Berechnung des Folgetages

Zusammenfassung der Problemstellung

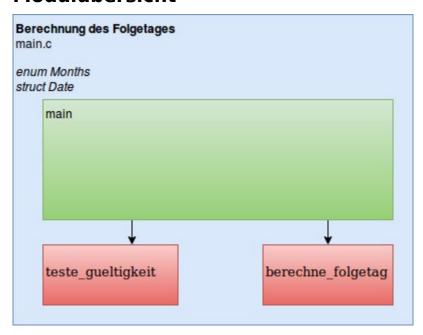
Das Programm soll von einem eingegebenen gültigen Datum das Datum des Folgetages berechnen und ausgeben.

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 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

Seite 1 10.10.2017

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

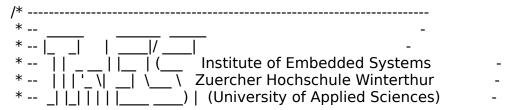
Alle Tests liefen erfolgreich durch.

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: Quelitext



Seite 2 10.10.2017

```
/**
* @brief calculation date of next day
#include <stdio.h>
#include <stdlib.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 given Date is invalid.
*/
int main(void)
  Date start date;
  int d, m, y;
  (void)printf("Geben Sie ein Datum ein: <Tag> <Monat> <|ahr> \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 der %d.%d.%d\n",
          next date.day, next date.month, next date.year);
        return EXIT SUCCESS;
      } else {
        return EXIT FAILURE;
      }
```

Seite 3 10.10.2017

```
}
/**
* @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 == |un ||
         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;
  //case: change of day
  } else {
    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
```

Seite 4 10.10.2017

Seite 5 10.10.2017