Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра вычислительных методов и программирования

Отчет по лабораторной работе №8
Алгоритмы вычисления интегралов
Вариант 11

Выполнил:
студент 1 курса
группы № 348602
Трошкин Дмитрий Сергеевич

Проверил:
Матюшкин Светослав Иванович

Минск 2023

1 АЛГОРИТМЫ ВЫЧИСЛЕНИЯ ИНТЕГРАЛОВ

Цель работы: изучить алгоритмы нахождения значений интегралов.

1.1 Условие

Написать и отладить программу вычисления интеграла указанным методом двумя способами – по заданному количеству разбиений n и заданной точности (метод 1) (задания табл. 8.1). Реализацию указанного метода оформить отдельной функцией, алгоритм которой описать в виде блок-схемы.

1.2 Исходный код

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

typedef double (*f_type)(double x);
typedef double (*f_inter_type)(double* x_arr, double* y_arr, double x, int m);

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;
}

double *gen_inter_val_table(f_inter_type f, double a, double b, double *max,
                    double *min, double *x_arr, double *y_arr, int m, 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(x_arr, y_arr, a, m);
        if(isinf(arr[i])) arr[i] = f(x_arr, y_arr, a-h, m);
        *min = (*min > arr[i]) ? arr[i] : *min;
        *max = (*max < arr[i]) ? arr[i] : *max;
    }
    return arr;
}

void to_inter(f_type f, int m, double a, double b, double **x_arr, double **y_arr)
{
    double step = (b - a) / (m - 1);
    *x_arr = malloc(m*sizeof(double));
    *y_arr = malloc(m*sizeof(double));
    for(int i = 0; i < m; i++, a+= step)
    {
        (*x_arr)[i] = a;
        (*y_arr)[i] = f(a);
    }
}


double gausa2(f_type f, double a, double b, int m)
{
    double result = 0;
```

```
    double h = (b - a) / m;
    double delta = h / sqrt(3);
    for(double x = a - (delta/2); x < b + h/2; x += h)
    {
        result += f(x) + f(x + delta);
    }
    return result * h / 2;
}
double gausa3(f_type f, double a, double b, int m)
{
    double result = 0;
    double h = (b - a) / m;
    double delta = (h/2) * sqrt(3./5);
    double c1 = 5./9;
    double c2 = 8./9;

    for(double x = a; x < b + h/2; x += h)
        result += c2*f(x) + c1*(f(x + delta) + f(x - delta));

    return result * h / 2;
}

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 *y_arr, *x_arr;
    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 - 1);

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

int main()
{
        double a = -5, b = 3, eps, s1, s2;
        int mode, n = 5;

    printf("Enter a, b: ");
    if(scanf("%lf %lf", &a, &b) < 2) {
        printf("Reading failed!\n");
        return -1;
    }

    printf("Choose mode:\n1 - by partitions count\n2 - by error\n");
    if(!scanf("%i", &mode)){
        printf("Reading failed!\n");
        return -1;
    }

    switch(mode)
    {
```

```
        case 1:
            printf("Enter n: ");
            while(scanf("%i", &n) < 1) printf("Try again!\n");
            s2 = gausa2(function, a, b, n);
            break;
        case 2:
            printf("Enter eps: ");
            while(scanf("%lf", &eps) < 1) printf("Try again!\n");

            s1 = gausa2(function, a, b, n);
            do {
                s2 = s1;
                n *= 2;
                s1 = gausa2(function, a, b, n);
            } while(fabs(s1 - s2) > eps);
            break;

        default:
            printf("Wrong mode. Try again!\n");
            return -1;
    }
    printf("Plotting function\n");
        plot_term(function, a, b, 128, 40);

    printf("Integral of funcion in [%lf, %lf], with n = %i: %lf\n", a, b, n/2, s1);
}
```