About Final Project

Tsan-sheng Hsu

tshsu@iis.sinica.edu.tw

http://www.iis.sinica.edu.tw/~tshsu
Language Definitions

- A static scoping language called $P$.
  - PASCAL-like;
  - lexical scoping;
  - block structure;
  - nested procedure with recursion;
  - case sensitive;

- A program contains
  - header: PROGRAM name
  - constant definitions: optional
    - $\textit{CONST}$ ... $\textit{ENDCONST}$
    - name = constant;
  - type definitions: optional
    - $\textit{TYPE}$ ... $\textit{ENDTYPE}$
    - name = typestring;
  - procedure/function definition: optional
  - statement: BEGIN ... END
    - variable declarations: optional
    - $\textit{VAR}$ ... $\textit{ENDVAR}$
Example

PRORGM main
CONST %% can be empty or completely missing
    cons360 = 360; %% a legal name on the left, a legal constant on the right
    myfloat = 3.6;
ENDCONST
TYPE %% can be empty or completely missing
mytype = ARRAY[1..10] of INTEGER;
ENDTYPE
FUNCTION foo(x : INTEGER): INTEGER;
BEGIN
    foo := x * x - 3;
END
BEGIN
VAR %% can be empty or completely missing
    x : ARRAY[-3 .. 5] of integer;
    y : mytype;
ENDVAR
BEGIN
    VAR
        w, x, z: INTEGER;
    ENDVAR
    x := foo(y[4]);
    WRITE(x);
    WRITESP();
    WRITE(y);
    WRITELN();
END
END
Data types and variables

- **elementary types:**
  - INTEGER: 32-bit signed
  - FLOAT: 32-bit
  - INTEGER and FLOAT are not compatible types
  - FLOAT constant must have a “dot”.

- **aggregate types:**
  - 1-D array: `ARRAY[low .. upper]` of elementary type;
  - multi-D array: row major
    `ARRAY[low1 .. upper1,low2 .. upper2,...]` of elementary type;
  - need to check array out of bound in compile and run time;

- **type equivalence:** name equivalence
- **check for incompatible types**

- **variables**
  - Names: legal C variable names;
  - Length of variable names: from 1 to 1024 characters;
  - using ASCII encoding;
I/O statements

- **READ**(single variable)
  - the variable must be of the type INTEGER or FLOAT;
- **WRITE**(single variable) — output a variable
- **WRITESP**() — output a single space
- **WRITELN**() — write a new line
- There is no space before and in between "()".
Procedure and function

- **Procedure**: one that does not return anything
- **Function**: one returns a value of the elementary type
- **Parameters**:
  - call-by-value or call-by-reference
  - check for incompatible types

```pascal
PROCEDURE p(x,y: INTEGER; VAR z: FLOAT);
    TYPE
    ENDTYPE
    FUNCTION foo(x:INTEGER): INTEGER; %% return value is INTEGER
    BEGIN
        foo := x * x;
    END
    BEGIN
        y := foo(x);
    END
END
```
Statements

- One line contains at most one statement.
  - comments: from % to the rest of the line
  - “;” is statement terminator
  - a blank line is legal, but a line with only “;” is not legal;

- Assignments and I/O statements.
- Procedure/function call statements.
  - p(100,200,w)
  - p()

- Return statement,
  - return;
Operators

- **assignment**: :=
  - variable := expression;
  - must be of the same type;
  - check for incompatible types;

- **swap**: < →>

  a <-> b; % swaps the content of two variables

  - swap two variables of identical types;
  - can be aggregate;

- **arithmetic**: +, −, *, /, mod, where mod is remainder;
  - check for divide by 0 in compile and run time;
  - mod is only for INTEGERS;

- **logical**: or, and, not, xor

- **comparison**: >, <, =, <=, >=, <>
  - Must between data of identical elementary type;
Expressions

- **arithmetic expression:**
  - operations on integers / floats
  - no auto-type conversion
  - detect incompatible types
  - can have “(” and “)”

- **boolean expression:** no short-circuited evaluation.
  - operations on boolean variables
  - can have “(” and “)”
Conditional statements

- if ... then ... else

IF boolean-expression
THEN
  statement
ENDIF;

IF boolean-expression
THEN
  statement 1
ELSE
  statement 2
ENDIF;
Looping statements

- **for loop**

  /* add 1 at a time */
  FOR var := int-expression-1 TO int-expression-2 DO
    statement

  /* minus 1 at a time */
  FOR var := int-expression-1 DOWNTO int-expression-2 DO
    statement

  ▶ the value of the looping variable at the end of a loop is the last looping value
  ▶ if the loop is not executed, then the value of the looping variable stays unchanged.

- **while loop**

  WHILE boolean-expression DO
    statement
Scores

- Teams
  - Two persons per team
  - One person per team: project score \( \times 1.1 \)

- Phases: in this order.
  - 1. (20%) float variables and expressions
  - 2. (30%) constant and typedef
  - 3. (50%) 1-D array and then multi-D array
  - 4. (70%) boolean expressions, conditional and looping statements
  - 5. (90%) procedure and function with call-by-value parameters
  - 6. (100%) call-by-reference parameters

- Bonus: do these only when everything above is done.
  - record: + 10%
    - type1 = RECORD a,b:INTEGER; END;
    - array of records
    - \( X.a \) to access a field
  - pointer: +10 %
    - ptr = ^INTEGER;
Submitted packages

- Format of your package: check out the TA’s web site.

- Your final project package must include
  
  - A make file that produces a compiler with the name equaling your team name, compiles and runs all test programs.
  
  - A collection of test programs, inputs and anticipated outputs.
    
    - programX.p: program.
    - inputX_Y: input test data.
    - outputX_Y: output data.
    - readmeX: documentation for programX, contains the purpose of having test programX.
    - Example: program1.p, input1_1, input1_2, output1_1, output1_2 and readme1.

  - Documentation (in PDF, PS, TXT or HTML format):
    
    - Language reference manual: language.xxx
    - List of features implemented and their corresponding test programs: features.xxx
    - Implementation manual: internal.xxx contains the implementation details.
    - Other helpful documents: otherX.xxx
Correctness (50%)
- 35%: produce right codes on correct programs in reasonable time.
- 15%: detect and report errors on incorrect programs.

Elegance (20%):
- 5%: algorithmic issues.
- 10%: nice, exact and helpful error reporting.
- 5%: coding.

Documentation and Testing (30%):
- 15%: manuals.
- 15%: test programs.