EEE243 – Applied Computer Programming

Algorithms and problem solving skills

ROYAL MILITARY COLLEGE OF CANADA
ELECTRICAL & COMPUTER
ENGINEERING

GÉNIE ÉLECTRIQUE
ET GÉNIE INFORMATIQUE
COLLÈGE MILITAIRE ROYAL DU CANADA

RMC CMR

Applied Programming?

Go beyond useless programming!

- Control
- React to an environment
- Data manipulation
- Games

Algorithm

An algorithm is a self-contained step-by-step set of operations to be performed. It is and effective method that can be expressed within a finite amount of space and time in a well-define formal language for calculating a function [1]

Algorithm

- Cooking recipe
- Instructions to get to a place
- Solution of a mathematical problem
- Instruction for solving a problem
- etc.

Example of an algorithm

Finding square roots manually:

1. Separate your number’s digits into pairs (from right to left).
2. Find the largest integer *n* whose square is lesser than or equal to the leftmost number (or pair).
3. Subtract the number you just calculated from the leftmost pair.

Example of an algorithm

Finding square roots manually:

5. Drop down the next pair.
6. Find the largest integer *a* that results in a number smaller than the result of step 5 that satisfies the equation *(2r)a* x *a* where (2*r*)*a* is a number which first number is the current *result* multiplied by 2 and the second is *a*.

Example of an algorithm

Finding square roots manually:

7. Subtract the result of step 6 from the number in step 5.
8. Repeat steps 5 to 7 taking note of the values of *n* and *a*.
9. To continue to calculate digits, drop a pair of 00 on the left, and repeat steps 5 to 7 until you obtain the number with the desired decimal places.

Solution tirée de http://www.wikihow.com/Calculate-a-Square-Root-by-Hand

$$\sqrt{1013.51}$$

10 13 .51

$$\sqrt{1013.51}$$

10 13 .51

-

9

1 13

3

3 x 3 = 9

$$\sqrt{1013.51}$$

10 13 .51

-

9 ↓

——

1 13

3

3 x 3 = 9

6_ x _ =

$$\sqrt{1013.51}$$

10 13 .51

-

9 ↓

——

1 13

31

3 x 3 = 9

61 x 1 = 61

$$\sqrt{1013.51}$$

10 13 .51

-

9 ↓

1 13

-

61

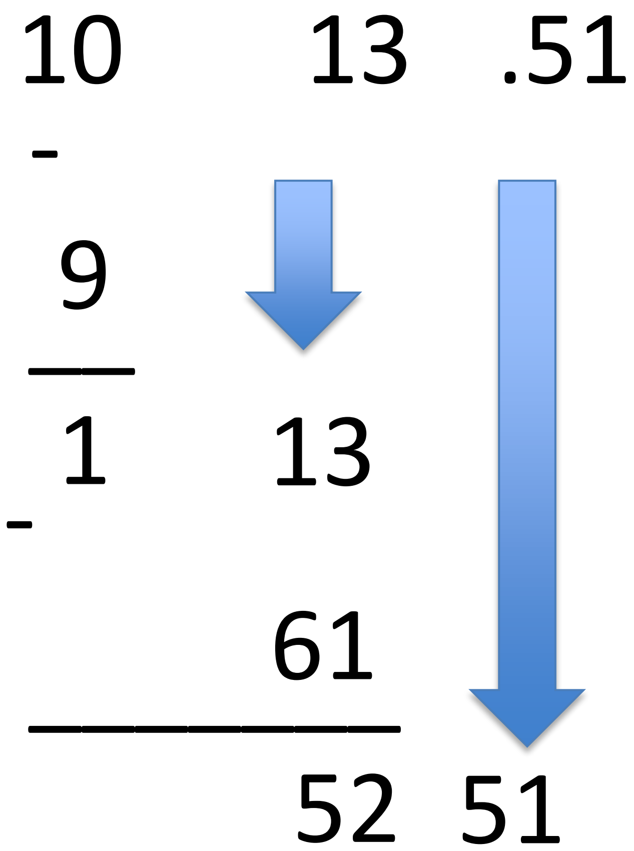
52

31

3 x 3 = 9

61 x 1 = 61

$$\sqrt{1013.51}$$



10 13 .51

-

9

1 13

-

61

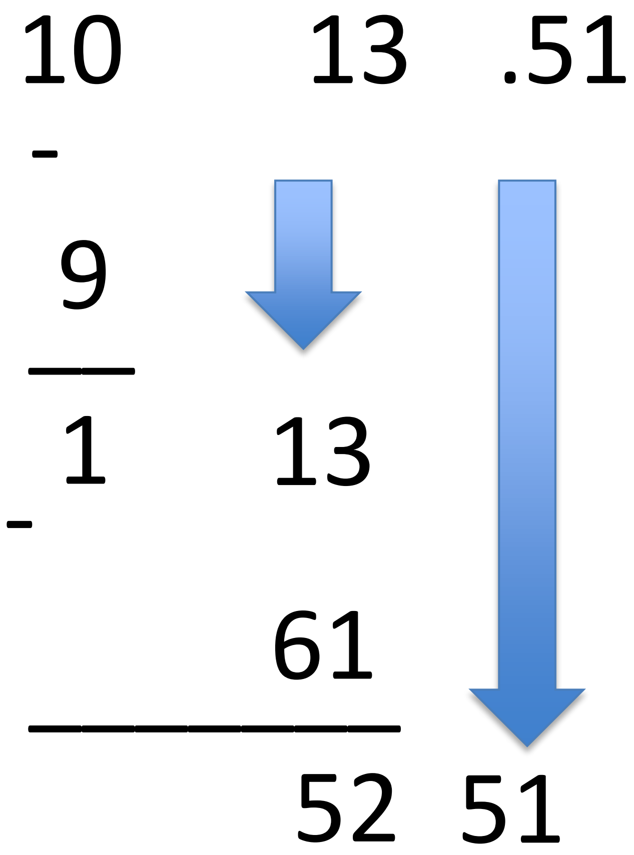
52 51

31.

3 x 3 = 9

61 x 1 = 61

$$\sqrt{1013.51}$$



10 13 .51

-

9

---

1 13

-

61

--------------

52 51

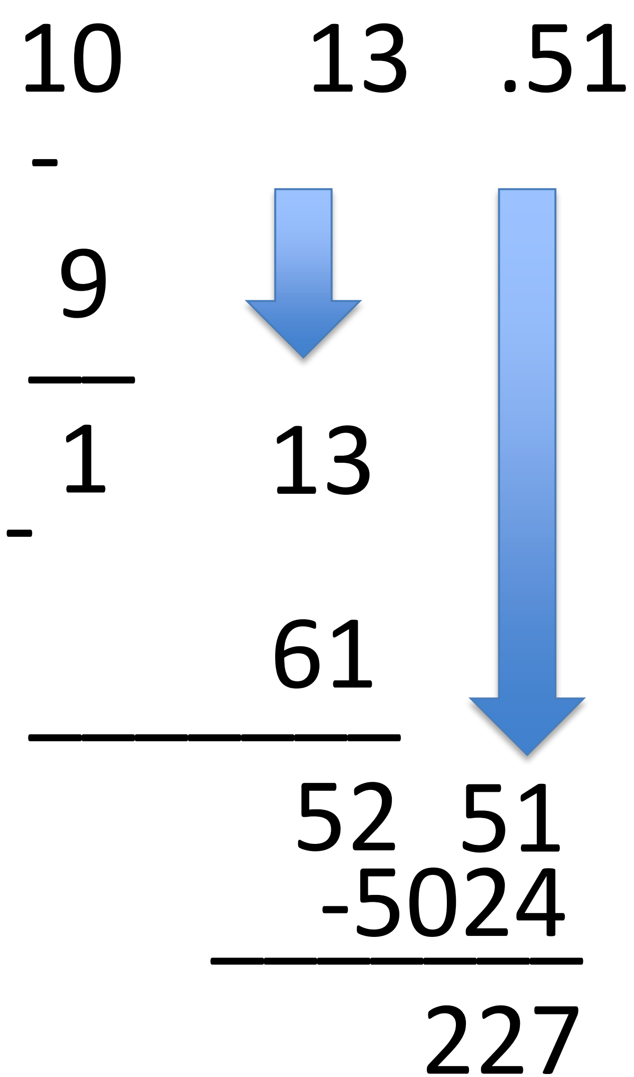
31.

3 x 3 = 9

61 x 1 = 61

62_ x _ =

$$\sqrt{1013.51}$$



10 13 .51

-

9

1 13

-

61

52 51

-5024

227

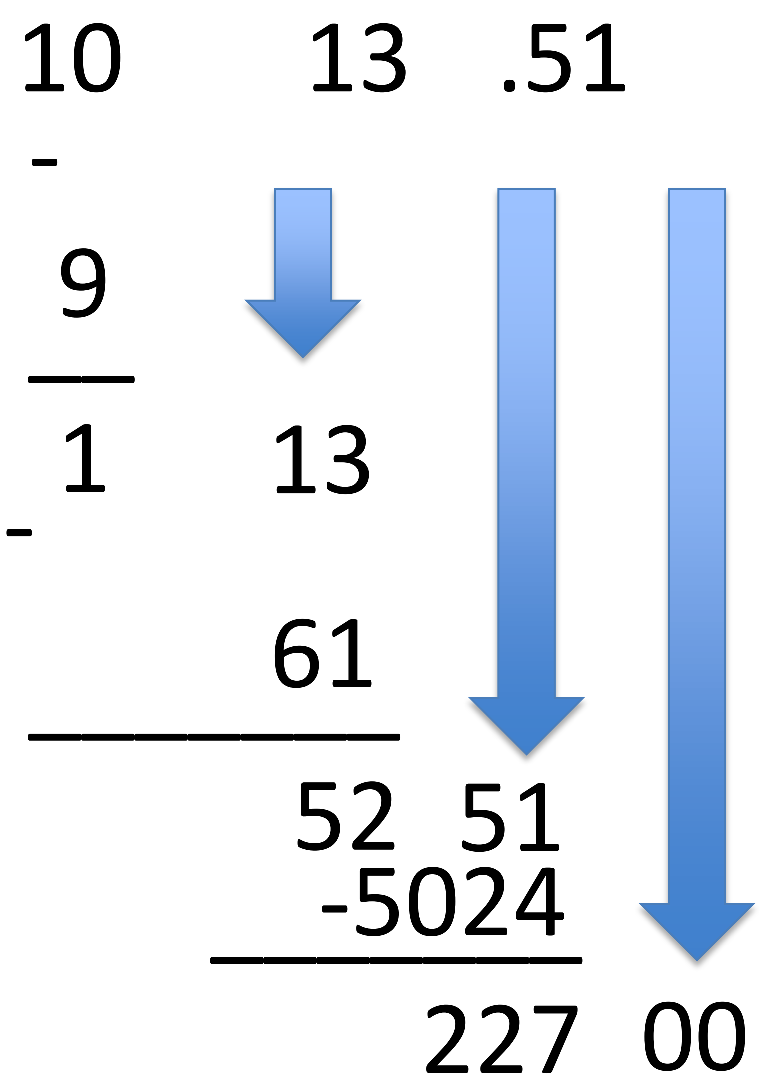
31.8

3 x 3 = 9

61 x 1 = 61

628 x 8 = 5024

$$\sqrt{1013.51}$$



10 13 .51

- 9

1 13

- 61

52 51

-5024

227 00

31.8

3 x 3 = 9

61 x 1 = 61

628 x 8 = 5024

$$\sqrt{1013.51}$$

22700

31.8

3 x 3 = 9

61 x 1 = 61

628 x 8 = 5024

636_ x _ =

$$\sqrt{1013.51}$$

$$\begin{array}{r} 22700 \\ -19089 \\ \hline 3611 \end{array}$$

31.83

3 x 3 = 9

61 x 1 = 61

628 x 8 = 5024

6363 x 3 = 19089

$$\sqrt{1013.51}$$

```
   22700
  -19089
__________
     3611 00
```

31.835

3 x 3 = 9

61 x 1 = 61

628 x 8 = 5024

6363 x 3 = 19089

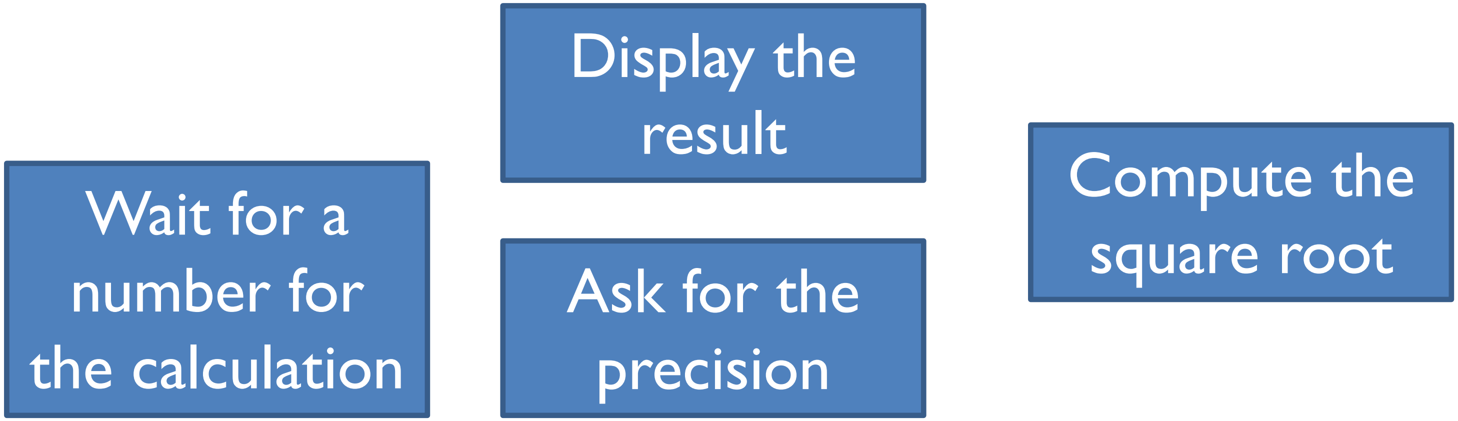
63665 x 5 = 318325

Example of a problem

Write a program that computes the square root of a real number entered by the user and displays the result. The accuracy of the result is specified by the user by the desired number of decimal places.

Example of a problem

Write a program that computes the square root of a real number entered by the user and displays the result. The accuracy of the result is specified by the user by the desired number of decimal places.



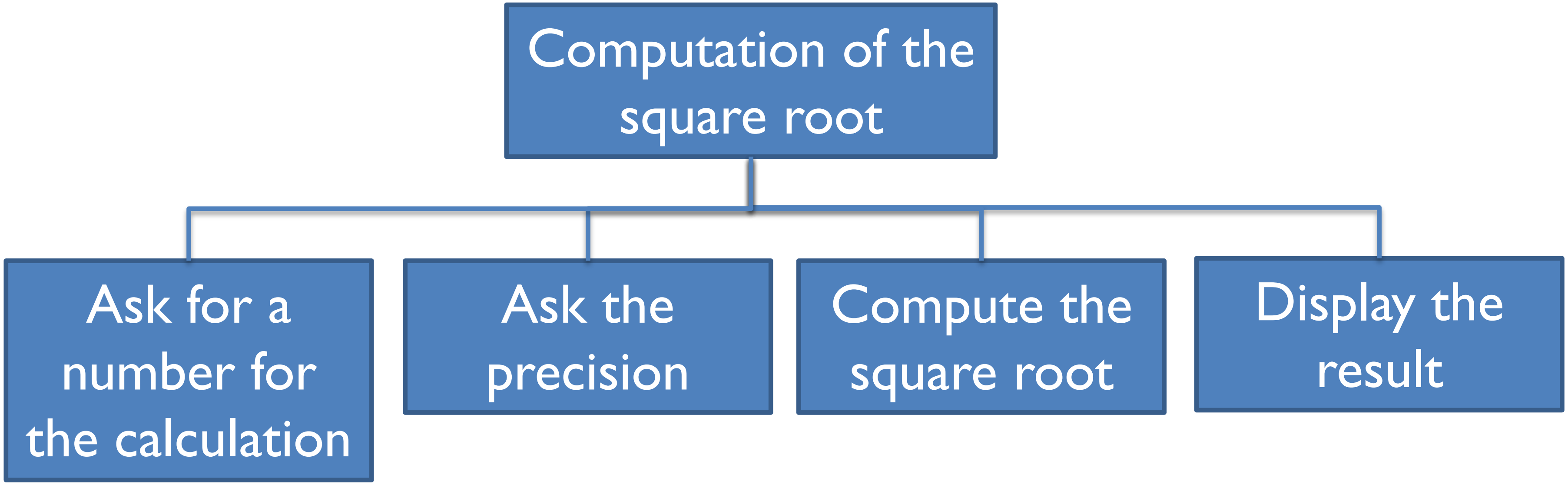
Wait for a number for the calculation

Display the result

Ask for the precision

Compute the square root

High-level solution



Computation of the square root

Ask for a number for the calculation

Ask the precision

Compute the square root

Display the result

Start with a small solution that works and develop on top of it

High-level algorithm

1. Ask the user for a number for the square root calculation
2. Ask for the number of decimal places required for the solution
3. Compute the square root
4. Display the result
5. Go back to step 1

Pseudo-code

```
while(1)
    ask_number
    ask_precision
    compute_square_root
    display_result
```

Solving a problem

- Define the problem
- Break down the problem
- Find an algorithm for solving the first part
- Write the solution in pseudo-code
- Write code
- Test
- Repeat until the problem is completely solved

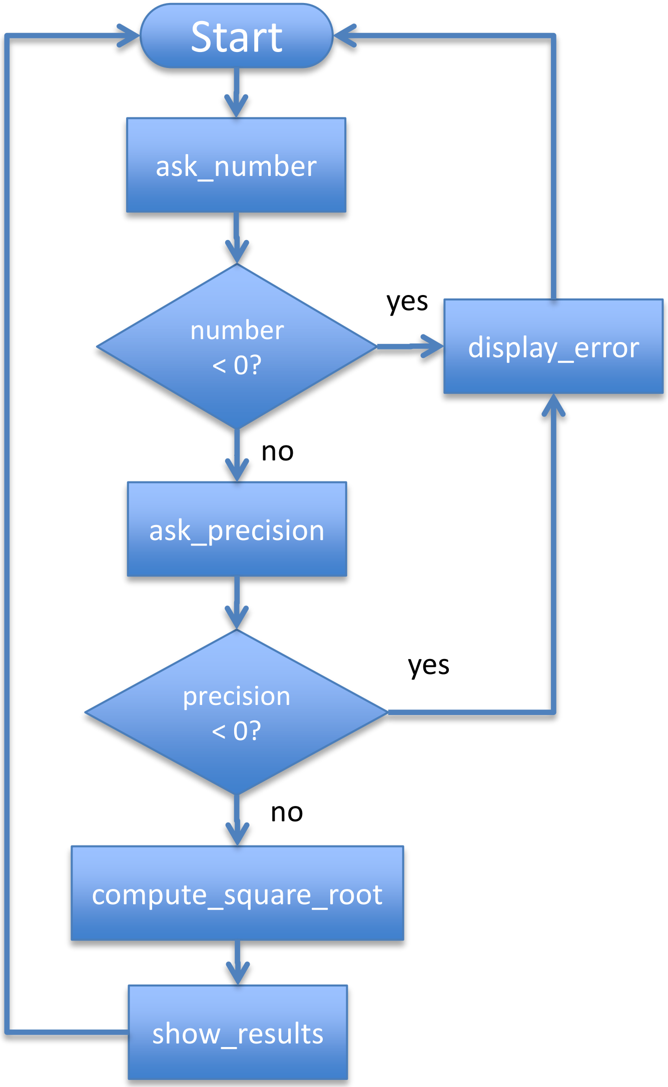
Tests

- Tests allow us to detect bugs and omissions
- Use limits
  - ex.: the square root of a negative number

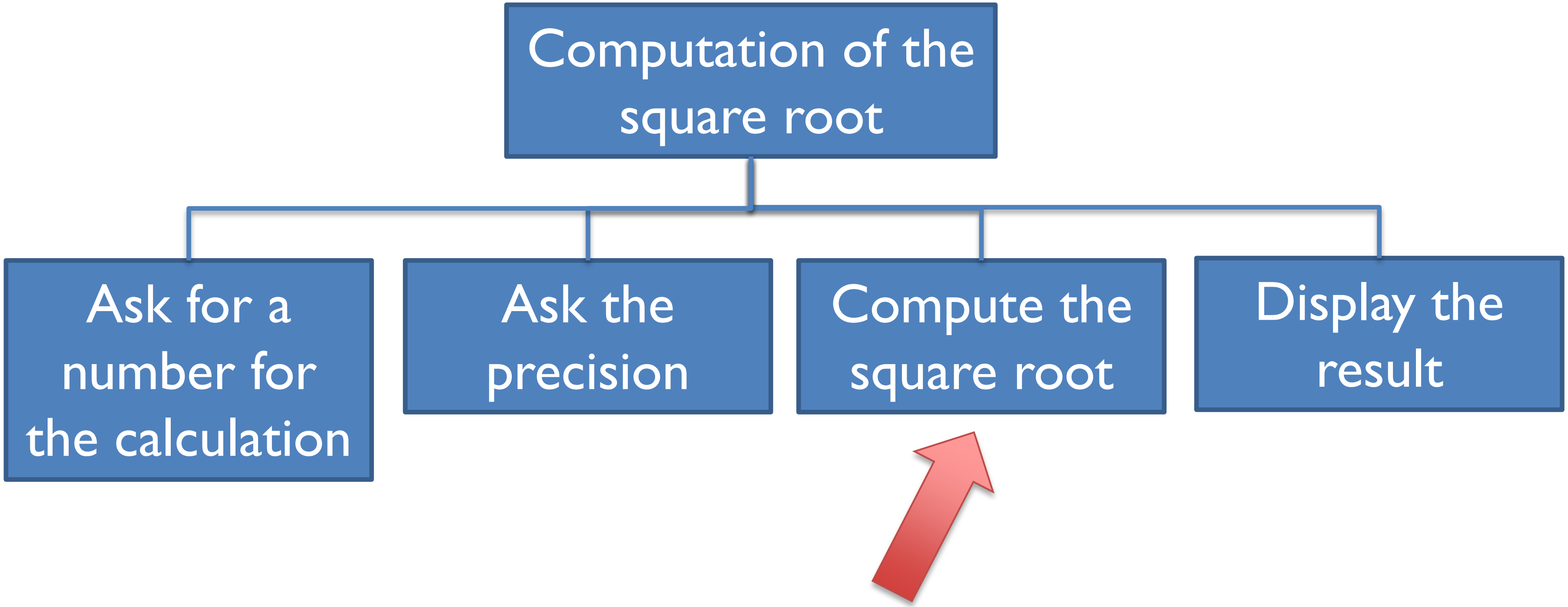
Revised pseudo-code

```
while(1)
  ask_number
  if(number < 0)
        display_error
        return to beginning
  ask_precision
  if(precision < 0)
        display_error
        return to beginning
  compute_square_root
  show_result
```

Flowcharts


Start
ask_number
number
< 0?
yes
display_error
no
ask_precision
precision
< 0?
yes
no
compute_square_root
show_results

Elaboration of the solution



Computation of the square root

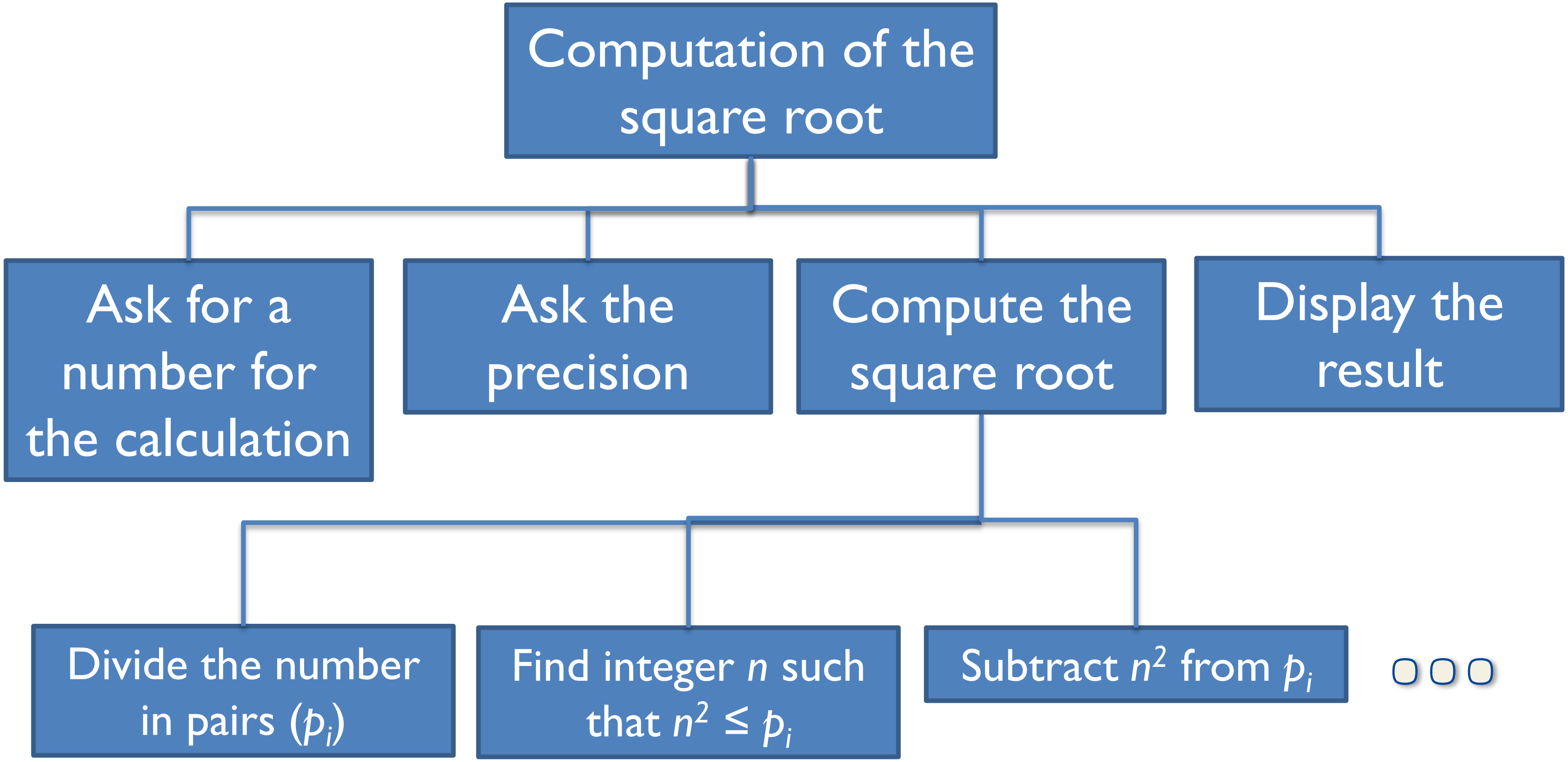
Ask for a number for the calculation

Ask the precision

Compute the square root

Display the result

Elaboration of the solution



Computation of the square root

Ask for a number for the calculation

Ask the precision

Compute the square root

Display the result

Divide the number in pairs (p_i)

Find integer n such that $n^2 \le p_i$

Subtract n^2 from p_i

▢ ▢ ▢

Problems

Write the algorithms to solve the following problems:

1. Converting temperatures in Fahrenheit to Celsius (0 C is 32 F and -40 C is -40 F)
2. Determining if a year is a leap year. Every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400. [2]

Your solution must include inputs and outputs.

Questions?

References

[1] Algorithm, Wikipedia, accessed 7 September 2017, https://en.wikipedia.org/wiki/Algorithm

[2] Leap Year, Wikipedia, accessed 29 August 2017, https://en.wikipedia.org/wiki/Leap_year