Wie löse ich Programmieraufgaben

Übungen in diesem Semester

Wie sehen diese aus – was ist das Ziel?

In diesem Semester sollen anhand von Problemstellungen relativ kleine und überschaubare Lösungen programmiert werden. Deshalb ist man vielleicht versucht, einfach drauf los zu programmieren. Aber glaubt mir, wenn Ihr die Aufgaben im kommenden Semester in folgender Art löst, werdet Ihr sehr viel Zeit sparen und auch ich habe nicht so aufwendige Korrekturarbeiten.

Die Ziele sind:

- die Grammatik einer Programmiersprache anhand der EBNF verstehen
- Auswirkungen von Code mit Hilfe des Hoare Triples eruieren
- Alle klassischen Konstrukte der Programmiersprache Oberon verstehen:
    1. Module & Prozeduren
    2. Anweisungen (If, While, usw.)
    3. Variablen & Zuweisungen & Scopes
    4. Konstanten
    5. Arrays & Records
    6. Parameter & Rückgabewerte
    7. Rekursion
    8. Pointer
- die Datenstrukturen lineare Liste und binärer Baum verstehen
- Programme schreiben, welche diese Funktionalitäten benutzen beziehungsweise zur Verfügung stellen

Schritt für Schritt

1. Aufgabenstellung lesen

Überlegt Euch während dem Lesen, welche Dinge Informationen sind, und welche Dinge Blabla sind. Streicht Stellen an, welche Anforderungen für Euer Programm sind.

Schreibt und denkt alles bis und mit Schritt 4 auf Papier und nicht auf dem Computer. Am Computer kann man nicht denken!

2. Abstrakte Lösung überlegen: welche Module, Prozeduren brauche ich?

Überlegt Euch, wie Ihr das Ganze programmieren würdet. Brauche ich nur ein Modul? Welche Prozeduren erledigen welche Teilaufgaben? Nehmen diese Prozeduren Parameter entgegen und geben sie Werte zurück?

Zeichnet dazu irgend eine Art Diagramm auf mit Kästchen und Pfeilen, welche zeigen, welches Kästchen welche anderen aufruft und benutzt.

3. Deklarationen überlegen: welche Variablen & Datenstruktur brauche ich?

Danach könnt Ihr Euch überlegen, wo wird was und wie gespeichert. Reichen normale Variablen oder brauche ich Records, Pointers, Listen oder Bäume?

Zeichnet immer alles auf.

4. Algorithmus in Pseudocode

Wenn Ihr wisst, was ein Modul oder eine Prozedur zu tun hat, erst dann schreibt Ihr im Textform, wie der Algorithmus dann schliesslich aussehen wird. So in der Art:

```
PROCEDURE Abspielen*(Liedtitel : ARRAY OF Char);
Teste, ob Kassette eingelegt ist
Wenn ja, dann
    Teste, ob Liedtitel auf der Kassette ist
    Wenn ja, dann
        Wiederhole so lange folgenden Abschnitt bis Lied fertig ist
            Lies Note an n-ter Stelle
            Spiel diese Note über den PC-Speaker
```

Erhöhe n um 1

Wenn Lied fertig ist, stell PC-Speaker aus

sonst,

sag dem Benutzer, das keine Kassette drinnen ist

Wie Ihr seht, ist dieser Pseudocode beinahe unser gewünschtes Programm. Man erkennt beispielsweise If-Anweisungen und While-Schlaufen.

Wenn Ihr alle Prozeduren auf dem Papier in Pseudocode skizziert habt, könnt Ihr zu Schritt 5 gehen.

5. Oberon starten

Startet Oberon, so wie es in Kapitel *Wie starte ich Oberon?* beschrieben ist.

6. Modul- und Prozedurdefinitionen und Deklarationen schreiben ohne Quellcode

Schreibt alle Module und Prozeduren mit allen Deklarationen in die entsprechenden Dokumente in Oberon und speichert sie. Füllt bis hier hin noch keinen Quellcode ein. Das heisst, alles sieht in dieser Art aus (Orte, wo noch Code hingehört werden mit `(* TODO *)` gekennzeichnet):

```
MODULE Bla;
CONST Gugus = 5;
PROCEDURE Foo*(x : INTEGER) : INTEGER;
   VAR x: INTEGER;
BEGIN
   (* TODO *)
   RETURN(x);
END Foo;
BEGIN
(* TODO *)
End Bla;
```

7. Reicht das bis hier hin? Kann ich compilieren? Fehlen Prozeduren?

Wenn Du jetzt kompilierst und es Fehler gibt, dann hast Du irgend Etwas falsch gemacht:

- Schreibfehler/Syntaxfehler, zum Beispiel END oder ; vergessen
- Prozeduren aufgerufen, die es so nicht gibt
- Prozeduren, die Werte zurückgeben sollten, aber nicht zurückgeben

Es sind natürlich noch eine Vielzahl anderer Fehler möglich. Darum geh den Code nochmals durch, und überprüfe, ob Du vielleicht sogar einen Fehler in den vorherigen Schritten gemacht hast.

Erst wenn Du fehlerfrei kompilieren konntest, gehe zu Schritt 4.

8. Module und Prozeduren mit Quellcode nach den Überlegungen aus Punkt 4 füllen

Du hast jetzt also eine Architektur gemacht, die noch gar nichts selbst ausführt, aber dennoch kompilierbar ist. Also kannst Du jetzt mit dem Auffüllen der Module und Prozeduren beginnen.

Übersetze dazu alle Pseudocode-Texte aus Schritt 4 in Oberoncode mit Variablen, Anweisungen und all den Dingen, die Oberon so anbietet.

9. Kompilieren

Wenn Du Dir sicher bist, dass sehr wahrscheinlich Alles läuft, dann darfst Du es ehrfürchtig wagen, das Ganze mit `Compiler.Compile` zu kompilieren...

10. Compilerfehler korrigieren

...und sehr wahrscheinlich hast Du auch hunderte Fehlermeldungen. Das ist ganz normal - Ruhe bewahren! Jetzt gehst Du ganz einfach zum ersten Fehler und schaust, was da passiert ist. Versuche in korrekt zu korrigieren und führe nochmals Schritt 9 aus. Korrigiere aber nie die Dinge aus Schritt 6 und 7, weil ja da alles perfekt wahr. Es müssen sich also alle Fehler an den Stellen befinden, die durch Schritt 8 entstanden sind.

Wenn dann alles fehlerfrei funktioniert, dann kannst Du zu Schritt 11 gehen.

11. Eigene Lösung mit Eingaben füttern und testen, testen, testen

Führe Dein Programm mit Eingaben aus. Teste allerlei Konstellationen von Eingaben, um zu sehen, ob sie richtig von den Prozeduren behandelt werden.

Wenn eine Prozedur einen Laufzeitfehler auslöst, dann reagiert sie nicht recht. Das nennt man dann einen Bug in einer Prozedur. Die Jagd auf diesen geht folgendermassen:

- Schauen, wo der Fehler ausgeführt wurde
- Anhand des Codes und des Pseudocodes überlegen, was falsch ging
- Versuchen, das Ganze mit einem `If` oder so zu beheben.
- Modul aus dem Speicher entfernen (mit `System.Free MeinModul`)
- Nochmals kompilieren

12. Erst wenn Du es kompilieren konntest, mir zusenden

Wenn dann alles funktioniert, kannst Du es mir zusenden. Auch wenn Du nicht alle Teile gelöst hast. Das Programm teilweise falsch reagiert und abstürzt. Hauptsache Du konntest es kompilieren.

Wenn Du irgendeinmal nicht die geringste Chance siehst eine Übung zu lösen, dann stell Dir eine einfachere Aufgabenstellung vor und versuche diese zu lösen. Das bringt mehr, als mir einfach ein vollständig nicht funktionierendes Programm, das nicht einmal kompiliert werden kann, abzugeben.

Du erhältst einen Punkt pro Übung, wenn Du mir ein Programm, das kompiliert werden konnte, abgegeben hast. Falls Du wirkliche Schwierigkeiten haben solltest, etwas zu kompilieren, kannst Du mich in der Übungsstunde oder per Email fragen, was da falsch ist.