Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра вычислительных методов и программирования

Отчет по лабораторной работе №6
Алгоритмы поиска корней уравнений
Вариант 11

Выполнил:
студент 1 курса
группы № 348602
Трошкин Дмитрий Сергеевич

Проверил:
Матюшкин Светослав Иванович

Минск 2023

1 АЛГОРИТМЫ ПОИСКА КОРНЕЙ УРАВНЕНИЙ

Цель работы: изучить алгоритмы поиска алгебраических уравнений с заданной точностью.

1.1 Условие

Написать и отладить программу поиска всех корней функции f(x) на отрезке [a, b] в соответствии с вариантом (табл. 6.1). Метод нахождения корня оформить в виде отдельной функции, алгоритм которой описать блок-схемой.

1.2 Исходный код

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef double (*f_type)(double x);

double calc_derivative(f_type f, double x, double dx)
{
    return (f(x + dx) - f(x)) / dx;
}

double func_max(f_type f, double a, double b, double eps)
{
    double max = 0, y;
    for(double x = a - eps; x < b + eps; x += eps)
    {
        y = f(x);
        max = max < y ? y : max;
    }
    return max;
}
double func_min(f_type f, double a, double b, double eps)
{
    double min = 0, y;
    for(double x = a - eps; x < b + eps; x += eps)
    {
        y = f(x);
        min = min > y ? y : min;
    }
    return min;
}

double *gen_val_table(f_type f, double a, double b, double *max, double *min, int size)
```

```
{
    double h = (b - a) / (size+1) ;
    //*min = *max = f(a);
    double *arr = malloc(size * sizeof(double));

    for(int i = 0; i < size; i++, a += h)
    {
        arr[i] = f(a);
        if(isinf(arr[i])) arr[i] = f(a-h);
        *min = (*min > arr[i]) ? arr[i] : *min;
        *max = (*max < arr[i]) ? arr[i] : *max;
    }
    return arr;
}

void draw_table(char *plot, char symbol, double *y, int size,
                int height, double min, double max)
{
    double stepY = (max - min) / height;

    for(int i = 1; i < size - 2; i++) {
        if(y[i] >= min && y[i] <= max)
            plot[i + (int)((max - y[i]) / stepY) * size] = symbol;
    }
}

void draw_axis(char *plot, char x, char y, double a, double b,
                int size, double min, double max, int height)
{
    double stepX = (b - a) / size;
    double stepY = (max - min) / height;

    for(int i = 1; i < size - 3; i++)
        plot[i + (int)trunc(max / stepY) * size] = x;
     plot[size - 3 + (int)trunc(max / stepY) * size] = '>';
    for(int i = 1; i < height - 1; i++)
        plot[size - (int)trunc(b / stepX) + i * size] = y;
    plot[size - (int)trunc(b / stepX) + size] = '^';
}

void draw_box(char *plot, int size, int height)
{
    for(int i = 0; i < height; i++)
        plot[i * size] = plot[size - 2 + i*size] = '|';

    for(int i = 0; i < size - 1; i++)
        plot[i] = plot[i + size*(height-1)] = '-';

    for(int i = 0; i < height - 1; i++)
        plot[size - 1 + i*size] = '\n';
```

```
    plot[size + (height-1)*size] = 0;
}

void plot_term(f_type f, double a, double b, int width, int height)
{
    double yMax  = f(a);
    double yMin  = f(a);
    char *res = malloc((width+1) * height * sizeof(char));
    memset(res, ' ', (width+1)* height - 1);
    double *y1 = gen_val_table(f, a, b, &yMax, &yMin, width);

    yMin -= 0.05 *(yMax - yMin);
    yMax += 0.05 *(yMax - yMin);

    draw_table(res, '1', y1, width , height, yMin, yMax);
    draw_axis(res, 'x', 'y', a, b, width, yMin, yMax, height);
    draw_box(res, width, height);
    puts(res);
    printf("Axis: xMin = %lf, xMax = %lf, yMin = %lf, yMax = %lf\n", a, b, yMin, yMax);
    free(y1);
}

double metod_n(f_type f, double x, double eps)
{
    return x - f(x) / calc_derivative(f, x, eps);
}

void calc(f_type f, double a, double b, double h, double eps)
{
    double xn = a - 1, y;
    int roots_count = 0;
    printf("--- Roots of function ---\n");
    for(double x = a - h; x < b + h; x += h)
    {
            y = f(x);
            if(y*f(x+h) < 0) {
                    roots_count++;
                    printf(" %i. %lf\n", roots_count, metod_n(f, x, eps));
            }
            else if(y < eps && xn < a) xn = x;
            else if(y > eps && xn > a) {
                    roots_count++;
                    printf(" %i. [%lf, %lf]\n", roots_count, xn, x);
                    xn = a - 1;
            }

    } if(!roots_count) printf("No roots.\n");
}

double function(double x) { return (x*x*x + 6*x*x - 0.02*pow(M_E, x) - 14); }
```

```
int main()
{
        double a = -6, b = 3, h = 1E-6, eps = 1E-5;
        calc(function, a, b, h, eps);
        printf("Plotting function\n");
        plot_term(function, a, b, 216, 41);

}
```