FT507 C++/C Programming

Algorithms

Carsten Svaneborg

FKF

March 10, 2026

Algorithm

- Algorithm is a step-by-step plan for solving problem.
- Multiply or add two numbers. $351 + 134$ or $273 * 854$
- Make coffee (boil water, ground coffee, ..)
- Knit sweater
- Build LEGO
- Borrowing book from library.
- Find lecture room
- Solve linear equation

Algorithm 1

Describe problem (often difficult!):

- What is the solution of the problem?
- What input is expected?
- What output is expected?

Algorithm 2

Analyze the problem:

- What are the types of input data?
- What coarse steps to execute to reach goal.
- How do we know, if we are done?
- Output: screen / file data?
- Identify and handle special cases.

Often its a good idea to simplify problem or solve several problems on a piece of paper, to get a feeling for the structure of the problem.

Algorithm 3

Don't start coding yet! First write down high-level solution:

- What are the big logical steps to solve problem
- What are the relations between these steps
- E.g. Pseudo-code
- E.g. Flow-chart
- :

Algorithm 4

Refine and Translate into code

- Turn high-level description to actual code (divide and conquer)
- Data are mapped to types / variables / arrays
- Steps = blocks of code or functions
- How does blocks of code / functions communicate.
- Output to screen / file

Algorithm 5

Often we need to iterate!

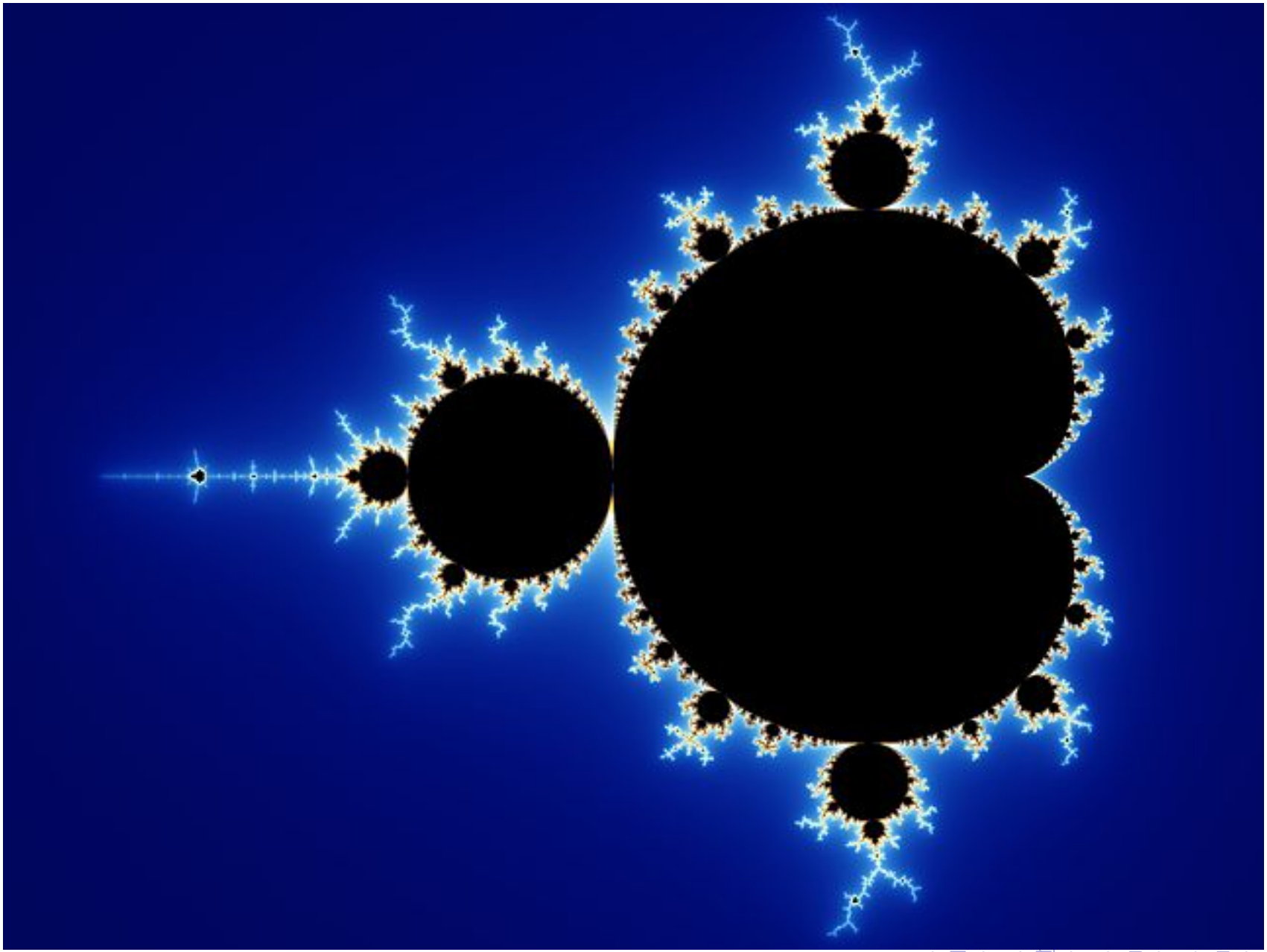
- Often we realise the problem was not clearly stated.
- Often we make implicit assumptions about the problem
- We didn't really understand the problem.
- Need to handle additional special cases.
- Make resilient code handling faulty user input.
- Optimize code to run faster / use less memory.
- Structure code for making algorithm more clear.
- Redefine function and variable names for improved semantics
- Generalize solution to work for a wider set of problems.

Algorithms

So far OK for imperative programming, BUT

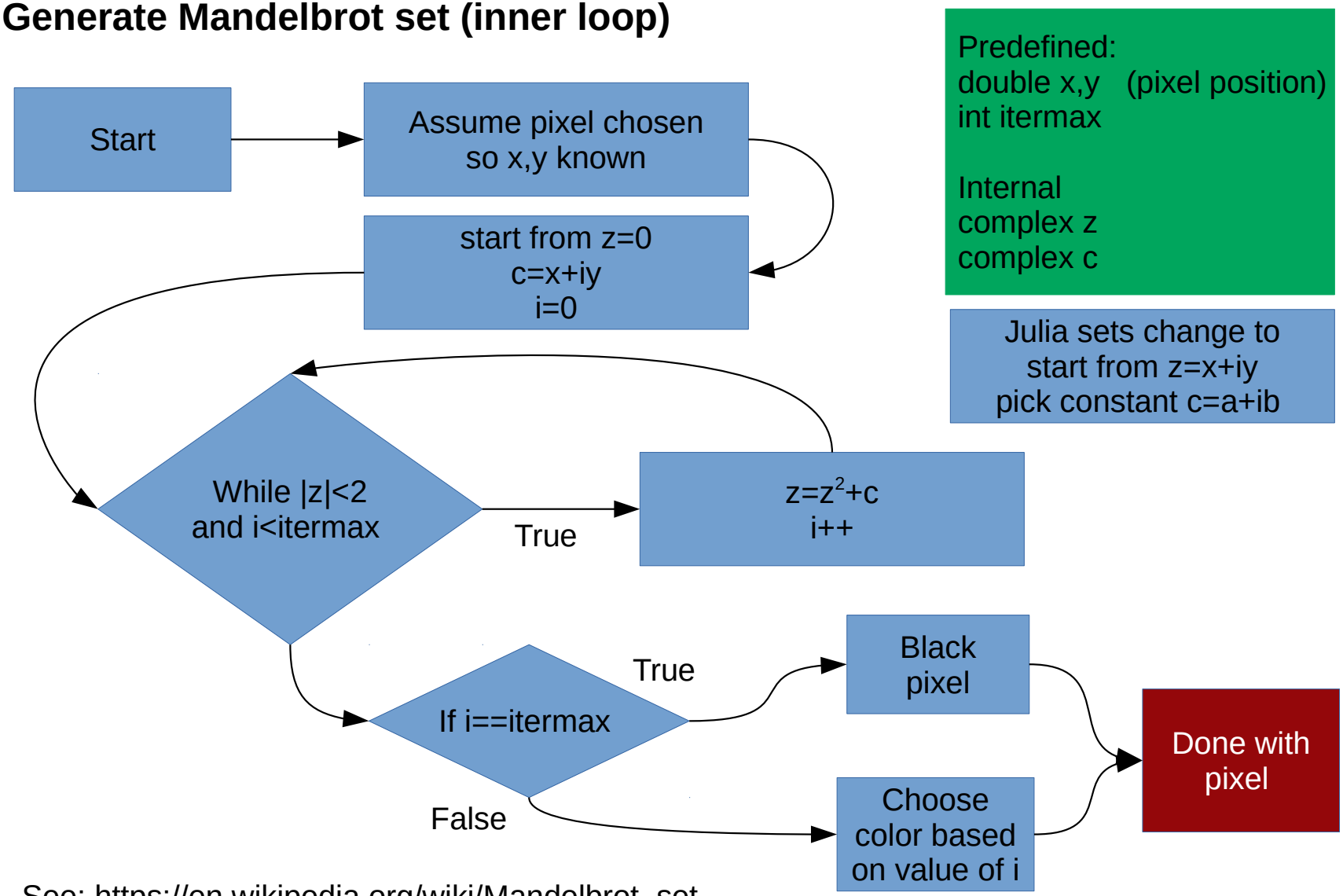
- What about stochastic/non deterministic algorithms?
- What about recursive algorithms?
- What about distributed algorithms?
- What about algorithms for heterogeneous hardware?
- What about quantum algorithms?

Example Mandelbrot



Flowchart

Generate Mandelbrot set (inner loop)



See: https://en.wikipedia.org/wiki/Mandelbrot_set

PseudoCode

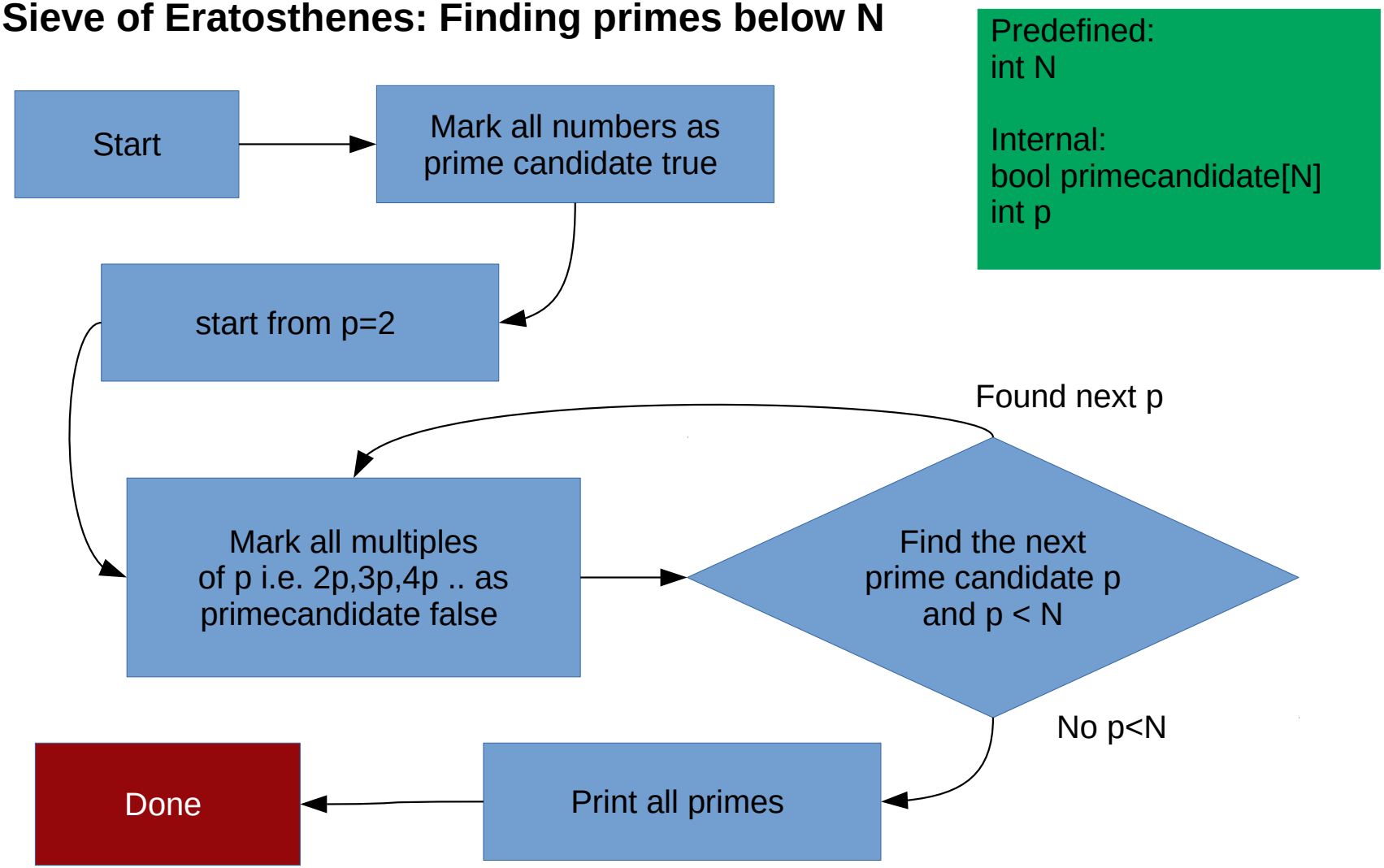
- set c=x+iy (x,y) pixel coordinate
- set z=0, counter=0
- while ($|z| < 2$ and counter<itermax)
  - $z = z^2 + c$
  - Increment counter
- if (counter==itermax)
  - print black pixel at x,y
- else
  - print pixel at x,y with color related to counter.

Example Eratosthenes

Primes: 1,2,3,5,7,11,13,..

Flowchart

Sieve of Eratosthenes: Finding primes below N



See video at https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

PseudoCode

- Read in N the maximal potential prime value.
- Define primecandidate array [0..N-1] with all true.
- Do
  - Starting at p=2
  - for j:
    - mark primecandidate[2p] false
    - mark primecandidate[3p] false
    - :
    - mark primecandidate[j*p] false
    - as long as $j * p < N$
  - Find next p, that is not marked or $p = N$.
- Repeat while $p < N$;
- Print primes that is i where primecandidate[i] is true.

Now

- Root finding: Bisection
- Root finding: Newton-Raphson
- Solving ODEs: Euler
- Integration: Box and trapetz