

Konzepte der Programmiersprachen

Sommersemester 2007

Eduard Wiebe

Übungsblatt 1

Organisatorisches

Dieses Übungsblatt wird am 4. Mai besprochen.

Aufgabe 1.1: Namensbindung

Gegeben sei das folgende Programm:

```
program Namensbindung;
  x, y : integer;

  procedure p1 (y : integer);
  begin
    print (x + y);
  end p1;

  procedure p2 (y : integer);
    x : integer;
  begin
    x := y;
    p1 (y);
  end p2;

begin
  x := 0; y := 2;
  p1 (y); p2 (y);
end.
```

1. Bestimmen Sie die Ausgabe des Programms bei
 - a) statischer Namensbindung
 - b) dynamischer Namensbindung
2. Charakterisieren Sie für die Ausführung dieses Programms den Gültigkeitsbereich und die Lebensdauer jeder Variable **x** bei
 - a) statischer Namensbindung
 - b) dynamischer Namensbindung
3. Welche wesentlichen Nachteile hat die dynamische Namensbindung gegenüber der statischen Namensbindung?

Aufgabe 1.2: Aktivierungsblöcke

Gegeben sei folgendes Programmfragment in einer Sprache mit statischer Namensbindung:

```
PROGRAM Schach

    PROCEDURE Bewerte (FUNCTION f: REAL)
    BEGIN
        ... := f();          (* Punkt 1 *)
    END Bewerte;

    PROCEDURE AlphaBeta

    FUNCTION eroeffnung: REAL
    BEGIN ... END eroeffnung;

    BEGIN
        ...
        Bewerte (eroeffnung)  (* Punkt 2 *)
        ...
    END AlphaBeta;

    PROCEDURE Start
    BEGIN ... END Start;

BEGIN
    Start;                  (* Punkt 3 *)
    AlphaBeta;              (* Punkt 4 *)
    ...
END Schach.
```

1. Wozu werden die statische und die dynamische Verkettung der Aktivierungsblöcke benötigt? Welche Informationen stellen sie dar?
2. Welche Struktur hat der Laufzeitstapel (inkl. statischer und dynamischer Verweise) jeweils nach folgenden Schritten der Programmausführung?
 - a) `Schach` ruft `Start` auf (Punkt 3).
 - b) `Start` endet.
 - c) `Schach` ruft `AlphaBeta` auf (Punkt 4).
 - d) `AlphaBeta` ruft `Bewerte` auf und übergibt dabei `eroeffnung` als Unterprogramm-Parameter. (Punkt 2)
 - e) `Bewerte` ruft sein Unterprogramm-Parameter `f` auf (Punkt 1).
 - f) `f` und `Bewerte` enden.

3. Welche Struktur ergibt sich für die Situationen aus der vorigen Teilaufgabe, wenn anstelle der statischen Verkettung die Display-Technik verwendet wird?
4. Welche Vor- oder Nachteile haben Verkettung und Display-Verfahren?
5. Warum ist es in vielen Sprachen nicht erlaubt, den Kontrollfluss mit Hilfe des `GOTO` Befehls in ein anderes Unterprogramm umzulenken? Welche Implementierungsprobleme würden sich ergeben? Könnte man diese durch eine modifizierte Semantik vermeiden?

Aufgabe 1.3: Deskriptoren für Namensbindung

In welchen Fällen (in Bezug auf die tatsächliche Verwendung von Sprachkonzepten in einem Programm) ist es unabdingbar, für die Namensbindung Laufzeitdeskriptoren einzusetzen? Wann sind sie verzichtbar?

Aufgabe 1.4: Deskriptoren für Adressbindung

1. Nennen Sie einige Situationen, in denen Laufzeitdeskriptoren für die Adressbindung von Variablen eingesetzt werden müssen, und andere in denen sie nicht benötigt werden.
2. Wann wird der Deskriptor üblicherweise in einen statischen und einen dynamischen Anteil aufgeteilt?
3. Kennen Sie Programmiersprachen, in denen die Adressbindung ganz ohne Laufzeitdeskriptoren auskommt?

Aufgabe 1.4: Indirekte Funktionsaufrufe

1. Welches Problem mit der Lebensdauer von Variablen kann in Zusammenhang mit indirekten Funktionsaufrufen entstehen?
2. Warum kann dieses Problem bei direkten Funktionsaufrufen in üblichen Programmiersprachen nicht entstehen?

Aufgabe 1.5: Sprünge

1. Kann es Probleme mit Sprüngen (`GOTO`) und Namensbindung geben? Mit Sprüngen und Adressbindung? Welche?
2. Welche Einschränkungen werden für Sprünge in Programmiersprachen gemacht, um diese Probleme zu vermeiden?
3. Welche Eigenschaften muss eine Programmiersprache haben, damit auch bei uneingeschränkten Sprüngen diese Probleme nicht auftreten können?