufgaben innerhalb des Programms klar aufzuteilen und damit Übersicht zu schaffen.

**Modulübersicht**

Da es sich um ein sehr kleines Programm handelt, befindet sich der ganze Code in main.c, aufgeteilt auf die 3 Funktionen main, teste\_gueltigkeit und berechne\_folgetag.

**Globale Variablen:**

- enum JAN, FEB, MAR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC

- struct Date

**Lokale Variablen:**

- int d, m, y (main)

- Date start\_date (main)

- int testwert (main)

- Date next\_date (main)

**Programmablauf**

Main > ruft teste\_gueltigkeit auf zur Prüfung der Korrektheit des eingegebenen Datums und gibt einen int testwert zurück > main ruft berechne\_folgetag auf > gibt das Datum des Folgetages auf der Konsole aus.

**Tests**

*test\_berechnung\_schaltjahr:* prüft die Berechnung des Folgetages im spezialfall Schaltjahr

*test\_berechnung\_monatwechsel:* prüft die Berechnung des Folgetages für einen normalen Monatswechsel

*test\_berechnung\_jahreswechsel:* prüft die Berechnung des Folgetages für einen normalen Jahreswechsel

*test\_berechnung\_erstes\_datum:* prüft die Berechnung des Folgetages für das erste gültige Datum

*test\_teste\_gueltigkeit\_vor\_erstem\_datum:* prüft, ob der Gültigkeitstest ein Datum vor dem ersten gültigen Datum ablehnt

*test\_teste\_gueltigkeit\_von\_datum\_tag:* prüft, ob der Gültigkeitstest ein ungültiges Datum ablehnt

*test\_teste\_gueltigkeit\_von\_datum\_monat:* prüft, ob der Gültigkeitstest ein ungültiges Datum ablehnt

*test\_teste\_gueltigkeit\_von\_datum\_in\_zukunft:* prüft, ob der Gültigkeitstest ein Datum in der Zukunft akzeptiert

Alle Tests liefen erfolgreich durch.

**Erkenntnisse**

Das Programm läuft fehlerfrei.

Das Datum wird vom Programm zwei Mal betrachtet. Einmal auf seine Gültigkeit im Allgemeinen (Methode teste\_gueltigkeit) und einmal spezifisch auf die Anzahl Tage pro Monat in der Methode, die den Folgetag berechnet. Hier wäre eine Vereinfachung möglich, indem beides kombiniert würde.

**Anhang A: Quelltext**

/\* ----------------------------------------------------------------------------

\* -- \_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_ -

\* -- |\_ \_| | \_\_\_\_|/ \_\_\_\_| -

\* -- | | \_ \_\_ | |\_\_ | (\_\_\_ Institute of Embedded Systems -

\* -- | | | '\_ \| \_\_| \\_\_\_ \ Zuercher Hochschule Winterthur -

\* -- \_| |\_| | | | |\_\_\_\_ \_\_\_\_) | (University of Applied Sciences) -

\* -- |\_\_\_\_\_|\_| |\_|\_\_\_\_\_\_|\_\_\_\_\_/ 8401 Winterthur, Switzerland -

\* ----------------------------------------------------------------------------

\*/

/\*\*

\*

\* @brief calculation date of next day

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

enum {Jan = 1, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec};

typedef struct {

int day;

int month;

int year;

} Date;

Date berechne\_folgetag(Date start\_date);

int teste\_gueltigkeit(Date start\_date);

/\*\*

\* @brief Read an input-date and prints out the date of the day

\* after the input-date.

\* @returns Returns EXIT\_SUCCESS (=0) on success,

\* EXIT\_FAILURE (=1) if more than one argument is given.

\*/

int main(void)

{

Date start\_date;

int d, m, y;

(void)printf("Geben Sie ein Datum ein: <Tag> <Monat> <Jahr> \n");

scanf("%d%d%d", &d, &m, &y);

start\_date.day = d;

start\_date.month = m;

start\_date.year = y;

//function call to test the input value

int testwert = teste\_gueltigkeit(start\_date);

//if the given date is ok > function call to calculate the date of the next day and print out the result

if (testwert == 0) {

Date next\_date = berechne\_folgetag(start\_date);

(void)printf("Der Folgetag ist %d %d %d\n", next\_date.day, next\_date.month, next\_date.year);

return EXIT\_SUCCESS;

} else {

return EXIT\_FAILURE;

}

return EXIT\_SUCCESS;

}

/\*\*

\* @brief calculates the date one day after the input-date

\* @param Date start\_date

\*/

Date berechne\_folgetag(Date start\_date)

{

//case: change of year

if (start\_date.month == Dec && start\_date.day == 31) {

start\_date.year++;

start\_date.month = 1;

start\_date.day = 1;

//case: change of month

} else if ((start\_date.month == Jan ||

start\_date.month == Mar ||

start\_date.month == May ||

start\_date.month == Jul ||

start\_date.month == Aug ||

start\_date.month == Oct) && start\_date.day == 31) {

start\_date.month++;

start\_date.day = 1;

} else if ((start\_date.month == Apr ||

start\_date.month == Jun ||

start\_date.month == Sep ||

start\_date.month == Nov) && start\_date.day == 30) {

start\_date.month++;

start\_date.day = 1;

//case: month february

} else if(start\_date.month == Feb && start\_date.day == 28) {

if ((start\_date.year%4 == 0 &&

start\_date.year%100 != 0)||

(start\_date.year%400 == 0)) {

start\_date.day++;

}else {

start\_date.month++;

start\_date.day = 1;

}

} else {

//case: change of day

start\_date.day++;

}

return start\_date;

}

/\*\*

\* @brief test, if the given input is a correct date

\* @param Date start\_date

\*/

int teste\_gueltigkeit(const Date start\_date)

{

//test, if it is in the range

if (start\_date.year >= 1583 && start\_date.day >= 1 &&

start\_date.day <= 31 && start\_date.month >= 1 &&

start\_date.month <= 12) {

return EXIT\_SUCCESS;

//test, if its out of the range

} else if (start\_date.day < 1 || start\_date.day > 31 ||

start\_date.month < 1 || start\_date.month > 12 ||

start\_date.year < 1583) {

return EXIT\_FAILURE;

//any other cases

} else {

return EXIT\_FAILURE;

}

}