L#3

Basics of Programming. Introduction

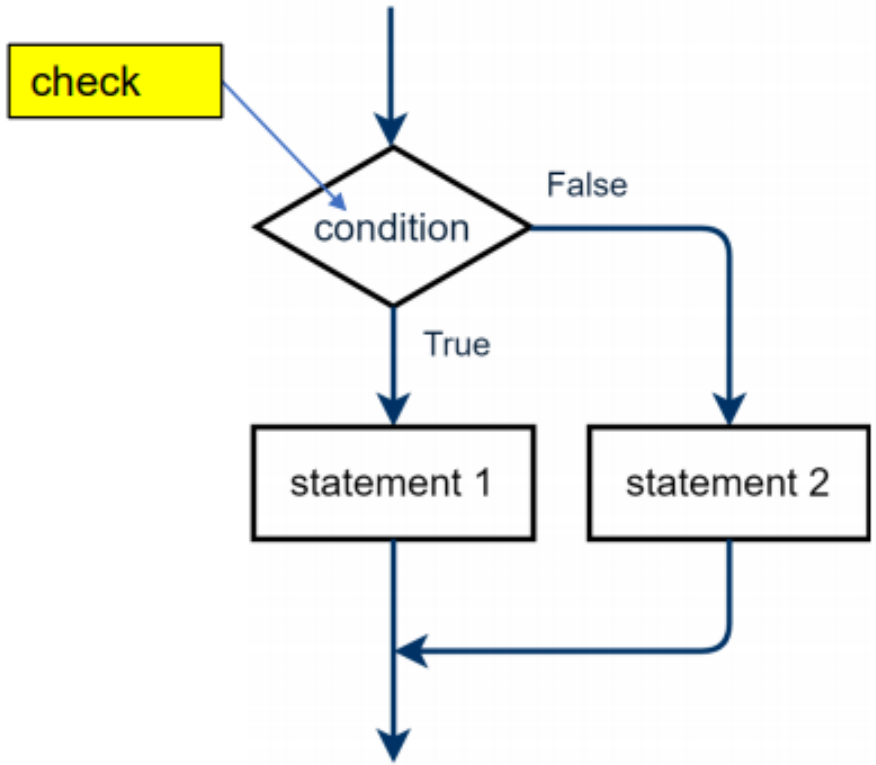
Course Basics of Programming Semester 1, FIIT

Mayer Svetlana Fyodorovna

Conditions

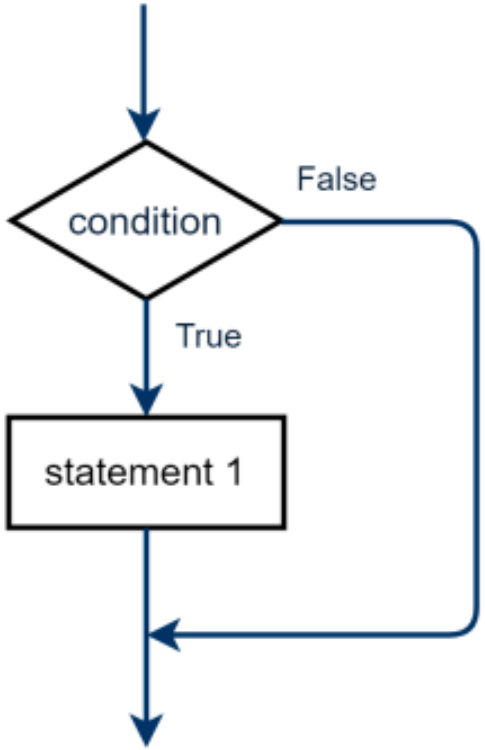
Complete form

```
if Mark > 2 then
   Print('OK')
else Print('Bad mark');
```



Incomplete form

```
if Mark > 2 then
   Print('OK');
```



Examples

- is the value of the variable even or odd?

```
if x mod 2 <> 0 then
   Print('Odd')
else Print('Even');
```

- Is it a one-digit number?

```
if n < 10 then
   Print('One-Digit number');
```

Tasks

- To do: tasks 6, 7, 8, 9

Boolean expressions

```
if Mark > 2 then
   Print('OK')
```

Expression Mark > 2 is a question

Mark > 2 ?

The answer on this question is **True** or **False**.

We say that the expression Mark > 2 has a type **boolean**.

This type has only two values: **True** or **False**.

- We can output the Boolean expressions:

```
var mark := 2;
Print($'is it true that mark >2? {mark > 2}') // output false
```

Boolean type

- We can define variables of Boolean type:

```
var Mark := ReadInteger('Enter Mark:');
var B: boolean;
B := Mark > 2;
if B then
  Print('OK')
```

Boolean operations: and, or, not

```
var A,B: boolean;
A := True;
B := False;
Print(A and B);
Print(A or B);

Print(not A);
```

- A and B is *True* if A is *True* and B is *True* at the same time. In other cases A and B is *False*
- A or B is *False* if A is *False* and B is *False*. In other cases A or B is *True*
- not A has opposite value: not A is *True* if A is *False*

Truth tables

Truth tables give us a values of boolean operations
A and B, A or B, not A
when A and B take all possible values

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

A	not A
True	False
False	True

Examples

```
var x := ReadReal;
var B := (x > 3) and (x < 5);
```

$$x \in (3,5)$$

```
var x := ReadInteger;
if (x = 1) or (x = 3) or (x = 5) then
  Print('x=1 or 3 or 5');
```

```
var x := ReadInteger;
if not (x > 2) then
  Print('x<=2');
```

Tasks

- To do: tasks 1, 2, 3, 4, 5

Nested If statements

Problem.

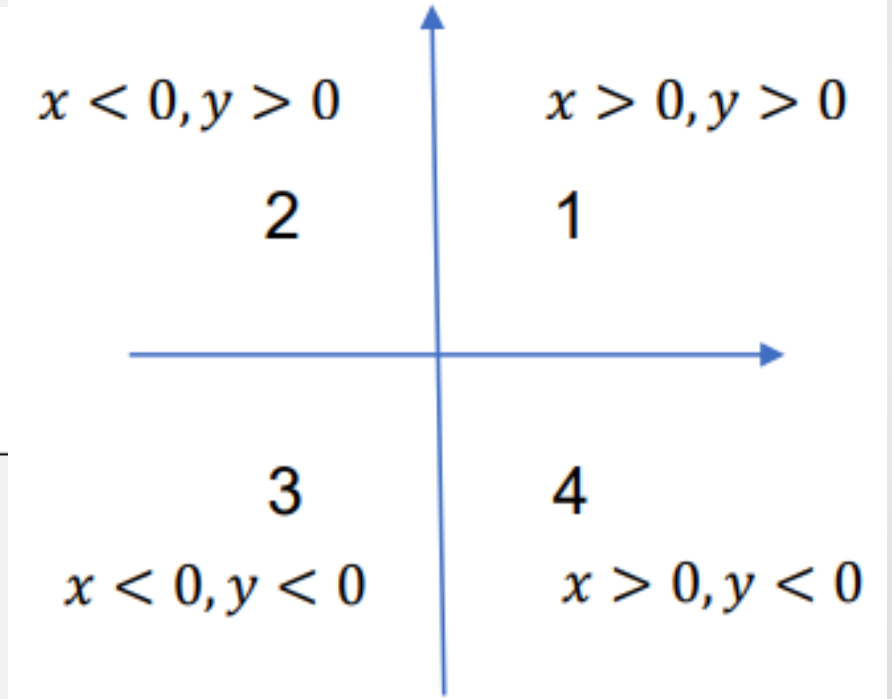
Given: Point (x,y) on a coordinate plane,

$x \neq 0, y \neq 0.$

Find: Number of quarter

Code:

```
var (x,y) := ReadInteger2;
var quater: integer;
Assert((x<>0) and (y<>0));
if x>0 then
  if y>0 then
    quarter := 1
  else quarter := 4;
else
  if y>0 then
    quarter := 2
  else quarter := 3;
```



Chained If statements

Given: a number of Season (Winter is first)

Output: Name of Season

Code:

```
var Season := ReadInteger('Enter Season (1..4):');
if Season=1 then
  Print('Winter')
else if Season=2 then
  Print('Spring')
else if Season=3 then
  Print('Summer')
else if Season=4 then
  Print('Autumn')
else Print('Wrong Season')
```

Logical (boolean) operations in if statements

- Conditions may consist of logic operations: not, or, and. If there are more than one condition, so each condition must be surrounded by round brackets.
- For example if we must check two conditions:

```
if (year < 20) or (year > 18)
then begin
// if body

end
```

If one of the conditions or both conditions are True then if body executes.

```
var a: = 5;
if (not (a<4)) and (7>5) then // ← True
begin
// if body

end
```

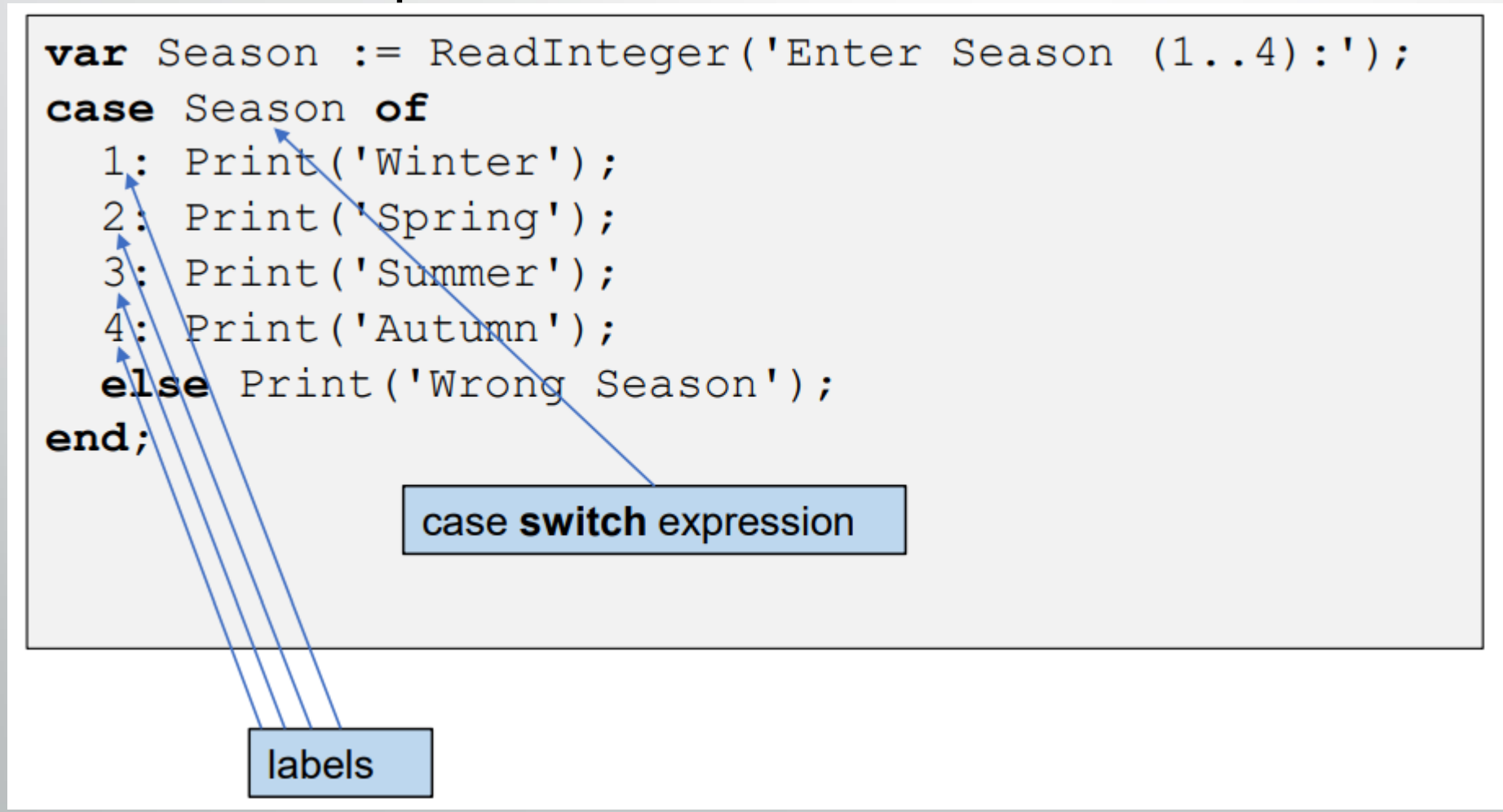
Tasks

- To do: tasks 10, 11, 12, 13, 14, 15, 16, 17

Case statement

- We use case statement in a situation of *multiple choice*.
- Previous example with chained if we can write as follows:

```
var Season := ReadInteger('Enter Season (1..4):');
case Season of
  1: Print('Winter');
  2: Print('Spring');
  3: Print('Summer');
  4: Print('Autumn');
  else Print('Wrong Season');
end;
```



Case: another examples

Example. Translator

```
var Word := ReadString;
var Translation: string;
case Word of
  'dog': Translation := 'собака';
  'use': Translation := 'использовать';
  'find': Translation := 'находить';
  else Translation := 'не знаю ☺';
end;
```

Tasks

- To do: tasks 18, 19, 20, 21, 22, 23

Case: using ranges and enumerations

Problem. Given a number of Month. Calculate a name of season

```
var Month := ReadInteger;
Assert((Month>=1) and (Month<=12));
var Season: string;
case Month of
  3..5: Season := 'Spring';
  6..8: Season := 'Summer';
  9..11: Season := 'Autumn';
  12,1,2: Season := 'Winter';
end;
```

Ranges & enumerations must **not overlap**

Tasks

- To do: tasks 24,25,26

Q & A