Wykonał: Radosław Smoter

Grupa: 14

Nr: 27

Numer zadania: 2

Przykład: 62

Prowadzący: Prof. dr hab. inż. Volodymyr Samotyy

Politechnika Krakowska

Wydział Inżynierii Elektrycznej i Komputerowej

Sprawozdanie: Wstęp do Programowania

Kraków 2021

Spis treści

Polecenie

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

Funkcje:

$y=1.7-\ln(x^2)+3.2lg(x^3), x\in(-\infty, 0.6)$,

$y=\sqrt{3x+\ln(x-1.2)}, x\in[0.6, 0.7)$,

$y=2.7+4x^2+2.3x^3, x\in[0.7, +\infty)$

Dziedzina: $x\in[0.5, 0.9]$.

$lg(x)$ przyjmuję za $\log_{10}(x)$,

$\ln(x)$ przyjmuję za $\log_e(x)$.

Kod programu

```
/*

  * Author: Radosław Smoter
  * Name: smoter_r_wp_2.c
  * OS:
  * Ubuntu 21.10
  * To compile:
  * gcc -Wall smoter_r_wp_2.c -o smoter_r_wp_2 -lm
  * To invoke:
  * ./smoter_r_wp_2

*/

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

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

// Count first expression
// Value
double countFirst(double);
// Count second expression
// Value
double countSecond(double);
// Count third expression
// Value
double countThird(double);

// Carry out equations
int main(void)
{
  // xp, xk - domain boundaries
  const double xp = .5; // xk - domain boundary low
  const double xk = .9; // xk - domain boundary high
  const double dx = 0.002; // Domain step

  getRecords(xp, xk, dx);

  return 0;
}

void getRecords(double xp, double xk, double dx)
{
  /* Create unique name for new file (time in sec + ext) */

  time_t timer; // System time
  char filename[32]; // File name string buffer
```

```
  // Get sys time
  time(&timer);
  // Convert time (from timer) to string and save it to filename
  sprintf(filename, "%li", timer);
  // Add ext
  strcat(filename, ".dat");

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

  // Ensure file exists
  if (file != NULL)
  {
    for (double x = xp; x < xk; x += dx)
    {
      double y; // Init step result var

      if (x < .6)
        y = countFirst(x);
      else if (x >= .6 && x < .7)
        y = countSecond(x);
      else
        y = countThird(x);

      // Save only if number encountered
      if (y != NAN)
        fprintf(file, "%f\t%f\n", x, y);
    }
  }

  // Close file
  fclose(file);

  return;
}

/*
  * If dangerous operations are encountered, verify wether
  their values are numbers; otherwise return NAN.
*/

double countFirst(double x)
{
  double log1 = log(pow(x, 2));
  double log2 = log(pow(x, 3));
  // Check if logarithms are numbers
  if (isnan(log1) || isnan(log2)) return NAN;
  return 1.7 - log1 + 3.2 * log2 / log(10);
}

double countSecond(double x)
{
  double log1 = log(x - 1.2);
  // Check if logarithm is a number
  if (isnan(log1)) return NAN;
  double sqrt1 = pow(3 * x + log1, 1/2);
```

```
  // Check if root is a number
  if (isnan(sqrt1)) return NAN;
  return sqrt1;
}

double countThird(double x)
{
  return 2.7 + 4 * pow(x, 2) + 2.3 * pow(x, 3);
}
```

Wyniki

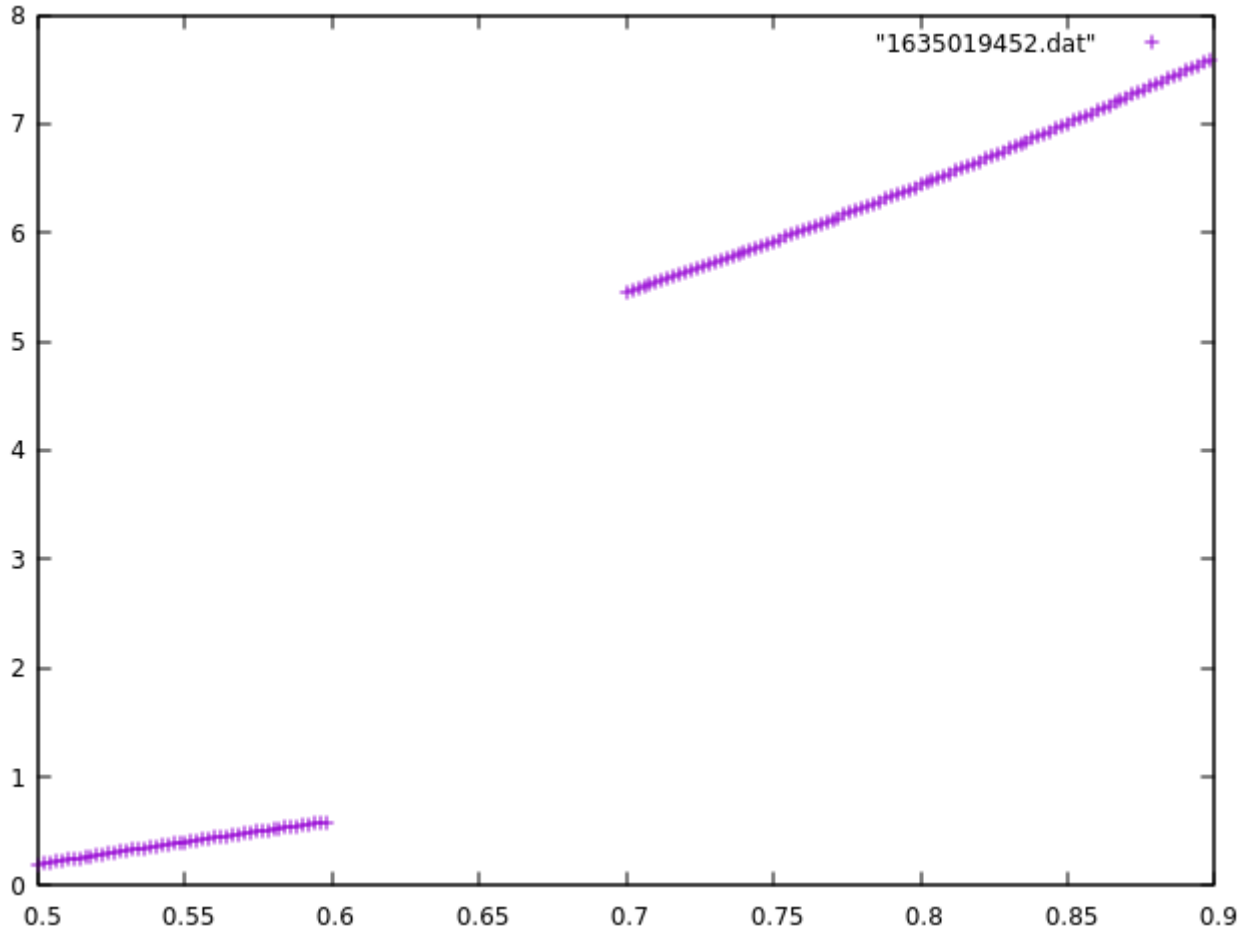
Nr	x	y
1	0.500000	0.196406
2	0.502000	0.205066
3	0.504000	0.213691
4	0.506000	0.222282
5	0.508000	0.230839
6	0.510000	0.239363
7	0.512000	0.247853
8	0.514000	0.256310
9	0.516000	0.264734
10	0.518000	0.273126
11	0.520000	0.281485
12	0.522000	0.289812
13	0.524000	0.298108
14	0.526000	0.306371
15	0.528000	0.314604
16	0.530000	0.322805
17	0.532000	0.330975
18	0.534000	0.339115
19	0.536000	0.347224
20	0.538000	0.355303
21	0.540000	0.363352
22	0.542000	0.371372
23	0.544000	0.379362
24	0.546000	0.387322
25	0.548000	0.395253
26	0.550000	0.403156
27	0.552000	0.411030
28	0.554000	0.418875
29	0.556000	0.426692
30	0.558000	0.434481
31	0.560000	0.442242
32	0.562000	0.449975
33	0.564000	0.457681
34	0.566000	0.465360
35	0.568000	0.473012
36	0.570000	0.480636
37	0.572000	0.488234
38	0.574000	0.495806
39	0.576000	0.503351
40	0.578000	0.510870
41	0.580000	0.518363
42	0.582000	0.525830
43	0.584000	0.533272
44	0.586000	0.540688
45	0.588000	0.548079
46	0.590000	0.555445

47	0.592000	0.562786
48	0.594000	0.570102
49	0.596000	0.577393
50	0.598000	0.584660
51	0.600000	nan
52	0.602000	nan
53	0.604000	nan
54	0.606000	nan
55	0.608000	nan
56	0.610000	nan
57	0.612000	nan
58	0.614000	nan
59	0.616000	nan
60	0.618000	nan
61	0.620000	nan
62	0.622000	nan
63	0.624000	nan
64	0.626000	nan
65	0.628000	nan
66	0.630000	nan
67	0.632000	nan
68	0.634000	nan
69	0.636000	nan
70	0.638000	nan
71	0.640000	nan
72	0.642000	nan
73	0.644000	nan
74	0.646000	nan
75	0.648000	nan
76	0.650000	nan
77	0.652000	nan
78	0.654000	nan
79	0.656000	nan
80	0.658000	nan
81	0.660000	nan
82	0.662000	nan
83	0.664000	nan
84	0.666000	nan
85	0.668000	nan
86	0.670000	nan
87	0.672000	nan
88	0.674000	nan
89	0.676000	nan
90	0.678000	nan
91	0.680000	nan
92	0.682000	nan
93	0.684000	nan
94	0.686000	nan
95	0.688000	nan
96	0.690000	nan

97	0.692000	nan
98	0.694000	nan
99	0.696000	nan
100	0.698000	nan
101	0.700000	5.448900
102	0.702000	5.466897
103	0.704000	5.484965
104	0.706000	5.503104
105	0.708000	5.521314
106	0.710000	5.539595
107	0.712000	5.557947
108	0.714000	5.576371
109	0.716000	5.594866
110	0.718000	5.613432
111	0.720000	5.632070
112	0.722000	5.650780
113	0.724000	5.669562
114	0.726000	5.688416
115	0.728000	5.707341
116	0.730000	5.726339
117	0.732000	5.745409
118	0.734000	5.764552
119	0.736000	5.783767
120	0.738000	5.803055
121	0.740000	5.822415
122	0.742000	5.841849
123	0.744000	5.861355
124	0.746000	5.880934
125	0.748000	5.900587
126	0.750000	5.920313
127	0.752000	5.940112
128	0.754000	5.959984
129	0.756000	5.979931
130	0.758000	5.999951
131	0.760000	6.020045
132	0.762000	6.040213
133	0.764000	6.060455
134	0.766000	6.080771
135	0.768000	6.101161
136	0.770000	6.121626
137	0.772000	6.142165
138	0.774000	6.162779
139	0.776000	6.183468
140	0.778000	6.204231
141	0.780000	6.225070
142	0.782000	6.245983
143	0.784000	6.266972
144	0.786000	6.288036
145	0.788000	6.309175
146	0.790000	6.330390

147	0.792000	6.351680
148	0.794000	6.373046
149	0.796000	6.394488
150	0.798000	6.416006
151	0.800000	6.437600
152	0.802000	6.459270
153	0.804000	6.481016
154	0.806000	6.502839
155	0.808000	6.524738
156	0.810000	6.546714
157	0.812000	6.568767
158	0.814000	6.590896
159	0.816000	6.613103
160	0.818000	6.635386
161	0.820000	6.657746
162	0.822000	6.680184
163	0.824000	6.702699
164	0.826000	6.725292
165	0.828000	6.747962
166	0.830000	6.770710
167	0.832000	6.793536
168	0.834000	6.816440
169	0.836000	6.839421
170	0.838000	6.862481
171	0.840000	6.885619
172	0.842000	6.908836
173	0.844000	6.932131
174	0.846000	6.955504
175	0.848000	6.978956
176	0.850000	7.002488
177	0.852000	7.026097
178	0.854000	7.049786
179	0.856000	7.073555
180	0.858000	7.097402
181	0.860000	7.121329
182	0.862000	7.145335
183	0.864000	7.169421
184	0.866000	7.193586
185	0.868000	7.217832
186	0.870000	7.242157
187	0.872000	7.266562
188	0.874000	7.291048
189	0.876000	7.315613
190	0.878000	7.340259
191	0.880000	7.364986
192	0.882000	7.389793
193	0.884000	7.414680
194	0.886000	7.439649
195	0.888000	7.464698
196	0.890000	7.489829

197	0.892000	7.515040
198	0.894000	7.540333
199	0.896000	7.565707
200	0.898000	7.591163



Wykres 1: Wykres powyższych punktów. Uzyskany za pomocą programu gnuplot, poleceniem plot "1634499036.dat"

Opis Programu

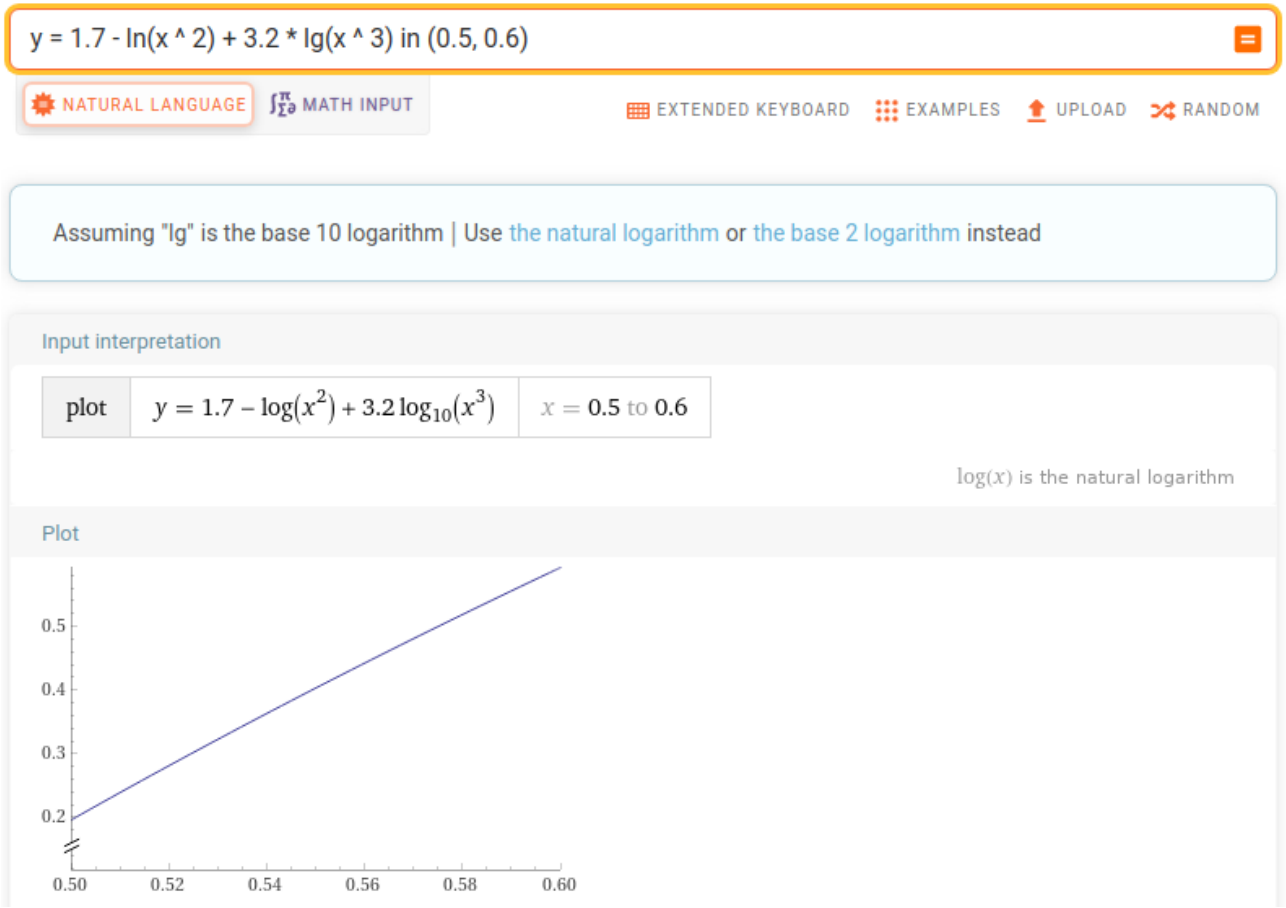
Program składa się z funkcji getRecords() wywoływanej wewnątrz funkcji main(). Przyjmuje trzy parametry typu double odpowiadające początkowi oraz końcowi określonej dziedziny wykonywanej wewnątrz funkcji. Trzeci argument to krok o jaki powinny posuwać się kolejne rekordy.

Funkcja getRecords() określa nazwę tworzonego przez siebie pliku poprzez połączenie czasu systemowego z rozszerzeniem typu “.dat”. Funkcja nie zwraca żadnej wartości. Jej zadaniem jest wykonanie polecenia matematycznego na zadanej dziedzinie oraz zapis do pliku o wygenerowanej nazwie. Na zadanej dziedzinie wykonuje się pętla for, w której instrukcja warunkowa określa jaka funkcja matematyczna powinna zostać wykonana dla konkretnej wartości argumentu.

Funkcje matematyczne są przechowywane w trzech bliźniaczych funkcjach, tj. countFirst(), countSecond(), countThird(), których jedynym zadaniem jest zwrot uzyskanych wartości przetworzonych przez zawarte w nich algorytmy. Wszystkie te funkcje przyjmują po jednym parametrze typu double, będącym argumentem dla zawartych w nich funkcji matematycznych. W funkcjach countFirst() I countSecond() sprawdzane są wartości pośrednie wykonywanych operacji matematycznych, pod kątem należności danego argumentu do dziedziny podanej operacji (jeśli argument nie należy do dziedziny, to zwrócona zostaje stała NAN I dalsze operacje nie zostają wykonane).

Wnioski

Dla poszczególnych funkcji składowych, w określonych dla nich dziedzinach , funkcje te przypominają wykresy utworzone dla nich za pomocą programu WolframAlpha.



WolframAlpha computational intelligence.

y = 2.7 + 4 * x ^ 2 + 2.3 * x ^ 3 in (0.7; 0.9]

NATURAL LANGUAGE | MATH INPUT | EXTENDED KEYBOARD | EXAMPLES | UPLOAD | RANDOM

Input interpretation

plot	$y = 2.7 + 4\,x^2 + 2.3\,x^3$	$x = 0.7$ to 0.9

Plot