The programming language PASCAL

Eine Seminararbeit im Rahmen des Proseminars
“Programmiersprachen”

Sezgi Seret

0227324

1) Kurzfassung :

Pascal ist eine Lehrsprache, welche leicht erlernbar ist und eine einfache Syntax hat. Zuerst werde ich einen historischen Einblick über die Entwicklung der Programmiersprache Pascal geben. Danach werde ich einen Kurzeinblick über die Syntax und Semantik geben, sowie über die Grundlagen des Pascals.
Ich werde noch versuchen den Programmfluss der Sprache zu erklären.
Zuletzt werde ich über die Vorteile und Nachteile des Pascals sprechen.

2)Motivation zur Entwicklung der Programmiersprache Pascal :

Die Geschichte von Pascal :

Pascal wurde Ende der sechziger, Anfang der siebziger Jahre von Niklaus Wirth (ETH Zürich) speziell als Ausbildungssprache entwickelt (erste Publikation 1971). Der Name Pascal wurde zu Ehren des französischen Mathematikers und Philosophen Blaise Pascal (1623-1662) gewählt.

Blaise arbeitete vorher mit einer anderen Sprache, namens Algol-68. Algol ist der Urvater fast aller heutigen prozeduralen Programmiersprachen. Die Sprachen, die vorher entwickelt wurden, waren weitgehend unstrukturiert. Sie hatten nur das GOTO um Bedingte Sprünge zu machen. Algol wird oft in dem Bestreben eine höhere Abstraktionsebene - von der Maschine zum Modell - als die erste Stufe angesehen : Die Strukturierung des Codes.

Algol hatte aber auch Schwächen. Die Sprache war wie Fortran vornehmlich auf Berechnungen ausgerichtet. Es fehlten sowohl leistungsfähige Stringfunktionen als auch Ein/Ausgabefunktionen. Die Syntax war noch nicht so eingängig. Dies wollte Wirth bei Pascal verbessern. Die Sprache sollte primär als Lehrsprache eingesetzt werden.

Dazu wurde die Syntax so entworfen, dass die Pascalprogramme sehr leicht lesbar und verständlich sind. Das zweite Ziel von Wirth war die Einführung des Typkonzepts. Alle Sprachen vorher kannten nur elementare Datentypen : Zeichen, Zahlen. In Pascal war es möglich eigene Typen zu definieren. Entweder indem die schon vorhandenen zusammengefasst wurden (Record), oder als eine Aufzählung oder ein Teilbereich eines schon existierten Typs. Als neuen Typ gab es den Mengentyp (Set) und den Zeigertyp.

Außerdem, der Compiler wachte auch auf die Typeinhaltung, auf Wunsch auch zur Laufzeit. Vorbei waren die Zeiten, als man einer Unterroutine die eine Ganzzahl erwartete eine Zeichenkette übergeben konnte. Pascal hatte verbesserte Zeichenkettenfunktionen, aber leider in der Urform wenige Möglichkeiten Dateien zu bearbeiten wie Algol.

3)Kurzüberblick über Syntax und Semantik :

3.1.Grundlagen :

3.1.a) Programmstruktur (Programm Structure) :

Die grundlegende Struktur eines Pascalprogramms ist :

```
PROGRAM ProgramName (FileList);
CONST
   (* Constant declarations *)
TYPE
   (* Type declarations *)
VAR
   (* Variable declarations *)
 (* Subprogram definitions *)
BEGIN
   (* Executable statements *)
END.
```

Die Elemente eines Programms müssen in der korrekten Reihenfolge sein, obwohl einige ausgelassen werden können, wenn sie nicht benötigt werden. Hier ist ein Programm, das nichts tut, aber alle erforderlichen Elemente hat:

```
program DoNothing;
begin
end.
```

Kommentaren sind Teile des Codes, die nicht kompiliert oder durchgeführt werden. Sie beginnen mit (* und beenden mit *).

3.1.b) Bezeichner (Identifiers) :

Bezeichner sind Namen, die Ihnen erlauben, gespeicherte Werte zu beziehen, wie Variablen und Konstanten. Auch jedes Programm und Maßeinheit müssen durch einen Bezeichner genannt werden.

Richtlinien für Bezeichner:

- muss mit einem Buchstaben vom englischen Alphabet anfangen.
- kann von den alphanumerischen Buchstaben (Buchstaben und den Ziffern) gefolgt werden und oder manchmal das Unterstreichen (_).
- enthalten möglicherweise nicht bestimmte Sonderzeichen, die meistens spezielle Bedeutungen im Pascal haben!

~ ! @ # $ % ^ & * ( ) + ` - = { } [ ] : " ; ' < > ? , . / |

Einige Bezeichner werden im Pascal als syntaktische Elemente aufgehoben. Ihnen werden nicht erlaubt, diese als Ihre Bezeichner zu verwenden. Diese schließen ein, aber werden nicht begrenzt:

```
and array begin case const div do downto else end file for forward
function goto if in label mod nil not of or packed procedure program
record repeat set then to type until var while with
```

Pascal ist nicht case-sensitive. `MyProgram`, `MYPROGRAM`, und `mYpRoGrAm` sind gleichwertig.

3.1.c) Konstanten (Constants) :

Konstanten werden durch Bezeichner bezogen und können einen Wert am Anfang des Programms zugewiesen werden. Der Wert, der in einer Konstante gespeichert wird, kann nicht geändert werden.

Wir können einige Konstanten der verschiedenen Datentypen definieren: String, char, integer, real und boolean.

```
const
   Name = 'Niklaus Wirth';
   FirstLetter = 'N';
   Year = 1971;
   pi = 3.1415926535897932;
   UsingNCSAMosaic = TRUE;
```

3.1. d) Variablen und Datentypen (Variables and Data Types):

Variablen sind ähnlich wie Konstanten, aber ihre Werte können geändert werden, wenn das Programm läuft. Variablen müssen im Pascal zuerst deklariert werden, bevor sie verwendet werden können.

IdentifierList ist eine Reihe vom Bezeichnern, getrennt durch Kommas (,). Alle Bezeichner in der Liste müssen vom gleichen Datentyp sein.

Die grundlegenden Datentypen in Pascal sind: integer, real, char, boolean.

```
var
   age, year, grade : integer;
   circumference : real;
   LetterGrade : char;
   DidYouFail : Boolean;
```

3.1.e) Operatoren (Operations) :

- Arithmetische Operatoren : + - * / div mod
- Logische Operatoren : not and or

- Vergleichsoperatoren : = <> < <= > >=

- Zuweisungsoperator : :=

- Mengenoperatoren : * + -

3.1.f) Standardfunktionen(Standard functions): Funktionen werden benannt, indem man den Funktionnamen verwendet, der in Klammern von den Argumenten gefolgt wird. Standardfunktionen in Pascal sind:

abs(x)	sqr(x)	sin(x)	cos(x)
exp(x)	ln(x)	sqrt(x)	arctan(x)
trunc(x)	round(x)	odd(x)	

3.2. Programm Fluss :

3.2.a) Anweisungen :

1) Einfache Anweisungen :

- Wertzuweisung: Die Wertzuweisung hat die Aufgabe, den momentanen Wert einer Variablen durch einen neuen Wert zu ersetzen, der sich durch die Auswertung eines Ausdrucks ergibt.

```
Bsp. :   y : = x + cos(x/3)
```

- Sprunganweisung(Goto-Anweisung): Mit einer Sprunganweisung kann man die hingeschriebene Reihenfolge durchbrechen, indem man in einer Sprunganweisung die nächste auszuführende Anweisung schreibt.

2) Struktierte Anweisungen :

- Verbungsanweisung: ist eine Folge von Anweisungen, die in der Reihenfolge ausgeführt werden, in der sie niedergeschrieben sind.

```
Bsp. begin h : = a ; a : = b ; b : = h end
```

- Bedingte Anweisung :

if Anweisung: Die if – Anweisung dient zur Beschreibung einer Alternative, wenn sich das Programm in zwei Äste verzweigen soll.

```
if BooleanExpression then
    StatementIfTrue
else
    StatementIfFalse;
```

case Anweisung: Soll sich ein Programm an einer Stelle in mehrere Äste verzweigen, ist es eine von mehreren Möglichkeiten auszuwählen, so ist dafür die **case**-Anweisung geeignet.

```
  case selector of
        List1:    Statement1;
        List2:    Statement2;
        ...
        Listn:    Statementn;
        otherwise Statement
    end;
```

- Wiederholungsanweisung :

while - Anweisung: ist eine Wiederholungsanweisung, also eine Schleifenanweisung, bei der die Anzahl der Wiederholungen von einer Bedingung abhängig ist. Dabei wird die Bedingung zu Beginn eines neuen Durchlaufes geprüft.

```
    while BooleanExpression do
       statement;
```

repeat....until Anweisung: Die Anweisungen zwischen **repeat** und **until** werden solange ausgeführt, solange der logische Ausdruck den Wert **false** hat.

```
    repeat
       statement1;
       statement2
    until BooleanExpression;
```

for – Anweisung: Die for – Anweisung dient zur Formalierung von Schleifen, wenn die Anzahl der Durchläufe vor Eintritt in die Schleife festliegt.

Die Allgemeine Form von einer Fixen Wiederholungsform(fixed repetition form):

```
    for index := StartingLow to EndingHigh do
       statement;
```

In **for-to-do** Loop, der Anfangswert muss niedriger als der Endewert sein, oder die Schleife wird nicht durchgeführt! Wenn Sie rückwärts zählen möchten, sollten Sie **for-downto-do** Loop verwenden.

```
     for index := StartingHigh downto EndingLow do
        statement;
```

3.3.Prozeduren und Funktionen :

Prozeduren sind in Pascal die allgemeinste Form von Unterprogrammen.Ein Prozeduraufruf in einem Programm wird notiert durch Benennung des Prozedurnamens, gefolgt von der in Klammern eingeschlossenen Liste der aktuellen Parameter.

```
procedure Name;
  const (* Constants *)
  var (* Variables *)
begin (* Statements *) end;
```

Funktionen sind analog zu Prozeduren aufgebaut.Sie sind ebenfalls die programmtechnische Realisierung einer funktionell abgeschlossenen Teilaufgabe.Der untergeordnete Algorithmus wird aber nicht durch eine Anweisung aktiviert, sondern durch Benennen des Funktionsnamens in einem Ausdruck.

```
function Name (parameter_list) : return_type;
```

4)Besonderheiten und Anwendungsbeispiele :

Die Programmiersprache Pascal hat einen relativ starken Systemtyp, teils wegen der Tatsache, daß es ursprünglich eine Sprache für Anweisung und Typ Überprüfung sein sollte, kann helfen, sich fast alle Fehler der Programmieranfänger zu verfangen. Pascal erlaubt Rekursion, eine Verbesserung über vielen früheren Programmiersprachen.

Pascal ist eine strukturierte Sprache, das Verwenden if-then-else, while, repeat-until und for-to/downto Steuerstrukturen. Es unterscheidet sich hauptsächlich von fortfahrenden Sprachen dadurch, daß Datenstrukturen auch eingeschlossen waren, mit Aufzeichnungen (eine Eigenschaft geborgt von Cobol), Reihen, Akten, Sätzen und Zeigern. Pascal ist auch für das Schmieden eines wirkungsvollen Kompromisses zwischen Spracheneinfachheit, Energie und dem Zusammenbringen der Sprachstrukturen zu zugrundeliegender Maschine Implementierung ungewöhnlich.

Pascal hat auch viele Eigenschaften für Compilerverfasser. Die Sprache wird konstruiert, um ein Minimum Mehrdeutigkeit zu haben. Pascal, mit wenigen Ausnahmen, kann mit allen kleineren Elemente verarbeitetes "Vorwärts" sein (wie Konstanten, Arten, usw.), die definiert werden, bevor sie benutzt werden. Pascal erfordert die Arten und die genauen Größen der bekannt Rechengrößen, bevor sie an bearbeitet werden und wieder zu die vereinfachte Sprachenverabeitung führen und des leistungsfähigen Ausgang Codes (obgleich diese Eigenschaft häufig als ein Problem gesehen ist).

Aus diesem Grund bleibt Pascal weiterhin eine populäre Sprache, zum der Compiler für als Teil eine Compilerwissenschaft Kategorie einzuführen.

5)Zusammenfassung : Vorteile und Nachteile :

☹ Nachteile

Systemzugriffe können im allgemeinen nicht ganz so tief wie bei C oder Assembler erfolgen (Adresszugriffe).Die Abarbeitungsgeschwindigkeit ist durch die Programmstruktur bei den ausführbaren Dateien nicht ganz so hoch wie bei C / C++ und bei Assembler. Es gibt leider nicht so viele Units auf dem Markt, wie beispielsweise von C. Pascal ist keine Objektorientierte Sprache.

☺ Vorteile

Pascal hat relativ höhe Geschwindigkeit, ist leicht und gut strukturierbar. Es ist leicht zu erlernen, besonders für Menschen mit Programmiererfahrung in C. Assembler Quelltexte lassen sich in viele neue Pascal Programme einbinden. Da es sich um eine Compilersprache handelt lassen sich ausführbare relativ kleine Dateien erstellen. Pascal lässt sich mit Units , ähnlich wie bei C (includes), erweitern. Pascal wird oft in Schulen im Informatikunterricht behandelt. Es gibt deshalb viele Lektüre für Einsteiger.

6)Literaturliste :

[1] Rudolf Herschel , Friedrich Pieper
Pascal
Systematische Darstellung von Pascal und Concurrent Pascal für den Anwender

[2] http://www.taoyue.com/tutorials/pascal/contents.html

[3] Hopfer
Informationsverarbeitung mit Pascal

[4] http://www.db.informatik.uni-kassel.de/Help/pascal/einfuehrung/