

# The Basics of Programming

Wolfgang Schreiner

Research Institute for Symbolic Computation (RISC)

Johannes Kepler University, Linz, Austria

[Wolfgang.Schreiner@risc.jku.at](mailto:Wolfgang.Schreiner@risc.jku.at)

<http://www.risc.jku.at>

## Basic Notions

- **Program:** formal description of a method solving a problem.
  - Consists of **instructions** (also called **commands** or **statements**).
  - Operates on **data** (values like numbers, characters, text files, images, ...).

We will now investigate those elements of programming that are valid for (almost) any programming language.

## Variables

Data are stored in variables.

- **Variable:** a labelled box holding some content.
  - The label is the **name** of the variable.
  - The content is the **value** of the variable.

- Example: variable named  $x$  with value 17:

$x$ : 17

- Variables may hold different kinds of values.
  - Integer numbers:  $x$ : 17
  - Characters:  $ch$ : '+'
  - Strings of characters:  $str$ : "hello"

**Think of variables as boxes.**

## Changing Variable Values

A program operates on variables and changes its values.

- Variables **before** the execution of a program.

*x*: 17  
*ch*: '+'  
*str*: "hello"

- Variables **after** the execution of the program.

*x*: 18  
*ch*: '\*'  
*str*: "world"

The name "variable" comes from "vary" (changing).

## Variable Types

Variable types may not change arbitrarily

- Every variable has an associated **type**.
  - The set of values that it may hold during its whole life-time.
  - The type remains fixed during its life-time.
  - The type may be attached to the visual representation.

*x*: 17 *integer*

*ch*: '+' *character*

*str*: "hello" *string*

For the moment, we will only use variables of type “integer”.

## Variable Assignments

The simplest and most important instruction.

$$x \leftarrow E$$

- Read as “ $x$  becomes  $E$ ” .
  - Variable  $x$ .
  - Mathematical expression  $E$ .
- Compute value of  $E$  and store it in  $x$ .
  - Overwrites previous value of  $x$ .

Assignments change variable values.

## Example

•  $x: \boxed{5}, y: \boxed{1}$

$y \leftarrow 2$

•  $x: \boxed{5}, y: \boxed{2}$

$y \leftarrow x + 1$

•  $x: \boxed{5}, y: \boxed{6}$

$x \leftarrow x + 1$

•  $x: \boxed{6}, y: \boxed{6}$

$x \leftarrow x * y$

•  $x: \boxed{36}, y: \boxed{6}$

The execution of a program is a sequence of variable assignments.

## Sequences

A program may consist of a sequence of commands:

- $x: \boxed{5}, y: \boxed{1}$

$$y \leftarrow 2$$

$$y \leftarrow x + 1$$

$$x \leftarrow x + 1$$

$$x \leftarrow x * y$$

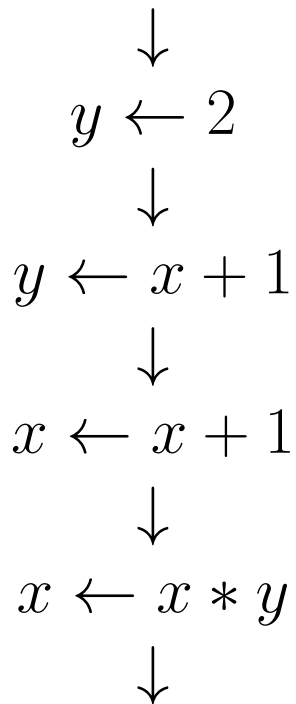
- $x: \boxed{36}, y: \boxed{6}$

Multiple commands may be listed one after another.



## Control Flow Diagrams

Arrows may indicate the order in which commands are executed.



Such diagrams will become important to visualize control structures.

## Assertions

Diagrams may be annotated by assertions.

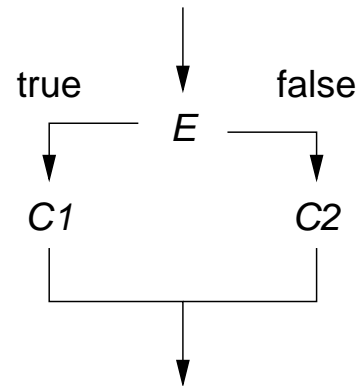
- **Assertion:** a proposition about the variable values.
  - Must hold at the denoted position of the program execution.

↓  
 $y \leftarrow 2 * x$   
↓ --  $y$  is even  
 $z \leftarrow y + 1$   
↓ --  $z$  is odd

Dashed line -- indicates position of assertion in control flow.

## Conditionals

A statement may be only executed if a condition holds.



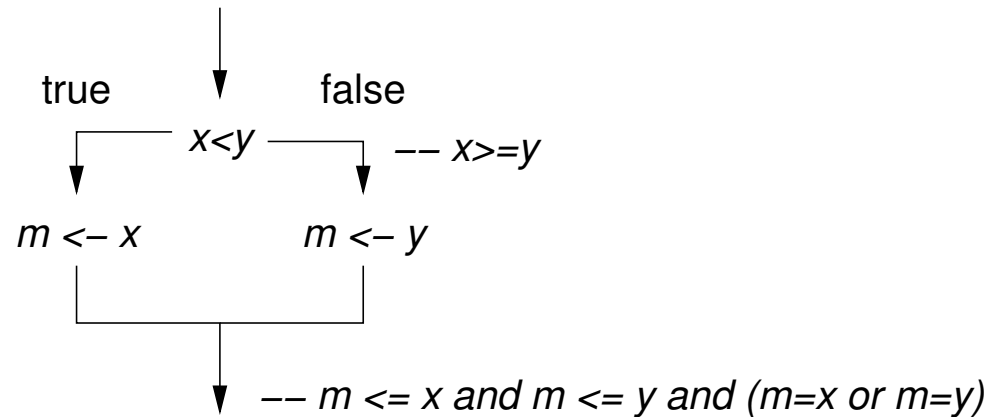
- **Conditional statement**

- Condition  $E$ .
- Commands  $C_1$  and  $C_2$  (the **branches**).
- If  $E$  holds, then  $C_1$  is executed, else  $C_2$  is executed.

**The control flow may branch.**

## Example

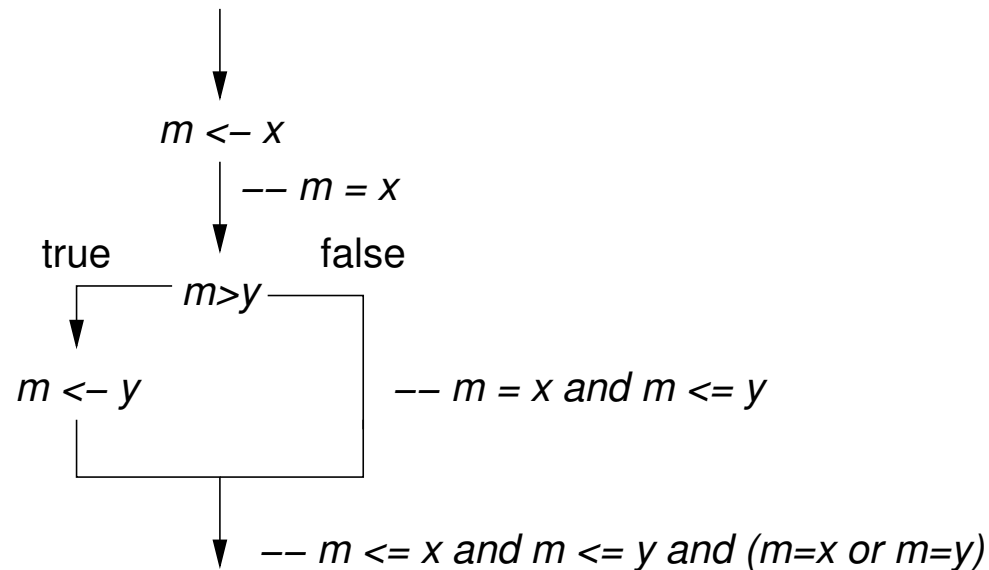
Computation of the minimum of two values.



Variable  $m$  is set to the smaller of the values of  $x$  and  $y$ .

## Example

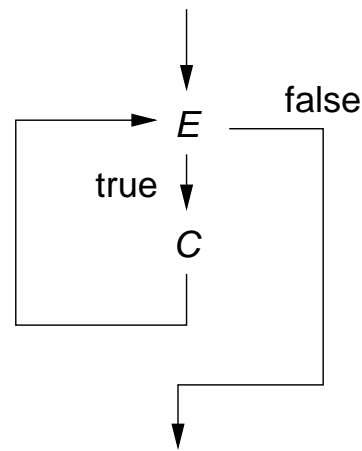
Alternative version of minimum computation.



A conditional may also have only one branch.

## Loops

A statement may be repeatedly executed (iterated).



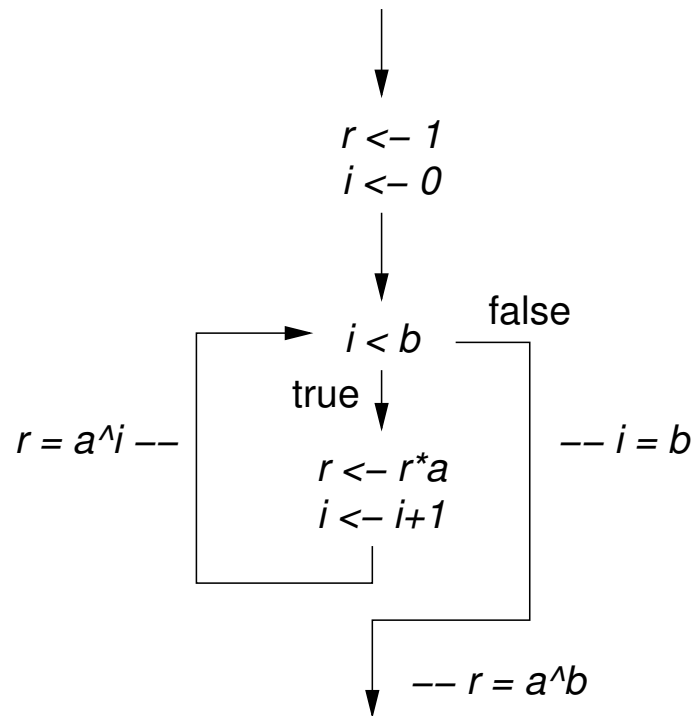
- Loop statement

- Condition  $E$  (the loop condition).
- Command  $C$  (the loop body).
- While  $E$  holds,  $C$  is executed.

The control flow may form loops.

## Example

The computation of an exponentiation value.

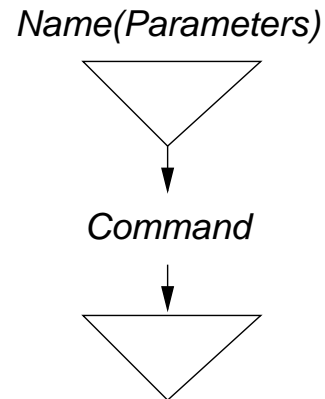


	$r$	$i$
Before first iteration	1	0
After first iteration	4	1
After second iteration	16	2
After third iteration	64	3

The variable  $r$  is set to  $a^b$ .

## Algorithm Descriptions

Format for describing algorithms.



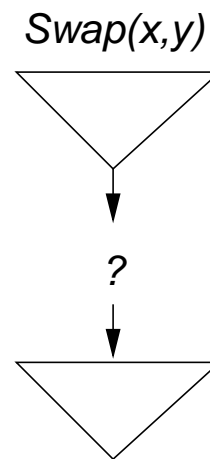
- Every algorithm has a **name** and **parameters**.
  - **Parameters**: variables provided by the user.
  - Algorithm may use these variables and internal ones.
- Triangles initiate and terminate the control flow.



## Swapping two Variables

Exchange the values of two variables.

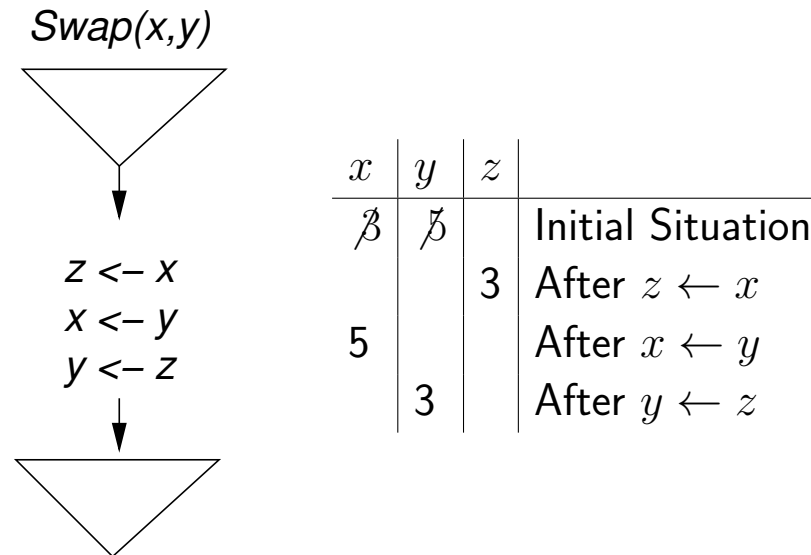
$$x: \boxed{3}, y: \boxed{5} \rightsquigarrow x: \boxed{5}, y: \boxed{3}$$



We have to develop the body of the algorithm skeleton.

## Swapping two Variables

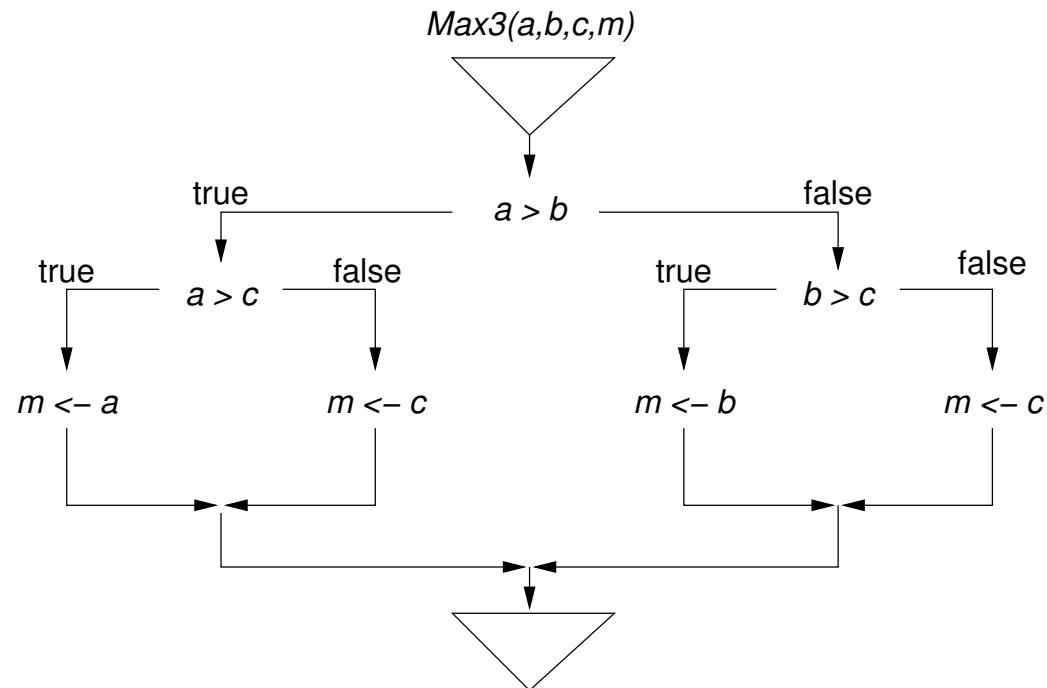
Algorithm needs an auxiliary variable.



Every algorithm can be simulated with paper and pencil.

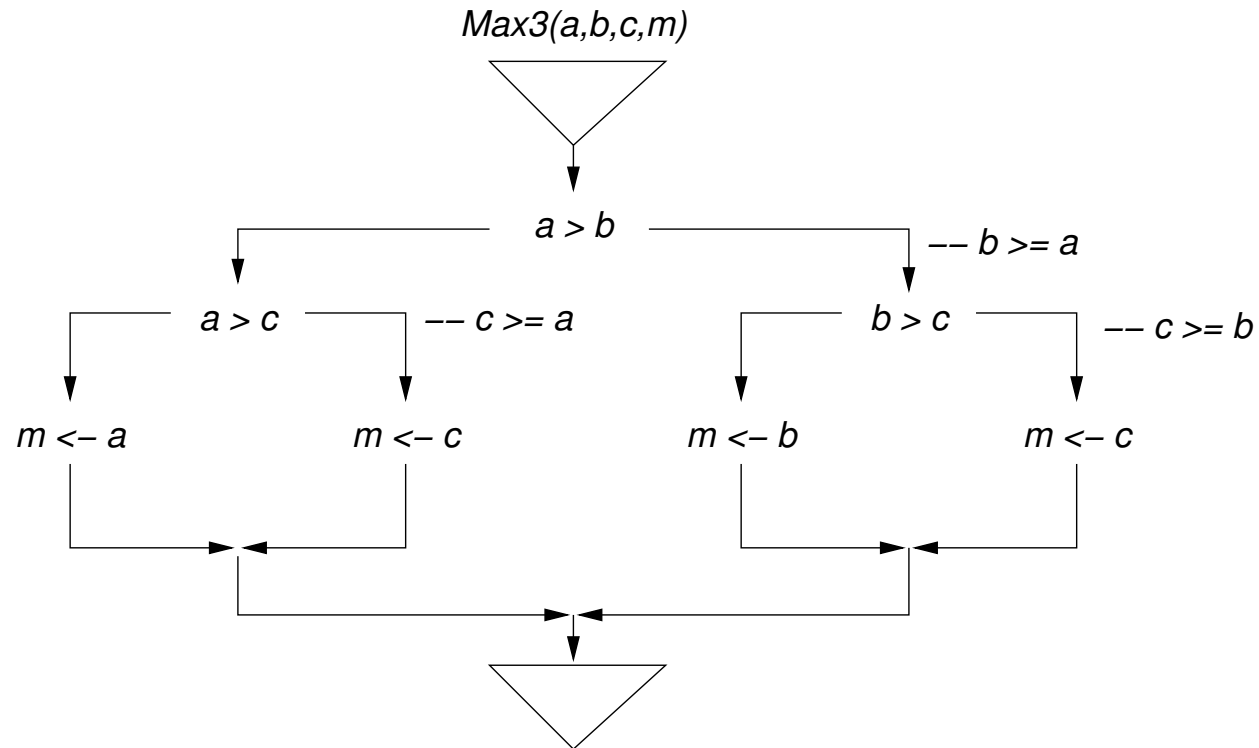
## Maximum of Three Numbers

Find the largest of three numbers.



How to verify the correctness of the algorithm?

# Maximum of Three Numbers



Use assertions to support the reasoning.

## Primality Test

Test whether a given natural number  $n$  is a prime number.

- **Prime number:** a natural number  $n$  such that
  - $n$  is greater than 1 and
  - $n$  is only divisible by 1 and itself.
    - \*  $n$  is divisible by  $m$  (written as  $m|n$ ) if and only if  $n = m \cdot o$  for some natural number  $o$ .
    - \* For example, 10 is divisible by 5 ( $5|10$ ) because  $10 = 5 \cdot 2$ .
- **Prime numbers:** 2, 3, 5, 7, 11, 13, 17, ...
  - 2 is the only even prime number (why?).

Signal success by setting a variable  $p$  to “true” or “false”.

# Primality Test

