Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

# Отчет по заданию №6

## «Сборка многомодульных программ. Вычисление корней уравнений и определенных интегралов.»

## Вариант 2 / 3* / 3

Выполнил:
студент 101 группы
Александров М. С.

Преподаватель:
Дудина И. А.

Москва
2021

# Содержание

# Постановка задачи

Задачей является написание и сборка программы, реализующей численный метод, позволяющий вычислять площадь плоской фигуры, ограниченной тремя кривыми.

- методы приближенного решения уравнений (для нахождения вершин фигуры) **методом касательных (Ньютона)** либо **методом бисекции**,
- квадратурная формула (для вычисления интеграла) - **формула Симпсона**,
- три уравнения кривых надо реализовать на языке ассемблера с соглашением вызова cdecl,
- все остальные функции программы реализуются на языке Си,
- программа должна поддерживать различные ключи (в том числе для запуска тестирования на дополнительных функциях),
- необходимо аналитически найти отрезок для поиска точек пересечения и обосновать его выбор,
- выбор погрешностей $\varepsilon_1$ и $\varepsilon_2$ для вычисления корня и интеграла соответственно должен гарантировать общую точность $\varepsilon = 0.001$,
- данные функции: $y = 3(\frac{0.5}{x+1} + 1), y = 2.5x - 9.5, y = \frac{5}{x}$ при $x > 0$.

## Математическое обоснование

Найдем отрезки, на которых будет осуществляться поиск вершин фигуры: Для метода бисекции требуется [1] (существование единственного корня на отрезке) непрерывность, неравенство знаков на концах отрезка, сохранение знака производной на отрезке. Для метода Ньютона [1] вдобавок требуется неизменение знака второй производной на отрезке.

Заметим, что при $x > 0$ все наши функции непрерывны и непрерывно дифференцируемы (в том числе два раза), а значит и разность каждых двух функций непрерывна и непрерывно дифференцируема (в т. ч. два раза)

- $y = 3(\frac{0.5}{x+1} + 1) - (2.5x - 9.5)$ на отрезке [4, 6] монотонно убывает, ее первая производная $\frac{-6}{(2x+2)^2} - \frac{5}{2}$ не меняет знак и не обращается на нем в ноль (она всегда меньше нуля при x не равном -1), а ее вторая производная $\frac{24}{(2x+2)^3}$ не меняет знак (числитель и знаменатель больше нуля при положительных x). Значения на концах отрезка имеют разный знак: $3 * (\frac{0.5}{4+1} + 1) - 2.5 * 4 + 9.5 = 3.1, 3 * (\frac{0.5}{6+1} + 1) - 2.5 * 6 + 9.5 = -\frac{16}{7}$, значит на нем существует единственный корень и применимы методы Ньютона и деления отрезка пополам.

- $y = 3(\frac{0.5}{x+1} + 1) - \frac{5}{x}$ на отрезке [1, 2] монотонно возрастает, ее первая производная $\frac{-6}{(2x+2)^2} + \frac{5}{x^2} = \frac{14x^2+40x+20}{x^2(2x+2)^2}$ не меняет знак (она больше нуля при $x > \frac{-10+\sqrt{30}}{7}$, а это число меньше 1) и не обращается на нем в ноль, а ее вторая производная $\frac{24}{(2x+2)^3} - \frac{10}{x^3} = \frac{-56x^3-240x^2-240x-80}{(2x^2+2x)^3}$ не меняет знак (меньше нуля). Значения на концах отрезка имеют разный знак: $3 * \frac{0.5}{1+1} + 3 - \frac{5}{1} = \frac{-5}{4}, 3 * \frac{0.5}{2+1} + 3 - \frac{5}{2} = 1$, значит на нем существует единственный корень и применимы методы Ньютона и деления отрезка пополам.

- $y = 2.5x - 9.5 - \frac{5}{x}$ на отрезке [4, 5] монотонно возрастает, ее первая производная $\frac{5}{2} + \frac{5}{x^2}$ не меняет знак и не обращается на нем в ноль (оба слагаемых больше нуля на отрезке), а ее вторая производная $\frac{-10}{x^3}$ не меняет знак (числитель меньше нуля, знаменатель больше нуля). Значения на концах отрезка имеют разный знак: $2.5*4-9.5-\frac{5}{4} = \frac{-3}{4}, 2.5*5-9.5-\frac{5}{5} = 2$, значит на нем существует единственный корень и применимы методы Ньютона и деления отрезка пополам.

Нужно определить, при каких $\varepsilon_1$ и $\varepsilon_2$ общая точность будет не хуже $\varepsilon = 0.001$. Нахождение корня происходит с погрешностью $\varepsilon_1$, значит отрезок интегрирования находится с погрешностью $2\varepsilon_1$. Вклад погрешности нахождения пределов интегрирования в погрешность вычисления интеграла (в качестве квадратурной формулы используется формула Симпсона) :
$\int_a^b f(x)dx \approx \frac{b-a}{6}(f(a) + f(b) + 4f(\frac{a+b}{2}))$
Отрезок нашелся с погрешностью $b - a = 2\varepsilon_1$, значение функции можно ограничить наибольшим значением функции на отрезке [1, 6], (пределы интегрирования лежат на нем, см. выше) – 5.5. Значит вклад равен $\frac{2\varepsilon_1}{6}(5.5 + 5.5 + 4 * 5.5) = 11\varepsilon_1$.
Тогда интеграл каждой функции будет вычислен с погрешностью $11\varepsilon_1 + \varepsilon_2$. Так

как вычисляется три интеграла, то итоговая погрешность равна $33\varepsilon_1 + 3\varepsilon_2$. Тогда выбор $\varepsilon_1 = 0.000001$ и $\varepsilon_2 = 0.000001$ обеспечит погрешность вычислений $\varepsilon = 0.001$

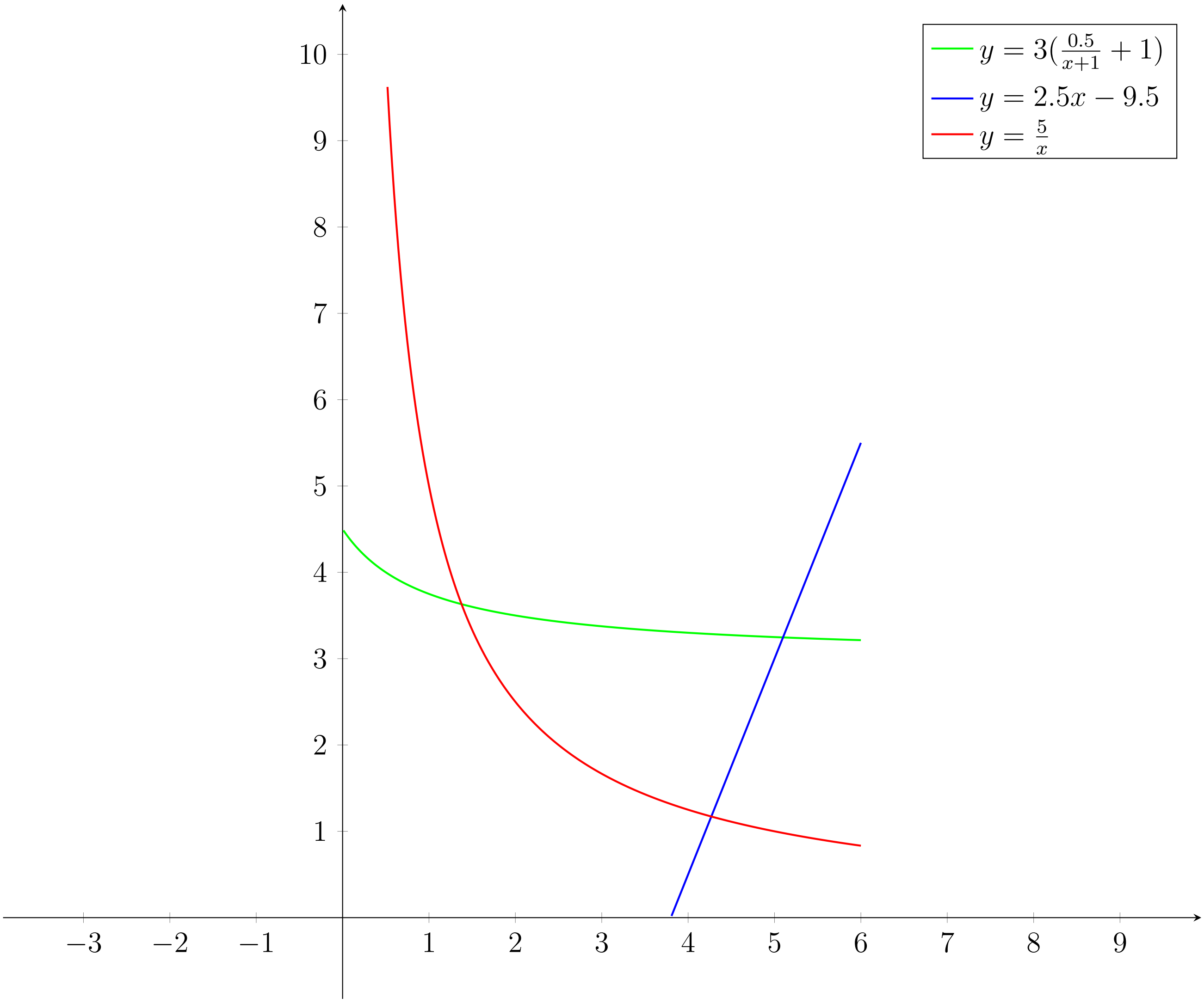


Рис. 1: Плоская фигура, ограниченная графиками заданных уравнений

# Результаты экспериментов

| Кривые | $x$ | $y$ |
|---|---|---|
| 1 и 2 | 5.098387 | 3.245967 |
| 2 и 3 | 4.268544 | 1.171360 |
| 1 и 3 | 1.377015 | 3.631044 |

Таблица 1: Координаты точек пересечения

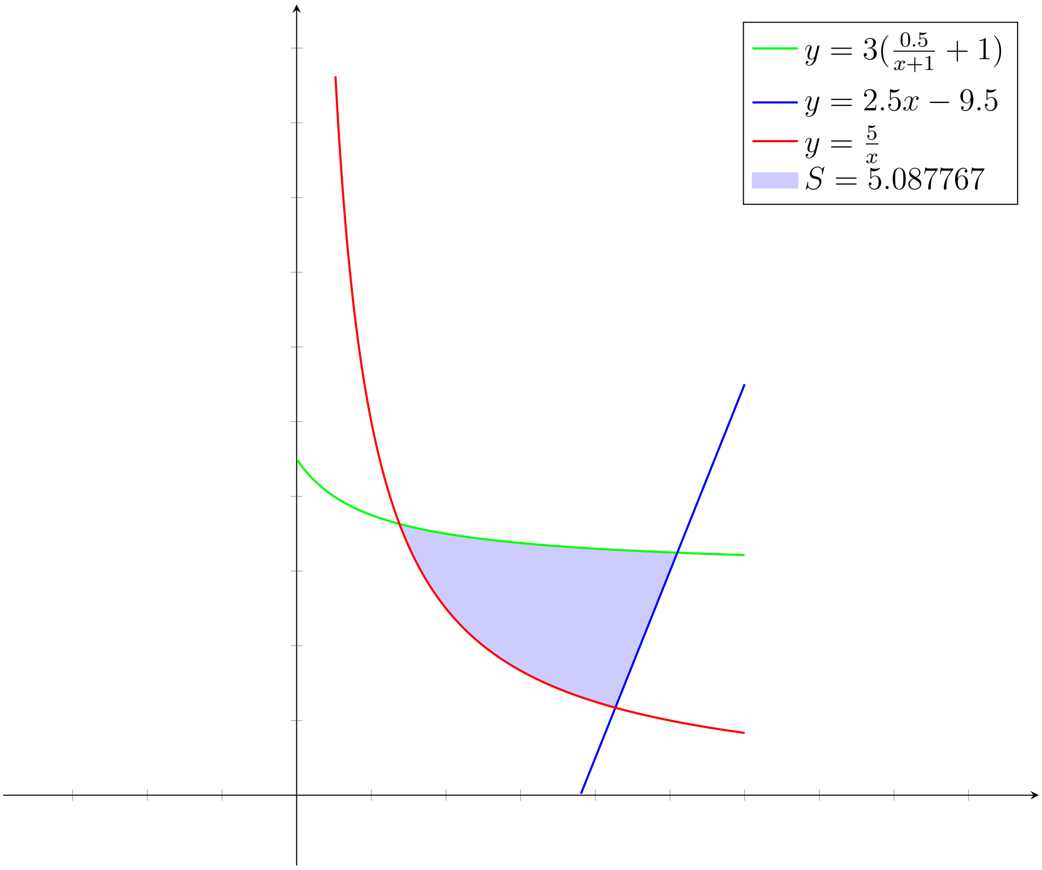


Рис. 2: Плоская фигура, ограниченная графиками заданных уравнений

# Структура программы и спецификация функций

1. `double f1(double x)`, `double f2(double x)`, `double f3(double x)` - сами заданные функции, написанные на ассемблере.

2. `double df1(double x)`, `double df2(double x)`, `double df3(double x)` - производные заданных функций, написанные на ассемблере.

3. `double ddf1(double x)`, `double ddf2(double x)`, `double ddf3(double x)` - вторые производные заданных функций, написанные на ассемблере.

4. `double root_newton(double (*f)(double x), double (*df)(double x), double (*ddf)(double x), double (*g)(double x), double (*dg)(double x), double (*ddg)(double x), double a, double b, double eps1)` - функция нахождения точки пересечения методом касательных.

5. `double root_bisection(double (*f)(double x), double (*g)(double x), double a, double b, double eps1)` - функция нахождения точки пересечения методом деления отрезка пополам.

6. `double root(double (*f)(double x), double (*df)(double x), double (*ddf)(double x), double (*g)(double x), double (*dg)(double x), double (*ddg)(double x), double a, double b, double eps1)` - функция, вызывающая метод деления отрезка пополам или метод касательных в зависимости от передачи ключа -Dbi.

7. `double calc(double (*f)(double x), double a, double b)` - формула Симпсона вычисления интеграла на отрезке.

8. `double integral(double (*f)(double x), double a, double b, double eps2)` - функция вычисления интеграла делением отрезка на более мелкие и вызовом формулы Симпсона для каждого.

9. `void print_help(void)` - вывод всех допустимых ключей.

10. `int main(int argc, char *argv[])` - главная функция, осуществляющая основной функционал.

11. `double f4(double x)`, `double f5(double x)`, `double f6(double x)` - дополнительные функции, нужные для тестирования программы.

12. `double df4(double x)`, `double df5(double x)`, `double df6(double x)` - производные дополнительных функций.

13. `double ddf4(double x)`, `double ddf5(double x)`, `double ddf6(double x)` - вторые производные дополнительных функций.

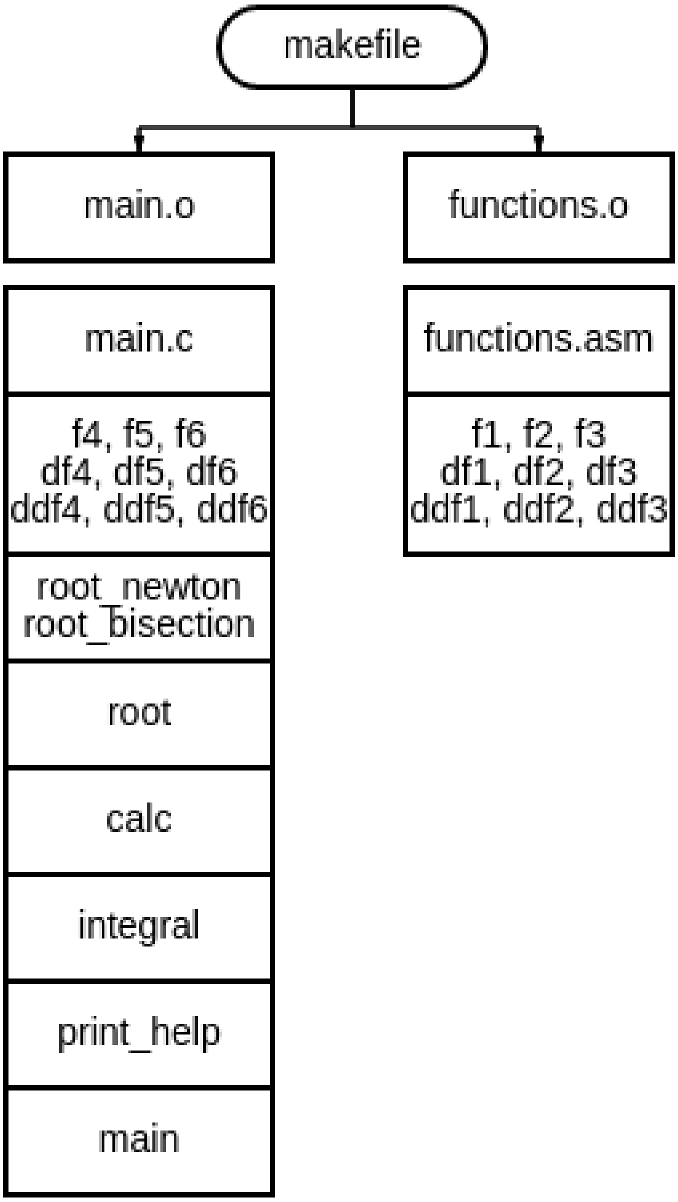


Рис. 3: Схема взаимодействия функций программы

# Сборка программы (Make-файл)

Так как был выбран вариант с реализацией дополнительного метода нахождения корня, было необходимо менять используемый метод на этапе препроцессирования. Были заданы дополнительные команды сборки, передающие ключ `-Dbi`, который даст возможность использовать `#ifdef bi` в коде основной программы для переключения используемого метода.

```
all: prog
use_bisection: prog_bisection

prog: main.o functions.o
    gcc -m32 functions.o main.o -lm -o prog

prog_bisection: main.o_bisection func.o
    gcc -m32 functions.o main.o -lm -o prog

main.o: main.c
    gcc -m32 -c main.c -o main.o

main.o_bisection: main.c
    gcc -m32 -c main.c -Dbi -o main.o

functions.o: functions.asm
    nasm -f elf32 -o functions.o functions.asm

clean:
    rm -rf *.o prog
```

# Отладка программы, тестирование функций

Для тестирования есть ключи `-test_root` и `-test_int` для отладки функций `root` и `integral` соответственно. Для заданных функций надо передавать номера 1, 2, 3, а для дополнительных - $3x^3$, $x^2$, $x$ - номера 4, 5, 6. Также были добавлены их первые и вторые производные.
Поиск корня: Для тестирования вводятся левая граница отрезка, правая граница отрезка, требуемая точность вычислений, номер первой функции, номер второй функции.
Для функций под номерами 4, 5, 6 (первые производные: $9x^2, 2x, 1$, вторые производные $18x, 2, 0$ соответственно) на выбранных отрезка выполнены условия для применения как метода деления отрезка пополам, так и метода Ньютона, а именно: разности функций непрерывны на этих отрезках (так как сами функции непрерывны), непрерывно дифференцируемы (так как их производные $9x^2, 2x, 1$ непрерывны), причем первая и вторая производные не меняют знака и не обращаются в нуль ($9x^2 - 2x, 9x^2 - 1, 2x - 1$ больше нуля на каждом из выбранных отрезков, $18x - 2, 18x - 0, 2 - 0$ больше нуля на каждом из отрезков).
Результаты тестирования:

Метод Ньютона:
`0.25 10 0.0001 4 6`
`Output: 0.577350`
(правильный ответ: $\frac{\sqrt{3}}{3}$)

`0.25 1.0 0.0001 4 5`
`Output: 0.333333`
(правильный ответ: $\frac{1}{3}$)

`0.5 1.5 0.0001 5 6`
`Output: 1.000000`
(правильный ответ: 1)

Метод деления отрезка пополам:
`0.25 10 0.0001 4 6`
`Output: 0.577338`
(правильный ответ: $\frac{\sqrt{3}}{3}$)

`0.25 1.0 0.0001 4 5`
`Output: 0.333313`
(правильный ответ: $\frac{1}{3}$)

`0.5 1.5 0.0001 5 6`
`Output: 1.000000`
(правильный ответ: 1)

Вычисление интеграла: Для тестирования вводятся левая граница отрезка, правая граница отрезка, требуемая точность вычислений, номер функции. Результаты тестирования:

```
1.0 2.5 0.0001 4
Output: 28.546875
```
(правильный ответ: $\frac{1827}{64}$)

```
-1.0 1.0 0.0001 5
Output: 0.666667
```
(правильный ответ: $\frac{2}{3}$)

```
-5.0 100.0 0.0001 6
Output: 4987.500000
```
(правильный ответ: $\frac{9975}{2}$)

# Программа на Си и на Ассемблере

Весь код выложен на репозиторий в GitHub. Для запуска надо перенести все файлы в одну папку, открыть в ней терминал и ввести такие команды:

```
make all
./prog
```

# Анализ допущенных ошибок

1. Первые версии дополнительных функций были выбраны случайно так, что было невозможно найти точное аналитическое решение.

# Список литературы

[1] Ильин В.А., Садовничий В.А., Сендов Бл.Х. Математический анализ. Т.1 — М.: Наука, 1985.