Пояснительная записка

**Архитектура Вс**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2021

Оглавление

[Описание задание 2](#_Toc83572692)

[Условия 3](#_Toc83572693)

[Модульная структура 4](#_Toc83572694)

[Структура входного и выходного файла 5](#_Toc83572695)

[Шейкерная сортировка 5](#_Toc83572696)

Описание задание

В ходе выполнения каждого задания необходимо написать программу, которая должна быть оформлена в виде консольного приложения, удовлетворяющего следующим требованиям: 1. Запуск программы осуществляется из командной строки, в которой указываются: имя запускаемой программы; имя файла с исходными данными; имя файла с выходными данным

Для каждого программного объекта, загружаемого в контейнер (на основе массива максимальной размерности), исходный файл с тестовым набором должен содержать, признак альтернативы, а также список параметров, необходимых этой альтернативе. Этот список должен быть представлен в формате, удобном для обработки компьютером. При больших данных во входном файле должны быть указаны только параметры для генератора случайных наборов данных, который и заполняет контейнер

В выходной файл необходимо вывести введенные в контейнер данные. Помимо, этого необходимо вывести информацию об общем количестве объектов, содержащихся в контейнере. После этого в тот же файл необходимо вывести новые данные в соответствии с результатами, полученными в ходе работы программы. Информация для вывода должна быть представлена в форме, удобной для восприятия пользователем.

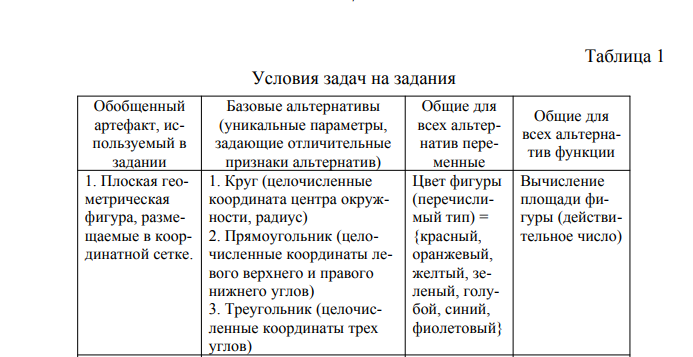
Программа должна иметь модульную структуру, соответствующую выданному варианту задания

Для представления символьных данных, обрабатываемых в программе, предлагается использовать только латиницу. Это обусловлено стремлением упростить их обработку. Дело в том, что в Linux в качестве основной кодировки символов принята UTF-8, которая за пределами кодировки ASCII использует двухбайтовый формат, в том числе и для русских букв. Комментарии, пояснительный текст, не требующий обработки можно писать с использованием как русского, так и английского языков, а также транлитом

Условия

Каждый из вариантов собирается из двух независимых компонент: условия задачи и функции обработки данных, загруженных в контейнер. Выбор этих составляющих осуществляется на основе номера варианта задания (от 1 до 336), выданного преподавателем. Пусть Nvar – номер варианта задания, div – операция целочисленного деления, mod – операция выделения остатка от целочисленного деления. Тогда номер условия задачи Ntask вычисляется следующим образом: Ntask = (Nvar - 1) mod 14 + 1. Номер допоолнительной функции Ncont определяется по формуле: Nfunc = ((Nvar - 1) div 14) mod 25 + 1.  
 **Мой вариант 169 Итого  
Номер задание из таблицы - 1  
Номер задание из функции - 13**





**13**  Упорядочить элементы контейнера по убыванию используя шейкерную сортировку (Shaker Sort). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

Модульное структура

**Точка и Цвет(Point.h и Color.h)  
самый низки уровень**

**Фигуры (Rectangle.h,Triangle.h,Circle.h)**

**Объединение фигур (Figure.h)**

**Контейнер фигур (Vector.h)**

Уровень ВС “Плоских фигур”

Color.h и Point.h реализовано отдельно оно не зависимо от фигур так как все фигуры создаются из точки .

Структура входного и выходного файла

Входной файл :Сперва тип фигуры 1-Прямоугольник 2-Триугольник 3-Окружность  
Потом параметры этой фигуры например для Окружность будет выглядеть как:  
3

1 2 3 Red

Цвет задается на английском всего 6 цветы “Red” ,“Green” ,”Blue” ,”Orange”,”Violet”,”Yellow”.

Выходной файл  
Сведение о количество фигур и полное сведение о каждый фигуре

Можно смотреть пример из Каталога “/tests”-тестовый входные файлы а из   
“/testanswer” пример выходных

Шейкерная сортировка

***Шейкер-сортировка*** является усовершенствованным методом пузырьковой сортировки.  
Анализируя метод пузырьковой сортировки, можно отметить два обстоятельства:

* если при движении по части массива перестановки не происходят, то эта часть массива уже отсортирована и, следовательно, ее можно исключить из рассмотрения.
* при движении от конца массива к началу минимальный элемент «всплывает» на первую позицию, а максимальный элемент сдвигается только на одну позицию вправо.

Эти две идеи приводят к модификациям в методе пузырьковой сортировки.

От последней перестановки до конца (начала) массива находятся отсортированные элементы. Учитывая данный факт, просмотр осуществляется не до конца (начала) массива, а до конкретной позиции. Границы сортируемой части массива сдвигаются на 1 позицию на каждой итерации.

Массив просматривается поочередно справа налево и слева направо.

Просмотр массива осуществляется до тех пор, пока все элементы не встанут в порядке возрастания (убывания).