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 5 / 4 / 3

Выполнил:
студент 101 группы
Антипов Г. О.

Преподаватель:
Кузьменкова Е. А.

Москва
2025

Содержание

1. Постановка задачи **2**

2. Математическое обоснование **3**
2..1 Комбинированный метод (метод хорд и касательных) 3
2..2 Метод Симпсона 5
2..3 Выбор ε_1 и ε_2 6
2..4 Обоснование выбора внутренних допусков ε_1 и ε_2 6
2..4.1 Обоснование для ε_1 7
2..4.2 Обоснование для ε_2 7

3. Результаты экспериментов **8**

4. Структура программы и спецификация функций **9**
4..1 Общая архитектура 9
4..2 Детальное описание файлов проекта 9
4..2.1 compute.c 9
4..2.2 functions.h 10
4..2.3 test_mode.c 10
4..2.4 main.c 10
4..2.5 func.asm 10
4..2.6 Графическое представление структуры 10

5. Сборка программы (Make-файл) **12**
5..1 Компиляторы и флаги 12
5..1.1 Флаги компиляции C-кода 12
5..1.2 Флаги NASM 12
5..2 Цели сборки 13
5..3 Правила сборки 13
5..4 Особенности 13
5..5 Зависимости модулей 13
5..6 Особенности сборки 14

6. Отладка и тестирование программы **15**
6..1 Функция $f_1(x) = x^2 + 5x + 6$ 15
6..2 Функция $f_2(x) = \sin x$ 15
6..3 Функция $f_3(x) = x^3 - 1$ 15
6..4 Выбор отрезков для поиска корней 16
6..5 Выводы 16

7. Анализ допущенных ошибок **17**

Список цитируемой литературы **18**

1. Постановка задачи

В данной работе решается задача вычисления площади плоской фигуры, ограниченной тремя кривыми:

$$f_1(x) = 0.35x^2 - 0.95x + 2.7 \qquad f_2(x) = 3x + 1 \qquad f_3(x) = \frac{1}{x+2}$$

Для решения поставленной задачи необходимо выполнить следующие этапы:

- Аналитически определить пределы интегрирования, найдя отрезки пересечения для каждой пары функций;
- Реализовать комбинированный метод (метод хорд и касательных) для нахождения абсцисс точек пересечения кривых;
- Реализовать метод Симпсона для численного вычисления площади заданной фигуры.
- Протестировать написанные функции, для понимания их работоспособности

2. Математическое обоснование

В этом разделе проводится анализ используемых методов: комбинированного метода (метода хорд и касательных) для поиска корней уравнения и метода Симпсона для численного интегрирования.

2..1 Комбинированный метод (метод хорд и касательных)

Пусть необходимо найти корень уравнения $F(x) = f_1(x) - f_2(x) = 0$ на отрезке $[a, b]$. Согласно [1] для того чтобы комбинированный метод сходился необходимо, чтобы функция на выбранном отрезке удовлетворяла следующим условиям:

- $F(x)$ непрерывна и непрерывно дифференцируема на $[a, b]$;
- $F(a)F(b) < 0$ (корень лежит внутри отрезка);
- $F'(x)$ монотонна и не меняет знак на $[a, b]$.

Алгоритм метода:

На каждом шаге k имеем текущий интервал $[a_k, b_k]$ с $F(a_k)F(b_k) < 0$. Для построения следующего отрезка необходимо:

1. Построить приближение по методу хорд:

$$x_k^{(\text{chor})} = \frac{a_k\,F(b_k) - b_k\,F(a_k)}{F(b_k) - F(a_k)} \tag{1}$$

2. Определить точку d_k для метода касательных:

$$d_k = \begin{cases} b_k, F'(x)F''(x) > 0 \text{ на } [a_k, b_k] \\ a_k, \text{в противном случае} \end{cases} \tag{2}$$

   Вычислить приближение

$$x_k^{(\text{sec})} = d_k - \frac{F(d_k)}{F'(d_k)} \tag{3}$$

3. Выбрать следующий интервал $[a_{k+1}, b_{k+1}]$ так, чтобы $F(a_{k+1})\,F(b_{k+1}) < 0$, например

$$[a_{k+1}, b_{k+1}] = \left[\min(x_k^{(\text{chor})}, x_k^{(\text{sec})}), \max(x_k^{(\text{chor})}, x_k^{(\text{sec})})\right]$$

Процесс останавливается, когда $|b_{k+1} - a_{k+1}| < \varepsilon_1$.

Выбор начальных отрезков:

Интервалы для поиска корней определяются анализом графиков функций (рис. 1) и проверкой условий:

1. $F(a)F(b) < 0$
2. $F'(x) \neq 0$
3. $F'(x)$ монотонна на $[a, b]$

Где $F(x) = f_i(x) - f_j(x), i \neq j$. Проанализировав графики, получили следующие интервалы для каждой из пар функций:

$$\begin{aligned} f_1(x) = f_2(x) &\ \rightarrow [0, 1] \\ f_1(x) = f_3(x) &\ \rightarrow [-1.96, 0] \\ f_2(x) = f_3(x) &\ \rightarrow [-1, 0] \end{aligned}$$

Проверим, отвечают ли они всем необходими условиям метода.

Математическое обоснование выбора отрезков для поиска корней

Для применения комбинированного метода (метода хорд и касательных) требуется выбрать такие отрезки $[a, b]$, на которых:

1. Функция $F(x) = f_i(x) - f_j(x)$ непрерывна на $[a, b]$.
2. $F(a)F(b) < 0$, то есть значения функции на концах отрезка имеют разные знаки - по теореме Больцано-Коши на этом отрезке существует хотя бы один корень.
3. $F'(x)$ не обращается в ноль и не меняет знак на $[a, b]$ (монотонность производной).

Проверим эти условия для каждой пары функций:

1. $f_1(x)$ и $f_2(x)$

Рассмотрим $F_{12}(x) = f_1(x) - f_2(x) = 0.35x^2 - 3.95x + 1.7$.

- $F_{12}(x)$ - непрерывна на всей $\mathbb{R}$.
- Проверим знаки на концах отрезка $[0, 1]$:
  $F_{12}(0) = 0.35 \cdot 0^2 - 3.95 \cdot 0 + 1.7 = 1.7$
  $F_{12}(1) = 0.35 \cdot 1^2 - 3.95 \cdot 1 + 1.7 = 0.35 - 3.95 + 1.7 = -1.9$
- $F_{12}(0) \cdot F_{12}(1) = 1.7 \cdot (-1.9) < 0$, значит, на $[0, 1]$ есть корень.
- Производная $F'_{12}(x) = 0.7x - 3.95$ на $[0, 1]$:
  $F'_{12}(0) = -3.95$,
  $F'_{12}(1) = 0.7 - 3.95 = -3.25$
  Производная отрицательна и не меняет знак, функция строго убывает.

2. $f_1(x)$ **и** $f_3(x)$

Рассмотрим $F_{13}(x) = f_1(x) - f_3(x) = 0.35x^2 - 0.95x + 2.7 - \frac{1}{x+2}$.

- $F_{13}(x)$ непрерывна при $x > -2$.
- Проверим на $[-1.96, 0]$:
$F_{13}(-1.96) \approx 0.35 \cdot (-1.96)^2 - 0.95 \cdot (-1.96) + 2.7 - \frac{1}{-1.96+2} \approx 0.35 \cdot 3.8416 + 1.862 + 2.7 - \frac{1}{0.04} \approx 1.3446 + 1.862 + 2.7 - 25 \approx 5.9066 - 25 \approx -19.0934$
$F_{13}(0) = 0.35 \cdot 0^2 - 0.95 \cdot 0 + 2.7 - \frac{1}{2} = 2.7 - 0.5 = 2.2$
- $F_{13}(-1.96) \cdot F_{13}(0) < 0$.
- Производная $F'_{13}(x) = 0.7x - 0.95 + \frac{1}{(x+2)^2}$ на $[-1.96, 0]$: знаменатель положителен, x от -1.96 до 0, поэтому $F'_{13}(x)$ не обращается в ноль и не меняет знак.

3. $f_2(x)$ **и** $f_3(x)$

$F_{23}(x) = f_2(x) - f_3(x) = 3x + 1 - \frac{1}{x+2}$.

- $F_{23}(x)$ непрерывна при $x > -2$.
- Проверим на $[-1, 0]$: $F_{23}(-1) = 3 \cdot (-1) + 1 - \frac{1}{-1+2} = -3 + 1 - 1 = -3$
$F_{23}(0) = 0 + 1 - \frac{1}{2} = 1 - 0.5 = 0.5$
- $F_{23}(-1) \cdot F_{23}(0) = -3 \cdot 0.5 = -1.5 < 0$.
- Производная: $F'_{23}(x) = 3 + \frac{1}{(x+2)^2}$ на $[-1, 0]$ производная всегда положительна, функция строго возрастает.

Таким образом, для всех выбранных отрезков выполнены необходимые условия: функция непрерывна, значения на концах имеют разные знаки, производная не обращается в ноль и не меняет знак. Это гарантирует существование и единственность корня на каждом отрезке.

2..2 Метод Симпсона

Пусть необходимо приближенно вычислить интеграл $I = \int_a^b f(x)dx$ используя метод Симпсона. Для этого, согласно [1], необходимо, чтобы $f \in C^4[a, b]$. Для вычисления интеграла необходимо разбить отрезок $[a, b]$ на чётное число n равных подотрезков длины

$$h = \frac{b-a}{n}, \quad x_i = a + i\,h,\ i = 0, 1, \ldots, n$$

На каждом втором подотрезке $[x_{i-1}, x_{i+1}]$ аппроксимируем $f(x)$ параболой, проходящей через точки (x_{i-1}, f_{i-1}), (x_i, f_i), (x_{i+1}, f_{i+1}) и получаем итоговую формулу Симпсона:

$$\int_a^b f(x)\,dx = \frac{h}{3}\Big[f_0 + 4 \sum_{\substack{i=1 \\ i \text{ - нечетные}}}^{n-1} f_i + 2 \sum_{\substack{i=2 \\ i \text{ - четные}}}^{n-2} f_i + f_n\Big] + R \qquad (4)$$

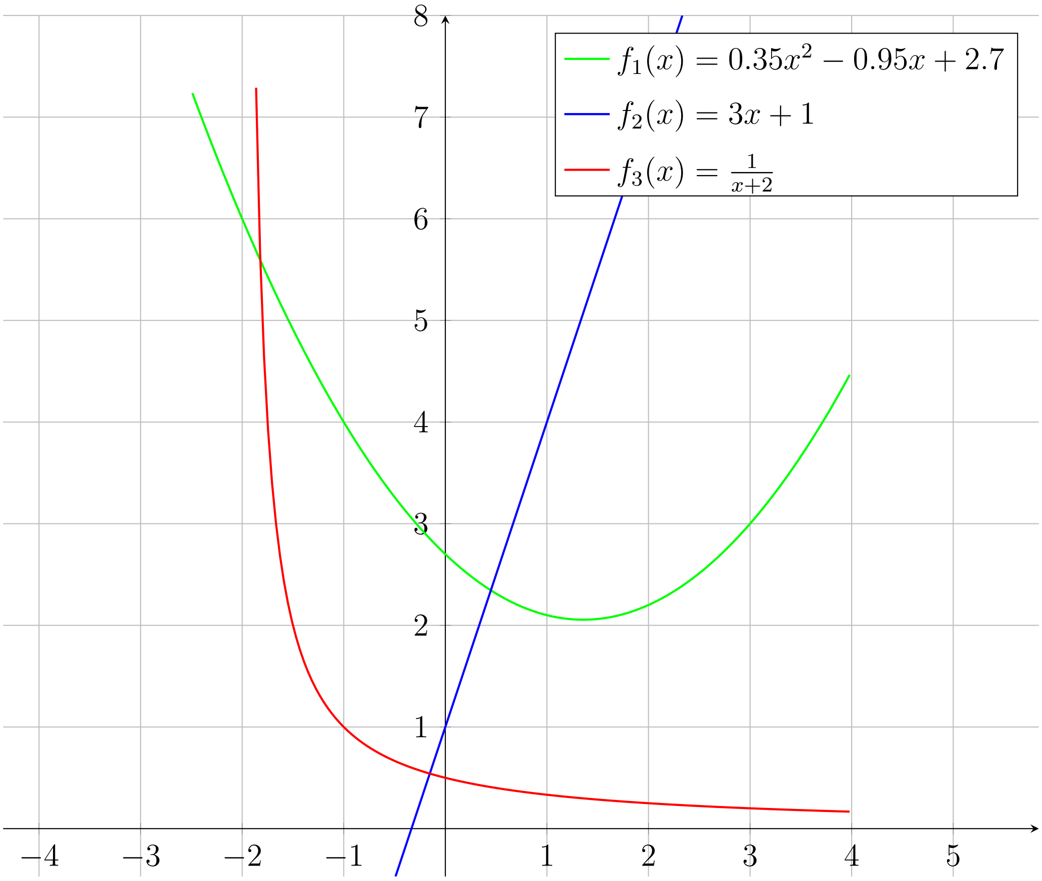


Рис. 1: Плоская фигура, ограниченная графиками заданных уравнений

где R - остаточный член который равен:

$$R = -\frac{(b-a)}{180}\, h^4\, f^{(4)}(\xi), \qquad \xi \in [a,b] \tag{5}$$

или в эквивалентной форме:

$$R = -\frac{(b-a)^5}{2880\, n^4}\, f^{(4)}(\xi) \tag{6}$$

2..3 Выбор ε_1 и ε_2

В программе требуемая итоговая точность равна 10^{-4}. Чтобы гарантировать, что итоговые ошибки при поиске корней и численном интегрировании действительно не превысят этот порог, мы вводим более строгие внутренние допуски:

$$\varepsilon_1 = \varepsilon_2 = 10^{-6}.$$

2..4 Обоснование выбора внутренних допусков ε_1 и ε_2

Для обеспечения общей точности вычислений не хуже 10^{-4}, необходимо учитывать, что ошибки, возникающие на каждом этапе, могут накапливаться. Поэтому выбираются более строгие внутренние допуски ε_1 и ε_2, чтобы гарантировать выполнение требований итоговой точности.

2..4.1 Обоснование для ε_1

Пусть x^* — точное решение уравнения $F(x) = 0$, а x_{k+1} — приближение после (k+1)-й итерации. Тогда, согласно свойствам метода, ошибка оценки корня связана с длиной интервала:

$$|x_{k+1} - x^*| \leq \frac{b_{k+1} - a_{k+1}}{2}.$$

Чтобы обеспечить ошибку не более ε_x, необходимо:

$$\frac{b_{k+1} - a_{k+1}}{2} < \varepsilon_x,$$

где ε_x — допустимая погрешность в корне. Учитывая, что итоговая точность должна быть 10^{-4}, а внутренние допуски выбраны как $\varepsilon_1 = 10^{-6}$, то:

$$|x_{k+1} - x^*| < \frac{\varepsilon_1}{2} = 5 \times 10^{-7}.$$

Это сильно меньше требуемой итоговой погрешности, что обеспечивает буфер для компенсации ошибок арифметики с плавающей точкой и накопления ошибок при итерациях.

2..4.2 Обоснование для ε_2

При численном интегрировании методом Симпсона остаточный член имеет вид:

$$R = -\frac{(b-a)^5}{2880\, n^4} f^{(4)}(\xi),$$

где $M = \max_{x \in [a,b]} |f^{(4)}(x)|$. Для гарантии, что ошибка интегрирования не превысит ε_2, необходимо:

$$|R| \leq \frac{(b-a)^5 M}{2880\, n^4} < \varepsilon_2.$$

Отсюда следует минимальное число разбиений:

$$n \geq \left(\frac{(b-a)^5 M}{2880\, \varepsilon_2} \right)^{1/4}.$$

При неизвестном M используют правило Рунге:

$$|I_n - I_{2n}| \approx \frac{1}{15} |I_{2n} - I_{4n}| < \varepsilon_2,$$

что обеспечивает контроль над ошибками и гарантирует, что итоговая погрешность не превысит 10^{-4}, учитывая возможное накопление ошибок.

3. Результаты экспериментов

Используя написанную программу вычислим координаты точек пересечения кривых и составим таблицу а также отобразим это на графике (рис. 2)

Кривые	x	y
1 и 2	0.4482	2.3445
2 и 3	-0.1529	0.5414
1 и 3	-1.8211	5.5909

Таблица 1: Координаты точек пересечения

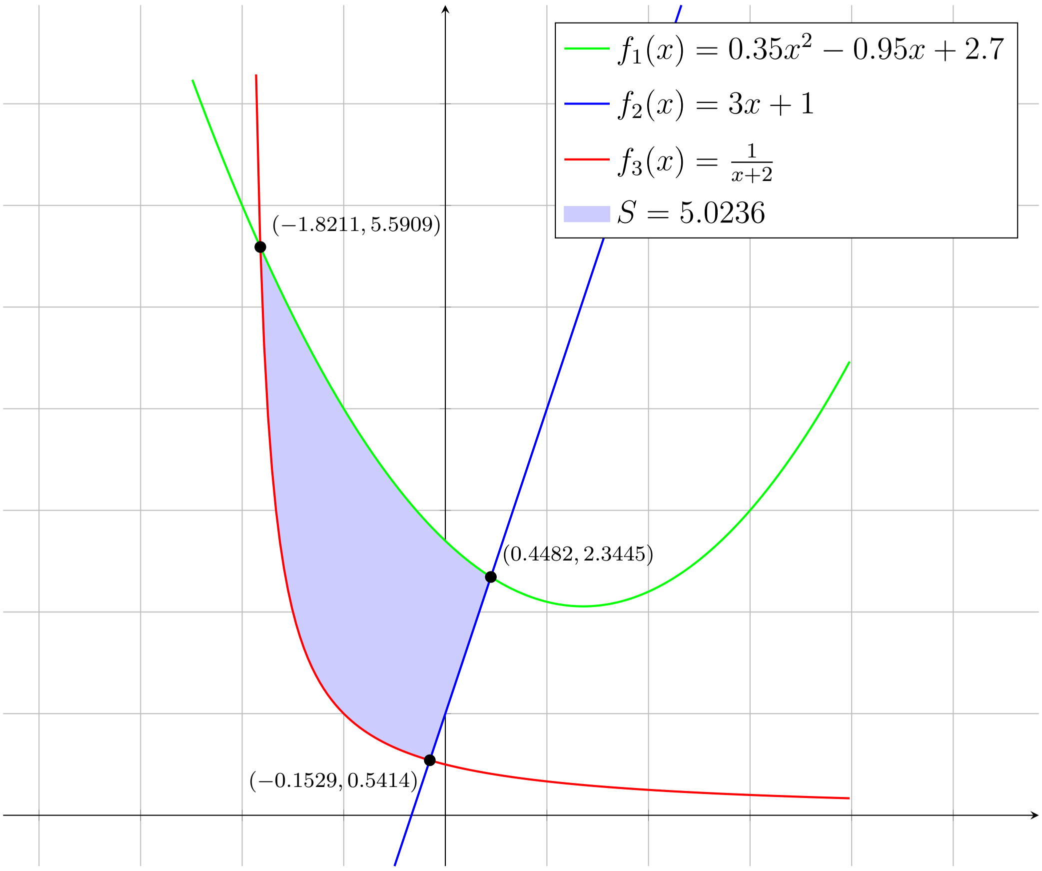


Рис. 2: Плоская фигура, ограниченная графиками заданных уравнений

4. Структура программы и спецификация функций

В данном разделе описывается архитектура программы, её компоненты и их взаимодействие. Программа разделена на несколько модулей, каждый из которых отвечает за определённый аспект функциональности.

4..1 Общая архитектура

Программа состоит из следующих модулей:

- Заголовочный файл с объявлениями (`functions.h`)
- Модуль для основной сборки программы (`main.c`)
- Модуль с реализованными численными методами (`compute.c`)
- Модуль для сборки программы в режиме тестирования (`test_mode.c`)
- Ассемблерный модуль с реализацией подсчета значений функций из задания и их производных в точке (`func.asm`)

4..2 Детальное описание файлов проекта

4..2.1 compute.c

Назначение: Реализация численных методов и тестирования.

- `test_root()`: для тестирования функции root
  - Автоматически определяет смещение для доступа к производным через `funcs[0](0)`
  - Формат вывода: абсолютная и относительная погрешность
- `test_integral()`: для тестирования функции itegral
  - Сравнивает результат с эталоном с точностью до 1e-4
- `root()` функция реализующая нахождение корня с помощью комбинированного метода
- `integral()` реализует адаптивный Симпсон для подсчет инграла. Особенности функции:
  - Начальное разбиение: $n = 2$
  - Коэффициент Рунге: $p = 1/15$
  - Максимальное разбиение: 5×10^6 (для защиты от бесконечного цикла)

4..2.2 functions.h

- глобальные переменные:
  - массив `funcs`:
    - В основном режиме: 7 элементов $[idx, f1, f2, f3, df1, df2, df3]$
    - В тестовом режиме: 9 элементов $[idx, test\_f1-4, test\_df1-4]$
  - `EPSILON`
  - `root_iteration` — для подсчета числа итераций в функции root
- прототипы функций

4..2.3 test_mode.c

- тестовые функции
- `main()` — вывод результатов тестирования

4..2.4 main.c

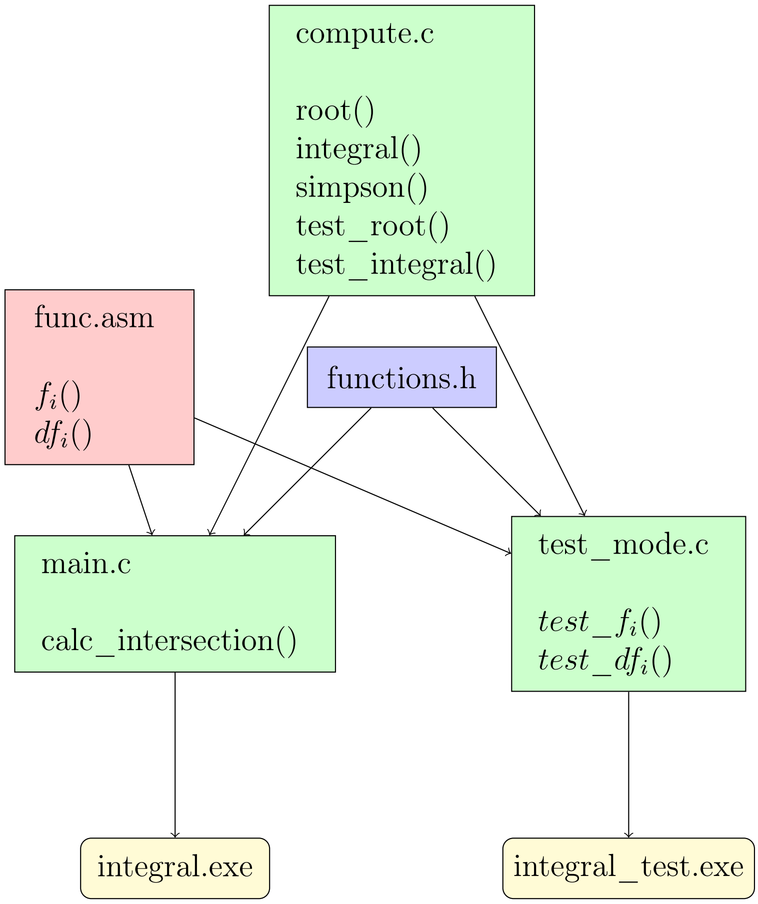
- `calc_intersection()` — функция для подсчета точек пересечения функций в основном режиме работы
- `main()` — основная функция, отвечающая за работу программы, поддерживающая работу с командной строкой

4..2.5 func.asm

- функции для подсчета значения функций $f1$, $f2$, $f3$ в точке
- функции для подсчета значения производных функций $f1$, $f2$, $f3$ в точке

4..2.6 Графическое представление структуры

Структуру программы и взаимодействие её компонентов можно представить в виде следующей диаграммы:


compute.c
root()
integral()
simpson()
test_root()
test_integral()
func.asm
$f_i()$
$df_i()$
functions.h
main.c
calc_intersection()
test_mode.c
$test\_f_i()$
$test\_df_i()$
integral.exe
integral_test.exe

5. Сборка программы (Make-файл)

Make-файл проекта управляет процессом компиляции, определяя используемые компиляторы, флаги и зависимости между модулями. Ниже приведено описание его ключевых компонентов с учетом структуры вашего Make-файла.

5..1 Компиляторы и флаги

В начале Make-файла задаются используемые компиляторы и их параметры:

- `CC = gcc -m32 -no-pie -fno-pie` — компилятор C с опциями:
  - `-m32` — компиляция в 32-битный код для совместимости с ассемблерным модулем;
  - `-no-pie`, `-fno-pie` — отключение поддержки позиционно-независимых исполняемых файлов (PIE), что важно для корректной линковки с ассемблером.
- `NASM = nasm` — ассемблер NASM для сборки ассемблерных файлов.

5..1.1 Флаги компиляции C-кода

Используются несколько групп флагов:

- `CFLAGS_COMMON` — включает широкий набор предупреждений компилятора (`-Wall`, `-Werror`, `-Wformat-security` и др.), что повышает качество и безопасность кода.
- `CFLAGS_SANITIZE` — включает санитайзеры для выявления неопределенного поведения (`-fsanitize=undefined`, `-fsanitize-undefined-trap-on-error`).
- `CFLAGS` — объединяет общие флаги и санитайзеры.
- `CFLAGS_TEST` — аналогичен `CFLAGS_COMMON`, но добавляет определение `TEST_MODE` для компиляции тестов.
- `-g` — добавление отладочной информации.
- `-std=gnu99` — стандарт языка C.
- `-O2` — оптимизация второго уровня.

5..1.2 Флаги NASM

- `NASMFLAGS = -fwin32` — сборка ассемблерного кода в формате 32-бит Windows-объектных файлов.

5..2 Цели сборки

Make-файл определяет следующие основные цели:

- `all` — сборка основной программы и тестовой программы, а затем удаление объектных файлов.
- `main` — сборка основной программы (`integral`).
- `test` — сборка и запуск тестовой программы (`integral_test`).
- `clean` — удаление всех объектных файлов (`*.o`).

5..3 Правила сборки

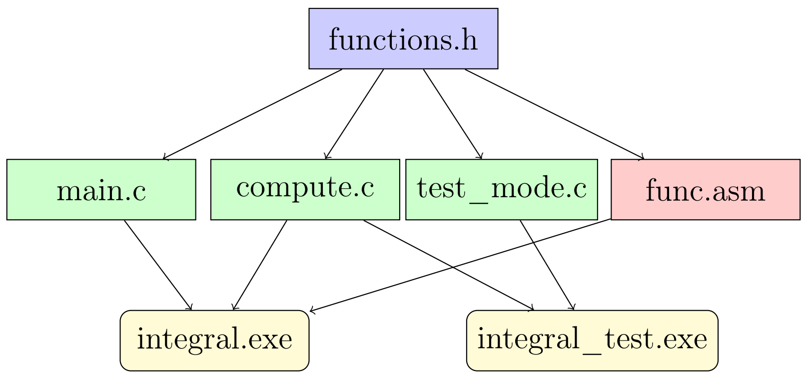
- `integral`: собирается из `main.o`, `compute.o`, `func.o` с помощью `CC` и `CFLAGS`, линковка с математической библиотекой (`-lm`).
- `integral_test`: собирается из `test_mode.o`, `compute.o` с использованием `CFLAGS_TEST`.
- `func.o`: собирается из `func.asm` с помощью NASM.

5..4 Особенности

- Использование строгих флагов компиляции и санитайзеров повышает надежность и безопасность кода.
- Автоматизация запуска тестов через цель `test`.
- Очистка промежуточных файлов после сборки основной и тестовой программ.

5..5 Зависимости модулей

Основные зависимости между модулями можно представить в виде следующей диаграммы:



Где:

- `functions.h` — заголовочный файл с объявлениями функций:
  - Объявления основных функций: `root`, `integral`
  - Объявления функций `f1`, `f2`, `f3` и их производных
  - Определение типа `func` как указателя на функцию
  - Константа `EPSILON` для точности вычислений
- `main.c` — основной модуль программы:
  - `main` — точка входа, обработка аргументов командной строки
  - `calc_intersection` — вычисление точек пересечения кривых
  - `test_root` — тестирование поиска корней
  - `test_integral` — тестирование вычисления интегралов
- `compute.c` — модуль с реализацией численных методов:
  - `root` — комбинированный метод поиска корня уравнения
  - `integral` — вычисление определённого интеграла
  - `simpson` — метод Симпсона для численного интегрирования
  - Глобальная переменная `root_iterations` для подсчёта итераций
- `test_mode.c` — модуль для тестирования:
  - Тестовые функции: `test_f1`, `test_f2`, `test_f3`, `test_f4`
  - Их производные: `test_df1`, `test_df2`, `test_df3`, `test_df4`
- `func.asm` — ассемблерный модуль:
  - Основные функции: `f1`, `f2`, `f3`
  - Первые производные: `df1`, `df2`, `df3`
- `integral.exe` — основная исполняемая программа
- `integral_test.exe` — тестовая программа

5..6 Особенности сборки

При сборке тестового режима добавляется специальный флаг `-DTEST_MODE`, который включает тестовые функции в программу. Это позволяет иметь единую кодовую базу как для основной программы, так и для тестов.

6. Отладка и тестирование программы

Для проверки корректности работы программы рассмотрим три тестовые функции, для которых аналитически вычисляются корни и определённые интегралы на заданных отрезках:

$$f_1(x) = x^2 + 5x + 6 \tag{7}$$
$$f_2(x) = \sin x \tag{8}$$
$$f_3(x) = x^3 - 1 \tag{9}$$

Для каждой функции найдём точки пересечения с осью Ox (т.е. нули функции), а также аналитически вычислим определённый интеграл на соответствующем интервале.

6..1 Функция $f_1(x) = x^2 + 5x + 6$

Решим уравнение:

$$x^2 + 5x + 6 = (x+2)(x+3) = 0 \Rightarrow x = -2, -3$$

Вычислим определённый интеграл на отрезке $[-4, -2]$:

$$\int_{-4}^{-2} (x^2 + 5x + 6)\, dx = \left(\frac{x^3}{3} + \frac{5x^2}{2} + 6x \right) \Bigg|_{-4}^{-2}$$

Подставим значения:

$$= \left(-\frac{8}{3} + 10 - 12 \right) - \left(-\frac{64}{3} + 40 - 24 \right) = \frac{2}{3} \approx 0.6667$$

6..2 Функция $f_2(x) = \sin x$

Нули функции определяются из уравнения:

$$\sin x = 0 \Rightarrow x = \pi n, \quad n \in \mathbb{Z}$$

Вычислим интеграл на отрезке $[1, 2]$:

$$\int_0^{\pi} \sin x\, dx = -\cos x \Big|_0^{\pi} = -(-1) - (-1) = 2$$

6..3 Функция $f_3(x) = x^3 - 1$

Нуль функции:

$$x^3 - 1 = 0 \Rightarrow x = 1$$

Вычислим интеграл на отрезке $[0, 2]$:

$$\int_0^2 (x^3 - 1)\, dx = \left(\frac{x^4}{4} - x \right) \Bigg|_0^2 = \left(\frac{16}{4} - 2 \right) - \left(\frac{0}{4} - 0 \right) = (4 - 2) - 0 = 2$$

6..4 Выбор отрезков для поиска корней

Для каждой тестовой функции приведём аналитическое выражение производной, определим её знак на соответствующем отрезке и укажем используемый отрезок поиска корня.

- $f_1(x) = x^2 + 5x + 6$
  Производная: $f_1'(x) = 2x + 5$
  На отрезке $[-4, -3]$:
  $f_1'(-4) = 2 \cdot (-4) + 5 = -3$
  $f_1'(-3) = 2 \cdot (-3) + 5 = -1$
  Производная отрицательна. На отрезке $[-2.4, -1.5]$:
  $f_1'(-2.4) = 2 \cdot (-2.4) + 5 = 0.2$
  $f_1'(-1.5) = 2 \cdot (-1.5) + 5 = 2.0$
  Производная положительна. На каждом подотрезке знак не меняется.

- $f_2(x) = \sin x$
  Производная: $f_2'(x) = \cos x$
  На отрезке $[3, 4]$:
  $f_2'(3) = \cos 3 \approx -0.990$
  $f_2'(4) = \cos 4 \approx -0.654$
  Производная отрицательна на всём отрезке.

- $f_3(x) = x^3 - 1$
  Производная: $f_3'(x) = 3x^2$
  На отрезке $[0.5, 1.5]$:
  $f_3'(0.5) = 3 \cdot (0.5)^2 = 0.75$
  $f_3'(1.5) = 3 \cdot (1.5)^2 = 6.75$
  Производная строго положительна.

Используемые отрезки поиска корней:

- Для $f_1(x)$: $[-4, -3]$ (корень $x = -3$), $[-2.5, -1.5]$ (корень $x = -2$)
- Для $f_2(x)$: $[3, 4]$ (корень $x = \pi \approx 3.1416$)
- Для $f_3(x)$: $[0.5, 1.5]$ (корень $x = 1$)

Таким образом, для каждого теста на выбранном отрезке производная не меняет знак, что соответствует условиям применения комбинированного метода.

6..5 Выводы

Сравним результаты полученные в программе и теоретически и сравним их, записав в таблицу.

Таблица 2: Результаты теоретических расчётов для тестирования

Функция	**Корни (теор.)**	**Корни (прог.)**	**Интеграл (теор.)**	**Интеграл (прог.)**
x^2+5x+6	$-3,-2$	$-3,-2$	$\frac{2}{3}$	0.6667
$\sin x$	$\pi n, n \in \mathbb{Z}$	3.1416	2	2
x^3-1	1	1	2	2

Функции *root* и *integral* работают корректно и выдают требуемую точность.

7. Анализ допущенных ошибок

- отсутствие команды `finit` в начале ассемблерных функций для вычисления значений и производных заданных функций.
- доработка функции integral, чтобы она учитывала ранее подсчитанные значения
- отсутсвие прототипов функций
- использование вторых производный в функции root

Список литературы

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