TALLINNA TEHNIKAULIKOOL
Faculty of Information Technology
Institute of computer Systems

Mikuláš Šveda 244729

FUNCTION TABULATION

1. Homework in subject IAX0583

Lecturer: Vladimir Viies

Tallinn 2024

Copyright declaration

I confirm that I have prepared this homework independently and it is not from someone else previously submitted to the defense. All works of other authors used in the preparation of the work, important points of view, data from literary sources and elsewhere are cited in the work.

Author: Mikuláš Šveda

13.9.2024

Contents

List of Figures

List of Tables

1 Task definition

Create an algorithm (as a diagram or pseudocode) for solving the task and C code that would correspond to the algorithm description. All of the input data should be inserted from the keyboard and it can be any real number. The results should be shown on the terminal screen as tables, which has the columns for argument x and function y=f(x) values.

The value of the function should be displayed only if it exists i.e. it is final and real number. If the function value is not defined (it is infinite) or it is complex number, program should display 'not avilable' or 'complex number' accordingly.

1.1 Task variant

The method and function were based on my student number, which is 244729.

Method 5: User inputs a starting value A, step H and the lower limit of function value YM. The following conditions have to be true:
H $>$ 0.

The function value y will be calculated in the following points:
A
A + H
A + 2H
A + 3H
while the condition y $>$ YM holds true, however not more than 15 times.

Function

$$y = x^2 + \frac{x}{2} - \sqrt{\frac{1}{2x}} \tag{1}$$

2 Function y = f(x) analysis

In order to compute the parameters entered by the user, we consider the function determination region. The function is defined only when the expression under the square root is larger than 0.

The domain of determination is $x \in (0, \infty)$.

2.1 Graph of the function y = f(x)

To better understand the function, we will use the tools available on the web for the function to display the graph. From the graph, it can be seen that the region of determination we found is true.

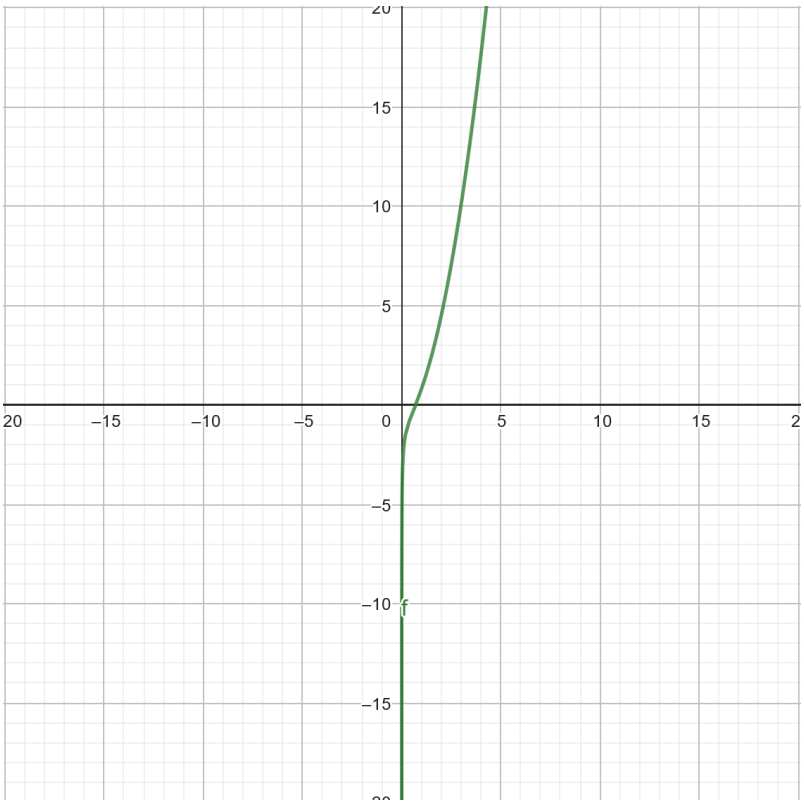

Figure 1: Graph of the function [2]

3 Solution

The goal of the task is to solve function (1) for several argument values. The argument follows a simple rule

$$x_i = A + (i - 1) * H, \tag{2}$$

where i goes from 1 to 15. Unless function value is below lower limit YM.

The user is asked to enter values for A, H and YM individually. With each input the program checks whether valid input has been given. This means input is a number and not a symbol, and number value follows meets all conditions ($H > 0$). All variables are doubles, as there is no upper limit for inputs.

Thanks to the square root at the end of the function, only positive arguments ($x_i > 0$) can lead to values in $\Re$. For all negative arguments ($x_i < 0$) the value of the function will be from complex plane $\Im$. The most problematic argument value is 0, where such function is undefined even in $\Im$. The program still calculates the result as $-\infty$.

As complex solutions are in my field of studies (Mechanical Engineering) often ignored as they do not present any actual solution for out problems, I have decided not implement support for imaginary numbers. Involving them would also create an issue for comparing function value y to lower limit YM. Where we would have to decide if we compare only the $\Re$ part of the solution, or we use it's absolute value.

Nevertheless I have also decided not to restrict user from entering input resulting in complex function solutions. The answer will simply be "not available". This also applies for the argument equal to 0.

The output of the program is a table of values in a form of

$$x_i = "argument" \; y_i = "solution"$$

with 6 decimal places.

3.1 Workflow

A very simple description of the program:

1) Defines variables

2) Asks user for input

3) Checks whether inputs meet condiditons

4) Calculates

5) Outputs results

3.2 Algorithm

For user's better understanding of the code and the whole process, here is an UML diagram.

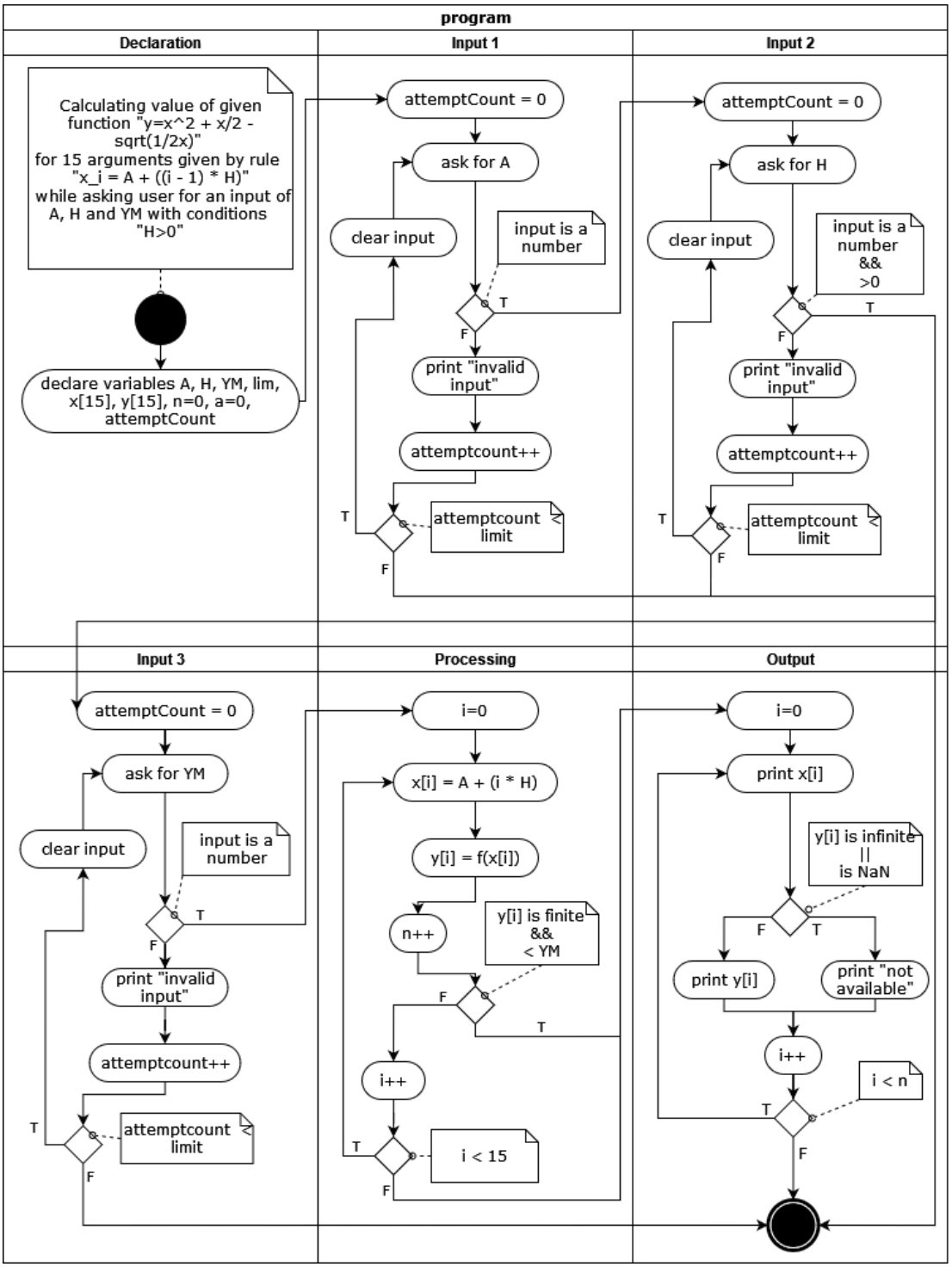

Figure 2: UML diagram [1]

3.3 Special cases

#	**Description**	**Status**	**Solution**
1	Input does not meet $H > 0$ condition	Resolved	Program will recognise such input as invalid. See #2
2	Input is invalid / not number	Resolved	Program will recognise faulty input and will ask user for input again. However not more than 3 times. (Limit can be changed easily at the beginning of the code.)
3	Too large numbers	Unresolved	User can put numbers too large leading to loss of precision.
4	Complex plane	Unresolved	Support for complex numbers is not added. Calculation leading to imaginary number will simply output "not available".
5	Undefined operation	Resolved	Undefined operations will lead to the output of "not available".

Table 1: Table with special cases

4 Summary

After getting to know the assignment, I have started writing code right away. Slowly, step by step, making sure that every addition to the code works as intended.

Starting with declaring variables and asking for input. It was important to keep the condition $H > 0$ in mind. However, the bigger issue was to ensure that program will not take characters and other symbols as valid input. Of course, an limit for possible input attempts was necessary.

Moving on to the calculation process itself, this was fairly easy part, as the only important condition was the $y > YM$. However, another issue arose. Undefined function value would sometimes trigger this condition, even though "NaN" cannot be bigger or smaller than real number.

Although it might be shorter and easier to read to have output as a part of calculation process, I have decided to keep them separate. This is because most of my programming experience is from matlab, where we would like to keep the values in an array, instead of just being printed out. Nevertheless I have also decided to include 2in1 (process and output) code in a form of a text comment.

Finally, I have done UML diagram, so the whole process can be clearer for the reader / user.

References

[1] draw.io. *diagrams.net*. URL: https://app.diagrams.net/.

[2] Geogebra. *Graphing calculator*. URL: https://www.geogebra.org/graphing?lang=en.

Annex - Screenshots

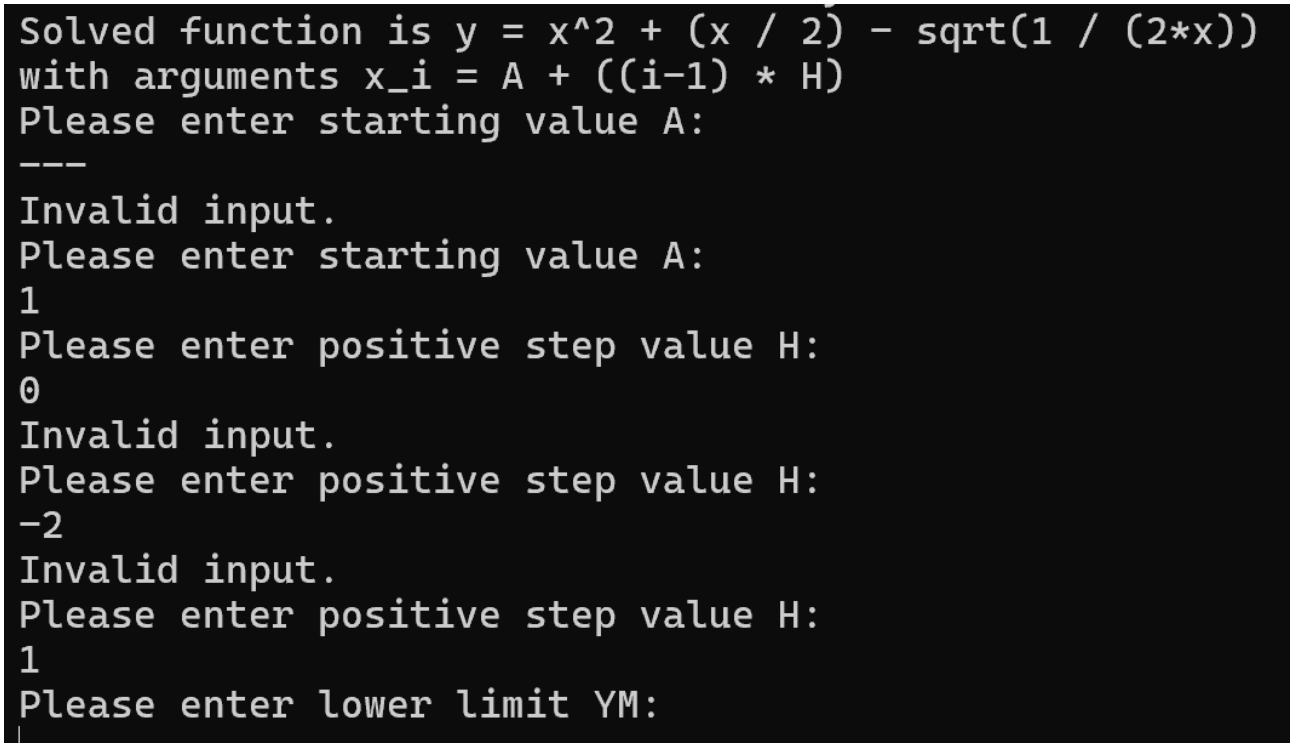

Figure 3: Programs reaction to invalid input

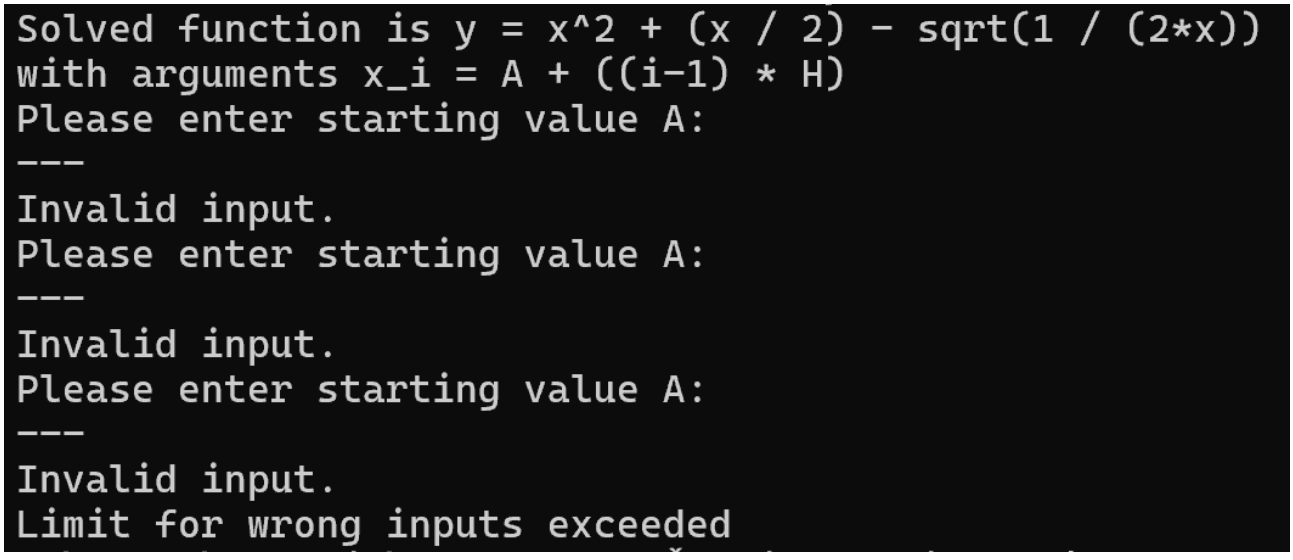

Figure 4: Programs reaction to too many invalid inputs

```
Solved function is y = x^2 + (x / 2) - sqrt(1 / (2*x))
with arguments x_i = A + ((i-1) * H)
Please enter starting value A:
-15
Please enter positive step value H:
3
Please enter lower limit YM:
1
x_01 = -15.000000          y_01 = not available
x_02 = -12.000000          y_02 = not available
x_03 = -9.000000           y_03 = not available
x_04 = -6.000000           y_04 = not available
x_05 = -3.000000           y_05 = not available
x_06 = 0.000000            y_06 = not available
x_07 = 3.000000            y_07 = 10.091752
x_08 = 6.000000            y_08 = 38.711325
x_09 = 9.000000            y_09 = 85.264298
x_10 = 12.000000           y_10 = 149.795876
x_11 = 15.000000           y_11 = 232.317426
x_12 = 18.000000           y_12 = 332.833333
x_13 = 21.000000           y_13 = 451.345697
x_14 = 24.000000           y_14 = 587.855662
x_15 = 27.000000           y_15 = 742.363917
```

Figure 5: Programs reaction to undefined calculation

```
Solved function is y = x^2 + (x / 2) - sqrt(1 / (2*x))
with arguments x_i = A + ((i-1) * H)
Please enter starting value A:
-10
Please enter positive step value H:
1
Please enter lower limit YM:
2
x_01 = -10.000000           y_01 = not available
x_02 = -9.000000            y_02 = not available
x_03 = -8.000000            y_03 = not available
x_04 = -7.000000            y_04 = not available
x_05 = -6.000000            y_05 = not available
x_06 = -5.000000            y_06 = not available
x_07 = -4.000000            y_07 = not available
x_08 = -3.000000            y_08 = not available
x_09 = -2.000000            y_09 = not available
x_10 = -1.000000            y_10 = not available
x_11 = 0.000000             y_11 = not available
Value y_12 (0.792893) is lower than YM.
```

Figure 6: Programs reaction to y $<$ YM.

```
Solved function is y = x^2 + (x / 2) - sqrt(1 / (2*x))
with arguments x_i = A + ((i-1) * H)
Please enter starting value A:
111111111111111111111111
Please enter positive step value H:
111111111111111111111111
Please enter lower limit YM:
111111111111111111111111
x_01 = 111111111111111092469760.000000        y_01 = 12345679012345678852532051294463211945453630259 20.000000
x_02 = 222222222222222184939520.000000        y_02 = 49382716049382715410128205177852847781814521036 80.000000
x_03 = 333333333333333277409280.000000        y_03 = 11111111111111110805019569335803527359330256945152.000000
x_04 = 444444444444444369879040.000000        y_04 = 19753086419753086164051282071141139112725808414720.000000
x_05 = 555555555555555462348800.000000        y_05 = 30864197530864194372922422139530852206807900749824.000000
x_06 = 666666666666666554818560.000000        y_06 = 44444444444444443220078277343214109437321027780608.000000
x_07 = 777777777777777647288320.000000        y_07 = 60493827160493830109370418414777096539017024897024.000000
x_08 = 888888888888888739758080.000000        y_08 = 79012345679012344656205128284564556450903233658880.000000
x_09 = 999999999999998758486016.000000        y_09 = 99999999999999966091394972813265975050994337185792.000000
x_10 = 111111111111111108777213952.000000     y_10 = 12345679012345673595331482027950238058326096923852 8.000000
x_11 = 122222222222222223090909184.000000     y_11 = 14938271604938272693412069017086057247465173889843 2.000000
x_12 = 133333333333333333109637120.000000     y_12 = 17777777777777777728803131093728564377492841111224 32.000000
x_13 = 144444444444444443128365056.000000     y_13 = 20864197530864195686864181444273203289509935343206 4.000000
x_14 = 155555555555555553147092992.000000     y_14 = 24197530864197523736073193710186632966812683206656 0.000000
x_15 = 166666666666666667460788224.000000     y_15 = 27777777777777778220484578187433644692882197158297 60.000000
```

Figure 7: Programs reaction to very large numbers.

```c
#include "stdio.h"
#include "math.h"
#include "stdlib.h"

double in(int lim, char [], char[]);

int cal(double A, double H, double YM, double x[], double y[]);

void out(int n, double x[], double y[]);

int main() {

    double A;
    double H;
    double YM;
    double x[15];
    double y[15];
    int n;
    int lim = 3;

    printf( format: "Solved function is y = x^2 + (x / 2) - sqrt(1 / (2*x)) \nwith arguments x_i = A + ((i-1) * H)\n");

    A = in(lim,  type: "A",  w: "starting value");
    H = in(lim,  type: "H",  w: "positive step value");
    YM = in(lim,  type: "YM",  w: "lower limit");

    n = cal(A, H, YM, x, y);

    out(n, x, y);
    return 0;
}

double in(int lim, char type[], char w[]) {
    double I;
    for (int att = 0; att < lim; att++) {
        printf( format: "Please enter %s %s: \n", w, type);

        switch ((int) type[0]) {

            case 'A':
            case 'Y':
                if (scanf( format: "%lf", &I) == 1) { // Check if the input is valid
                    return I;  // Valid input, exit the loop
                }

            case 'H':
                // Check if the input is valid (scanf returns 1 if a valid double is read)
                if (scanf( format: "%lf", &I) == 1 && I > 0) {
                    return I;  // Valid input, exit the loop
                }

        }
        printf( format: "Invalid input.\n");
        while (getchar() != '\n');  // Clear the input buffer to discard invalid input

    }
    printf( format: "Limit exceeded");
    exit( Code: 1); //ends process if limit of inputs has been reached
}

```

Figure 8: Code part 1

```c
int cal(double A, double H, double YM, double x[], double y[]) {
    int n = 0;
    for (int i = 0; i < 15; i++) {
        x[i] = A + (i * H);
        y[i] = (x[i] * x[i]) + (x[i] / 2) - sqrt( x: 1 / (2 * x[i]));

        if (isfinite(y[i]) && (y[i] < YM)) {
            return n; //ends calculation in case of y < YM
        }
        n++; //keeps track of how many calculations have been done ?????
    }
    return n;
}

void out(int n, double x[], double y[]) {
    for (int i = 0; i < n; i++) {
        printf( format: "x_%02i = %-20lf ", i + 1, x[i]); //prints x value

        if (isnan(y[i]) || isinf(y[i])) { //checks for inf or nan
            printf( format: "y_%02i = not available\n", i + 1);
        } else {
            printf( format: "y_%02i = %lf\n", i + 1, y[i]); //otherwise prints y value
        }
    }

    if (n < 15) {
        printf( format: "Value y_%i is lower than YM.\n", n + 1); //lets user know about the breaking of condition y>YM
    }
}
```

Figure 9: Code part 2