# SE2052 – Programming Paradigms

# ALGOL 68 & Pascal Programming

Lab sheet 6: IT23746114

## ALGOL 68 Discussion

1. How did ALGOL 68 improve structured programming?

ALGOL 68 improved structured programming by providing clear **block structures, nested procedures**, and **consistent control constructs (like FOR, IF, CASE)** instead of relying on unstructured jumps such as GOTO. This made programs more readable, maintainable, and modular, reducing errors and making the flow of execution easier to understand.

2. Show an example of lexical scoping in ALGOL 68.

Lexical scoping means that a variable is accessible only within the block it is declared in, including nested blocks.

```
BEGIN
  INT x := 10;
  BEGIN
    INT y := x + 5;
    print((y, new line))
  END;
END
```

3. Why is recursion significant in ALGOL 68?

Recursion allows **functions or procedures to call themselves**, which is useful for tasks like factorials, tree **traversal, or mathematical sequences**. ALGOL 68 supports recursion naturally, enabling elegant solutions to complex problems without iterative loops for certain algorithms.

4. Compare ALGOL 68's structure with label-based control in earlier languages.

Earlier languages like FORTRAN and early ALGOL 60 often relied on labels and GOTO statements for control flow, which could lead to “spaghetti code.” ALGOL 68 replaced this with structured loops, conditionals, and block scopes, making code more predictable, readable, and maintainable.

5. How did ALGOL 68 influence modern languages?

    ALGOL 68 introduced concepts like strong typing, flexible arrays, and structured data types. These ideas influenced languages such as Pascal, C, Ada, and even Java, particularly in terms of modularity, scoping rules, and structured programming principles.

## Pascal Discussion

1. How did Pascal encourage structured programming compared to earlier languages like FORTRAN?

    Pascal enforced clear block structures, strong typing, and nested procedures, unlike FORTRAN which often relied on GOTO statements and loose variable scoping. This promoted readable, maintainable code and logical program flow, especially for educational purposes.

2. Show an example of lexical scoping in Pascal using nested procedures.

```
program LexicalScopeExample;
var
  x: integer;

procedure Outer;
var
  y: integer;
  procedure Inner;
  var
    z: integer;
  begin
    z := x + y;  { x from global, y from Outer }
    writeln('z = ', z);
  end;
begin
  y := 5;
  Inner;
end;

begin
  x := 10;
  Outer;
end.
```

Here, Inner can access y from Outer and x from global scope, but variables declared inside Inner aren't accessible outside—lexical scoping in action

3. Why are procedures and functions important in Pascal's modular programming?

   Procedures and functions allow breaking programs into reusable modules. This supports code readability, easier debugging, and logical separation of concerns, which is essential for large programs.

4. Compare Pascal's type safety with ALGOL 68's strong typing.

   Both languages enforce strong typing, meaning variables must be declared with specific types and type mismatches are caught at compile-time. Pascal emphasizes this for educational clarity, while ALGOL 68 provides additional flexibility with structured types and arrays, making it more powerful but slightly more complex.

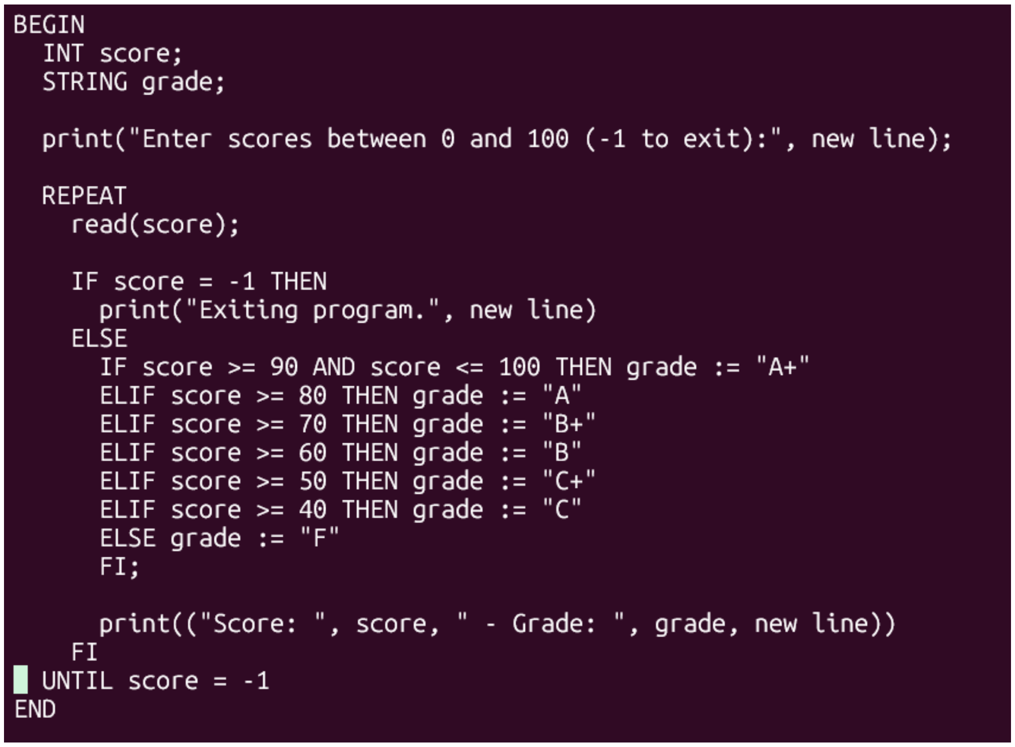
5. How did Pascal influence later languages like Modula, Ada, or Delphi?

   Pascal's design principles—structured programming, strong typing, and modularity—were adopted in Modula (modules), Ada (safety and reliability), and Delphi (object-oriented extensions). It provided a foundation for modern programming practices, especially in teaching and system programming.

## Comprehensive Exercise

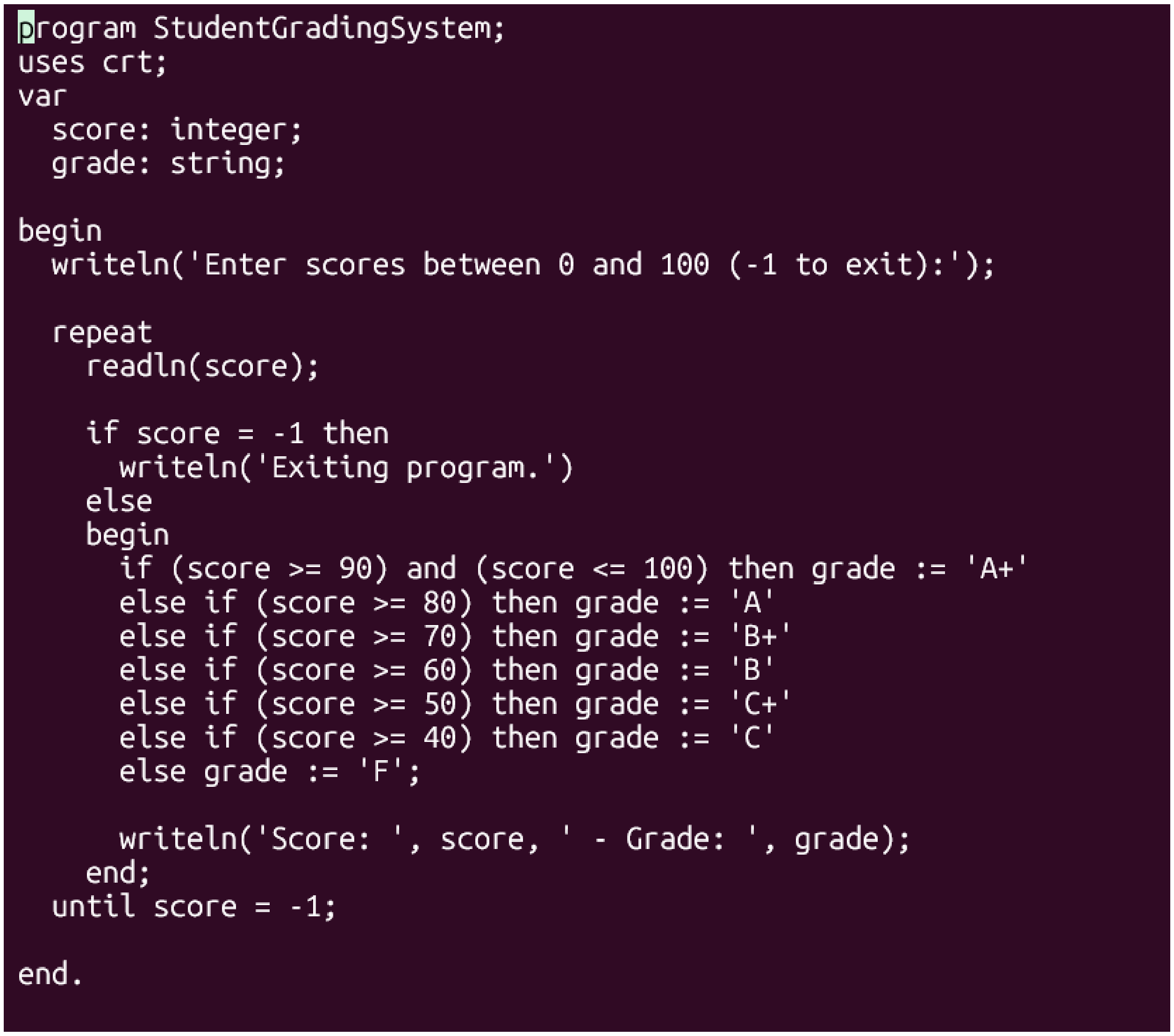
Task: Student Grading System

ALGOL 68 Version



```
BEGIN
  INT score;
  STRING grade;

  print("Enter scores between 0 and 100 (-1 to exit):", new line);

  REPEAT
    read(score);

    IF score = -1 THEN
      print("Exiting program.", new line)
    ELSE
      IF score >= 90 AND score <= 100 THEN grade := "A+"
      ELIF score >= 80 THEN grade := "A"
      ELIF score >= 70 THEN grade := "B+"
      ELIF score >= 60 THEN grade := "B"
      ELIF score >= 50 THEN grade := "C+"
      ELIF score >= 40 THEN grade := "C"
      ELSE grade := "F"
      FI;

      print(("Score: ", score, " - Grade: ", grade, new line))
    FI
  UNTIL score = -1
END
```

Pascal Version



```
program StudentGradingSystem;
uses crt;
var
  score: integer;
  grade: string;

begin
  writeln('Enter scores between 0 and 100 (-1 to exit):');

  repeat
    readln(score);

    if score = -1 then
      writeln('Exiting program.')
    else
    begin
      if (score >= 90) and (score <= 100) then grade := 'A+'
      else if (score >= 80) then grade := 'A'
      else if (score >= 70) then grade := 'B+'
      else if (score >= 60) then grade := 'B'
      else if (score >= 50) then grade := 'C+'
      else if (score >= 40) then grade := 'C'
      else grade := 'F';

      writeln('Score: ', score, ' - Grade: ', grade);
    end;
  until score = -1;

end.
```