# РЕФЕРАТ

\_\_\_с., \_\_\_ рис., \_\_\_ табл., \_\_\_ приложений, \_\_\_ источников

Целью курсовой работы является практическое закрепление теоретической части курса и приобретение навыков алгоритмизации и программирования с использованием современных программных и технических средств вычислительной техники.

В результате выполнения курсовой работы будут приобретены следующие практические умения и навыки:

* выбора и использования математических методов решения задачи;
* формализации и алгоритмизации задачи;
* разработки и отладки программ с использованием алгоритмического языка высокого уровня;
* проведения и анализа результатов вычислительных экспериментов;
* оформления программной документации.

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

АЛГОРИТМИЗАЦИЯ, СТРУКТУРА, ОБРАБОТКА ТЕКСТОВЫХ ФАЙЛОВ, ГРАФИЧЕСКИЕ ФИГУРЫ, ОБРАБОТКА МАССИВОВ

# ВВЕДЕНИЕ

Актуальность данной работы обуславливается получением теоретических и практических навыков по курсу программирования и закреплением знаний, полученных на нём. Для реализации поставленных задач был использован язык программирования СИ и среда разработки Visual Studio.

В наше время компьютеры являются неотъемлемой частью жизни и помощниками в различных занятиях. Для того, чтобы использовать их возможности полностью, существуют специальные программы, имеющие свой функционал и предназначенные для выполнения определённых операций. Программы создаются при помощи языков программирования (в дальнейшем – ЯП). Одним из таких и является СИ.

Главной особенностью этого ЯП является скорость, экономичность и эффективность. Будучи низкоуровневым языком, СИ отлично справляется с большим количеством математических вычислений. Эти черты обеспечивают разработку качественного программного обеспечения почти любого вида, работающего быстро и надёжно. В противовес этим преимуществам можно поставить сложность: для эффективного использования от программиста требуется знание спецификации и опыт.

Таким образом, использование СИ в данной работе обусловлено родом решаемых задач (математические вычисления по большей части) и эффективностью его работы.

Это решение позволяет изолировать языковые особенности от специфики процессора, на котором выполняется результирующая программа. Строгое определение языка делает его независимым от любых деталей операционной системы или машины. В то же время программисты могут добавить в библиотеку специфические системные программы, чтобы более эффективно использовать конкретной особенности машины.

# ПОСТАНОВКА ЗАДАЧИ

Программа должна иметь модульную структуру. Каждое отдельное задание должно быть реализовано в отдельном модуле (пара файлов). Вызов отдельных частей программы осуществляется с помощью интерактивного меню, находящегося в основной программе. Интерактивное меню представляет собой графический элемент.

В процессе выполнения данной курсовой работы, являющейся завершением изучения курса программирования, необходимо применить все полученные знания и навыки.

Эксплуатация происходит в диалогом режиме, все операции с программой имеют удобный и понятный интерфейс. Отображение всех данных структурировано, представлено в понятном и оформленном виде: таблица, строка, графическое представление. В зависимости от задания варианты вывода могут комбинироваться или отсутствовать.

Интерактивное меню включает в себя следующий функционал: вывод названия задачи, вывод её условия, подтверждение выбора и, собственно, переход к решению задачи.

Метод решения, вывода или другой вариативной особенности задачи выбирается пользователем в удобной для него форме. Так, пользователь может:

* выбирать источник исходных данных
* вводить исходные данные самостоятельно
* выбирать способ отображения ответа (в зависимости от задачи: в файл, в консоль, в графическом виде)

Конкретный состав меню выбирается студентом по согласованию с руководителем. Место запроса в программе для выбора метода вычисления, способа отображения результатов вычислений также может быть определено самостоятельно в зависимости от индивидуального задания.

Сортировка

При решении задач должны выполняться следующие требования:

* Размер массива чисел должен вводиться с клавиатуры;
* При формировании массива предусмотреть выбор варианта: случайным образом, вводом с клавиатуры или вводом из файла;
* При выборе ввода чисел с клавиатуры, должен быть предусмотрен удобный интерфейс ввода данных;
* При вводе из файла при недостаточном объеме данных элементы массива, для которых не хватило значений, заполняются случайными значениями;
* Сформированный массив чисел отобразить на экране;
* Предусмотреть пошаговый вывод результатов сортировки на экран и/или в файл и выделение цветом элементов, изменивших позицию на текущем шаге.

Индивидуальное задание: задана последовательность чисел длиной N. Первые N/2 чисел упорядочить по убыванию, а последние N/2 - по возрастанию методом обмена.

Обработка бинарных файлов

При решении задач должны выполняться следующие требования:

* Размер массива чисел должен вводиться с клавиатуры;
* При формировании массива предусмотреть выбор варианта: случайным образом или вводом с клавиатуры;
* При выборе ввода чисел с клавиатуры, должен быть предусмотрен удобный интерфейс ввода данных;
* Сформированный массив чисел отобразить на экране;
* Сформированные числа должны быть сохранены в бинарном файле, имя которого вводится в режиме диалога;
* Выполнить обработку сформированного бинарного файла в режиме прямого доступа к данным файла, при этом из файла разрешается считывать не более двух значений;
* Организовать просмотр созданного, а затем обработанного файла на экране.

Индивидуальное задание: удалить элементы, значения которых лежат в диапазоне между средним арифметическим значением и значением (max+min)/2.

Задача с использованием геометрических данных

При решении задач должны выполняться следующие требования:

* Размер множества точек должен вводиться с клавиатуры;
* При формировании координат точек предусмотреть выбор варианта: случайным образом, вводом с клавиатуры или вводом из файла;
* При выборе ввода с клавиатуры, должен быть предусмотрен удобный интерфейс ввода данных;
* При вводе из файла при недостаточном объеме данных элементы массивов, для которых не хватило значений, заполняются случайным образом;
* Сформированное множество точек отобразить на экране в графическом режиме;
* Решение задачи должно сопровождаться графической интерпретацией с построением осей координат.
* Графическое изображение должно менять масштаб, цвет, тип линии и заливок.

Индивидуальное задание: на плоскости заданы множества точек А и множества окружностей В. Найти две такие различные точки из А, что проходящая через них прямая пересекается с минимальным количеством окружностей из В.

# РАЗРАБОТКА СТРУКТУРЫ РЕШЕНИЯ ОБЩЕЙ ЗАДАЧИ И АЛГОРИТМА, ОПРЕДЕЛЯЮЩЕГО ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ДЕЙСТВИЙ

Структура выполнения программы для решения поставленной задачи будет реализована при помощи интерактивного меню. Меню позволяет выбрать, с каким заданием работать, а также внутри отдельных заданий в диалоговом режиме можно выбрать способ ввода исходных данных (ввод с клавиатуры, ввод из файла, заполнение случайными значениями), способ вывода результатов (в консоль, в файл). Выход из программы осуществляется при помощи нажатия клавиши Esc на клавиатуре. Общая структура изображена на рисунке 2.1.

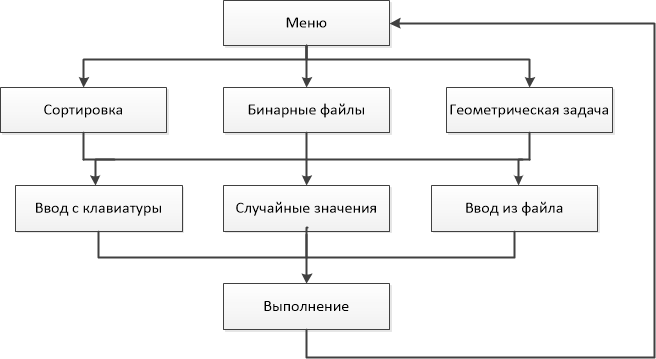


Рисунок 2.1

Соответственно заданию, программа будет состоять из 4х частей по варианту: сортировка массива, обработка бинарных файлов, решение геометрической задачи.

# Анализ существующих методов решения задачи

Метод решения для сортировки массива

Алгоритм сортировки – сортировка пузырьком (метод обмена). Среднее время выполнения – O(n \* ln n), при условии, что сравнения делаются за постоянное время.

Данная сортировка является одной из многочисленных, работающих по методу обмена. Среди них есть глупая, вялая, быстрая, гномья, сортировка расчёской и шейкерная сортировка. Общий принцип метода обмена можно описать так:

1. Попарно сравниваются элементы массива
2. В зависимости от условия элементы меняются или не меняются местами
3. Пункты 1-2 повторяются до тех пор, пока массив не отсортируется

Окончание сортировки может быть отслежено различными способами, самый распространённый – переменная-флаг.

Для реализации этого алгоритма в данной задаче было интегрировано две вещи: дифференцирование первой и второй половины массива перед п. 1 и сохранение позиций элементов, изменивших своё положение на текущем шаге для дальнейшей корректной закраски.

Сортировки обменом являются одними из самых быстрых относительно своей несложной реализации и понятного алгоритма действий. Они уступают другим методам по абсолютной скорости, но выигрывают за счёт компактности и простоты (все операции сводятся к разным способам отбора и сортировки отдельных пар, что само по себе не ресурсоёмко).

Отдельно стоит отметить сортировки, в которых элементы меняются местами на случайной или псевдослучайной основе: в справочниках указано, что такие сортировки также причисляются к сортировкам метода обмена (меняются соседние элементы), однако, руководствуясь здравым смыслом, можно утверждать, что сама по себе генерация случайного числа не является критерием выбора.

Метод решения для обработки бинарных файлов

Метод решения задачи с обработкой бинарных файлов можно разбить на две части: непосредственно работа с файлами и алгоритм выполнения сути задания.

Метод работы с файлом основан на интегрированных в СИ операторах с соответствующим функционалом. В частности, используются операторы fwrite и fread, предназначенные для работы с бинарными документами. Используется принцип прямого доступа к файлам.

Метод изменения массива базируется на нахождении максимального элемента, минимального элемента, среднего арифмитического элементов и проверке принадлежности конкретного элемента к заданному интервалу.

Метод нахождения максимального и минимального элемента основан на запоминании соответствующего значения и перебором всех элементов массива с последующим переприсвоением значения.

Метод нахождения среднего арифметического значения основан на математическом определении этого термина. Сумма всех элементов делится на их количество.

После нахождения границ интервала следует определение порядка их следования и проверки приндлежности элемента к данному интервалу.

Метод решения геометрической задачи

Метод решения геометрической задачи базируется на нахождении расстояния от точки до прямой и сравнении значений, основанном на геометрическом смысле.

Для нахождения расстояния от точки до прямой используется соответствующая формула из линейной алгебры:

,

где d – расстояние,

A, B, C – коэффициенты уравнения прямой,

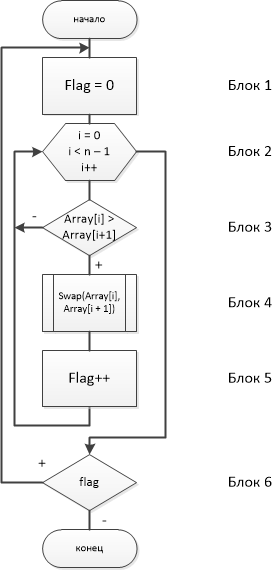
– координаты точки.

После нахождения расстояния полученное значение сравнивается с радиусом выбранной окружности. Если расстояние больше радиуса, то прямая не пересекает окружность. Если расстояние равно радиусу, то прямая касается окружности (соответствует условию отбора). Если расстояние меньше радиуса, прямая пересекает окружность в двух точках, что также соответствует условию отбора.

# Разработка и описание алгоритма решения поставленной задачи

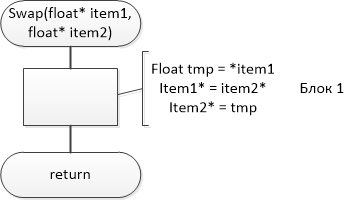
Для решения задачи сортировки массива я использовал следующие алгоритмы.

Алгоритм сортировки пузырьком:



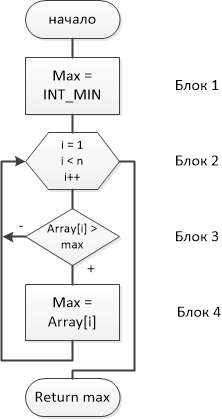
В блоке 1 обнуляется значение флага перед каждой проходкой по массиву. В блоке 2 открывается цикл for, в котором происходит попарное сравнение и обмен элементов (блоки 3, 4). Если взаиморасположение элементов удовлетворяет условию сортировки, вычисляется следующая итерация. В противном случае происходит обмен и также вычисляется следующая итерация. В случае совершения обмена значение флага инкрементируется. Таким образом, если после проходки по массиву флаг равен 0, программа выйдет из цикла (0 в условии) и закончит сортировку.

Функция Swap меняет местами два переданных в неё значения и имеет следующую реализацию:



В блоке один происходят все 3 действия: создаётся новая временная переменная соответствующего типа, инициализируется значением первого параметра. Первому параметру присваивается значение второго параметра, и второму параметру присваивается значения временной переменной. В результате значения соответствующих элементов поменяны местами. Параметр имеет тип указателя.

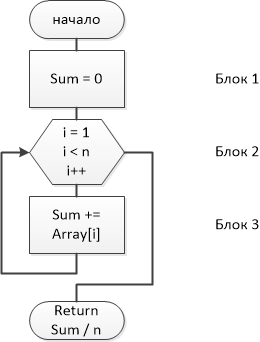
Для решения задачи по удалению элементов из массива, я использовал алгоритм поиска максимального элемента:



В блоке 1 будущая переменная с максимальным элементом инициализируется минимально возможным значением типа (это условность: если бы алгоритм был реализован на JavaScript, переменная инициализировалась бы значением -Infinity, т.е. бесконечно малым числом, что соответствует логике в большей степени). В блоке 2 открывается цикл, в каждой итерации которого сравнивается элемент массива с максимальным значением из предыдущих элементов (бесконечно малая инициализация будет замещена первым же сравнением). В блоке 3 при соответствии условию отбора max значение либо перезаписывается (блок 4), либо нет. По завершению цикла возвращается максимальное значение.

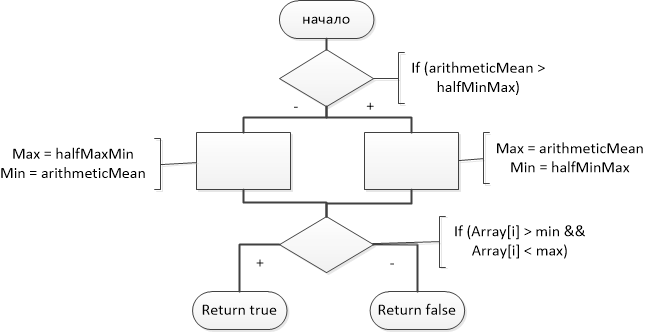
Алгоритм поиска минимального значения имеет аналогичную реализацию с той лишь разницей, что запоминающая переменная инициализируется бесконечно большим значением, а знак > заменяется знаком <.

Также используется алгоритм нахождения среднего арифметического значения элементов:



В блоке 1 инициализируется сумма нулём, в блоке 2 открывается цикл по массиву. В блоке 3 к переменной прибавляются все значения элементов массива, и возвращает алгоритм сумму всех элементов, делённую на их количество.

Остаётся лишь определить принадлежность элемента массива к полученному интервалу:



ОПИСАНИЕ ПРОГРАММЫ

Программа имеет модульную структуру и состоит из 4х заголовочных файлов и 5ти файлов исходного кода. 3 заголовочных файла для отдельных заданий по варианту, содержащие необходимые для этих заданий функции и объекты. 1 заголовочный файл, общий для этих заданий – он подключается в них непосредственно через команду препроцессора #include. 4 файла с исходным кодом, содержащие реализацию описанных выше заголовочных файлов. 5й файл с исходным кодом – основная функция, содержащая интерактивное меню и управляющая всей программой, которая написана на языке СИ.

Структура состоит из пунктов интерактивного меню, отображения доп. информации согласно выделенным пунктам меню и диалоговому меню внутри каждого задания.

Описание заголовочного файла ArrayForming.h

Данный файл содержит декларации общих функций и структур для остальных заданий. Примечание: в качестве параметра в функции в этом файле передаётся копия объекта массива, и при этом в функциях заполняется массив. Это работает корректно, так как вместе с копией объекта передаётся копия адреса памяти, в которой содержатся элементы массива.

Структуры в файле: MyArray. MyArray содержит размеры массива и указатель на участок памяти со значениями массива.

Функции в файле: FormArray, EnterArrayByKeyboard, RandomArray, GetArrayFromFile, PrintArray, FloatRandom.

Функция FormArray формирует массив указанным пользователем способом, вызывая одну из прочих функций. В качестве параметров передаётся:

* MyArray\* array - указатель на объект
* int chosenMethod – метод заполнения, выбранный пользователем в меню прежде

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| array | MyArray\* | Указатель на объект структуры, содержащий информацию о массиве |
| int | chosenMethod | Выбор способа заполнения |

Функция EnterArrayByKeyboard осуществляет ввод значений массива напрямую с клавиатуры при помощи диалогового окна. В качестве параметров передаётся:

* MyArray array – копия объекта

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| array | MyArray | Копия объекта структуры, содержащая информацию о массиве |

Функция RandomArray заполняет массив случайными значениями и выводит массив на экран. В качестве параметров перадётся:

* MyArray array – копия объекта

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| array | MyArray | Копия объекта структуры, содержащая информацию о массиве |

Функция GetArrayFromFile считывает данные из файла и заполняет ими массив, а также выводит его на экран. В качестве параметров передаётся:

* MyArray array – копия объекта

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| array | MyArray | Копия объекта структуры, содержащая информацию о массиве |

Функция PrintArray выводит данные, хранящиеся на данный момент в памяти, на экран. В качестве параметров передаётся:

* MyArray array – копия объекта

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| array | MyArray | Копия объекта структуры, содержащая информацию о массиве |

Функция FloatRandom генерирует случайное вещественное число в заданном интервале. В качестве параметров передаётся:

* float leftBorder – левая граница интервала
* float rightBorder – правая граница интервала

Функция имеет тип float и возвращает сгенерированное случайное значение.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| leftBorder | float | Левая граница интервала для генерации |
| rightBorder | float | Правая граница интервала для генерации |

Описание заголовочного файла Sorting.h

Данный файл содержит декларации функций для решения задачи сортировки.

Функции в файле: Sorting, SwapTwoItems, SortArray, SortByStep.

Функция Sorting является основной функцией этого задания, в которой выполняются все необходимые действия. В качестве параметров передаётся:

* int chosenMethod – способ заполнения массива, выбранный пользователем

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| int | chosenMethod | Выбор способа заполнения |

Имеет тип void и не возвращает ничего.

Функция SwapTwoItems меняет местами значения двух переданных ей элементов. В качестве параметров передаётся:

* float\* item1 – указатель на первый элемент
* float\* item2 – указатель на второй элемент

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| item1 | float\* | Ссылка на первый элемент |
| item2 | float\* | Ссылка на второй элемент |

Функция SortArray сортирует массив. В качестве параметров передаётся:

* MyArray array – копия объекта

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| array | MyArray | Копия объекта структуры, содержащая информацию о массиве |

Функция SortByStep также сортирует массив, но при этом пошагово выводя его состояние после каждой проходки, и подсвечивая цветом элементы, изменившие свои места с прошлой итерации. В качестве параметров передаётся:

* MyArray array – копия объекта

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| array | MyArray | Копия объекта структуры, содержащая информацию о массиве |

Описание заголовочного файла HandleFile.h

Данный файл содержит декларации функций для решения задачи обработки бинарного файла.

Функции в файле: HandleFile, WriteArrayToFile, MyTask, ArithmeticMean, HalfOfMaxPlusMin, FileLinearShift, PrintFile.

Функция HandleFile является основной функцией этого задания, в которой выполняются все необходимые действия. В качестве параметров передаётся:

* int chosenMethod – способ заполнения массива, выбранный пользователем

Имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| int | chosenMethod | Выбор способа заполнения |

Функция WriteArrayToFile записывает массив из памяти в бинарный файл. В качестве параметров передаётся:

* MyArray array – копия объекта
* FILE\* file – указатель на поток файла

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| array | MyArray | Копия объекта структуры, содержащая информацию о массиве |
| file | FILE\* | Указатель на поток файла |

Функция MyTask выполняет непосредственно необходиммые действия для решения задачи. В ней находится ядро вычислений. В качестве параметров передаётся:

* FILE\* file – указатель на поток файла
* int\* itemsNumber – указатель на количество элементов в массиве

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| array | MyArray | Копия объекта структуры, содержащая информацию о массиве |
| file | FILE\* | Указатель на поток файла |

Функция ArithmeticMean считывает данные из файла и находит их среднее арифметическое. В качестве параметров передаётся:

* FILE\* file – указатель на поток файла

Функция имеет тип float и возвращает вычисленное среднее арифметическое.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| file | FILE\* | Указатель на поток файла |

Функция HalfOfMaxPlusMin считывает данные из файла и находит половину от суммы минимального и максимального элемента массива. В качестве параметров передаётся:

* FILE\* file – указатель на поток файла

Функция имеет тип float и возвращает вычисленное значение.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| file | FILE\* | Указатель на поток файла |

Функция FileLinearShift производит линейный сдвиг в файле, т.е. удаляет элемент и уменьшает общее кол-во элементов в массиве. В качестве параметров передаётся:

* FILE\* file – указатель на поток файла
* int itemToDelete – номер элемента, который необходимо удалить
* int\* itemsNumber – количество элементов, которое после произведения операции будет декрементировано.

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| file | FILE\* | Указатель на поток файла |
| int | itemToDelete | Номер элемента, который будет удалён |
| Int\* | itemsNumber | Количество элементов в массиве. |

Функция PrintFile выводит содержимое файла на экран в режиме прямого доступа. В качестве параметров передаётся:

* FILE\* file – указатель на поток файла

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| file | FILE\* | Указатель на поток файла |

Описание заголовочного файла Geometry.h

Данный файл содержит декларации функций для решения задачи обработки бинарного файла и соответствующие структуры. Примечание: в этом файле содержатся функции, аналогичные функциям из ArrayForming.h, но модифицированные под заполнения массива объектов точек и окружностей соответственно. В качестве параметров передаются указатели на массивы данных объектов, а также их количесво. Тип данных функций – void, они не возвращают ничего.

Структуры в файле: Dot, Circle, Line, Borders, Parameter.

Структура Dot описывает объект точка и содержит её кординаты на декартовой координатной плоскости: абсцисс и ординат.

Структура Circle описывает объект окружность и содержит координаты центра и длину радиуса окружности.

Структура Line содержит координаты двух точек, через которые проходит данная прямая. Формально является уравнением прямой.

Структура Borders содержит минимумы и максимумы абсцисс и ординат данного множества точек и окружностей.

Структура Parameter содержит два поля для абсциссы и ординаты, в которых содержаться некие параметры для А. и О. соответственно. Может иметь разный смысл в зависимости от контекста.

Функции в файле: Geometry, MyTask, Draw, MakeGrid, MakeAxis, MakeElements, CalcScale, CalcGraphBorders.

Функция Geometry является основной функций этого задания, в которой выполняются все действия. В качестве параметров передаётся:

* int chosenMethod – способ заполнения массива, выбранный пользователем

Имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| int | chosenMethod | Выбор способа заполнения |

Функция MyTask производит все необходимые вычисления для непосредственного решения математической стороны задачи. В качестве параметров передаётся:

* Dot\* dots – массив точек
* Circle\* circles – массив окружностей
* int dotsNumber – количество точек
* int circlesNumber – количество окружностей
* Line\* - массив уравнений прямых

Функция имеет тип Line и возвращает уравнение прямой, проходящей через наименьшее кол-во окружностей.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| dots | Dot\* | Указатель на массив точек |
| circles | Circle\* | Указатель на массив окружностей |
| dotsNumber | int | Количество точек |
| circlesNumber | int | Количество окружностей |
| lines | Line\* | Массив прямых |

Функция Draw рисует графический ответ на эту задачу. В качестве параметров передаётся:

* Dot\* dots – массив точек
* int dotsNumber – количество точек
* Circle\* circles – массив окружностей
* int circlesNumber – количество окружностей
* Line\* - массив уравнений прямых
* int linesNumber – количество прямых

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| dots | Dot\* | Указатель на массив точек |
| dotsNumber | int | Количество точек |
| circles | Circle\* | Указатель на массив окружностей |
| circlesNumber | int | Количество окружностей |
| lines | Line\* | Массив прямых |
| linesNumber | int | Количество прямых |

Функция MakeGrid рисует сетку координатной плоскости. В качестве параметров передаётся:

* Borders drawBorders – границы отрисовки
* Parameter axis – координаты осей
* Parameter scale – масштаб абсцисс и ординат

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| Borders | drawBorders | Границы отрисовки |
| Parameter | axis | Координаты осей |
| Parameter | scale | Масштаб |

Функция MakeAxis рисует оси координатной плоскости. В качестве параметров передаётся:

* Borders drawBorders – границы отрисовки
* Parameter scale – масштаб отрисовки

Функция имеет тип Parameter и возвращает координаты осей координат.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| Borders | drawBorders | Границы отрисовки |
| Parameter | scale | Масштаб |

Функция MakeElements рисует прямые, точки и отрезки. В качестве параметров передаётся:

* RECT windowSize – размеры консольного окна
* Dot\* dots – массив точек
* int dotsNumber – количество точек
* Circle\* circles – массив окружностей
* int circlesNumber – количество окружностей
* Borders drawBorders – границы отрисовки
* Line\* - массив уравнений прямых
* int linesNumber – количество прямых

Функция имеет тип void и не возвращает ничего.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| RECT | windowSize | Размеры консольного окна |
| dots | Dot\* | Указатель на массив точек |
| dotsNumber | int | Количество точек |
| circles | Circle\* | Указатель на массив окружностей |
| circlesNumber | int | Количество окружностей |
| Borders | drawBorders | Границы отрисовки |
| lines | Line\* | Массив прямых |
| linesNumber | int | Количество прямых |

Функция CalcScale вычисляет масштаб отрисовки относительно осей абсцисс и ординат. В качестве параметров передаётся:

* Borders drawBorders – границы отрисовки

Функция имеет тип Parameter и возвращает структуру с масштабом по осям.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| Borders | drawBorders | Границы отрисовки |

Функция CalcGraphBorders вычисляет границы отрисовки графического ответа. В качестве параметров передаётся:

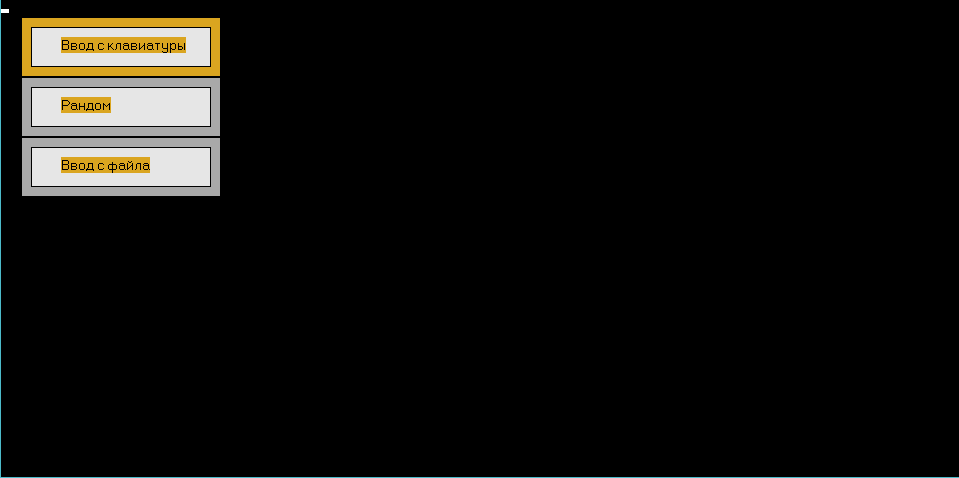
* Dot\* dots – массив точек
* int dotsNumber – количество точек
* Circle\* circles – массив окружностей
* int circlesNumber – количество окружностей

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| dots | Dot\* | Указатель на массив точек |
| dotsNumber | int | Количество точек |
| circles | Circle\* | Указатель на массив окружностей |
| circlesNumber | int | Количество окружностей |

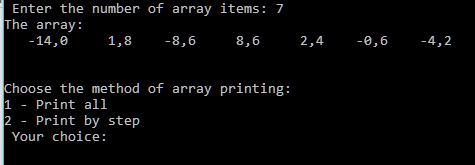
ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

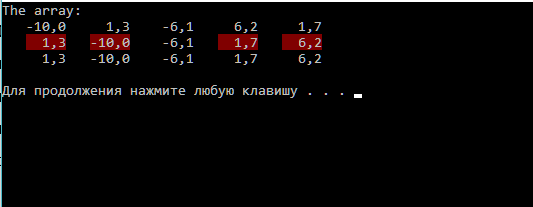
Запуск программы реализуется с помощью файла «CW-AnP.exe». Клавишами «2» и «8» пользователь может переключаться между вариантами выбора, подтверждение клавишей «Enter», выход на клавишу «Esc». При выборе ввода данных из файла можно ввести любое разрешенное имя файла. Считывание происходит в строчку, считывание точек – парами, считывание окружностей – тройками.

После запуска на экране отображается пользовательское меню:

Меню состоит из 3х пнуктов: Сортировка, Бинарные файлы, Геометрия. При выборе любой плитки будет предложено выбрать способ заполнения:

После выбора способа заполнения запуститься задача, выбранная на этапе 1, с указанным способ ввода данных:

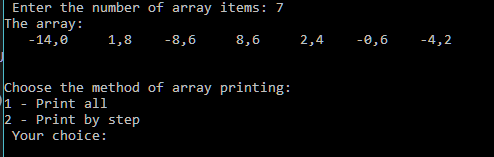




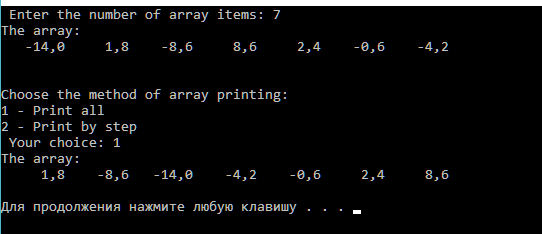
АНАЛИЗ РЕЗУЛЬТАТОВ РАБОТЫ ПРОГРАММЫ

Сортировка массива

В первой задаче необходимо отсортировать массив так, чтобы первая его половина шла от большего к меньшему, а вторая половина от меньшего к большему. Зададим размер массива 7, способ заполнения случайными данными:

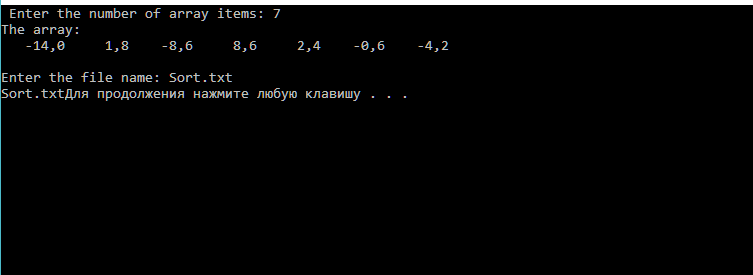


Сразу выведем отсортированный массив:



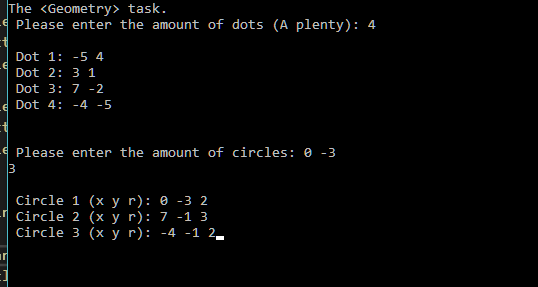
Первые три числа соблюдают условие: 1.8 > -8.6 > -14.0. Вторая половина также соблюдает условие задачи: -0.6 < 2.4 < 8.6. Результатом работы программы является отсортированный по заданию массив.

Обработка бинарных файлов

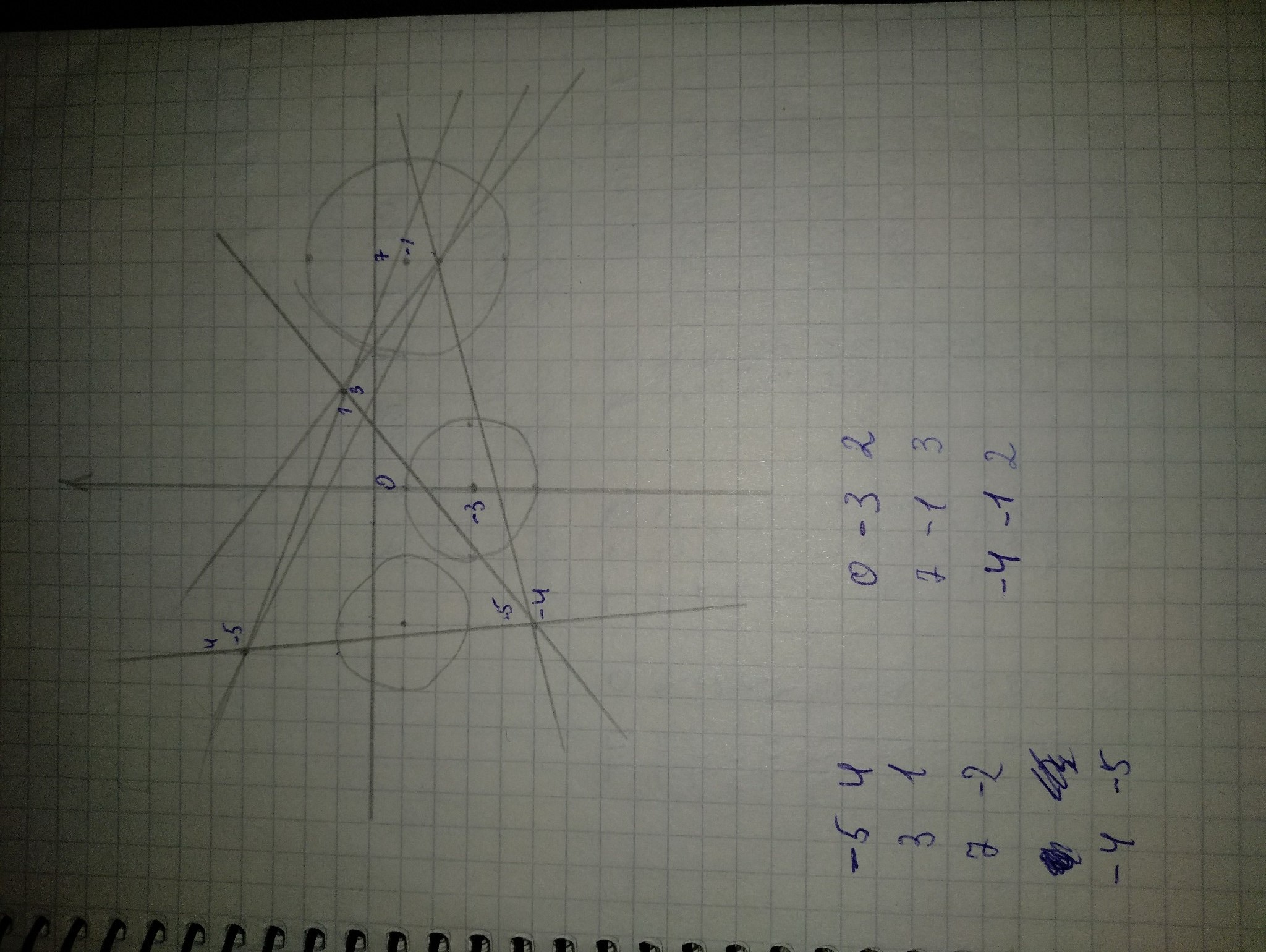
Во второй задаче необходимо записать массив в бинарный файл, считать оттуда данные в режиме прямого доступа, и удалить элементы, удовлетворяющие условию. Сгенерируем массив, введём имя фалйа:

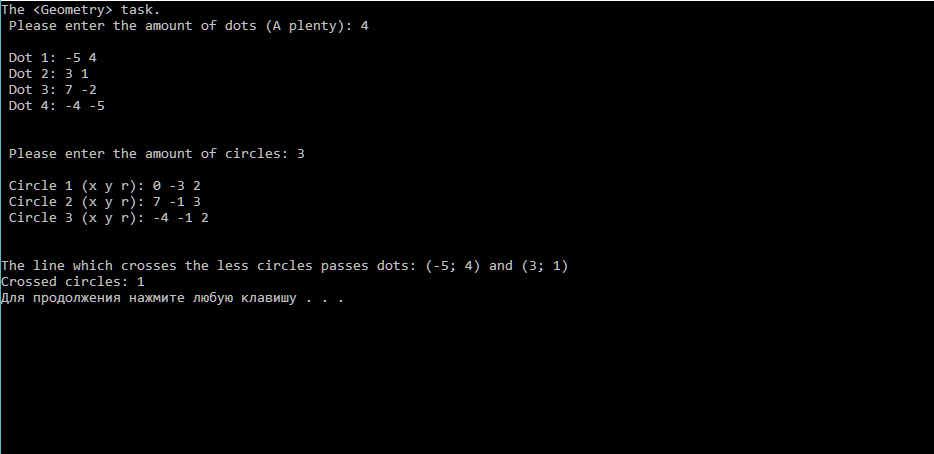
Геометрическая задача

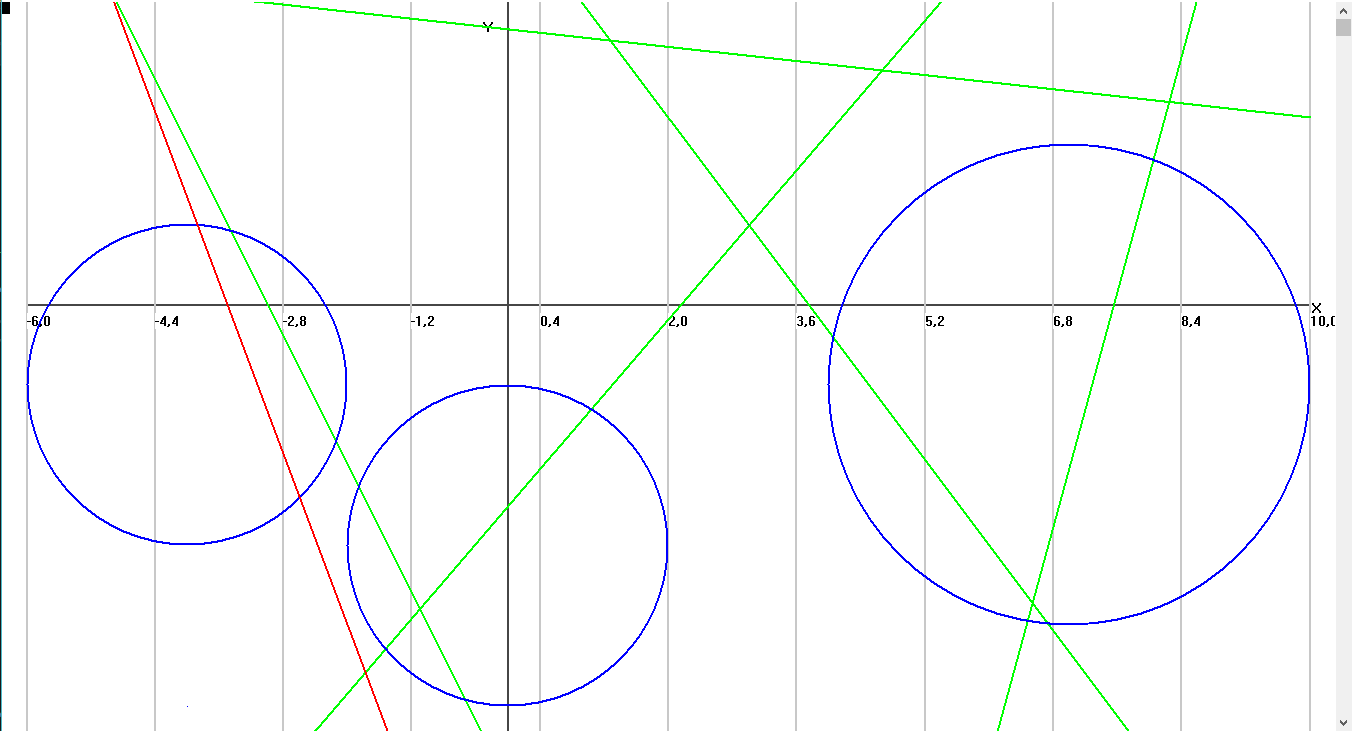
В третьей задаче необходимо из множества прямых и окружностей выбрать такую прямую, которая пересекает наименьшее количество окружностей. Выберем ввод с клавиатуры, введём контрольный пакет данных:



Этот пакет был протестирован и проверен, правильный ответ – прямая (3; 1) и (-5; 4). Она пересекает одну окружность:



Ответ «прямая (4; -5) и (-4; -5)» также будет верным, однако в силу особенностей вычисления в ответ пойдёт первый верный встреченный результат.

Представление ответа на задачу в графическом виде. Красная прямая – ответ задачи. Не вошедшие в ответ прямые – зелёные:

# ЗАКЛЮЧЕНИЕ

В результате работы над данной программой были созданы и освоены навыки в обработке одномерных и двумерных массивов (матриц), работа с построением графических фигур.

Программа создана и работает в соответствии с требованиями и готова для реализации на сторонних устройствах.

Программа проста в использовании, поскольку простое интерактивное меню и консольный интерфейс. Достаточно быстро реализует поставленные ей задачи, с легкостью производит нужные математические расчеты.

В результате выполнения курсовой работы были приобретены следующие практические умения и навыки:

* выбор и использование математических методов решения задачи;
* формализация и алгоритмизация задачи;
* разработка и отладка программ с использованием алгоритмического языка высокого уровня;
* проведение и анализ результатов вычислительных экспериментов;
* оформление программной документации.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Климова Л.М.. Основы практического программирования на языке Си. М .: Издательство ПРИОР, 1999, 464 с..
2. Романов E.Л. Информатика. Основы анализа и проектирования программ. Конспект лекций и методические указания к лабораторным работам по дисциплине "Информатика". - Новосибирск, Изд-во НГТУ, 1999, -80 с. (" методичка" для 1-го семестра, библиотека НГТУ - 73Р693)
3. Романов E.Л. Язык Си. Типы данных и управление памятью. Конспект лекций.
4. . В.В. Подбельский, С.С.Фомин. Программирование на языке Си. М .: ФиС, 1999, 600 с.
5. Х.М.Дейтел., П.Дж.Дейтел. Как программировать на Си++. М .: ЗАО БИНОМ, 1999, 1000 с.
6. У.Топп, У.Форд . Структуры данных в Си++. М .: ЗАО БИНОМ, 1999,
7. Б. Страуступ, Принципы и практика с использованием С++, 2016