Introduction To Defining New Types in Pascal

In this section of notes you how and why programmers can define new types.

Declaring Types

Why bother?

- Creating your own type of variable
- Making a synonym for an existing type

Format:

Type

Name(1) = Type for name (1);

Name(2) = Type for name (2);

: : : :

Name(n) = Type for name (n);

Declaring Types (2)

Can be used to provide alternative names for existing types

Example:

```
type
    FloatingPoint = real;
var
    gpa      : FloatingPoint;
    income: real;
```

Declaring Types (2)

Can be used to provide alternative names for existing types

Example:

```
type
    FloatingPoint = real;
var
    gpa      : FloatingPoint;
    income: real;
```

Declaring the type - defining what the type consists of (creating type)

Declaring a variable of the new type (creating instances)

efining new types in Pascal: Enumerated

Declaring Types (3)

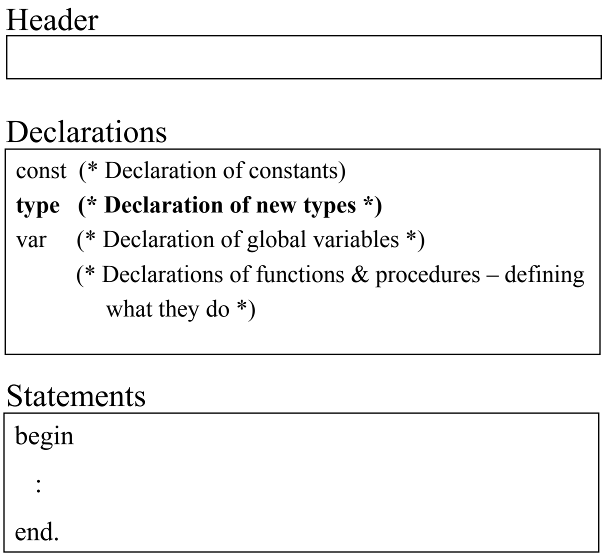
Can be used to provide alternative names for existing types

Example:

```
type
    FloatingPoint = real;
var
    gpa      : FloatingPoint;
    income: real;
```

Original type still usable

Where Type Declarations Fit Into Pascal Programs



Header

Declarations

```
const  (* Declaration of constants)
type   (* Declaration of new types *)
var    (* Declaration of global variables *)
       (* Declarations of functions & procedures – defining
           what they do *)
```

Statements

```
begin
   :
end.
```

efining new types in Pascal: Enumerated

Programmer-Defined Ordinal Types

Enumerated types

Subrange

Enumerated Types

Format:

Type

Name of type = (*identifier 1*, *identifier 2*..*identifier n*);

Example:

```
type
    Months =  (January, February, March, April, May, June, July,
               August, September, October, November, December);
    Budget = array [January..December] of real;
begin
    var tamjBudget   : Budget;
    var monthsIndex : Months;
    for monthsIndex := January to December do
        tamjBudget[monthsIndex] := 2000 + random(2000);
end.
```

Operations On Enumerated Types

Operation	Name
Equity	=
Inequity	<>
Less than	<
Less than, equal to	<=
Greater than	>
Greater than, equal to	>=
Predecessor	pred
Successor	succ
Ordinal Number	ord

Examples Of Operations On Enumerated Types

Pred

```
if ((pred(February)) = January) then
      writeln('Jan comes before Feb');
```

Ord

```
writeln(ord(January));

(As an ASCII converter)
  writeln(ord('A'));
  writeln(ord (chr(65)));
  writeln(chr(65));
```

Subrange

Used to define a new type which is a subset of an existing type.

Syntax:

```
type
    Subrange name = first value..last value;
```

Example:

```
type
    Months =  (January, February, March, April, May, June, July,
               August, September, October, November, December);

    FallTerm = September..December;
    WinterTerm = January..April;
```

You Should Now Know

- Why you need to create your own types
- How and where programmer defined types are created
- How to define and declare instances of enumerated types and subranges
- How to use enumerated types and subranges
- What operations are valid on enumerated types and how does each one work