Lecture 13

Programming in Modula-P, examples including rank sort and an overview of language features.

Modula-P

- Based on Modula-II with additional language features to control parallelism.
- Very similar in syntax to Pascal.
- Developed by the Concurrency Research Group at La Trobe University,
  - Dr. Rhys Francis
  - Dr. Ian Mathieson
  - Dr. Arnold Pears.

Notes on syntax

- The key-word begin is not used to commence blocks of statements, except when defining procedures.

Eg.

```
FOR I := 12 TO 65 DO
        <statementlist>
END;

WHILE (Index < 16) DO
        <statementlist>
END;
```


Notes on syntax

- Procedures and Modules must terminate with a named end statement.

Eg.

```
PROCEDURE XMin ( <parameterlist>);
VAR
        <declist>
BEGIN
        <statementlist>
END XMin;
```

Generating parallelism

- Modula-P provides three main language features for generating parallel execution.
  - FORALL loops
  - Parallel PROCEDURES
  - The & operator for executing dissimilar procedures in parallel.

Modules

- A Module is an object which can export data types and procedure definitions (or method interfaces) to other modules.
- There are two types of Module.
- Definition modules contain the data type and formal interface definitions that can be imported and used in other modules.
- Implementation modules contain the implementation code for the methods, and local data types and variables.

FORALL

- Operates on the local data of the procedure in which it is used.
- Has access to global data structures directly.
- Provides each copy of the loop body with a unique value in the index range.
- This is not a good technique to use to generate large scale parallelism, since the majority of the variables are public.

Module structure

```
MODULE <name>;
TYPE
   <typedecls>
VAR
   <vardecls>

<procedures>

BEGIN
   <statementlist>
END <name>.
```

Rank sorting

- Many of the existing sorting algorithms are inherently sequential. That is each iteration depends on the preceeding iterations in order to realise the sort.
- If we want to sort values in parallel new approaches are required.
- One such approach computes the final position of each value in a sequence simultaneously and then places all the values in a solution array.

Parallel procedures

- Defined in the same way as normal procedures.
- Called using a range in one or more actual parameter.

  Xmin(1..Degree, <other paramaters>);
- Have access to global variables, but also have local variables and parameters which are private to each copy of the procedure.

Rank sort algorithm

select a unique value

determine its final position

place that value into that position

Using range parameters

- Formal parameters whose actual parameter is a range are bound to a unique value in the range.
- This allows different copies of the procedure to work on different pieces of global data.
- Range parameters can also be used by the procedure copies to determine how they should co-operate with their siblings.

Rank sort in Modula-P

```
PROCDEDURE RSort (Id, N:INTEGER, VAR A, F: IntVector);
VAR
   Val, Rank, Index: INTEGER;
BEGIN
   (* Get my value *)
   Val:= A[Id];
   (* Find final position *)
   Rank:= 1;
   FOR Index:= 1 TO N DO
         IF (Val > A[Index]) THEN INC(Rank) END;
   END;
   (* Place in sorted array *)
   F[Rank]:= Val;
END RSort;
```


What happens?

A	0	6	9	1	5	3	2	8	7

What happens?

A	0	6	9	1	5	3	2	8	7

Val	Val	Val	Val	Val	Val	Val	Val	Val
0	0	0	0	0	0	0	0	0
Rank	Rank	Rank	Rank	Rank	Rank	Rank	Rank	Rank
0	0	0	0	0	0	0	0	0
Id	Id	Id	Id	Id	Id	Id	Id	Id
1	2	3	4	5	6	7	8	9

F	0	0	0	0	0	0	0	0	0

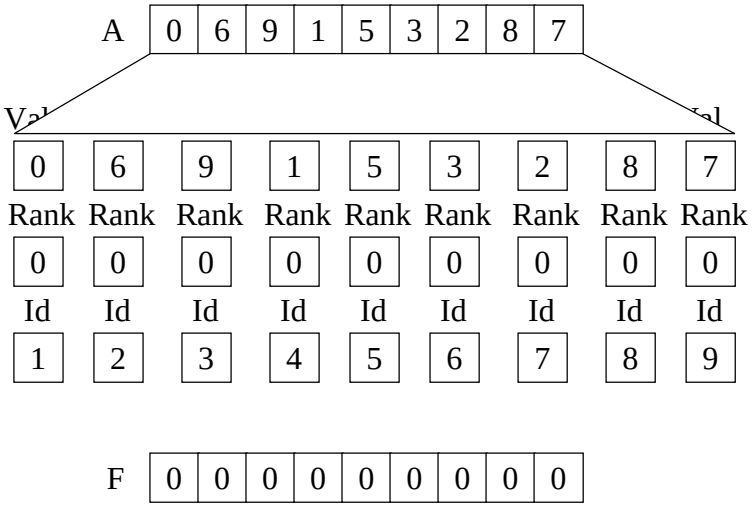
What happens?

A	0	6	9	1	5	3	2	8	7

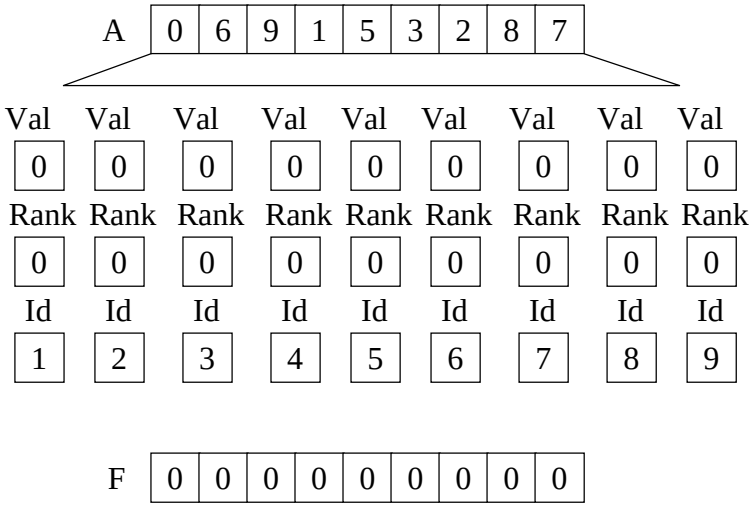
Val	Val	Val	Val	Val	Val	Val	Val	Val
0	0	0	0	0	0	0	0	0
Rank	Rank	Rank	Rank	Rank	Rank	Rank	Rank	Rank
0	0	0	0	0	0	0	0	0
Id	Id	Id	Id	Id	Id	Id	Id	Id
1	2	3	4	5	6	7	8	9

F	0	0	0	0	0	0	0	0	0

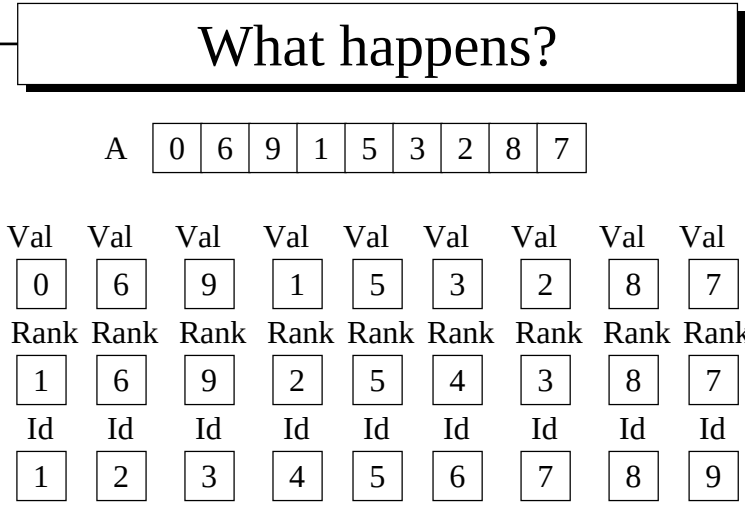
What happens?



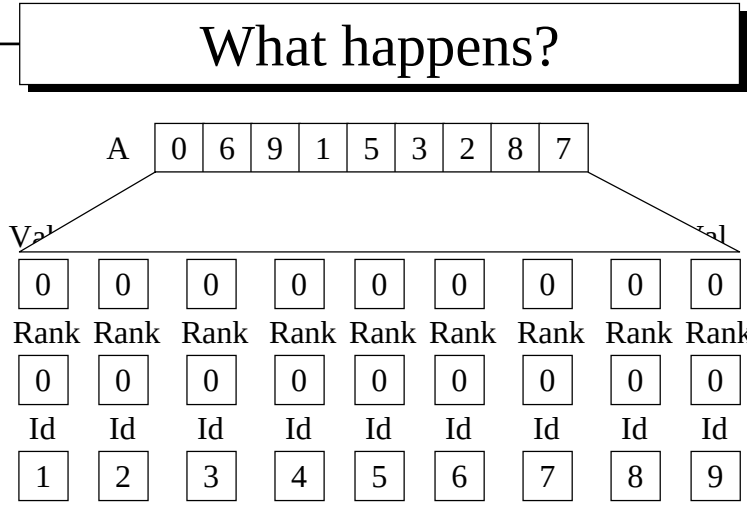
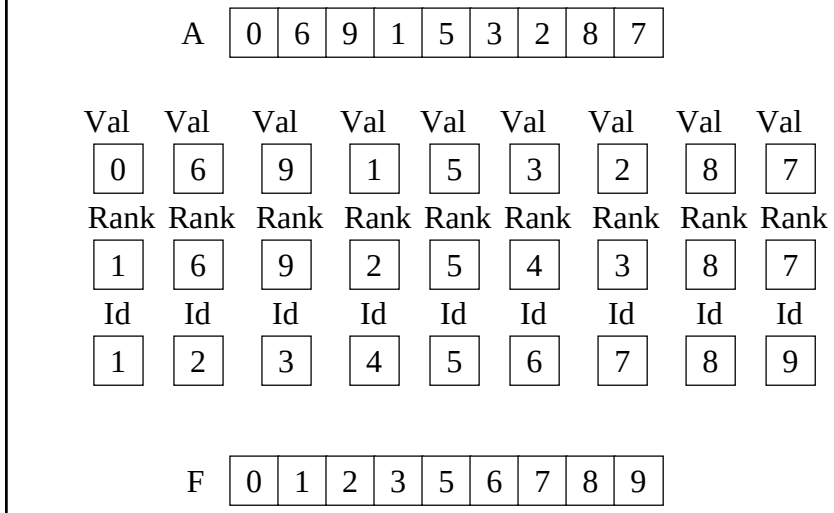
What happens?



What happens?



What happens?



What happens?