



Computational Thinking

Prof. Madhavan Mukund

Department of Computer Science
Chennai Mathematical Institute

Prof. G. Venkatesh

Indian Institute of Technology Madras

Mr. Omkar Joshi

Course Instructor
IITM Online Degree Programme



Computational Thinking

Tutorial on pseudocode
for fundamentals of programming

Content

- Filtering
 - if block
 - if – else blocks
 - if – else ladder
 - Nested if – else blocks
- Iteration
 - Multiple iterations (non-nested)
 - Nested iterations
- Procedures
- return vs. exitloop

Find the number of students based on their total marks

A = 0

```
while (Pile 1 has more cards) {  
    Read the top card X from Pile 1  
    if (X.Total > 220) {  
        A = A + 1  
    }  
    Move X to Pile 2  
}
```

Find the number of students based on their total marks

```
A = 0
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    if (X.Total > 220) {
        A = A + 1
    }
    Move X to Pile 2
}

A = 0, B = 0
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    if (X.Total > 220) {
        A = A + 1
    }
    else {
        B = B + 1
    }
    Move X to Pile 2
}
```

Find the number of students based on their total marks

```
A = 0
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    if (X.Total > 220) {
        A = A + 1
    }
    Move X to Pile 2
}
```

```
A = 0, B = 0
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    if (X.Total > 220) {
        A = A + 1
    }
    else {
        B = B + 1
    }
    Move X to Pile 2
}
```

```
A = 0, B = 0, C = 0, D = 0
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    if (X.Total >= 250) {
        A = A + 1
    }
    if (X.Total > 220) {
        B = B + 1
    }
    if (X.Total > 200) {
        C = C + 1
    }
    else {
        D = D + 1
    }
    Move X to Pile 2
}
```

Find the number of students based on their total marks and City/Town

```
A = 0, B = 0, C = 0, D = 0
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    if (X.CityTown == "Chennai") {
        if (X.Total >= 210) {
            A = A + 1
        }
        else {
            B = B + 1
        }
    }
    else {
        if (X.Total >= 210) {
            C = C + 1
        }
        else {
            D = D + 1
        }
    }
    Move X to Pile 2
}
```

Find the number of students above average based on their total marks

```
count = 0, sum = 0, avg = 0, A = 0
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    count = count + 1
    sum = sum + X.Total
    Move X to Pile 2
}
avg = sum / count
while (Pile 2 has more cards) {
    Read the top card X from Pile 2
    if (X.Total > avg) {
        A = A + 1
    }
    Move X to Pile 1
}
```


Find the number of pairs of students who scored same marks in Mathematics

```
pair = 0
while (Table 1 has more rows) {
    Read the top row X from Table 1
    Move X to Table 2
    while (Table 1 has more rows) {
        Read the top card Y from Table 1
        if (X.Mathematics == Y.Mathematics) {
            pair = pair + 1
        }
        Move Y to Table 3
    }
    Move all rows from Pile 3 to Table 1
}
```

Table 1	
SeqNo	Mathematics
1	62
6	81
17	62
21	78
27	81

Table 3	
SeqNo	Mathematics

Table 2	
SeqNo	Mathematics

Find the number of pairs of students who scores same marks in Mathematics

```
pair = 0
while (Table 1 has more rows) {
    Read the top row X from Table 1
    Move X to Table 2
    while (Table 1 has more rows) {
        Read the top card Y from Table 1
        if (X.Mathematics == Y.Mathematics) {
            pair = pair + 1
        }
        Move Y to Table 3
    }
    Move all rows from Pile 3 to Table 1
}
```

Table 1	
SeqNo	Mathematics
6	81
17	62
21	78
27	81

Table 3	
SeqNo	Mathematics

Table 2	
SeqNo	Mathematics
1	62

Find the number of pairs of students who scores same marks in Mathematics

```
pair = 0
while (Table 1 has more rows) {
    Read the top row X from Table 1
    Move X to Table 2
    while (Table 1 has more rows) {
        Read the top card Y from Table 1
        if (X.Mathematics == Y.Mathematics) {
            pair = pair + 1
        }
        Move Y to Table 3
    }
    Move all rows from Pile 3 to Table 1
}
```

Table 1	
SeqNo	Mathematics
17	62
21	78
27	81

Table 3	
SeqNo	Mathematics
6	81

Table 2	
SeqNo	Mathematics
1	62

Find the number of pairs of students who scores same marks in Mathematics

```
pair = 0
while (Table 1 has more rows) {
    Read the top row X from Table 1
    Move X to Table 2
    while (Table 1 has more rows) {
        Read the top card Y from Table 1
        if (X.Mathematics == Y.Mathematics) {
            pair = pair + 1
        }
        Move Y to Table 3
    }
    Move all rows from Pile 3 to Table 1
}
```

Table 1	
SeqNo	Mathematics
21	78
27	81

Table 3	
SeqNo	Mathematics
6	81
17	62

Table 2	
SeqNo	Mathematics
1	62

Find the number of pairs of students who scores same marks in Mathematics

pair = 0

while (Table 1 has more rows) {

 Read the top row X from Table 1

 Move X to Table 2

 while (Table 1 has more rows) {

 Read the top card Y from Table 1

 if (X.Mathematics == Y.Mathematics) {

 pair = pair + 1

 }

 Move Y to Table 3

 }

 Move all rows from Pile 3 to Table 1

}

Table 1	
SeqNo	Mathematics

Table 3	
SeqNo	Mathematics
6	81
17	62
21	78
27	81

Table 2	
SeqNo	Mathematics
1	62

Find the number of pairs of students who scores same marks in Mathematics

```
pair = 0
while (Table 1 has more rows) {
    Read the top row X from Table 1
    Move X to Table 2
    while (Table 1 has more rows) {
        Read the top card Y from Table 1
        if (X.Mathematics == Y.Mathematics) {
            pair = pair + 1
        }
        Move Y to Table 3
    }
    Move all rows from Pile 3 to Table 1
}
```

Table 1	
SeqNo	Mathematics
6	81
17	62
21	78
27	81

Table 3	
SeqNo	Mathematics

Table 2	
SeqNo	Mathematics
1	62

Find the number of pairs of students who scores same marks in Mathematics

```
pair = 0
while (Table 1 has more rows) {
    Read the top row X from Table 1
    Move X to Table 2
    while (Table 1 has more rows) {
        Read the top card Y from Table 1
        if (X.Mathematics == Y.Mathematics) {
            pair = pair + 1
        }
        Move Y to Table 3
    }
    Move all rows from Pile 3 to Table 1
}
```

Table 1	
SeqNo	Mathematics
17	62
21	78
27	81

Table 3	
SeqNo	Mathematics

Table 2	
SeqNo	Mathematics
1	62
6	81

Find the number of pairs of students who scores same marks in Mathematics

```
pair = 0
```

```
while (Table 1 has more rows) {
```

```
    Read the top row X from Table 1
```

```
    Move X to Table 2
```

```
    while (Table 1 has more rows) {
```

```
        Read the top card Y from Table 1
```

```
        if (X.Mathematics == Y.Mathematics) {
```

```
            pair = pair + 1
```

```
        }
```

```
        Move Y to Table 3
```

```
    }
```

```
    Move all rows from Pile 3 to Table 1
```

```
}
```

Table 1	
SeqNo	Mathematics

Table 3	
SeqNo	Mathematics

Table 2	
SeqNo	Mathematics
1	62
6	81
17	62
21	78
27	81

Find the subject topper of Mathematics, Physics and Chemistry

```
mMax = 0, pMax = 0, cMax = 0
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    if (X.Mathematics > mMax) {
        mMax = X.Mathematics
    }
    Move X to Pile 2
}
while (Pile 2 has more cards) {
    Read the top card X from Pile 2
    if (X.Physics > pMax) {
        pMax = X. Physics
    }
    Move X to Pile 1
}
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    if (X.Chemistry > cMax) {
        cMax = X.Chemistry
    }
    Move X to Pile 2
}
```

Find the subject topper of Mathematics, Physics and Chemistry

mMax = 0, pMax = 0, cMax = 0

```
while (Pile 1 has more cards) {  
    Read the top card X from Pile 1  
    if (X.Mathematics > mMax) {  
        mMax = X.Mathematics  
    }  
    Move X to Pile 2  
}
```

```
while (Pile 2 has more cards) {  
    Read the top card X from Pile 2  
    if (X.Physics > pMax) {  
        pMax = X. Physics  
    }  
    Move X to Pile 1  
}
```

```
while (Pile 1 has more cards) {  
    Read the top card X from Pile 1  
    if (X.Chemistry > cMax) {  
        cMax = X.Chemistry  
    }  
    Move X to Pile 2  
}
```

mMax = 0, pMax = 0, cMax = 0

mMax = findMax ("Mathematics")

pMax = findMax ("Physics")

cMax = findMax ("Chemistry")

Procedure findMax (subject)

max = 0

```
while (Pile 1 has more cards) {  
    Read the top card X from Pile 1  
    if (X.subject > max) {  
        max = X.subject  
    }  
    Move X to Pile 2  
}
```

```
Move all cards from Pile 2 to Pile 1  
return (max)
```

End findMax

Find the subject topper of Mathematics, Physics and Chemistry

Procedure call

```
mMax = 0, pMax = 0, cMax = 0  
mMax = findMax ("Mathematics")  
pMax = findMax ("Physics")  
cMax = findMax ("Chemistry")
```

Procedure definition

```
Procedure findMax (subject)  
    max = 0  
    while (Pile 1 has more cards) {  
        Read the top card X from Pile 1  
        if (X.subject > max) {  
            max = X.subject  
        }  
        Move X to Pile 2  
    }  
    Move all cards from Pile 2 to Pile 1  
    return (max)  
End findMax
```

Advantages of procedures

- Avoids repetition of code
- Divides the complex problem into smaller ones
- Makes it easy to read the code
- Modifying the pseudocode becomes easier

return vs. exitloop

return	exitloop
return statement terminates the procedures and flow of the code goes back to procedure call.	exitloop terminates the only loop from which the statement has been called.
<pre>A = isEven (num) Procedure isEven (n) if (n % 2 == 0) { return (True) } return (False) End isEven</pre>	<pre>i = 1, j = 1, n = 4, sum = 0 while (i < n) { while (j < n) { if (i == j) { exitloop } sum = sum + j j = j + 1 } i = i + 1 }</pre>

```

i = 1, j = 1, n = 4, sum = 0
while (i < n) {
    while (j < n) {
        if (i == j) {
            exitloop
        }
        sum = sum + j
        j = j + 1
    }
    i = i + 1
}

```

Outer iteration	Inner iteration	Variables and values			Remark
		i	j	sum	
1	1	1	1	0	exitloop
	2	Not applicable			
	3				
2	1	2	1	1	sum updated
	2	2	2	1	exitloop
	3	Not applicable			
3	1	3	1	2	sum updated
	2	3	2	4	sum updated
	3	3	3	4	exitloop