ПНИПУ

Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Лабораторная работа №2. Численные методы решения уравнений.

Вариант 23.

Выполнил студент группы РИС-23-3Б

Буковский Денис Владимирович

Проверила доцент кафедры ИТАС О.А. Полякова

2023

1. Постановка задачи.

1. Написать функцию с умалчиваемыми параметрами в соответствии с вариантом, продемонстрировать различные способы вызова функции:

* с параметрами заданными явно,
* с опущенными параметрами
* часть параметров задана явно, а часть опущена.

Задача: написать функцию для вывода фамилии, имени и отчества

2. Написать функцию с переменным числом параметров в соответствии с вариантом, продемонстрировать вызов функции с различным числом параметров.

Задача: найти количество четных элементов в списке параметров.

3. Написать перегруженные функции в соответствии с вариантом. Написать демонстрационную программу для вызова этих функций.

Задача: поиск заданного элемента в отсортированном массиве

4. Написать шаблон функций вместо перегруженных функций из задания 3. Написать демонстрационную программу для вызова этих функций. списка параметров

5. Решить уравнение указанным в варианте методом. Уравнение передать в функцию как параметр с помощью указателя.

Задача:

метод Ньютона

Метод итераций

ex - e-x -2 = 0

Отрезок, содержащий корень: [0;1]

Точное значение: 0,8814

1. Словесный алгоритм.

1. Для решения этой задачи воспользуемся возможностью C++ принимать опущенные аргументы. В определении функции укажем 3 принимаемых аргумента: surname, name и last\_name. Зададим им значения по умолчанию: “not defined”, “” и “” соответственно.

Затем просто выведем их, один за другим, используя cout.

Задача решена.

2. Для решения следующей задачи определим функцию FindElements, в которой используем возможность C++ принимать неявные аргументы. В качестве первого и единственного явного аргумента будем передавать ей количество чисел, которые будем передавать после через запятую – k. Затем через запятую пишем многоточие – это и позволит нам принять k аргументов.

В теле функции определим счётчик count = 0 и указатель p, хранящий в себе адрес k. Напишем цикл, в котором переберём k ячеек памяти после самой ячейки k, на каждом шаге меняя указатель p используя конструкцию ++p.

Напишем условие, по которому каждое следующее полученное число проверяется на чётность, и если оно чётное, то увеличим счётчик на 1.

Задача решена.

3. Для решения этой задачи с наибольшей эффективностью, воспользуемся тем, что он отсортирован, и используем алгоритм бинарного поиска.

Определим функцию binSearch, принимающую в качестве аргументов массив, произвольный элемент (цель поиска) и длину ранее переданного массива.

В теле функции определим целочисленные переменные mid, low = 0 и high = len – 1 - границы действия алгоритма, которые мы будем сужать по мере выполнения программы.

Запустим цикл, в котором мы будем менять значение mid = (low + high)/2, с помощью которого мы будем получать элементы массива. Напишем условия, по которым: если элемент массива с индексом mid будет равен значению цели поиска, то мы вернём цель поиска, если он будет больше цели поиска, то мы изменим high на mid-1, иначе low на mid+1.

Если пройдя все шаги цикла элемент не будет найден, то вернём значение NULL (по сути 0).

Перегрузим эту функцию просто меняя типы данных в ней, например int arr на float arr, int item на char item и т.д.

Задача решена.

4. Для решения этой задачи используем то, что у нас уже есть с прошлого раза.

Определим функцию binSearchTemplate, а затем просто перед определением функции зададим шаблон с классами <class Arr, class Item, class I>

Подставим эти классы вместо типов в функции: Item binSearchTemplate(Arr arr, Item item, I len){…}. Теперь нам даже не нужно перегружать её, компилятор всё сделал за нас!

5. Для решения последней задачи воспользуемся методом Ньютона и итерационным методом. Эти методы позволяют найти решение уравнения x с определённой точностью на заданном интервале [a, b] для некоторой функции f(x).

Начнём с метода Ньютона. Подключим библиотеку cmath для использования математических функций.

Далее, определим и запишем данную нам функцию как func с аргументом double x. В первой же строке вернём выражение exp(x)-exp(-x)-2.0. Функция готова. Затем, нам нужна производная от этой функции. Определим функцию diff1 с аргументом double x0 и вернём выражение exp(x0) + exp(-x0) (производная). Далее по той же схеме определим вторую производную – diff2 и вернём exp(x0) – exp(-x0).

Перейдём к написанию самого метода. В качестве аргументов этой функции примем ссылку на функцию, границы a и b. Определим переменную eps вещественного типа двойной точности для задания точности алгоритма, сразу же присвоим ей значение 10-6. Далее – xn и xn-1 их пока определять не будем. Проверим, будет ли правая граница b > 0, если да – присвоим xn-1 значение b, в противном случае – a. Сразу после условия определим xn по формуле xn = xn-1 – (f(xn-1)/f’(xn-1)).

Напишем цикл. Условием будет проверка точности корня с заданной в переменной eps точности, если она будет больше – прерываем выполнение цикла. В теле цикла пишем: xn-1­ = xn и уже знакомое нам выражение xn = xn-1 – (f(xn-1)/f’(xn-1)).

После завершения выполнения вернём xn.

Переходим к итерационному методу. Определяем новую функцию на основе старой, прологарифмировав выражение и приведя его к виду x = f(x). Получаем функцию xfunc с вещественным аргументом double x, которая возвращает ln 2-x.

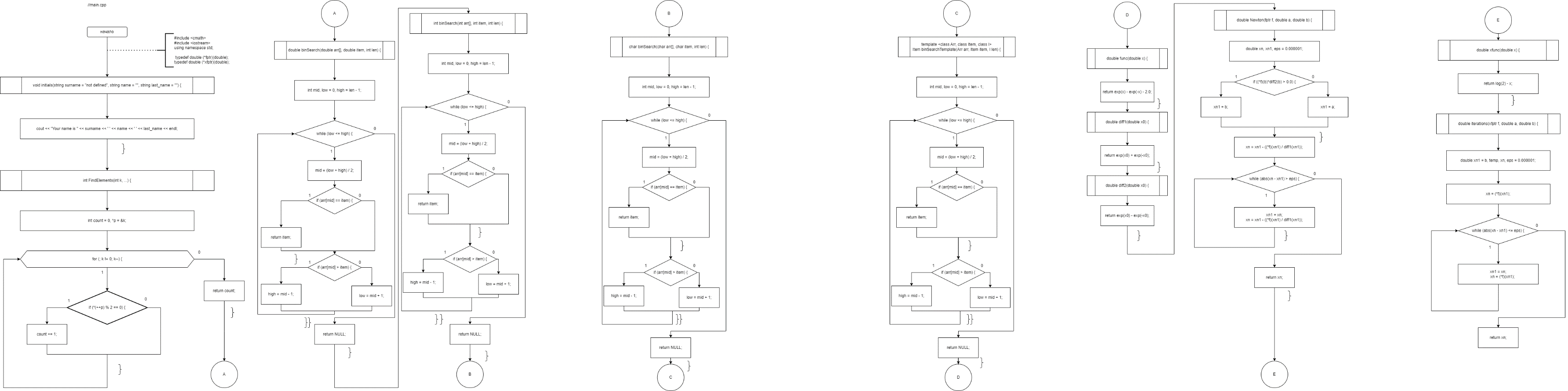
Наконец, пишем итерационный метод в функции iterations. В качестве аргументов будем принимать ссылку на функцию f, вещественные числа двойной точности a и b.

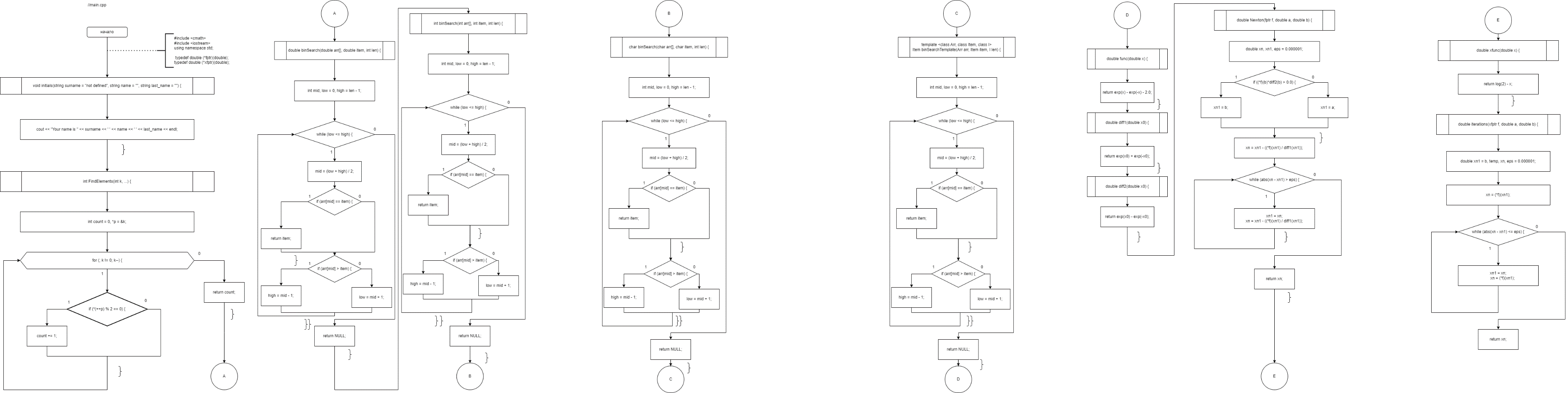
Определим вещественные переменные xn-1 = b, temp, xn, eps = 10-6. Присвоим переменной xn значение функции в точке xn-1.

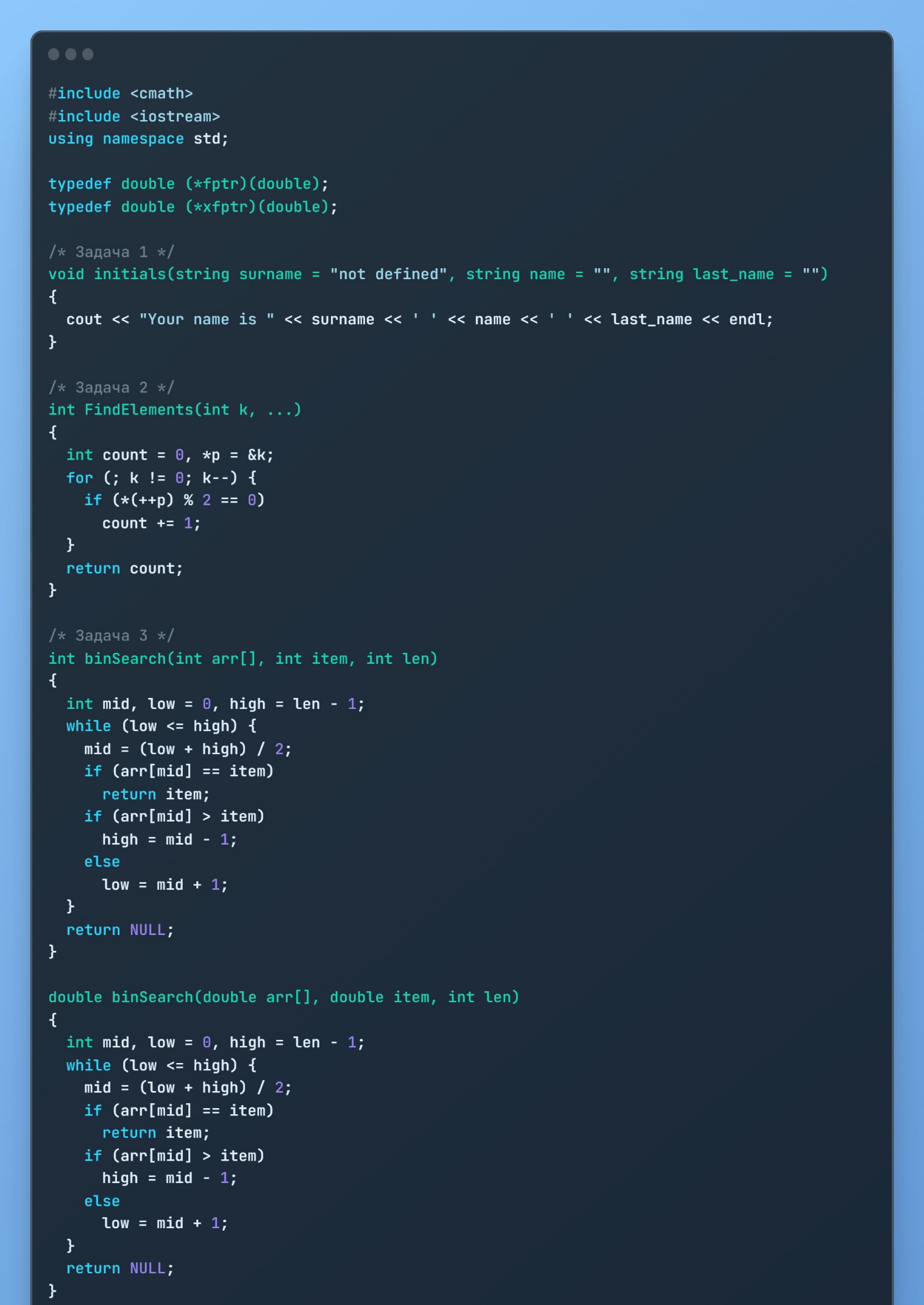
Создадим цикл с условием abs(xn-xn-1) <= eps (то есть пока точность x не превысит заданную). В теле цикла будем присваивать переменной xn-1 значение xn и вычислять следующее значение xn, подставляя его в качестве аргумента функции. По завершении цикла вернём значение xn.

Для всех функций напишем тесты и вызовем их в функции main().

1. Блок-схемы всех функций представлены ниже





4. Код, тесты, main().

5. Результаты выполнения программы.

