

Aalto 2023

Index	Contest	Submissions	Pre	0	CP	1	2a	2b	2c	3a	3b	4	5	9a	9c	IS	4	6a	6b	9a
MF	1	2	9a	SO	4	5	6	P	9a	X	0a	0b	9a	9b						

CP9c: fast solution with doubles ★★

Please note that you can still submit, but as the course is already closed, your submissions will not be graded.

To get started with the development, [download the code templates](#), unzip the file, edit `cp.cc`, and run `./grading test` or `./grading benchmark` to try it out – see the [instructions](#) for more details!

Upload your solution as a file here...

Please upload here the file **cp.cc** that contains your solution to task CP9c.

Choose File

No file chosen

... or copy-paste your code here

Submit

Your submissions

Your submissions to CP9c will appear here; you can simply [reload](#) this page to see the latest updates.

What you will need to do in this task

Please read the [general instructions for this exercise](#) first. Here are the additional instructions specific to this task:

This is a version of CP3a which has extended, **experimental** benchmarks that also try to measure cache traffic. This is a somewhat tricky endeavour, as the exact meaning of cache performance events can be CPU specific. We would therefore like to hear your feedback on how this works **on your local system**, as well as any cases where the reported numbers seem to differ from your expectations.

What I will try to do with your code

I will first run all kinds of tests to see that your code works correctly. You can try it out locally by running `./grading test`, but please note that your code has to compile and work correctly not only on your own computer but also on our machines.

If all is fine, I will run the benchmarks. You can try it out on your own computer by running `./grading benchmark`, but of course the precise running time on your own computer might be different from the performance on our grading hardware.

Benchmarks

Name	Operations	Parameters
benchmarks/1	1,004,000,000	nx = 1000, ny = 1000
the input contains 1000 × 1000 pixels, and the output should contain 1000 × 1000 pixels		
benchmarks/2a	16,016,000,000	nx = 1000, ny = 4000
the input contains 4000 × 1000 pixels, and the output should contain 4000 × 4000 pixels		
benchmarks/2b	16,016,000,000	nx = 1000, ny = 4000
the input contains 4000 × 1000 pixels, and the output should contain 4000 × 4000 pixels		
benchmarks/2c	15,991,989,003	nx = 999, ny = 3999
the input contains 3999 × 999 pixels, and the output should contain 3999 × 3999 pixels		
benchmarks/2d	16,040,029,005	nx = 1001, ny = 4001
the input contains 4001 × 1001 pixels, and the output should contain 4001 × 4001 pixels		
benchmarks/3	216,144,000,000	nx = 6000, ny = 6000
the input contains 6000 × 6000 pixels, and the output should contain 6000 × 6000 pixels		
benchmarks/4	729,324,000,000	nx = 9000, ny = 9000
the input contains 9000 × 9000 pixels, and the output should contain 9000 × 9000 pixels		

Here “operations” is our rough estimate of how many useful arithmetic operations you will **at least** need to perform in this benchmark, but of course this will depend on exactly what kind of an algorithm you are using.

Grading

In this task your submission will be graded using **benchmarks/4**: the input contains 9000 × 9000 pixels, and the output should contain 9000 × 9000 pixels.

There are no points available for submissions to this task, but you can freely use this task for experimentation.