



Extensions: Syntax

```
• Syntax macros allowed general syntax
extension
real syntax sum from i = lb to ub of elem;
value lb, ub;
integer lb, ub, i; real elem;
begin real s; s := 0;
for i := lb step 1 until ub do
        s := s + elem;
return s;
end;
- Allows:
    total := sum from k = 1 to N of Wages[k];
```

Issues with Extensibility

- Inefficiency
 - New syntax is translated to kernel constructs
 - Inefficiencies are magnified

Poor diagnostics

- Compiler errors are issued at kernel-level, which may be confusing to programmer
- Language is hard to read, since people make up their own syntax
- Upside
 - Research on minimal requirement for PL's

Move Toward Simplicity

- Niklaus Wirth suggests changes to Algol-60
 - Non-numeric data types
 - Removing baroque features
 - Maintain efficiency (compile and run-time)
 - Can be taught systematically
- Implements Algol-W (after changes are rejected by Algol committee)
 - Evolves into Pascal, competed in 1970

Pascal - 3rd Generation

- Developed 1968-1970 - 29 page report
- Revised 1972
- International Standard 1982
- Popular teaching language

Pascal's Syntax • Pascal's syntax is like Algol's (p. 171) • Major changes – program ... end. – procedure <declarations> begin <statements> end;

- var, const, type
- for-loop: simplified
- case-statement

var, const, type

- const – Constant parameter declaration
- const Max = 900;
- type
- Type declarations introduced by "type"
 type index = 1 .. Max;
- var
 - Variables declared after "var"
 - var
 - i: index;

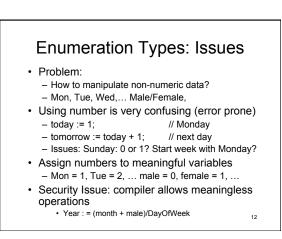
sum, ave, val: real;

10

Data Structures

11

- Primitives are like Algol's
 - real, integer, Boolean, char
 - Char holds one character
 Strings are arrays of chars



Enumeration Types

Pascal introduces enumeration types

type month = (Jan, Feb, Mar, Apr, May, ...); sex = (male, female); var thisMonth : month; gender : sex; begin thisMonth := Apr;

gender := female;

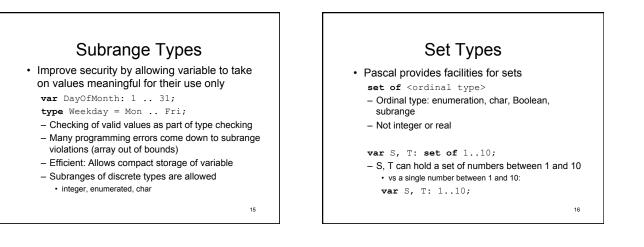
Supported operations for all enumerated types
 :=, succ, pred, =, <>, <, =, >, <=, >=

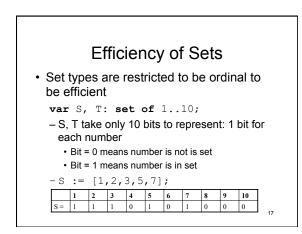
13

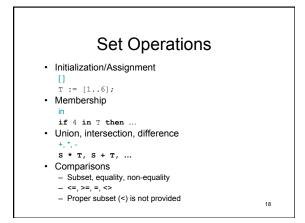
Enumeration Types

- Advantages
 - High level
 - · Lets programmers write what they mean
 - Secure
 - · Type checking is performed
 - No meaningless operations
 - Efficient
 - Allows optimization of storage
 - E.g.: Days of week can be stored in 3 bits

14

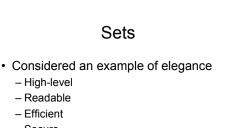






Efficiency of Sets

- Sets are implemented using bit masks
 - Therefore, operations on sets can be implemented using logical operations
 - Intersection: logical and
 - Union: logical or
 - Difference: logical exclusive or
- Logical operations are the fastest a computer can do
- Memory efficiency: 1 bit per element

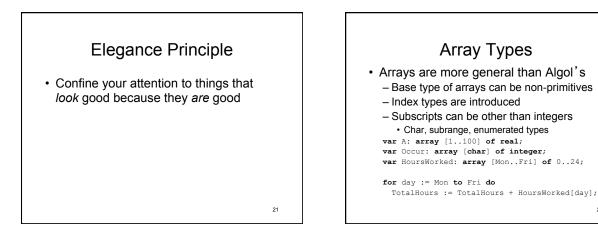


20

22

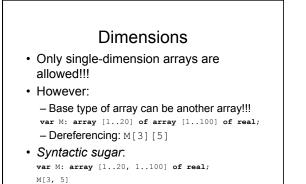
24

- Secure

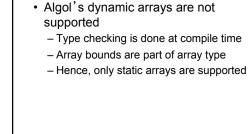


23

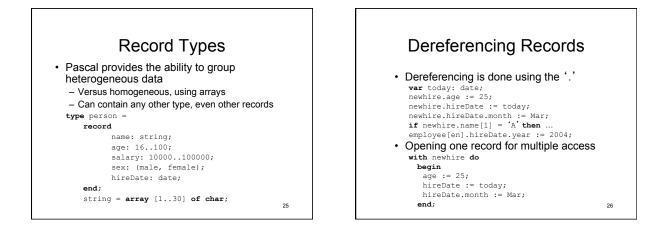
19

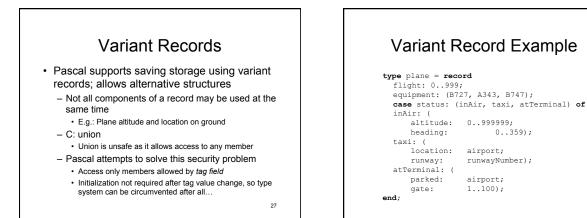


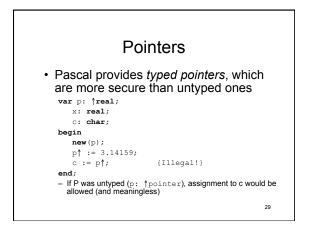
(Doesn't affect functionality, sweeter for human use.)

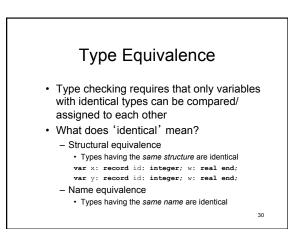


Static Arrays Only



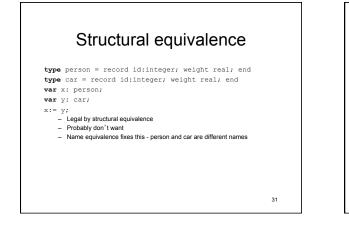






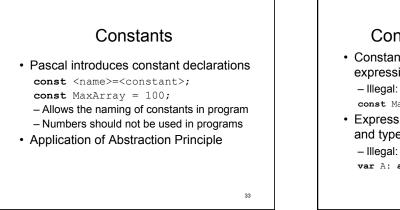
32

34



Name Structures

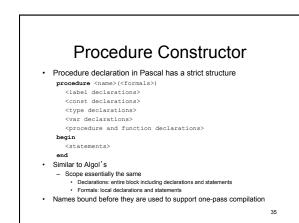
- · Name binding mechanisms in Pascal
 - Constant bindings
 - Type bindings
 - Variable bindings
 - Procedure and function bindings
 - Implicit enumeration bindings
 - Label bindings

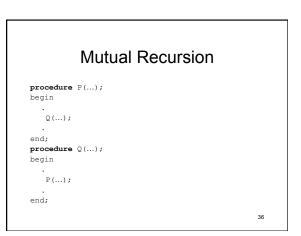




and type declarations

var A: array [0.. MaxData - 1] of real;





Procedure Constructor

- · Opposite of top-down
 - Uppermost procedures first, then lower ones they call
- · Mutual recursion - Cannot define both procedures before one is called
- · Pascal's solution
 - "forward" declaration of procedures allows recursion, and observation of structure principle procedure Q(...); forward;

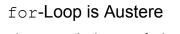
37

39



- Pascal eliminates Algol's blocks
 - Compound statements but no blocks
 - Variable declarations are only allowed before begin in procedures and functions
 - Simplifies name structures
 - Complicates efficient use of memory Storage shared only between disjoint procedures

Control Structures · Pascal includes more control structures than Algol-60, but they are simpler - Provides simple I/O - Introduces more structured control structures (structure principle) · 1-entry point 1-exit point controls - Includes goto (rarely needed) - Includes recursive procedures



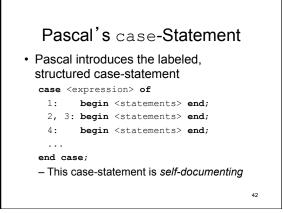
- Pascal removes the barogue for loop, in favor of one simpler than Algol's
 - for <name> := <exp> {to|downto} <exp> do <statement>
 - Only step size of 1 is allowed (+1 & -1) · May be too restrictive
 - Bounds are computed once, on entry • Called definite iterator - Always executes a definite number of times unless goto

40

38

Leading & Trailing Decision Loops

- · Indefinite iterators:
 - Loop is controlled by condition, not counter - Condition is tested each time
 - · Versus pre-computed in for-loop
- Leading Decision loop
- while <condition> do <statement>
- Trailing Decision loop
- repeat <statement>* until <condition> Mid-Decision loop
- - Can be implemented using "while true do" and goto



44

46

Labels in case-Statement · Case labels can be labels from

enumeration types case nextFlight.status of

inAir: begin <statements> end; onGround: begin <statements> end; atTerminal:**begin** <statements> **end**; end case;

Parameter Passing

- · Pass by value - Exactly like before, in Algol-60
- · Pass by reference
 - Allows output parameters
 - Replaces pass by name
 - Only allows meaningful variables to be written into (unlike Fortran)

Pass as Constant · Pass as constant was originally specified instead of pass by value - Like pass by value, but parameter could not be modified in callee Safe - Implemented as pass by reference Efficient

- Replace by pass by value, since pass as constant can be circumvented using scoping (p 202)
 - C++ provides this functionality by explicit pass by reference and *const* definitions (f(const int &a))

45

43

Two Orthogonal Issues

- · Input vs output parameters
- · Copy value vs pass address
- · Decisions should be separated

Goals

- · Main goal: good teaching language
 - Reliability
 - Simplicity
 - Efficiency
- · Successful!
- · Third Generation