

Getting Started With Pascal **Programming**

What is the basic structure of a Pascal Program

Variables in Pascal

Performing input and output with Pascal

Useful mathematical functions in Pascal

Basic Structure Of Pascal Programs

```
(*  
  
                                Header  
  
                                *)  
  
program name (input, output)  
  
    declarations  
  
        var  
  
        const  
  
begin  
  
  
  
end.
```

Variables

Set aside a location in memory

Used to store information (temporary)

Types:

- integer – whole numbers
- real – whole numbers and fractions
 - Can't end or start with a decimal
- char – alphabetic, numeric and miscellaneous symbols
- boolean – true or false values

Usage:

- Declaration
- Using values stored

Declaring Variables

Sets aside memory

Memory locations addressed through the name

Naming conventions

- Should be meaningful
- Any combination of letters, numbers or underscore (can't begin with a number and shouldn't begin with an underscore)
- Can't be a reserved word e.g., program, begin, end (see Appendix B)
- Avoid using words with an existing meaning e.g., integer, real, boolean, write, writeln, read, readln
- Avoid distinguishing variable names only by case
- Okay:
 - tax_rate
 - firstName
- Not Okay
 - labc
 - x
 - test.msg
 - good-day

Declaring Variables (2)

Typically occurs in the variable declaration ("var") section

i.e.,

var

name of first variable, name of second variable...: type of variables;

e.g.,

var

height, weight: real;

age: integer;

Using Values Stored In Variables

Assignment

- Performed via the assignment operator :=
- Usage:
 - Destination := Source;₁
- Example:
 - x := 5;
 - x := y;
 - interest := principle * rate;
 - Initial := 'j';
- Avoid assigning mixed types

e.g.,

var

num1: integer;

num2: real;

Begin

num1 = 12;

num2 = 12.5;

num2 := num1;

num1 := num2;

Not allowed!



¹ The source can be any expression (constant, variable or formula)

Named Constants

A memory location that is assigned a value that cannot be changed

Format:

const

name of first constant = value of first constant;

name of second constant = value of second constant;

etc.

Location

Anywhere after the "program" statement and before the "begin"

Purpose of Named Constants

1) Makes the program easier to understand

e.g.,

begin

```
population_change := (0.1758 - 0.1257) * current_population;
```

Vs.

const

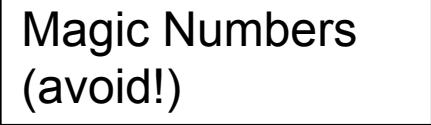
```
BIRTHRATE = 0.1758;
```

```
DEATHRATE = 0.1257;
```

begin

```
population_change := (BIRTHRATE - DEATHRATE) * current_population;
```

2) Makes the program easier to maintain



Magic Numbers
(avoid!)

Output

Displaying information onscreen

Done via the write and writeln statements

Formats (either write or writeln):

```
write ('text message');  
    or  
writeln('text message');
```

```
write(name of variable or constant);  
    or  
writeln (name of variable or constant);
```

```
write('message', name of variable, 'message'...);  
    or  
writeln('message', name of variable, 'message'...);
```

Formatting Output

Computer often inserts spaces as it thinks is necessary in order to display output.

Manually formatting of output:

- write or writeln (data to output: field width for data: no. of decimal places)
- e.g., writeln (num1:6:2);

If the field width doesn't match the actual size of the field

- Field width too small – extra spaces will be added for numerical variables.
- Field width too large – the data will be right justified (extra spaces will be put in front of the data).

Formatting Output (2)

If the number of decimal places doesn't match the actual number of decimal places.

- Set number of decimal places less than the actual number of decimal places – number will be rounded up.
- Set number of decimal places greater than the actual number of decimal places – number will be padded with zeros.

Formatting Output: Examples

For the complete program and executable look under /home/231/examples/getting_started/out1.p (out1 for the compiled version)

```
num1 := 123;
```

```
num2 := 123.456;
```

```
writeln('Auto formatted by Pascal ', num1, num2);
```

```
writeln('Manual format':13, num1:3, num2:7:3);
```

```
writeln('Manual not enough':13, num1:2, num2:6:3);
```

```
writeln('Manual too much':16, num1:4, num2:8:4);
```

Input

The computer program getting information from the user

Done via the read and readln statements

Formats:

(single input)

read (name of variable);

or

readln (name of variable);

(multiple inputs)

read (nv1, nv2...);

or

readln (nv2, nv3...);

Input: Read Vs. Readln

Both:

- Reads each value inputted and matches it the corresponding variable.

Read

- If the user inputs additional values they will remain

Readln

- Any additional values inputted will be discarded

Input: Read Vs. Readln (An example)

For the complete version of this program look in Unix under:
/home/231/examples/getting_started/read1.p (or read1 and read2 for the
compiled version)

e.g., read1.p

```
write('Input some integers making sure to separate each one with a space ');
```

```
write('or a new line: ');
```

```
read (num1, num2);
```

```
write('Input some integers making sure to separate each one with a space ');
```

```
write('or a newline: ');
```

```
read(num3, num4);
```

Input: Read Vs. Readln (An example (2))

For the complete version of this program look in Unix under:
/home/231/examples/getting_started/read2.p (or read2 for the compiled
version)

e.g., read2.p

```
write('Input some integers making sure to separate each one with a space ');
```

```
write('or a newline: ');
```

```
readln (num1, num2);
```

```
write('Input some integers making sure to separate each one with a space ');
```

```
write('or a newline: ');
```

```
readln(num3, num4);
```


Uses Of Readln

To filter out extraneous input

As an input prompt

e.g.,

```
writeln('To continue press return');
```

```
readln;
```

Some Useful Functions

See also Appendix D in Pascal Programming and Problem Solving by Leestma S. and Nyhoff L.

Name	Description	Input type	Type of result	Example
abs	absolute value	integer	integer	$\text{abs}(-2) = 2$
		real	real	$\text{abs}(-2.2) = 2.2$
round	rounding	real	integer	$\text{round}(2.6) = 3$
trunc	truncation	real	integer	$\text{trunc}(2.6) = 2$
sqr	squaring	integer	integer	$\text{sqr}(2) = 4$
		real	real	$\text{sqr}(1.1) = 1.21$
sqrt	square root	integer	real	$\text{sqrt}(4) = 2.00$
		or real		

Summary

What are the fundamental parts of a Pascal program

What are the basic types of variables employed in Pascal and how are they used

How to output information with the write and writeln statements

Getting information from the user through the read and readln statements

How are some common mathematical functions performed in Pascal