

Hardwarenahe Programmierung

Musterlösung zu den Übungsaufgaben – 24. Oktober 2019

Aufgabe 1: Seltsame Programme

Unter <https://gitlab.cvh-server.de/pgerwinski/hp/tree/master/20191024> finden Sie (unter anderem) die Programme `test-1.c`, `test-2.c` und `test-3.c`.

Was bewirken diese Programme, und warum verhalten sie sich so?

Lösung

- `test-1.c`

Hinter `return` steht ein Ausdruck mit dem Komma-Operator. Dieser bewirkt, daß der Wert vor dem Komma berechnet und ignoriert und danach der Wert nach dem Komma zurückgegeben wird.

In diesem Fall wird vor dem Komma der Wert des `printf()`-Aufrufs berechnet und ignoriert. Als Seiteneffekt gibt das Programm die Zeile `Hello, world!` aus. Anschließend wird der Wert `0` an `return` übergeben und daher `return 0` ausgeführt.

- `test-2.c`

Das Programm gibt die Zeile `Die Antwort lautet: 42` aus.

Die `if`-Bedingung ist eine Zuweisung `b = 42`, die den zugewiesenen Wert `42` zurückgibt. Weil dieser Wert ungleich Null ist, interpretiert `if` ihn als Wahrheitswert „wahr“, führt also den `if`-Zweig aus und überspringt den `else`-Zweig.

- `test-3.c`

Das Programm stürzt mit einer Fehlermeldung „Speicherzugriffsfehler“ oder „Schutzverletzung“ ab.

Der Funktionsaufruf `printf(42)` übergibt den Zahlenwert `42` als String, also als einen Zeiger auf `char`-Variable, an die Funktion `printf()`. Diese versucht, auf den Speicher ab Adresse 42 zuzugreifen, wofür aber das Programm keine Zugriffsrechte hat. Das Betriebssystem beendet daraufhin das Programm mit der o. a. Fehlermeldung.

Der String `"Die_Antwort_lautet:_"` wird nicht ausgegeben, weil Schreiboperationen aus Effizienzgründen erst nach einer abgeschlossenen Zeile (`"\n"`) durchgeführt werden.

Aufgabe 2: Kalender-Berechnung

Am 3. 1. 2009 meldete *heise online*:

Kunden des ersten mobilen Media-Players von Microsoft erlebten zum Jahresende eine böse Überraschung: Am 31. Dezember 2008 fielen weltweit alle Zune-Geräte der ersten Generation aus. Ursache war ein interner Fehler bei der Handhabung von Schaltjahren.

<http://heise.de/-193332>,

Der Artikel verweist auf ein Quelltextfragment (Datei: [aufgabe-2.c](#)), das für einen gegebenen Wert `days` das Jahr und den Tag innerhalb des Jahres für den `days`-ten Tag nach dem 1. 1. 1980 berechnen soll:

```
year = ORIGINYEAR; /* = 1980 */

while (days > 365)
{
    if (IsLeapYear (year))
    {
        if (days > 366)
        {
            days -= 366;
            year += 1;
        }
    }
    else
    {
        days -= 365;
        year += 1;
    }
}
```

Dieses Quelltextfragment enthält schlechten Programmierstil, nämlich mehrere Code-Verdopplungen:

- Die Anweisung `year += 1` taucht an zwei Stellen auf.
- Es gibt zwei unabhängige Abfragen `days > 365` und `days > 366`: eine in einer `while`- und die andere in einer `if`-Bedingung.
- Die Länge eines Jahres wird nicht durch eine Funktion berechnet oder in einer Variablen gespeichert; stattdessen werden an mehreren Stellen die expliziten numerischen Konstanten 365 und 366 verwendet.

Diese Probleme führten am 31. Dezember 2008 zu einer Endlosschleife.

Gut hingegen ist die Verwendung einer Konstanten `ORIGINYEAR` anstelle der Zahl 1980 sowie die Kapselung der Berechnung der Schaltjahr-Bedingung in einer Funktion `IsLeapYear()`.

- (a) Erklären Sie das Zustandekommen der Endlosschleife.
- (b) Schreiben Sie das Quelltextfragment so um, daß es die beschriebenen Probleme nicht mehr enthält.

Hinweis 1: Verwenden Sie Ihre eigene Funktion `IsLeapYear()`.

Hinweis 2: Schreiben Sie zusätzlich eine Funktion `DaysInYear()`.

Lösung

(a) Erklären Sie das Zustandekommen der Endlosschleife.

Das Programm startet mit demjenigen Wert für `days`, der der Anzahl der Tage vom 1. 1. 1980 bis zum 31. 12. 2008 entspricht. Die `while`-Schleife läuft zunächst solange korrekt durch, bis `year` den Wert 2008 und `days` den Wert 366 hat. (Der 31. 12. des Schaltjahres 2008 ist der 366. Tag seines Jahres.)

Die Bedingung der `while`-Schleife ist damit weiterhin erfüllt; das Programm läuft weiter.

Da 2008 ein Schaltjahr ist, ist auch die Bedingung der äußeren `if`-Anweisung erfüllt.

Da `days` den Wert 366 hat und dieser nicht größer als 366 ist, ist die innere `if`-Bedingung nicht erfüllt. Somit wird innerhalb der `while`-Schleife kein weiterer Code ausgeführt, die `while`-Bedingung bleibt erfüllt, und das Programm führt eine Endlosschleife aus.

(b) Schreiben Sie das Quelltextfragment so um, daß es die beschriebenen Probleme nicht mehr enthält.

Um das Programm zu testen, genügt es, das Datum auf den 31. 12. 1980 zu stellen, also `days` auf den Wert 366 zu setzen. Darüberhinaus muß man die Funktion `IsLeapYear()` bereitstellen (vgl. Aufgabe 1 vom 17. 10. 2019).

Der Quelltext `loesung-2-f1.c` ist eine lauffähige Version des Programms, die den Fehler (Endlosschleife) reproduziert.

Es liegt nahe, den Fehler in der `while`-Bedingung zu korrigieren, so daß diese Schaltjahre berücksichtigt. Der Quelltext `loesung-2-f2.c` behebt den Fehler auf diese Weise mit Hilfe von Und- (`&&`) und Oder-Verknüpfungen (`||`) in der `while`-Bedingung.

Der Quelltext `loesung-2-f3.c` vermeidet die umständliche Formulierung mit `&&` und `||` durch Verwendung des ternären Operators `?:`. Dieser stellt eine „`if`-Anweisung für Ausdrücke“ bereit. In diesem Fall liefert er für die rechte Seite des Vergleichs `days > den Wert 366` im Falle eines Schaltjahrs bzw. ansonsten den Wert 365.

Beide Lösungen `loesung-2-f2.c` und `loesung-2-f3.c` sind jedoch im Sinne der Aufgabenstellung **falsch**. Diese lautet: „Schreiben Sie das Quelltextfragment so um, daß es die beschriebenen Probleme nicht mehr enthält.“ Mit den beschriebenen Problemen sind die genannten drei Code-Verdopplungen gemeint, und diese befinden sich weiterhin im Quelltext. Damit ist der Fehler zwar „korrigiert“, aber das Programm ist eher noch unübersichtlicher geworden, so daß nicht klar ist, ob es nicht noch weitere Fehler enthält.

Eine richtige Lösung liefert `loesung-2-4.c`. Dieses Programm speichert den Wert der Tage im Jahr in einer Variablen `DaysInYear`. Damit erübrigen sich die `if`-Anweisungen innerhalb der `while`-Schleife, und die damit verbundenen Code-Verdopplungen verschwinden.

Etwas unschön ist hierbei die neu hinzugekommene Code-Verdopplung bei der Berechnung von `DaysInYear`. Diese ist allerdings weniger kritisch als die vorherigen, da sie nur einmal innerhalb der `while`-Schleife vorkommt und das andere Mal außerhalb derselben.

Um diese Code-Verdopplung loszuwerden, kann man das `if` durch den `?:`-Operator ersetzen und die Zuweisung innerhalb der `while`-Bedingung vornehmen – siehe `loesung-2-5.c`. Dies ist einer der seltenen Fälle, in denen ein Programm *übersichtlicher* wird, wenn eine Zuweisung innerhalb einer Bedingung stattfindet.

Alternativ kann `DaysInYear()` auch eine Funktion sein – siehe `loesung-2-6.c`. Diese Version ist wahrscheinlich die übersichtlichste, hat jedoch den Nachteil, daß die Berechnung von `DaysInYear()` zweimal statt nur einmal pro Schleifendurchlauf erfolgt, wodurch Rechenzeit verschwendet wird.

`loesung-2-7.c` und `loesung-2-8.c` beseitigen dieses Problem durch eine Zuweisung des Funktionsergebnisses an eine Variable – einmal innerhalb der `while`-Bedingung und einmal außerhalb. Der zweimalige Aufruf der Funktion `DaysInYear()` in `loesung-2-8.c` zählt nicht als Code-Verdopplung, denn der Code ist ja in einer Funktion gekapselt. (Genau dazu sind Funktionen ja da: daß man sie mehrfach aufrufen kann.)

Fazit: Wenn Sie sich beim Programmieren bei Cut-And-Paste-Aktionen erwischen, sollten Sie die Struktur Ihres Programms noch einmal überdenken.

Wahrscheinlich gibt es dann eine elegantere Lösung, deren Korrektheit man auf den ersten Blick sieht.