

**Wykonał:** Radosław Smoter

**Grupa:** 14

**Nr:** 27

**Numer zadania:** 1

**Przykład:** 62

**Prowadzący:** Prof. dr hab. inż.  
Volodymyr Samotyy

# **Politechnika Krakowska**

**Wydział Inżynierii Elektrycznej i Komputerowej**

**Sprawozdanie: Wstęp do Programowania**

## Spis treści

Polecenie.....	1
Kod programu.....	2
Wyniki.....	4
Opis Programu.....	7
Wnioski.....	8

## Polecenie

Obliczyć wartości funkcji jednoargumentowej. Wyniki obliczeń zapisać do pliku tekstowego. Narysować wykres  $y(x)$  .

Funkcja:  $y = \frac{5}{3 + \lg^2(x)}$  ,

Dziedzina:  $x \in [1.2; 3.1]$  .

$\lg(x)$  przyjmuję za  $\log_{10}(x)$  .

## Kod programu

```
/*
 * Author: Radosław Smoter
 * Name: Wstep do programowania - projekt 1
 * OS:
 * Ubuntu 20.04.3 LTS
 * To compile:
 * gcc -Wall smoter_r_wp_1.c -o smoter_r_wp_1 -lm
 * To invoke:
 * ./smoter_r_wp_1
 */

// Add libraries
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include "time.h"
#include "string.h"

// Do math; print to file
// xp - domain start, xk - domain end
void getRecords(double, double);

// Carry out equations
int main(void)
{
    const double xp = 1.2; // xp - domain boundary low
    const double xk = 3.1; // xk - domain boundary high

    getRecords(xp, xk);

    return 0;
}

void getRecords(double xp, double xk)
{
    const double a = 5.; // Equation const
    const double b = 3.; // Equation const

    /* Create unique name for new file (time in sec + ext) */

    // System time
    time_t timer;
    time(&timer);

    // File name string buffer
    char filename[32];
    // Convert time (from timer) to string and save it to filename
    sprintf(filename, "%li", timer);
    // Add ext
    strcat(filename, ".dat");
}
```

```
// Domain step
double dx = (xk - xp) / 100;

// File pointer
FILE *file = fopen(filename, "w");

// Ensure file exists
if (file != NULL)
{
    for (double x = xp; x < xk; x += dx) // Step
    {
        // Step result
        double y = a / (b + pow(log(x) / log(10), 2));

        fprintf(file, "%f\t%f\n", x, y);
    }
}

// Close file
fclose(file);

return;
}
```

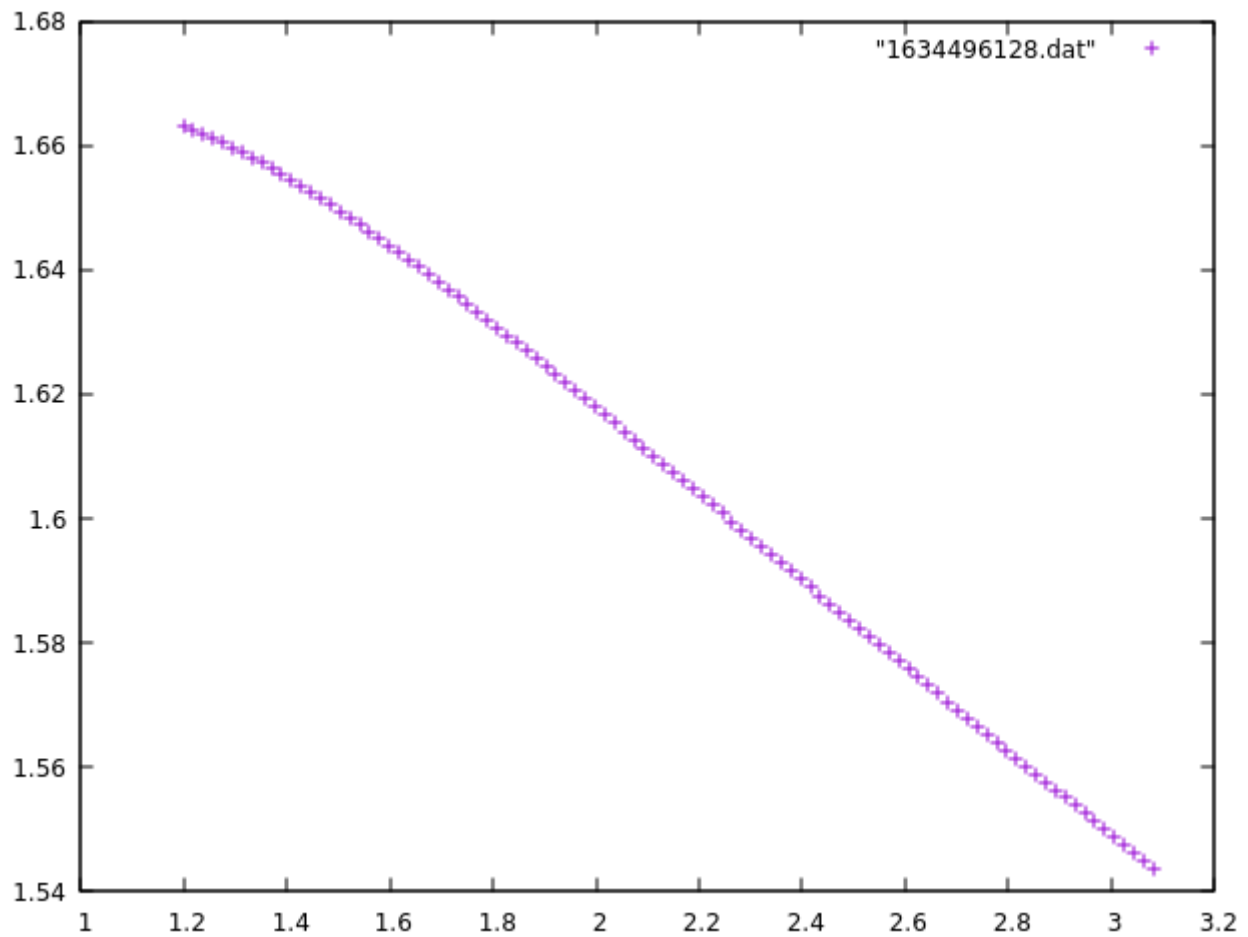
## Wyniki

Nr	x	y
1	1.200000	1.663191
2	1.219000	1.662568
3	1.238000	1.661904
4	1.257000	1.661203
5	1.276000	1.660465
6	1.295000	1.659693
7	1.314000	1.658889
8	1.333000	1.658054
9	1.352000	1.657190
10	1.371000	1.656299
11	1.390000	1.655381
12	1.409000	1.654438
13	1.428000	1.653472
14	1.447000	1.652483
15	1.466000	1.651473
16	1.485000	1.650443
17	1.504000	1.649394
18	1.523000	1.648327
19	1.542000	1.647243
20	1.561000	1.646142
21	1.580000	1.645027
22	1.599000	1.643896
23	1.618000	1.642753
24	1.637000	1.641596
25	1.656000	1.640427
26	1.675000	1.639246
27	1.694000	1.638055
28	1.713000	1.636853
29	1.732000	1.635641
30	1.751000	1.634421
31	1.770000	1.633191
32	1.789000	1.631954
33	1.808000	1.630709
34	1.827000	1.629457
35	1.846000	1.628198
36	1.865000	1.626933
37	1.884000	1.625662
38	1.903000	1.624386
39	1.922000	1.623105
40	1.941000	1.621819
41	1.960000	1.620528
42	1.979000	1.619234
43	1.998000	1.617936
44	2.017000	1.616634
45	2.036000	1.615329
46	2.055000	1.614022

47	2.074000	1.612712
48	2.093000	1.611399
49	2.112000	1.610085
50	2.131000	1.608769
51	2.150000	1.607451
52	2.169000	1.606131
53	2.188000	1.604811
54	2.207000	1.603489
55	2.226000	1.602167
56	2.245000	1.600844
57	2.264000	1.599520
58	2.283000	1.598196
59	2.302000	1.596873
60	2.321000	1.595549
61	2.340000	1.594225
62	2.359000	1.592901
63	2.378000	1.591578
64	2.397000	1.590256
65	2.416000	1.588934
66	2.435000	1.587613
67	2.454000	1.586293
68	2.473000	1.584974
69	2.492000	1.583656
70	2.511000	1.582340
71	2.530000	1.581025
72	2.549000	1.579711
73	2.568000	1.578398
74	2.587000	1.577088
75	2.606000	1.575779
76	2.625000	1.574471
77	2.644000	1.573166
78	2.663000	1.571862
79	2.682000	1.570560
80	2.701000	1.569261
81	2.720000	1.567963
82	2.739000	1.566668
83	2.758000	1.565375
84	2.777000	1.564084
85	2.796000	1.562795
86	2.815000	1.561509
87	2.834000	1.560225
88	2.853000	1.558944
89	2.872000	1.557665
90	2.891000	1.556389
91	2.910000	1.555115
92	2.929000	1.553844
93	2.948000	1.552575
94	2.967000	1.551309
95	2.986000	1.550046
96	3.005000	1.548786

97	3.024000	1.547528
98	3.043000	1.546273
99	3.062000	1.545021
100	3.081000	1.543772

Wykres 1: Wykres powyższych punktów; wykonany programem gnuplot, poleceniem:  
`plot "1634233768.dat"`





## Opis Programu

Program składa się z funkcji `getRecords()` wywoływanej wewnątrz funkcji `main()`. Przyjmuje dwa parametry typu `double` odpowiadające początkowi oraz końcowi określonej dziedziny wykonywanej wewnątrz funkcji.

Funkcja `getRecords()` zawiera dwie stałe typu `double` będące częścią wykonywanej funkcji matematycznej. Dalej określa nazwę tworzonego przez siebie pliku poprzez połączenie czasu systemowego z predefiniowanym rozszerzeniem typu `".dat"`.

Funkcja `getRecords()` nie zwraca żadnej wartości. Jej zadaniem jest wykonanie polecenia matematycznego:  $\frac{5}{3+\lg^2(x)}$ , na zadanej dziedzinie oraz zapis do pliku o wygenerowanej nazwie. Funkcja operuje na kroku dziedziny ustalonym wyrażeniem dzielącym dziedzinę na części:  $dx = \frac{xk - xp}{100}$ , gdzie:

$dx$  - to krok dziedziny,

$xp$  - wartość początkowa dziedziny,

$xk$  - wartość końcowa dziedziny.

## Wnioski

Wykres wykonany powyższym programem przypomina wykres uzyskany za pomocą programu WolframAlpha dla zadanej funkcji. Rozbieżności wynikają z innego umieszczenia skal na osiach.



$y = 5 / (3 + \lg(x)^2)$  in  $[1.2; 3.1]$

NATURAL LANGUAGE

MATH INPUT

EXTENDED KEYBOARD

EXAMPLES

UPLOAD

RANDOM

Assuming "lg" is the base 10 logarithm | Use [the natural logarithm](#) or [the base 2 logarithm](#) instead

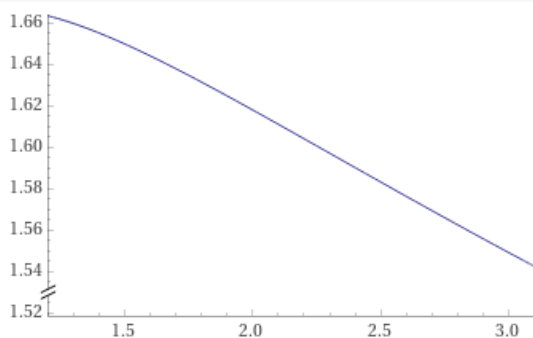
Input interpretation

plot

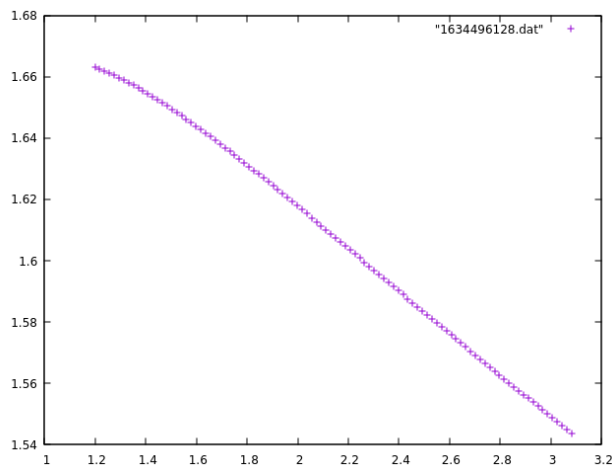
$$y = \frac{5}{3 + \log_{10}^2(x)}$$

$x = 1.2$  to  $3.1$

Plot



Wykres 2: Wykres uzyskany programem WolframAlpha.



Wykres 3: Wykres uzyskany przeze mnie.