INF222 Monday Group sessions

Procedures - v2023

Sander Wiig

Institute for Informatics
University of Bergen

20 February 2023

UNIVERSITY OF BERGEN
Faculty of Mathematics and Natural Scien

Script and Presentation

Figure: https://tinyurl.com/2t8xc34h

What is a Procedure

- Procedures are "programs within programs"
- Procedures have their own environment
- Functions $\neq$ Procedures

Anatomy of a Procedure

```
1
2 procedure <procedure_name> (<params>)
3     <procedure code>
```

Figure: Procedure structure

Declaration vs. Calling

```
1  program Proc_Example
2  begin
3     procedure swap (upd x: integer, upd y:integer)
4     begin
5        var tmp : integer; //
6        tmp := x;
7        x := y;
8        y := tmp;
9     end
10
11    procedure main ()
12    begin
13       var a = 4;
14       var b = 5;
15       call tmp(4,5);
16    end
17 end
```

Figure: Swap Procedure

Parameter Semantics

- **OBS** - "read only"
- **UPD** - "read/write"
- **OUT** - "write only"

Reference Semantics

- Parameters become aliased to arguments
- Points to same memory address
- Unsafe, but sometimes useful

Running a procedure with reference semantics

1. Get stackframe
2. Wipe environment
3. Add parameters to the environment with same address as arg
4. run the procedure code
5. restore the environment

Copy Semantics

- Parameters are declared as variables and initialized with args' value
- Safer
- More intuitive behavior
- More complicated to implement

Running a procedure with copy semantics

1. Get stackframe
2. Get values of args
3. Wipe environment
4. Add parameters to environment
5. init those parameters with the arg values
6. run the procedure code
7. get the values of the parameters
8. restore the environment
9. copy the parameter values back to the args

Swap example v2

```
1  procedure GroupSwap (upd x: integer, upd y :integer)
2  begin
3      y := x + y;
4      x := y - x;
5      y := y - x;
6  end;
7
8  procedure SelfSwap();
9  begin
10     var a = 5;
11     call GroupSwap (a, a);
12 end;
```

Reference semantics

```
1  procedure GroupSwap (upd x: integer, upd y :integer)
2  begin
3      y := x + y;
4      x := y - x;
5      y := y - x;
6  end;
7
8  procedure SelfSwap();
9  begin
10     var a = 5;
11     call GroupSwap (a, a); //x =5, y = 5
12 end;
```

Reference semantics

```
1  procedure GroupSwap (upd x: integer, upd y :integer)
2  begin
3      y := x + y;// y = x + y = 5+5 => y = 10
4      x := y - x;
5      y := y - x;
6  end;
7
8  procedure SelfSwap();
9  begin
10     var a = 5;
11     call GroupSwap (a, a); //x =5, y = 5
12 end;
```

Reference semantics

```
1  procedure GroupSwap (upd x: integer, upd y :integer)
2  begin
3      y := x + y;// y = x + y = 5+5 => y = 10
4      x := y - x;// x = y - x = 10 - 5 => x = 5
5      y := y - x;
6  end;
7
8  procedure SelfSwap();
9  begin
10     var a = 5;
11     call GroupSwap (a, a); //x =5, y = 5
12 end;
```

Reference semantics

```
1  procedure GroupSwap (upd x: integer, upd y :integer)
2  begin
3     y := x + y;// y = x + y = 5+5 => y = 10
4     x := y - x;// x = y - x = 10 - 5 => x = 5
5     y := y - x;// y = y - x = 10 - 5 => y = 5
6  end;
7
8  procedure SelfSwap();
9  begin
10    var a = 5;
11    call GroupSwap (a, a); //x =5, y = 5
12 end;
```

Copy semantics

```
1  procedure GroupSwap (upd x: integer, upd y :integer)
2  begin
3      y := x + y;
4      x := y - x;
5      y := y - x;
6  end;
7
8  procedure SelfSwap();
9  begin
10     var a = 5;
11     call GroupSwap (a, a); //x =a, y = a
12 end;
```

Copy semantics

```
1  procedure GroupSwap (upd x: integer, upd y :integer)
2  begin
3      y := x + y;// a = a + a = 5+5 => a = 10
4      x := y - x;
5      y := y - x;
6  end;
7
8  procedure SelfSwap();
9  begin
10     var a = 5;
11     call GroupSwap (a, a); //x =a, y = a
12 end;
```

Copy semantics

```
1  procedure GroupSwap (upd x: integer, upd y :integer)
2  begin
3      y := x + y;// a = a + a = 5+5 => a = 10
4      x := y - x;// a = a - a = 10 - 10 => a = 0
5      y := y - x;
6  end;
7
8  procedure SelfSwap();
9  begin
10     var a = 5;
11     call GroupSwap (a, a); //x =a, y = a
12 end;
```

Copy semantics

```
1  procedure GroupSwap (upd x: integer, upd y :integer)
2  begin
3      y := x + y;// a = a + a = 5+5 => a = 10
4      x := y - x;// a = a - a = 10 - 10 => a = 0
5      y := y - x;// a = a - a = 0 - 0 => a = 0
6  end;
7
8  procedure SelfSwap();
9  begin
10     var a = 5;
11     call GroupSwap (a, a); //x =a, y = a
12 end;
```