Die Teufelchen im Detail

Werkstatt Multiplikation
Posten: **Die Teufelchen im Detail**

Informationsblatt für die Lehrkraft

Thema:	Was ist ein korrektes Programm?
Schultyp:	Mittelschule, technische Berufsschule, Fachhochschule
Vorkenntnisse:	Grundlagen in einer höheren Programmiersprache (PASCAL)
Bearbeitungsdauer:	ca. 25 Minuten
Fassung vom:	14.10.95
Schulerprobung:	nein

Übersicht

Software-Experten der NASA programmieren im Schnitt etwa zehn Zeilen Code pro Tag. Die meiste Zeit verbringen Sie mit Analysen, Tests und Dokumentation. Das Ziel ist klar: möglichst zuverlässige und korrekte Programme.

An diesem Posten wird ein Programm untersucht, welches auf den ersten Blick korrekte Resultate liefert. Man soll sich darüber Gedanken machen, welche Spitzfindigkeiten allenfalls doch noch zu Abstürzen führen können.

Lernziele

Nach Bearbeiten diese Postens stehen die Schülerinnen und Schüler praktischen Softwareproblemen aufmerksamer gegenüber. Sie kennen einige typische Fehlerquellen in kleinen Programmen. Es wird ihnen bewusst, dass mindestens soviel Aufwand in die Implementierung eines gerissenen Algorithmus, wie in eine systematische Fehlersuche zu stecken ist.

Hinweise, Lösungen

Lösung Auftrag 1

Unter anderem stecken im vorgestellten Programm die folgenden Fehler:

- Ungültige Eingabedaten: Wird keine Zahl (z.B. ‘JUHUI’) oder eine Zahl ausserhalb des Rechenbereichs (z.B. 17345) eingegeben, stürzt das Programm kommentarlos ab.

- Randbedingungen vergessen: Für 0 und 1 liefert das Programm das falsche Resultat 2. Auch alle negativen Werte liefern das Resultat 2. Ob man das wohl so erwarten würde?

- Ungeeignete Datentypen: Grösstmögliche Eingabezahl ist 16383 mit Resultat 8192. Jede grössere Zahl produziert einen Überlauf. INTEGER (-32768 ... 32767) ist hier die falsche Wahl, da nie mit Zahlen unter Null gerechnet wird.

- Ungeeignetes Abbruchkriterium: Das Abbruchkriterium ‘UNTIL p*2>z;’ schränkt den verfügbaren Zahlenraum unnötig um die Hälfte ein. Besser wäre hier ‘UNTIL p > z div 2;’

- Schönheitsfehler: Ein- und Ausgabezeile erscheinen nicht untereinander. Gute Gestaltung hat auch etwas mit Software-Ergonomie zu tun...

Lehrer-Lernkontrolle / Test

Aufgabe 1

Du entwickelst ein kleines Hilfsprogramm, mit dem einige Kollegen arbeiten werden. Du möchtest Dir Deinen guten Ruf bewahren und prüfst Dein Programm sorgfältig. Auf welche Bereiche wirfst Du noch einen zweiten (dritten?!?) Blick, wenn Du glaubst, Dein Programm sei fertig. Nenne mindestens vier wichtige Kategorien von Softwarefehlern.

Aufgabe 2

Schreibe eine verbesserte Version des betrachteten Programmes. Du kannst zur Vereinfachung davon ausgehen, dass Eingaben nur natürliche ganze Zahlen im Bereich 0..50’000 sein werden, d.h. Du musst Dich hier nicht um Spitzfindigkeiten mit Buchstaben oder Bruchzahlen kümmern.

Lösungen und Taxierung

Aufgabe 1

Um Dir Deinen guten Ruf zu wahren achtest Du besonders auf Fehler in den folgenden Bereichen:

- Werden alle Eingabedaten auf Plausibilität getestet?
- Reagiert das Programm auch am Rand des Wertebereiches (bei 0, 1 und sonstige Spezialfälle) richtig?
- Sind die gewählten Datentypen dem Problem angepasst?
- Stimmen die Abbruchkriterien von Schlaufen? Beeinflussen sie allenfalls die Lösung oder können sie zu Endlosschlaufen führen?
- Hilft die Darstellung auf dem Bildschirm dem Benutzer, Dein Programm zu verstehen und problemlos bedienen zu können?

Die Aufgabe verlangt vom Schüler eine Reproduktion von erarbeitetem Wissen. Sie ist deshalb als **K1** zu taxieren.

Aufgabe 2

Untenstehend ein Lösungsbeispiel, in dem die untersuchten Probleme nicht mehr auftauchen. (Ob sich wohl wieder andere eingeschlichen haben?)

```
VAR
  z, p: WORD;

BEGIN
  WRITELN('Dieses Programm berechnet die grösste Zweierpotenz,');
  WRITELN('die nicht grösser als die Eingabezahl ist.');
  WRITELN;
  WRITE('--> Gib eine Zahl zwischen 1 und 50000 ein: '); READLN(z);
  IF z>0 THEN
    BEGIN
      p:=1;
      WHILE p <= z DIV 2 DO p:=p*2;
      IF z<>p THEN WRITELN('==> Die nächstkleinere Zweierpotenz von ',
                           z, ' ist ', p, '.')
              ELSE WRITELN('==> Die Eingabezahl ', z, ' ist bereits ',
                           'eine Zweierpotenz.');
    END
  ELSE
    BEGIN
      WRITELN('==> Es gibt keine Zweierpotenz unter 1.');
    END;
END.
```

Die Aufgabe fordert die Synthese eines neuen Programms aufgrund der Analyse des bereits bekannten. Sie ist als **K4** zu taxieren.

Werkstatt Multiplikation Auftragsblatt
Posten: **Die Teufelchen im Detail**

Was soll ich hier tun?

Software-Experten der NASA programmieren im Schnitt etwa zehn Zeilen Code pro Tag. Die meiste Zeit verbringen Sie mit Analysen, Tests und Dokumentation. Das Ziel ist klar: möglichst zuverlässige und korrekte Programme.

An diesem Posten untersuchst Du ein Programm, welches auf den ersten Blick korrekte Resultate liefert. Es stammt allerdings nicht von einem Programmierer der NASA. Ob es also doch noch Spitzfindigkeiten gibt, welche es zum Abstürzen bringen könnten?

Der Posten besteht aus den folgenden zwei Aufträgen:

(1) Studiere für Dich alleine das untenstehende Programm und versuche im Detail nachzuvollziehen, wie es ablaufen würde. Findest Du schon gewisse Probleme? Halte Deine Notizen stichwortartig fest.
(ca. 10 Minuten)

(2) Diskutiere das Resultat in einer Gruppe von zwei bis drei Personen. Versucht Eure Beobachtungen gewissen Kategorien von Softwareproblemen zuzuordnen.
(ca. 15 Minuten)

INPUT : Zahl z
OUTPUT : grösste Zweierpotenz, die nicht grösser als die vorgegebene Zahl z ist.

```
PROGRAM ZweiHoch;
VAR z, p: INTEGER;
BEGIN
  WRITE('? z='); READLN(z);
  p:=1;
  REPEAT p:=p*2 UNTIL p*2>z;
  WRITELN('!p= ', p);
END.
```