*Задания для курсовой работы по курсу **«Технологии программирования»***

Написать программу расчёта заданного вычислительного метода на языке, C#, C/C++, VB.NET, Python или других по согласованию с преподавателем.

Задание 1 – основная часть (обязательная).

Задание 2 – графическая часть.

Задание 3 – файлы последовательного доступа

Задание 4 – файлы произвольного доступа или файлы настроек.

ЗАДАНИЕ 1:

Задание подразумевает самостоятельную работу программиста над программой по приведённым ниже вариантам.

ВНИМАНИЕ! Использование библиотечных функций запрещено за исключением 25-26, 49 вариантов. Включение библиотечных функций допустимо в целях проверки рассчитанных алгоритмическим способом значений и регулируется с помощью настроек программы. Использование библиотек также разрешено для построения диаграмм.

Список вариантов:

1. Программа, отделяющая корни нелинейных уравнений на интервале [a,b]. Требуется определить количество корней и локализовать интервалы их нахождения с заданной пользователем точностью.
2. Решение нелинейного уравнения методом половинного деления.
3. Решение нелинейного уравнения методом секущих.
4. Решение нелинейного уравнения методом Стеффенсена.
5. Решение нелинейного уравнения методом простой итерации. Предусмотреть действия проверки сходимости.
6. Решение систем линейных уравнений с трёхдиагональной матрицей методом прогонки. Проверка на трёхдиагональность, корректность и устойчивость. Проверить точность расчётов.
7. Решение систем линейных уравнений методом Гаусса. Проверить точность расчётов.
8. Решение систем линейных уравнений методами простой итерации и Зейделя. Вычисление определителя (с использованием функции листа Excel или библиотеки). Проверить точность расчётов.
9. В заданной пользователем папке и её подпапках произвести поиск всех файлов и папок по заданной пользователем маске (включая её инверсию[1]). Подсчитать количество файлов каждого типа. Переместить или скопировать выбранные по маске файлы/папки в новую папку[2].
10. Решение систем линейных уравнений с помощью LU-разложения.
11. Нахождение минимума (максимума) функции методом золотого сечения.
12. Нахождение минимума (максимума) функции методом половинного деления[3].
13. Нахождение минимума (максимума) функции методом Фибоначчи.
14. Нахождение минимума (максимума) функции методом квадратичной интерполяции.
15. Аппроксимировать данные по методу наименьших квадратов алгебраическими многочленами.
16. Интерполяция полиномом Лагранжа. Оценить точность интерполяции для тестовой функции, заданной аналитически в зависимости от количества точек (10 – 50 точек. Для тестовой задачи можно использовать точки, вычисленные для какой-либо типовой функции).
17. Вычисление интегралов методами прямоугольников (левых и правых), трапеций и Симпсона (парабол). Сравнить точность расчёта по этим методам и определить влияние шага интегрирования.
18. Вычисление интегралов методами Чебышева и Гаусса. Сравнить точность расчёта по методам для разного количества узлов (2 – 5).
19. Решение задачи Коши для обыкновенных дифференциальных уравнений явными методами Эйлера, Эйлера-Коши, Рунге-Кутта 4-го порядка. Сравнить точность методов. Оценить влияние величины шага на точность расчётов.
20. Создать калькулятор, реализующий операции с комплексными числами (сложение, умножение, вычитание, деление, вычисление модуля).

---

[1] Т.е. если задана маска «*.txt», то в её инверсию войдут все файлы с любыми расширениями кроме .txt.

[2] Иерархию папок при этом сохранять не нужно. В случае совпадения имён папок/файлов переименовывать такие дубликаты, добавляя к ним суффиксы с порядковым номером.

[3] Не путать с методом для решения нелинейных уравнений!

21. Создание блока графического отображения информации. Блок должен обеспечивать: графическое отображение точечной диаграммы для любого набора данных и содержать следующие возможности по заданию форматирования диаграммы:
    a. Типы маркеров, каждого ряда данных, а также цвета их обводки и заливки
    b. Цвета и типы линии для каждого ряда данных
    c. Наличие, положение и форматирование легенды по выбору пользователя
    d. Наличие, цену делений и формат основных и вспомогательных осей
    e. Формат засечек на осях.
    f. Формат основных и вспомогательных линий сетки диаграммы
    g. Цвет фона области построения диаграммы.
    h. Возможность сохранения изображения диаграммы в файл.
    i. Форматы названий диаграммы, осей и меток.

    Вывести на форму все необходимые инструменты для управления данными параметрами, сохранения и считывания настроек в файл и из него.
22. Создание блока графического отображения информации. Блок должен обеспечивать: графическое отображение поверхностной диаграммы для любого набора данных и содержать следующие возможности по заданию форматирования диаграммы:
    a. Переключение между типами поверхностных диаграмм
    b. Наличие, положение и форматирование легенды по выбору пользователя
    c. Формат засечек на осях
    d. Форматы названий диаграммы и осей
    e. Наличие, цену делений и формат осей
    f. Угол поворота диаграммы и перспективу
    g. Диапазон значений по осям
    h. Цвета уровней (включая возможности градиентной и текстурной заливок)

Вывести на форму все необходимые инструменты для управления данными параметрами, сохранения и считывания настроек в файл и из него.

23. Создать модуль, реализующий основные матричные операции (сложение, умножение, транспонирование, обращение, вычисление определителя) с применением функций рабочего листа Excel или математических библиотек.
24. Создать модуль, который реализует разбиение одномерного массива на набор двухмерных массивов произвольной размерности M×N. Конечные двумерные массивы должны иметь одинаковые размеры. Последний массив получившегося набора может оставаться недозаполненным. Сделать возможным заполнение матриц, как по строкам, так и по столбцам в зависимости от выбора пользователя, а также заполнение «змейкой».
25. Создать программу, которая производит обход дерева папок, начиная с указанной пользователем, и выполняет следующие действия в этой папке и всех подпапках всех уровней:
    a. Собирает данные о размерах файлов.
    b. Проверяет наличие заданного пользователем файлов «_index.txt» и «errors.gif» во всех подпапках и удаляет последний при наличии пользовательского разрешения.
    c. Проверяет наличие пустых файлов «_index.txt» (с длиной 0) в каждой папке.
    d. Проверяет файлы .jpg/.JPG на наличие двух концевых байтов заданного вида (0xFF 0xD9).
26. Создать программу, которая производит обход дерева папок, начиная с указанной пользователем, и выполняет следующие действия в этой папке и всех подпапках всех уровней:
    a. Собирает данные об встречающихся типах файлов.
    b. Проверяет наличие пустых файлов (с длиной 0) и пустых папок.
    c. Проверяет наличие файлов с расширением «.jpg» или «.JPG».
    d. Проверяет наличие заданного пользователем имени файла (например, 00000002.jpg (или JPG)) во всех подпапках.
27. Решение нелинейного уравнения методом Ньютона с вариантами аппроксимации производной конечными разностями.
28. Сортировка числовых и текстовых значений методами всплывающего пузы**рька** (Bubble sort) и ш**ейкерной сортировкой (**также известна как сортировка перемешиванием и коктейльная сортировка).
29. Сортировка числовых и текстовых значений методами расчески (Comb sort) и Шелла (Shellsort)
30. Сортировка числовых и текстовых значений методами сортировки выбором (Selection sort) и пирамидальной сортировкой (Heapsort)

31. Сортировка числовых и текстовых значений методами быстрой сортировки (Quicksort) и сортировки слиянием (Merge sort)
32. В массиве 10×10 клеток разместите случайным образом корабли для игры «морской бой». Реализовать несколько стратегий их размещения. Реализовать алгоритм игры «морской бой» с 2 игроками и против программы.
33. –
34. Реализовать алгоритм игры в «крестики-нолики» на поле 3×3 клетки.
35. Реализовать алгоритм игры «Кости». Играющий называет любое число в диапазоне от 2 до 12 и ставку, которую он делает в этот ход. Программа с помощью генератора псевдослучайных чисел дважды выбирает числа от 1 до 6. Если сумма выпавших цифр меньше 7 и играющий задумал число меньшее 7, он выигрывает сделанную ставку. Если сумма выпавших цифр больше 7 и играющий задумал число большее 7, он также выигрывает сделанную ставку. Если играющий угадал сумму цифр, он получает в четыре раза больше очков, чем сделанная ставка. Ставка проиграна, если не имеет место ни одна из описанных ситуаций. В начальный момент у играющего 100 очков. Если количество очков становится меньше 0, то игрок проигрывает. Если больше заданного в начале игры максимального значения, то выигрывает.
36. Реализовать алгоритм игры «Ипподром». Играющий выбирает одну из заданного количества лошадей (2-5), состязающихся на бегах, и выигрывает, если его лошадь приходит первой. Скорость передвижения лошадей на разных этапах (не менее 10) выбирается программой с помощью генератора псевдослучайных чисел.
37. Реализовать алгоритм игры «поле чудес». Программа выбирает слово из произвольной базы и рисует в шаблоне на экране столько прочерков, сколько букв в этом слове. Отгадать, какое слово загадано программой. В каждый ход играющий указывает одну букву. Если названа буква, входящая в состав слова, она подставляется вместо соответствующего прочерка. В противном случае играющий теряет 1 очко. В начальный момент у играющего 15 очков. В любой момент он может заполнить всё слово, но если ошибается, то проигрывает.
38. Составить программу для тренировки памяти. Программа должна высветить на экране несколько точек, играющий – указать, в каком порядке эти точки были высвечены. Координаты точек выбираются в программе с помощью генератора псевдослучайных чисел.
39. Составить программу обучения работе с клавиатурой. Программа должна выдавать на экран буквы, цифры, слова и фразы, которые следует набрать на клавиатуре. Контролировать количество правильно выполненных заданий и время их выполнения. Регулировать тип и сложность заданий по уровням – 1. Буквы (числа, спецсимволы)/2. Слова/3. Фразы, а также выбор языка(ов) вода текста. Учесть, что буквы (а, о, е и др.) на разных языках могут выглядеть одинаково.
40. Создать турнирную таблицу. Каждая запись отражает счёт очередного проведённого матча. За победу команде даётся 3 очка, за ничью – 1. Обеспечить ранжирование команд в зависимости от количества набранных очков. При их равенстве первой будет та команда, у которой лучше разница забитых и пропущенных мячей. Если и эта разность одинакова, тогда лучшей будет та, у которой больше мячей забито.
41. Реализовать алгоритмы вычисления статистических функций для табличных значений *x* и *f(x)*: среднее значение, средневзвешенное значение, среднегармоническое, среднегармоническое взвешенное, среднее геометрическое, среднеквадратическое, мода, медиана, дисперсия.
42. Реализовать алгоритмы пересчёта концентраций растворов между массовой и мольной долями, титром и моляльностью.
43. Реализовать алгоритм игры «Жизнь» на поле 100×100 клеток.
44. Реализовать алгоритм клеточного автомата «Муравей Лэнгтона» на поле 100×100 клеток со случайным заполнением: Если в клетке единица – записать в нее ноль, повернуться на 90° влево, сделать шаг вперед на следующую клетку. Если в клетке ноль – записать в нее единицу, повернуться на 90° вправо, сделать шаг вперед на следующую клетку.
45. Задан набор химических реакций в виде $\sum_{i=0}^{N} a_i X_i = \sum_{i=0}^{M} b_i Y_i$, где X_i и Y_i заданные в текстовом виде вещества (например «H2O», «CO2» и т.п.), а a_i и b_i – стехиометрические коэффициенты. Причём N, и X_i заданы программно, а a_i, b_i и Y_i вводятся пользователем. Требуется определить количество несоответствий введённых пользователем a_i , b_i и Y_i их эталонным количествам и значениям, заданным программно. Наборы реакций требуется хранить так, чтобы их можно было динамически менять в зависимости от темы работы.

Примечание: Все пары $b_i Y_i$ могут вводиться пользователем в произвольном порядке и при произвольном количестве M>0. Регистр Y_i имеет значение.

46. Создать программный модуль, который обеспечит арифметические операции с целыми числами в 2, 8, 10, 16-чных системах, а также взаимную конвертацию из одной системы в другую.
47. Провести конверсию 8-битного сигнала в 2,4,6-битный и обратно Пример такой конверсии из 8 битного сигнала в 6-битный: 01101111 11000110 00010010… →011011 111100 0110000 010010….
48. Провести конверсию 8-битного сигнала в 3,5,7-битный и обратно. Пример такой конверсии из 8 битного сигнала в 3-битный: 01101111 11000110 00010010… →011 011 111 100 011 000 010 010…
49. Реализовать алгоритмы последовательного (линейного) и бинарного поиска для массивов данных, состоящих из строк. Для упорядочивания значений для бинарного поиска допустимо задействовать библиотечные функции. Сравнить их скорость работы.
50. Реализовать алгоритмы поиска на основе хэша (Hash-Based Search).

ЗАДАНИЕ 2:

В задании требуется провести ряд операций по построению диаграмм (в некоторых вариантах они могут быть напрямую не связаны с первым заданием). Требуется задать не меньше **трёх** вариантов каждого параметра форматирования диаграмм. (см. также методические указания ниже)

Список вариантов:

1. Построить точечную диаграмму функции на заданном интервале с форматированием легенды (наличие/отсутствие, шрифт).
2. Построить точечную диаграмму функции на заданном интервале с форматированием рядов данных (вид маркеров, толщина и цвет линий), и изменением порядка рядов данных. Обеспечить построение диаграмм по нескольким рядам данных.
3. Построить точечную диаграмму функции на заданном интервале с форматированием (линии сетки (наличие/отсутствие), пределы по оси Y). Обеспечить построение диаграмм по нескольким рядам данных.
4. Построить точечную диаграмму функции на заданном интервале с форматированием (Легенда, линии сетки, оси). Обеспечить построение диаграмм по нескольким рядам данных.
5. Построить точечную диаграмму функции на заданном интервале с форматированием (цвет и форма маркеров, цвета линий, тип диаграммы, линии сетки, отображение/скрытие легенды).
6. По имеющейся матрице построить поверхность функции с форматированием осей (цвет, засечки). Необходимые значения X и Y, требуемые для построения поверхностной диаграммы задать самостоятельно с нецелым шагом.
7. По имеющейся матрице построить поверхность функции с форматированием легенды (наличие/отсутствие, цвет текста). Необходимые значения X и Y, требуемые для построения поверхностной диаграммы задать самостоятельно с нецелым шагом.
8. По имеющейся матрице построить поверхность функции с форматированием заливки уровней (типы и цвета заливок). Необходимые значения X и Y, требуемые для построения поверхностной диаграммы задать самостоятельно с нецелым шагом.
9. Для найденных типов файлов построить диаграмму, показывающую распределение файлов по типам. Использовать не менее трёх типов диаграмм с форматированием осей и выбранных пользователем цветов линий/столбцов гистограммы.
10. По имеющейся матрице построить поверхность функции с форматированием сетки (наличие/отсутствие, варианты цвета)
11. Построить точечную диаграмму функции на заданном интервале с форматированием легенды (наличие/отсутствие, шрифт)
12. Построить точечную диаграмму функции на заданном интервале с форматированием (линии сетки, оси, цвет фона)
13. Построить точечную диаграмму функции на заданном интервале с форматированием (Пределы по оси Y, формат маркеров (вид, размер, заливка, контур)).
14. Построить точечную диаграмму функции на заданном интервале с форматированием (Форматы заголовков осей и графика, возможность сохранения графика в графический файл, формат линий).
15. Построить точечную диаграмму функции на заданном интервале с возможными линиями тренда.
16. Построить точечную диаграмму тестовой функции на заданном интервале с заданием погрешностей по X и Y.

17. Построить точечную диаграмму тестовой функции на заданном интервале.
18. Вычисление интегралов методами Чебышева и Гаусса. Сравнить точность расчёта по методам для разного количества узлов (2 – 5). Построить точечную диаграмму тестовой функции на заданном интервале с форматированием (Пределы по осям, цвет, тип и толщина линий).
19. Построить точечные диаграммы тестовых функций $f_i(x,y)$.
20. По логу операций построить гистограммы различного типа (по выбору пользователя) по действительным и мнимым частям использованных пользователем чисел.
21. Нет
22. Нет
23. По имеющейся матрице построить поверхность функции с форматированием цветовой палитры
24. По имеющейся одномерной матрице построить точечную диаграмму функции с форматированием маркеров и осей.
25. Построить диаграмму распределения всех найденных файлов по размерам (не меньше 10 диапазонов) с пользовательским переключением между возможными типами диаграмм (гистограмма, линейчатая, график) и форматированием цвета серий данных (столбцов или линий)
26. Построить диаграмму распределения всех найденных файлов по типам с пользовательским переключением между возможными типами диаграмм (гистограмма, линейчатая, график) и форматированием осей (толщина, цвет).
27. Построить точечную диаграмму функции на заданном интервале с форматированием легенды (наличие/отсутствие, цвет).
28. Построить гистограмму используемых чисел с форматированием оси значений (цвет, засечки)
29. Построить гистограмму используемых чисел с форматированием линий сетки, (цвет, тип).
30. Построить линейчатую диаграмму используемых чисел с форматированием оси категорий (формат числа, цвет)
31. Построить линейчатую диаграмму используемых чисел с форматированием рядов данных (заливка)
32. Построить поверхностные диаграммы игровых полей с цветовой кодировкой по уровням.
33. Построить поверхностные диаграммы игровых полей на плоскости с цветовой кодировкой по уровням.
34. Нет
35. Построить гистограмму используемых игроком и компьютером чисел с форматированием оси категорий (цвет, тип линии)
36. Вывести точечную диаграмму скоростей лошадей с форматированием линий данных.
37. Вывести статистику побед/поражений игрока в виде графика с форматированием сетки по обеим осям
38. Вывести статистику побед/поражений игрока в виде графика с форматированием маркеров данных.
39. Вывести данные в виде круговой диаграммы, показывающей процент правильных действий в разные моменты времени. Например, если разбить время, затраченное пользователем на выполнение задания, на 10 частей и для каждой части вывести процент верных действий пользователя.
40. Построить график движения задаваемых пользователем команд от 1-го до последнего тура.
41. Построить точечные диаграммы тестовых функций $f_i(x)$ с форматированием рядов данных (тип и цвет маркера и линий).
42. Построить гистограмму частоты вызовов по каждому типу концентрации с форматированием легенды (тип, начертание и цвет шрифта)
43. Нет
44. Построить динамическую поверхностную диаграмму поля клеток.
45. Вывести в виде гистограммы процент правильных ответов с форматированием оси категорий (цвет)
46. Вывести в виде линейчатой диаграммы значения частоты вызовов по каждому типу преобразований с форматированием легенды (цвет фона)
47. Гистограмма с форматированием осей (цвет, формат засечек) в координатах: номер байта – значение байта (в десятеричной системе)
48. График с форматированием линий (цвет, тип, наличие маркеров) в координатах: номер байта – значение байта (в десятеричной системе)
49. Вывести гистограмму времени выполнения для каждого типа поиска в зависимости от размерности массивов данных.

50. Вывести линейчатую диаграмму времени выполнения для каждого типа поиска в зависимости от размерности массивов данных.

ЗАДАНИЕ 3: Файлы последовательного доступа

Путь к файлу задаётся пользователем через графический пользовательский интерфейс. По умолчанию это путь к папке с исполнимым файлом.

Допустимо использовать любые форматы файлов – CSV, JSON, XML и пр.

Для случая большого объема выводимых данных можно использовать бинарные файлы. Однако, в этом случае необходим отдельный блок или программа для просмотра этих данных.

1. Запись всех значений вычисленных функций.
2. Запись исходных данных и результатов расчётов. Вывести в отдельный .TMP файл все значения функций, которые были вычислены в ходе выполнения программы.
3. Запись исходных данных и результатов расчётов. Вывести в отдельный .TMP файл все значения функций, которые были вычислены в ходе выполнения программы.
4. Запись исходных данных и результатов расчётов. Вывести в отдельный .TMP файл все значения функций, которые были вычислены в ходе выполнения программы.
5. Запись исходных данных и результатов расчётов. Вывести в отдельный .TMP файл все значения функций, которые были вычислены в ходе выполнения программы.
6. Предусмотреть вариант считывания исходных данных из выбираемого пользователем файла последовательного доступа. Запись результатов и исходных данных в итоговый файл.
7. Предусмотреть вариант считывания исходных данных из выбираемого пользователем файла последовательного доступа. Запись результатов и исходных данных в итоговый файл.
8. Предусмотреть вариант считывания исходных данных из выбираемого пользователем файла последовательного доступа. Запись результатов и исходных данных в итоговый файл.
9. Запись лога действий пользователя с перезаписью или дополнением этого файла.
10. Предусмотреть вариант считывания исходных данных из выбираемого пользователем файла последовательного доступа. Запись результатов и исходных данных в итоговый файл (включая L и U-матрицы).
11. Запись исходных данных и результатов расчётов. Вывести в отдельный файл все значения функций, которые были вычислены в ходе выполнения программы.
12. Запись всех значений вычисленных функций, а также исходных данных для расчётов.
13. Запись исходных данных и результатов расчётов. Вывести в отдельный файл все значения функций, которые были вычислены в ходе выполнения программы.
14. Запись исходных данных и результатов расчётов. Вывести в отдельный файл все значения функций, которые были вычислены в ходе выполнения программы.
15. Запись исходных данных и результатов расчётов.
16. Запись исходных данных и результатов расчётов.
17. Запись исходных данных и результатов расчётов. Вывести в отдельный файл все значения функций, которые были вычислены в ходе выполнения программы.
18. Запись исходных данных и результатов расчётов. Вывести в отдельный файл все значения функций, которые были вычислены в ходе выполнения программы.
19. Запись исходных данных и результатов расчётов. Для разных методов файлы должны отличаться по именам.
20. Запись лога операций.
21. Запись лога операций.
22. Запись лога операций.
23. Запись лога операций.
24. Запись исходных и итоговых матриц в отдельные файлы и их считывание на соответствующие листы книги Excel или элементы управления формы.
25. Протоколировать результаты работы.
26. Протоколировать результаты работы.
27. Запись исходных данных и результатов расчётов. Вывести в отдельный .TMP файл все значения функций, которые были вычислены в ходе выполнения программы.
28. Протоколировать результаты работы (количество перемещений элементов, время расчёта, конечное состояние массива данных, вид данных).

29. Протоколировать результаты работы (количество перемещений элементов, время расчёта, конечное состояние массива данных, вид данных).
30. Протоколировать результаты работы (количество перемещений элементов, время расчёта, конечное состояние массива данных, вид данных).
31. Протоколировать результаты работы (количество перемещений элементов, время расчёта, конечное состояние массива данных, вид данных).
32. Вывести результат размещения кораблей
33. Протоколировать ход игры. Стирать старый файл или дополнять его по выбору пользователя.
34. Протоколировать ход игры. Стирать старый файл или дополнять его по выбору пользователя.
35. Протоколировать ход игры. Стирать старый файл или дополнять его по выбору пользователя.
36. Протоколировать ход игры. Стирать старый файл или дополнять его по выбору пользователя.
37. Протоколировать ход игры. Стирать старый файл или дополнять его по выбору пользователя.
38. Протоколировать ход игры. Стирать старый файл или дополнять его по выбору пользователя.
39. Протоколировать ход игры. Стирать старый файл или дополнять его по выбору пользователя.
40. Сделать возможным считывание, редактирование и удаление данных из файла результатов.
41. Протоколировать ход расчётов.
42. Протоколировать ход расчётов.
43. Протоколировать ход игры в бинарный файл.
44. Протоколировать ход игры в бинарный файл.
45. Вывести итоговый результат в файл. В нём должна быть статистика, а также введенные уравнения.
46. Протоколировать результаты расчётов.
47. Протоколировать результаты расчётов.
48. Протоколировать результаты расчётов.
49. Протоколировать результаты (время расчёта, количество просмотренных элементов и т.п.)
50. Протоколировать результаты (время расчёта, количество просмотренных элементов и т.п.)

ЗАДАНИЕ 4: Файл настроек

Запись в файл всех пользовательских настроек и считывание их оттуда.

Алгоритм работы настроек включает в себя следующую стратегию:

- Перед закрытием программы все её настройки записываются в файл.
- При открытии программы, её настройки считываются из файла.
- Настройки могут записываться/считываться в произвольном порядке
- Если файл настроек не найден, то все настройки выставляются по умолчанию.
- Пользователь может сохранить свои настройки в файл и загрузить их оттуда при помощи соответствующих элементов управления или пунктов меню.
- Папка используется та же, что и для файлов в 3 задании.

Методические указания по выполнению работы:

к заданию 1

Исходные данные должны быть максимально гибкими. Т.е. пользователь может задавать исходные данные в максимально удобном виде. Возникающие при работе программы исключения должны быть обработаны.

к заданию 2

Термин *«Форматирование»* обозначает возможность вариации начертания и размеров шрифта, вариантов цвета, заливки, и пр. Например *«форматирование оси»* значит, что в соответствии с выбором пользователя, может изменяться тип, цвет и толщина линии, минимальные и максимальные значения, цена делений, вид засечек и другие настройки из свойств формата оси. Аналогично и с другими объектами.

к заданию 3

Хранение файлов данных можно по умолчанию осуществлять в папке проекта. В ином случае предусмотреть возможность выбора пользователем целевой папки. Формат файлов может быть любым. Для выходных данных в случаях, когда это не связано с хранением больших объёмов информации (матриц) предусмотреть вывод информации об исходных данных, необходимых для проведения расчёта.

Пример 1: Требуется решить систему уравнений. Пусть матрица и вектор правых частей задаются пользователем с помощью формы. Создаём 2 файла. Файл *input.dat* будет содержать всю инфор-

мацию о матрице и векторе правых частей. Файл *Output.dat* будет содержать искомый вектор неизвестных *x*, результаты проверки точности решения и информацию о файле исходных данных или же сами исходные данные. Настройки для построения диаграмм можно хранить в отдельном **.ini* файле.

Пример 2: Найти максимум функции *f(x)*. Исходные данные вводятся пользователем с формы Файл *input.dat* будет содержать данные по точке начального приближения, точности расчёта, методе расчёта и пр.

Файл *Output.dat* может содержать информацию об исходной точке, выбранном методе и точности расчёта, а так же найденном значении и значении функции в этой точке. При этом возможно реализовать вывод пути от начальной точки к точке оптимума в TMP-файл. Настройки для построения диаграмм можно хранить и в отдельном файле.

К заданию 4

Хранение и считывание структурированных данных удобно организовать с помощью переменной определенного класса или структуры. Вывод данных можно осуществлять в бинарный файл, а также использовать любые форматы для хранения структурированных данных (XML, JSON и т.п.).

ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА НАПИСАНИЕ ПРОГРАММЫ

1. Ввод исходных данных должен обеспечиваться с использованием графического пользовательского интерфейса. Настройки программы также осуществляются с пользовательской формы. Вывод данных должен осуществляться на пользовательскую форму. **Недопустим** вывод данных только в MessageBox-ы, или во множество пользовательских форм. Допустимо хранить ряд табличных параметров в структурированном текстовом (CSV, JSON, XML, XLS* и т.п.) файле.

   Требования к вводу/выводу при помощи пользовательской формы:

   a. Ко всем входным параметрам необходимо иметь ясные описания и комментарии на форме.
   b. Все элементы пользовательской формы должны быть выровнены и упорядочены.
   c. Обеспечивать проверку корректности ввода данных пользователем и перехват возможных ошибок ввода (нечисловые значения, отсутствие данных, превышение нижнего предела над верхним и т.п.).
2. Вывод промежуточной информации осуществляется в случае необходимости в специальный текстовой или бинарный TMP-файл или на лист Excel в зависимости от условий задачи.
3. Проект должен быть оформлен в модульном виде. Примерный состав модулей[4]:
   a. **Пользовательская форма** (интерфейсная часть),
   b. **головной** (управляющий) модуль (его роль может выполнять форма),
   c. **расчётный**(е) модуль(и),
   d. **модуль построения диаграмм,**
   e. модуль(и) **ввода/вывода**,
   f. модуль **расчёта функции** (если есть функция).
   g. Прочие модули – при необходимости.

   - Недопустимо производить в модуле или форме нетипичные для этой структурной единицы операции (например, выводить какую-либо информацию пользователю из модуля расчёта функции, или производить расчёты в коде формы)
   - Допустимо, когда модуль осуществляет протоколирование результатов своей работы в файл.
4. Расчётный модуль должен иметь универсальный характер. То есть, **абсолютно недопустимо** создавать модуль, ориентированный на какую-то одну рассчитываемую функцию или на матрицу какой-то одной размерности или интервала размерностей.
5. Там где это возможно по алгоритму метода(ов) пользователю должна предоставляться возможность варьировать точность расчётов. Недопустимо задавать фиксированную точность расчётов внутри кода программы.
6. Все константы должны фигурировать в коде программы в буквенном виде. Использования литералов и констант в числовом виде внутри кода следует максимально избегать.
7. Код программы **обязательно** должен содержать описание используемых входных и выходных переменных, а также необходимые заголовки и комментарии к используемым процедурам и функциям, поясняющие их функциональное назначение.
8. Тестирование программы должно быть проведено на нескольких функциях (матрицах) (не менее 3-х). Выбор тестовых примеров зависит от характера решаемой задачи. Тестовые функции должны быть из разных классов.
9. Отчёт по работе должен включать в себя:

[4] Можно вместо модулей использовать соответствующие классы.

a. **Постановка задачи**
b. **Блок-схема программы и описание алгоритма.**
c. **Результаты тестовых расчётов с графиками,**
d. **Анализ проведённых тестовых расчётов, доказательства их точности.**

Оценка работы:

№	Раздел	Критерий оценки	Баллы за этап
Контрольные точки. Предварительные результаты разработки (30 баллов)			
1	1 контрольная точка	Концепция. Интерфейс (Фронтенд).	10
2	2 контрольная точка	Алгоритм, расчётный блок и блок функций (бэкенд) (1-е задание)	10
3	3 контрольная точка[5]	Диаграммы или файлы[6]	10
Общая сдача работы* (70 баллов)			
4	Оценка с точки зрения пользователя (20 баллов)	Удобство и функционал интерфейса[7]	5+2+2+1=10
		Удобство работы с результатами[8]	2+1+1+1=5
		Гибкость настроек программы[9]	2+1+1+1=5
5	Оценка с точки зрения программиста (35 баллов)	Рациональность структуры программы и хранения данных[10]	5+2+2+1=10
		Оптимальность алгоритма	5+2+2+1=10
		Перехват ошибок	2+1+1+1=5
		Рациональное использование файлов	0+0+2+3=5
		Комментарии и имена, минимизация использования литералов[11]	=5
6	Оценка отчёта** (15 баллов)	Описание программы (блок-схема и т.п.)	=5
		Подбор тестовых данных	=3
		Анализ полученных результатов	=5
		Оформление	=2

* – Программа должна быть хотя бы частично работоспособна.
** – Отчёт **в печатном виде** является обязательной частью.

Используемые термины:

Рациональность структуры – код программы скомпонован по блочному принципу. Оптимальное сочетание глобальных и локальных переменных и констант. Деление программы на модули. Вынесение повторяющихся блоков в функции и методы и т.п.

Интерфейс – оценивается с точки зрения пользователя удобство работы с интерфейсной частью программы. Поля ввода/вывода должны быть рационально скомпонованы, размещены и описаны на формах и листах. Оценивается удобство ввода и вывода данных для пользователя и удобство запуска программы.

Перехват ошибок – Должен осуществляться перехват возможных ошибок пользователя при вводе исходных данных. Возможность возникновения неперехваченной ошибки считается недочётом.

Комментарии – комментарии к блокам, классам, и т.п.. Говорящие названия структурных единиц кода.

Работоспособность – Блок корректно и безошибочно выполняет поставленные расчётные задачи для заданного класса задач.

Рациональность – Данные во входных и результирующих файлах размещены рационально и отформатированы. Однотипные, повторяющиеся в нескольких местах действия оформлены в виде соответствующих функций/процедур. Также оценивается оптимальность хранения данных.

Подбор тестовых данных – подбор тестовых примеров, диапазонов, функций, матриц, систем уравнений и пр. (Для вариантов 21 – 22 баллы этого пункта начисляются за используемые возможности по форматированию диаграмм тестовых функций).

---

[5] Точка может отсутствовать. В таком случае, баллы за неё распределяются пропорционально по двум оставшимся.
[6] Для 21-22 вариантов – файлы
[7] Ввод и вывод данных. Компоновка формы, внутренняя структура файлов и т.п.
[8] В том числе файлы и диаграммы.
[9] Возможности по быстрому изменению начальных данных, функций, формата вывода данных
[10] Деление программы на модули. Вынесение повторяющихся блоков в функции и методы и т.п..

Анализ результатов – анализ и интерпретация полученных расчётной информации. Выводы. (Для вариантов 21 – 22 баллы этого пункта начисляются за используемые возможности по форматированию диаграмм тестовых функций.)

Рекомендуемая литература по численным методам:

1. А. В. Пантелеев, Т. А. Летова Методы оптимизации в примерах и задачах. Прикладная математика для ВТУЗов, Высшая школа, 2008
2. В. М. Вержбицкий, Основы численных методов, Высшая школа, 2005 г.
3. А.А. Амосов, Ю.А. Дубинский, Н.В. Копченова, Вычислительные методы для инженеров, Высшая школа, 1994