The programming language

PASCAL

Autorin

Name **:**KARACORLU Tuba

MatNr **:**0327381

Kennzahl **:**534

Universität **:** Technische Universität Wien

E-mail **:**tuba_karacorlu@yahoo.de

WAS UND WER IST PASCAL ?

1)Motivation zur Entwicklung der Programmiersprache Pascal :

Die Geschichte von Pascal

Pascal ist eine Programmiersprache, die 1968 bis 1972 von dem Schweizer Informatiker Niklaus Wirth entwickelt wurde.
Pascal dagegen ist kein Kunstwort, sondern erinnert an französischen Philosophen, Mathematiker und Physiker Blaise Pascal, der von 1623 bis 1662 lebte.

Pascal war ein kränkliches Kind, deshalb wurde er von seinem sehr gebildeten Vater und Hauslehrern unterrichtet. Spätestens mit zwölf erwies er sein mathematisches Talent. Er begann in Mathematikerkreisen zu verkehren und beeindruckte dort als 16-Jähriger mit einer grundlegenden Arbeit über die Berechnung von Kegelschnitten.

Pascal ist eine Weiterentwicklung von ALGOL.(ALGOL als Abkürzung für ALGOrithmic Language, meist als Wortschöpfung nicht in Großbuchstaben sondern Algol geschrieben.)

Das wichtigste Konstruktionsprinzip war, die Sprache so einfach wie möglich zu gestalten, damit sie in der Ausbildung genutzt werden konnte. Gleichzeitig sollte strukturierte Programmierung möglich sein. Alle Variablen müssen vor der Benutzung deklariert werden. Der erste Pascal-Compiler selbst entstand auf der CDC Cyber 6000 der ETH Zürich. Daraus entstand dann Pascal 6000, welches als erste operative Version eines Compilers der Sprache gesehen werden kann.

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.

Ein wichtiger Schritt für die große Verbreitung, die Pascal inzwischen erfahren hat, war die Standardisierung der Programmiersprache Pascal im Jahre 1983 (ISO 7185). (Eine Übersetzung dieser Norm wurde 1984 als DIN 66256 verabschiedet.)

Inzwischen gibt es auch einen Standard für ein erweitertes Pascal (Extended Pascal, ISO 10206 von 1991).

Pascal steht auf allen gängigen Hardwareplattformen zur Verfügung. Die größte Verbreitung ist im Bereich der PCs gegeben, wo Pascal neben C die wohl wichtigste Programmiersprache ist.

Standards

Es gibt 3 Standards, die sich auf Pascal beziehen:

- Standard Pascal: ANSI/IEEE770X3.97-1993 oder ISO 7185:1990
- Extended Pascal: ANSI/IEEE770X3.160-1989 oder ISO/IEC 10206:1991
- sowie einen Entwurf zu „Object-Oriented Extensions to Pascal“

und Turbo Pascal…

2)Syntax und Semantik:

- ***Programmstruktur (Programm Structure):***

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 *)

- ***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;
```

➢ ***Variablen und Datentypen (Variables and Data Types):***

Variablen sind Ihnen mit ziemlicher Sicherheit aus der Mathematik bekannt. Eine Konstante ist eine Variable mit festem Wert. 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.

Bevor wir auf eine Variable zugreifen können, müssen wir diese dem Compiler bekannt geben. Diese Bekanntgabe bezeichnet man als Deklaration. Weisen wir einer Variable einen Wert zu, definieren wir sie.

Deklaration	Bekanntgabe (z.B. dem Compiler bekannt geben, dass eine Variable existiert, welchen Bezeichner (Namen) sie trägt sowie von welchem Typ sie ist)
Definition	Wertzuweisung (z.B. eine zuvor deklarierten Variable wird nun ein Wert zugewiesen)

VAR Name: Typ[Längenbegrenzung]

Die Variablendeklaration (variable declaration part) ordnet dem Namen jeder dort auftretenden Variablen einen Typ zu.Sie beginnt stets mit dem Schlüsselwort ***VAR,***gefolgt von Listen von durch Kommata getrennten Namen,gefolgt von einem Doppelpunkt,hinter dem der Typ aller Variablen der Liste angegeben wird, vom folgenden abgetrennt durch ein Semikolon **.**

Es gibt in Pascal 4 einfache **Standardtypen** von Variablen,die mit den Standardnamen *integer,real,boolean und char* bezeichnet werden.

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

Hierbei unterscheiden wir 3 Gruppen: Typen zur Ganzzahlenspeicherung, Typen zur Speicherung von Fließkommazahlen und solche zur Speicherung von Zeichenketten. (Strings):

- ***Ganzzahlentypen:***

Typ	Bereich	Größe
SortInt	**-128 bis 127**	**1 Byte**
Byte	**0 bis 255**	**1 Byte**
Integer	**-32768 bis 32767**	**2 Byte**
Word	**0 bis 65535**	**2 Byte**
LongInt	**-2147483648 bis2147483647**	**4 Byte**

- ***Gleitkommatypen:***

Typ	Bereich	Grösse
Single	**$1.5*10^{-45}$ bis $3.4*10^{38}$**	**4 Byte**
Real	**$2.9*10^{-39}$ bis $1.7*10^{38}$**	**6 Byte**
Double	**$5.0*10^{-324}$ bis $1.7*101^{308}$**	**8 Byte**
Comp	**$9.2*10^{18}$**	**8 Byte**
Extended	**$3.4*10^{-4952}$ bis $1.1*10^{4952}$**	**10Byte**

- ***Zeichentypen:***

Typ	Bereich	Grösse
Char	**0 bis 255; erweiterter ASCII-Zeichensatz**	**1 Byte**
String	**bis zu 255 Zeichen**	**bis zu 255 Byte**

- ***Boolesche Typen:***

Typ	Grösse
Boolean	**1 Byte**
ByteBool	**1 Byte**
WordBool	**2 Byte**
LongBool	**2 Byte**

Der Datentyp *integer :*

Ein Integer-Typ repräsentiert eine Untermenge der ganzen Zahlen.Er umfasst eine Teilmenge der ganzen Zahlen,die durch integer-Literale dargestellt werden.

```
VAR
min,max :integer;
```

Der Datentyp *real :*

Er umfasst eine Teilmenge der reellen Zahlen (genauer sogar eine Teilmenge der rationalen Zahlen).die durch real-Literale dargestellt werden.

```
VAR
dinstanz: real;
```

Der Datentyp *boolean:*

Er umfasst genau die beiden Wahrheitswerte *wahr* und *falsch*,dargestellt durch die Standardkonstanten **true** und **false.**

```
VAR
gewonnen:  boolean;
```

Operatoren für einen bzw.zwei Operanden vom Typ *boolean* (logische,boolean-Operatoren),die als Ergebnis einen Wert vom Typ *boolean* liefern,sind die folgenden**:**

NOT (Negation,nur 1 Operand)
AND (Konjuktion)
OR (Disjuktion)

Der Datentyp *char:*

Er beschreibt die Menge der Verfügbaren Druckzeichen,bislang auch als character angebeben.Ein Wert also genau eines der Zeichen.Das Leerzeichen(blank) ist in diesem Sinn ein Druckzeichen.Steuerzeichen wie Zeilenvorschubzeichen und andere sind dagegen keine Druckzeichen.

```
VAR
anfangsbuchstabe : char ;
```

➢ ***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 diehingeschriebene 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 fordownto- do Loop verwenden

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

- ***Operatoren (Operations) :***

⇒ Logische Operatoren : not and or

⇒ Aritmetische Operatoren : + - * / div mod

⇒ Mengen Operatoren : * + -

⇒ Zuweisungoperator : :=

⇒ Vergleichsoperator : = < > < <= > >=

- ***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)**	**sgrt(x)**	**arctan(x)**
trunc(x)	**round(x)**	**odd(x)**	

- ***Prozeduren und Funktionen :*** Häufiger als Funktionen werden bei der Entwicklung gröserer Programme Prozeduren verwendet.Prozeduren haben zwar eine grosse formale Ähnlichkeit mit Funktionen, sie unterschieden sich jedoch in einem wesentlichen Punkt von Fuktionen:
Ein Prozeduraufruf ist eine selbständig ausführbare Anweisung und nicht Teil eines Ausdrucks wie ein Funktionsaufruf.Gerade diese Eigenschaft macht es möglich,Prozeduren als Unterprogramme aufzufassen.Sie erlauben damit eine klare Gliederung und schritweise Entwicklung eines Programms.

- ***Warum Pascal???***

Pascal wurde als Lernsprache konzipiert. Mit dem Ziel Studenten zu lernen saubere Programme zu schreiben. Daher ist die Sprache strenger in der Prüfung und insgesamt sicherer in der Anwendung. Als Lehrsprache wurde leider versäumt eine für die Praxis notwendige Bibliothek an Funktionen, insbesondere für hardwarenahe Operationen und Ein/Ausgabe mitzuliefern.

Die Programmiersprache Pascal hat einen relativ starken Systemtyp, teils wegen derTatsache, 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 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).

➢ ***Zusammenfassung:***

☺ *Vorteile* ☺

- ✓ Pascal hat relativ höhe Geschwindigkeit, ist leicht und gut strukturierbar. Es ist leicht zu erlernen, besonders für Menschen mit Programmiererfahrung in C.
- ✓ Da Pascal eine Sprache ist, die in der Lehre eingesetzt werden sollte, sind Pascal Programme sehr gut lesbar. Dazu tragen viele Dinge bei, die man bei C weggelassen hat.
- ✓ Pascal wird oft in Schulen im Informatikunterricht behandelt. Es gibt deshalb viele Lektüre für Einsteiger.
- ✓ Assembler Quelltextelassen 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.

☹ *Nachteile* ☹

- ✓ 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.

Systemzugriffe können im allgemeinen nicht ganz so tief wie bei C oder Assembler erfolgen (Adresszugriffe).

➢ ***Literatur:***

(1) *The programming language PASCAL*-Niklaus Wirth-

(2) *Programmierung mit Pascal* -Thomas Ottman,Peter Widmayer-

(3) http://www.tutorials.at/

(4)*Informationsverarbeitung mit PASCAL*-Dr.Ing.Reiner Hopfer-