PL0-Handbuch

Dieses Handbuch erklaert die Sprache PL/0 und bietet pro Sprachelement Beispiele in drei Lernstufen: kurz, mittel und ausfuehrlich.

Hinweis: TinyPl0 unterstuetzt die Dialekte `classic` und `extended`. Der `extended` Dialekt umfasst `?` (Eingabe) und `!` (Ausgabe).

Inhalte

- Sprachelemente (Konstanten, Variablen, Prozeduren, Bedingungen, Schleifen, Ein-/Ausgabe)
- Syntax-Referenz
- Best Practices und typische Fehler

Syntax-Referenz

Die offizielle EBNF fuer PL/0 befindet sich im kuratierten Kapitel aus `docs/LANGUAGE_EBNF.md`. Hier wird eine kurze Uebersicht der wichtigsten Konstrukte gegeben.

- `const`: Konstantenblock
- `var`: Variablenblock
- `procedure`: Prozedurdefinition
- `begin ... end`: Block
- `if ... then`: Bedingung
- `while ... do`: Schleife
- `? ident`: Eingabe (extended)
- `! expression`: Ausgabe (extended)

Konstanten

Konstanten werden mit `const` deklariert und koennen im weiteren Programmverlauf nicht geaendert werden.

Regeln

- Konstanten muessen vor Variablen deklariert werden.
- Zuweisungen an Konstanten sind nicht erlaubt.

Kurz

```
const pi = 3;
var r;
begin
  r := pi;
  ! r
end.
```

Mittel

```
const limit = 10, step = 2;
var x;
begin
  x := 0;
  while x < limit do
  begin
    ! x;
    x := x + step
  end
end.
```

Ausfuehrlich

```
const base = 2, max = 16;
var value;
procedure show;
begin
  ! value
end;
begin
  value := base;
  while value <= max do
  begin
    call show;
    value := value * base
```

```
  end
end.
```

Erklaerung

- Im kurzen Beispiel wird eine Konstante direkt verwendet.
- Im mittleren Beispiel zeigt `step`, wie feste Schrittweiten genutzt werden.
- Im ausfuehrlichen Beispiel demonstriert `base` die wiederholte Multiplikation.

Siehe auch

- Anhang: Math Functions
- Anhang: Kreis

Variablen

Variablen werden mit `var` deklariert und koennen spaeter per `:=` zugewiesen werden.

Regeln

- Variablen muessen vor ihrer Verwendung deklariert werden.
- Werte sind immer Integer.

Kurz

```
var x;
begin
  x := 1;
  ! x
end.
```

Mittel

```
var x, y;
begin
  x := 3;
  y := x * 2;
  ! y
end.
```

Ausfuehrlich

```
var a, b, c;
begin
  a := 5;
  b := 7;
  c := a * b + 3;
  ! a;
  ! b;
  ! c
end.
```

Erklaerung

- Variablen muessen vor der ersten Zuweisung deklariert sein.
- Die Beispiele zeigen einfache Berechnungen und Ausgabe.
- Im ausfuehrlichen Beispiel werden mehrere Variablen kombiniert.

Siehe auch

Prozeduren

Prozeduren kapseln wiederverwendbare Ablaufe. PL/0 kennt keine Parameter, aber Prozeduren koennen auf Variablen in aeusseren Bloecken zugreifen.

Regeln

- Prozeduren werden mit `procedure <name>;` eingefuehrt.
- Parameter und Rueckgabewerte gibt es nicht.

Kurz

```
var x;
procedure show;
begin
  ! x
end;
begin
  x := 1;
  call show
end.
```

Mittel

```
var x;
procedure inc;
begin
  x := x + 1
end;
begin
  x := 0;
  call inc;
  call inc;
  ! x
end.
```

Ausfuehrlich

```
var x;
procedure loop;
var i;
begin
  i := 0;
  while i < 3 do
  begin
    x := x + 1;
```

```
    i := i + 1
  end
end;
begin
  x := 5;
  call loop;
  ! x
end.
```

Erklaerung

- Prozeduren haben keinen Parameter, greifen aber auf aeussere Variablen zu.
- Das mittlere Beispiel erhoeht einen Wert mehrfach.
- Das ausfuehrliche Beispiel zeigt einen lokalen Zaehler innerhalb der Prozedur.

Siehe auch

- Anhang: Kgt
- Anhang: Ggt

Bedingungen

Bedingungen steuern den Programmfluss mit `if ... then` und Vergleichsoperatoren.

Regeln

- Jede Bedingung liefert 0 (false) oder 1 (true) in der VM.
- Vergleichsoperatoren: =, #, <, <=, >, >=

Kurz

```
var x;
begin
  x := 1;
  if x = 1 then
    ! x
end.
```

Mittel

```
var x;
begin
  x := 5;
  if x > 3 then
    ! x
end.
```

Ausfuehrlich

```
var a, b;
begin
  a := 4;
  b := 7;
  if a < b then
    ! b
end.
```

Erklaerung

- Bedingungen pruefen Relationen und steuern den Kontrollfluss.
- In PL/0 gibt es kein `else`; alternative Zweige werden mit weiteren `if` gebaut.

Siehe auch

- Anhang: Primzahltest
- Anhang: Zahlenvergleich

Schleifen

Schleifen werden mit `while ... do` gebildet.

Regeln

- Die Bedingung wird vor jeder Iteration geprueft.
- Der Schleifenkoerper kann ein einzelnes Statement oder ein `begin ... end` Block sein.

Kurz

```
var x;
begin
  x := 0;
  while x < 3 do
    x := x + 1;
  ! x
end.
```

Mittel

```
var x;
begin
  x := 1;
  while x <= 4 do
  begin
    ! x;
    x := x + 1
  end
end.
```

Ausfuehrlich

```
var x, sum;
begin
  x := 1;
  sum := 0;
  while x <= 5 do
  begin
    sum := sum + x;
    x := x + 1
  end;
  ! sum
end.
```

Erklaerung

- Die Schleife bricht ab, sobald die Bedingung false ist.
- Im ausfuehrlichen Beispiel wird eine laufende Summe gebildet.

Siehe auch

- Anhang: Countdown
- Anhang: Fibonacci

Ein- und Ausgabe

Im erweiterten Dialekt stehen `? ident` (Eingabe) und `! expression` (Ausgabe) zur Verfuegung.

Regeln

- Ein-/Ausgabe ist nur im Dialekt `extended` erlaubt.
- `?` liest einen Integer, `!` gibt einen Integer aus.

Kurz

```
var x;
begin
  ? x;
  ! x
end.
```

Mittel

```
var x;
begin
  ? x;
  x := x + 1;
  ! x
end.
```

Ausfuehrlich

```
var x, y;
var max;
begin
  ? x;
  ? y;
  if x > y then
    max := x;
  if y >= x then
    max := y;
  ! max
end.
```

Erklaerung

- `?` liest genau einen Integer von der Eingabe.
- `!` gibt einen Integer aus.
- Im ausfuehrlichen Beispiel wird das Maximum zweier Eingaben ausgegeben.

Siehe auch

- Anhang: Min Max
- Anhang: Statistik

Best Practices

- Kleine, klar strukturierte Prozeduren.
- Aussagekraeftige Bezeichner fuer Variablen und Konstanten.
- Einfache, gut lesbare Bedingungen in `if` und `while`.
- P-Code-Listen fuer Lernzwecke aktivieren.

Typische Fehler

- Fehlendes abschliessendes `.` am Programmende.
- Zuweisung an eine Konstante.
- Verwendung nicht deklarierter Bezeichner.
- Vergessene `end`-Markierung in Bloecken.