Контрольные работы по дисциплине

«Информационные технологии и программирование»

Преподаватели:

Иван Григорьевич Корниенко

Алексей Константинович Федин

Санкт-Петербург
2023

1 Назначение курса

1 Контрольные работы по дисциплине «Информационные технологии и программирование» предназначены для практического закрепления навыков, полученных при прослушивании лекционного материала по соответствующему курсу.

2 Общие требования

2.1 Порядок выполнения контрольной работы

1 Выполнение контрольной работы начинается с получения задания.
2 Студент должен ознакомиться с заданием на контрольную работу. В случае если задание непонятно, он может проконсультироваться у преподавателя.
3 Контрольная работа выполняется индивидуально. Номер варианта определяется порядковым номером студента в списке, выложенном в папке дисциплины на сервере кафедры. Первый в списке получает первый вариант. Если предположить, что вариантов шесть, то седьмой студент по списку получает снова первый вариант. Студент не может выбрать другой вариант.
4 После получения и согласования с преподавателем задания на контрольную работу студент приступает к выполнению теоретической части работы. В ходе теоретической части работы студент разрабатывает:
   – требования к программному продукту,
   – математические методы и алгоритмы,
   – форматы представления данных,
   – структуру программы.
5 После разработки теоретической части студент представляет преподавателю разработанный материал, и, получив разрешение преподавателя, приступает к непосредственному кодированию разработанной программы на компьютере.
6 Написав программу, студент должен её тщательно протестировать. После этого он должен защитить программу преподавателю, продемонстрировав её работу и исходный код.
7 По завершении работы студент оформляет и защищает отчёт, к которому прилагается исходный код программы.

2.2 Требования к программному коду

1 Работы выполняются в среде Microsoft Visual Studio на языке C++.
2 Все идентификаторы (названия переменных, функций, классов, модулей и т. д.) должны отражать назначение именуемых объектов. Все названия должны быть даны на английском языке. Не допускается использование транслитерации и других нестандартных приёмов именования.
3 Все константные литералы в коде должны быть объявлены как константы (`const`).
4 Стиль оформления кода (отступы, именование переменных, функций, классов и т. д.) должен последовательно соблюдаться в работе и соответствовать одному из общепринятых стандартов (например, Google C++ Style Guide).

2.3 Требования к программе

1 При запуске программа должна выводить приветствие. В приветствии указывается Ф. И. О., номер группы и вариант исполнителя, номер работы, номер и текст задания. Если программа выполнена с использованием графического интерфейса пользователя (приложение Windows) – программа должна выводить соответствующую информацию при запуске и в ответ на выбор соответствующего пункта меню. В этом случае должна быть возможность отключения вывода приветствия при запуске программы.
2 Выполнение консольной программы должно быть закольцовано. Завершение работы программы должно производиться только при выборе соответствующего пункта меню.

2.4 Общие требования к работам

1 Программы должны компилироваться без предупреждений. Уровень предупреждений, создаваемых компилятором, – Warning Level 4.
2 Все ошибки, которые могут возникнуть при использовании программы должны быть обработаны с выдачей соответствующих диагностических сообщений. Метод обработки ошибок должен быть единым для всей программы.
3 Интерфейс пользователя должен быть полностью отделён от вычислительных процедур программы.
4 Все интерфейсы должны быть документированы.

2.4.1 Типы данных

1 Все целые литералы по умолчанию должны иметь тип `int`.
2 Литералы, представляющие действительные числа, по умолчанию должны иметь тип `double`.
3 Явное приведение типов осуществляется с использованием `_cast` функций.

2.4.2 Операции ввода-вывода

1 Для реализации ввода-вывода (экранного и файлового) необходимо использовать потоковые библиотеки C++.
2 Консольный ввод-вывод осуществляется с помощью объектов `cin` и `cout`.
3 Файловый ввод-вывод осуществляется с помощью объектов `ifstream` и `ofstream`.

2.4.3 Операции выделения памяти

1 Для выделения и освобождения памяти необходимо использовать операторы `new` и `delete`.

2.4.4 Работа с файлами

1 Во всех работах кроме консольного или графического ввода-вывода (экран, клавиатура) необходимо предоставить возможность:
   – загружать исходные данные из файла,
   – сохранять исходные данные в файле (в подходящем для загрузки формате),

- сохранять результат работы программы (с указанием исходных данных, для которых он получен) в файле.

2 Полный путь к файлам в каждом случае задает пользователь.

3 При сохранении в существующий файл, пользователю предлагается перезаписать его или указать новый. Данная проверка продолжается до тех пор, пока пользователь не укажет уникальное имя или не выберет перезапись.

4 При сохранении в файл, доступный только для чтения, пользователю предлагается указать новый.

2.4.5 Разделение на модули

1 Программа разделяется (минимум) на три независимых *cpp*-файла.
   - основной модуль, осуществляющий запуск программы,
   - модуль с пользовательским интерфейсом,
   - модуль с используемыми алгоритмами.

2 Для каждого *cpp*-файла должен быть предоставлен соответствующий *h*-файл, содержащий интерфейс данного модуля.

2.4.6 Модульные тесты

1 Каждая программа должна сопровождаться набором модульных тестов, позволяющих проверить её работоспособность при различных вариантах использования на примерах с заранее известными результатами.

2 Модульные тесты должны запускаться автоматически или в случае выбора пользователем соответствующего пункта меню программы.

3 Модульные тесты должны содержать минимум пять различных тестовых наборов данных для автоматической проверки.

4 В случае успешного тестирования модуль должен возвращать результат тестирования. Вызывающая его программа должна выводить пользователю одно результирующее сообщение «Тестирование прошло успешно». В случае неуспешного тестирования должна быть выведена подробная информация о проваленном тесте с указанием набора данных, ожидаемого и полученного результата.

2.5 Содержание отчёта

1 В отчёт входит описание:
   - постановки задачи,
   - исходных данных,
   - особых ситуаций,
   - математических методов и алгоритмов решения задачи,
   - блок-схем алгоритмов программы,
   - форматов представления данных (классов, структур, констант, переменных) в памяти и на внешних носителях (файлах),
   - структуры (модулей, функций) программы,
   - хода выполнения работы,
   - полученных результатов.

2 Отчёт оформляется на листах формата А4 с обязательным титульным листом, на котором указываются название работы; Ф. И. О. исполнителя; Ф. И. О. преподавателей и т. д.

2.5.1 Постановка задачи

1 Постановка задачи указывает, какая цель должна быть достигнута при разработке программы. Какую задачу должна решать программа, и в каких условиях будет функционировать.

2.5.2 Исходные данные

1 Исходными данными являются любые данные, которые программа получает для обработки.
2 Описание исходных данных должно содержать:
   – семантику (назначение) данных,
   – единицы изменения,
   – представление в программе.

2.5.3 Особые ситуации

1 Под особыми ситуациями понимаются ситуации, в которых поведение программы может не соответствовать поведению, ожидаемому пользователем.
2 Все особые ситуации должны быть описаны и соответствующим образом обработаны в программе.
3 Примерами особых ситуаций являются:
   – некорректный ввод,
   – отсутствие ожидаемых программой файлов,
   – возможное деление на ноль в ходе вычислений,
   – нехватка оперативной памяти.

2.5.4 Математические методы и алгоритмы решения задач

1 Все используемые программой алгоритмы и математические методы решения задач должны быть описаны в форме и полноте, достаточной для восприятия другими разработчиками.

2.5.5 Блок-схемы алгоритмов программы

1 Построение блок-схем алгоритмов регламентируется ГОСТ 19.701-90 (ИСО 5807-85) «Единая система программной документации. Схемы алгоритмов программ, данных и систем. Условные обозначения и правила выполнения».

2.5.6 Форматы представления данных

1 Для всех пользовательских типов данных (не являющихся частью языка) должны быть документированы назначение и мотивация выбора конкретного типа данных.
2 Должны быть документированы форматы всех внешних ресурсов. Структура данных, сохраняемых в файлах и т. д.

2.5.7 Структура программы

1 Разработанная структура программы (разделение на модули, интерфейсы, шаблоны проектирования) должна быть документирована.
2 Должна быть описана основная последовательность работы программы (вызова функций, методов и т. д.).
3 Все модули, функции, методы и пользовательские типы данных должны быть соответствующим образом документированы в отчёте.

2.5.8 Описание хода выполнения контрольной работы

1 В отчёте должно содержаться подробное описание хода выполнения контрольной работы.
2 Основное внимание должно быть уделено выполнению работы за компьютером.
3 Должны быть описаны все новые методы и приёмы, использованные в ходе выполнения контрольной работы. Таковыми могут быть:
   – создание проекта и запуск программы,
   – работа с отладчиком (точки останова и протоколирования, просмотр значений переменных),
   – поиск ошибок в программе,
   – работа с устройствами ввода-вывода.
4 Из отчёта должно быть ясно:
   – как выполнялась контрольная работа,
   – какие были проблемы и как они были решены,
   – какие проблемы остались нерешёнными,
   – какие моменты были или остались непонятными.

2.5.9 Результаты работы программы

1 Необходимо указать, какие результаты производит программа.
2 Необходимо указать в каком формате пользователь получает результат.
3 Должны быть приведены скриншоты не менее пяти вариантов результата выполнения программы с различными исходными данными (корректными и некорректными).

2.5.10 Исходный текст программы

1 Исходный текст программы оформляется моноширинным шрифтом, форматируется студентом в текстовом редакторе и прилагается к отчёту.

2.5.11 Документирование и комментирование исходного текста

1 Все пользовательские типы данных должны быть прокомментированы.
2 Все функции, классы и модули должны быть прокомментированы.
3 Каждый модуль (*h* или *cpp*) должен начинаться с комментария, указывающего его назначение, автора, используемые алгоритмы.
4 Каждая нетривиальная функция должна предваряться комментарием, описывающим:
   – назначение,
   – входные данные,

– результаты.

5 В функциях, где соответствующее описание будет полезным, также следует описать:
   – предусловия,
   – постусловия,
   – инварианты.

2.6 Защита и сдача контрольной работы

1 В весеннем семестре выполняются четыре контрольные работы из данного документа. График сдачи контрольных работ:

Контрольная работа №	Период защиты
1	27.02 – 10.03
2	27.03 – 07.04
3	24.04 – 05.05
4	29.05 – 09.06

2 Контрольная работа защищается преподавателям, ведущим лекционные и практические занятия.

3 Для защиты необходимо иметь отчёт о проделанной работе, продемонстрировать работоспособность проекта на компьютере кафедры, ответить на вопросы.

4 Подпись за контрольную работу выставляет преподаватель, которому работа была защищена.

5 Окончательная сдача всех контрольных работ является допуском к экзамену по дисциплине.

2.6.1 Окончательная сдача контрольных работ

1 Для сдачи контрольных работ, необходимо представить:
   – комплект отчётов по контрольным работам,
   – папку с исходными кодами программ и выполняемыми модулями (без лишних, автогенерируемых файлов).

2 В папке должен содержаться файл *readme.txt*, в котором должно быть указано:
   – Ф. И. О. студента, выполнившего работы,
   – год, название дисциплины, названия работ.

2.7 Пример выполнения задания (часть пунктов опущена)

При выполнении контрольной работы будет использоваться компилятор Microsoft Visual Studio C++ 2017.

2.7.1 Постановка задачи

Методом Монте-Карло вычислить число пи.

2.7.2 Исходные данные

В качестве исходных данных программа использует вводимое пользователем число испытаний при проведении эксперимента.

2.7.3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

- если пользователь ввёл число испытаний меньше одного, то эксперимент провести невозможно,
- если пользователь ввёл число испытаний меньше разумного предела для проведения статистического эксперимента, результаты могут быть недостоверными,
- если пользователь ввёл очень большое число испытаний, то эксперимент может занять значительное время, о чём пользователь должен быть предупреждён.

2.7.4 Математические методы и алгоритмы решения задач

Согласно постановке задачи для составления программы будет использован метод Монте-Карло, который заключается в случайном выборе координат точек внутри квадрата заданного размера.

Для каждой точки будет проверяться попадание во вписанную в квадрат окружность. Таким образом, по окончании эксперимента мы будем располагать двумя числами:

- N – число случайно выбранных точек,
- M – число точек, попавших внутрь вписанной окружности.

Поскольку нам известна площадь квадрата и площадь вписанной окружности, то мы можем вычислить отношение площадей этих фигур. Если принять сторону квадрата за единицу, то его площадь будет равна одной квадратной единице.

Площадь круга, вписанного в этот квадрат, может быть определена по формуле:

$$S = \pi r^2 = \frac{\pi d^2}{4} \quad (1)$$

Таким образом, число π можно выразить через площадь и диаметр вписанного круга следующим образом:

$$\pi = 4\frac{S}{d^2} \quad (2)$$

Площадь вписанного круга можно найти из отношения M к N. Они относятся друг к другу так, как относится площадь вписанной окружности к площади квадрата. А площадь квадрата нам известна – она единична. Исходя из этого, получаем полную формулу для вычисления числа π по известным нам числам M и N.

$$\pi = \frac{4}{d^2} \cdot \frac{M}{N} \quad (3)$$

Все величины в данной формуле нам известны. Однако, как было указано выше, M является статистической величиной, которая будет рассчитана в результате проверки попадания случайных точек в окружность. Для генерации набора точек и проверки их попадания в круг необходимо написать программу.

2.7.5 Форматы представления данных

Программа использует следующие основные переменные:

Таблица 1 – Основные переменные, используемые в программе

Имя	Тип	Описание
`tries`	`int`	Число попыток в эксперименте
`inCircle`	`int`	Число точек, попавших в окружность

Для задания минимального и максимального пределов числа экспериментов используются следующие константы

Таблица 2 – Константы, используемые в программе

Имя	Тип	Значение	Описание
`lowerLimit`	`const int`	100	Минимальное число экспериментов, обеспечивающих достоверность
`upperLimit`	`const int`	10000000	Число экспериментов, при превышении которого вычисления могут быть долгими

Для задания координат точки на плоскости используется структура `Point2D`, в которой задаются X и Y координаты точки.

2.7.6 Структура программы

В силу своей простоты программа помещена в одном исполняемом модуле. Программа разделена на несколько функций:

Таблица 3 – Функции, составляющие программу

Имя	Описание
`GeneratePoint`	Создание точки в заданных координатах
`IsInsideCircle`	Проверка на нахождение точки внутри окружности
`ProcessInput`	Обработка ввода пользователя

2.7.7 Описание хода выполнения контрольной работы

1 В ходе контрольной работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio C++ 2017. В нём был создан проект.

2 После набора текста программы выяснилось, что вывод текста на экран консольного приложения работает неправильно из-за различия кодировок консольного приложения и среды разработки. Для решения этой проблемы была использована функция `setlocale(LC_ALL, "Russian")`, которая обеспечивает работу приложения с символами кириллицы.

3 Программа после запуска выдавала одни и те же результаты, хотя в коде использовался вызов функции `rand`, возвращающей случайное число. После изучения справочной системы выяснилось, что необходимо использовать функцию `srand` для начальной инициализации генератора случайных чисел. После этого программа стала работать правильно.

4 В начальном варианте программы переменная `inputSuccess` не была инициализирована, о чём свидетельствовало соответствующее предупреждение отладчика. Учитывая, что это автоматическая переменная, создаваемая на стеке, при выполнении программы могла возникать неоднозначность. Ошибка была исправлена инициализацией переменной в значение `false`.
5 В начальном варианте программы было перепутано условие выхода из цикла `while`. Было указано `while (inputSuccess)` в результате чего тело цикла ни разу не выполнялось. Для выявления проблемы было использовано пошаговое выполнение программы, в ходе которого выяснилась причина ошибки, и программа была исправлена.

2.7.8 Результаты работы программы

В результате вычислений программа выводит примерное значение числа пи. Точность вычислений определяется числом проведённых экспериментов и качеством используемого генератора случайных чисел.

2.7.9 Исходный текст программы

```
[Начало программы ---]
// Lab0.cpp
// Контрольная работа № 0.
// Использование языка C++ для математических расчётов
// Расчёт числа пи методом Монте-Карло
// Студент группы NNN, Фамилия Имя Отчество. 2023 год

#include <iostream>
#include <clocale>
#include <cmath>
#include <ctime>

using namespace std;

// Структура, описывающая точку на плоскости
struct Point2D {
  double x, y;
};

// Генерация случайной точки с координатами X и Y
// в интервале от нуля до единицы
Point2D GeneratePoint() {
  Point2D pt;
  pt.x = rand() / static_cast<double>(RAND_MAX);
  pt.y = rand() / static_cast<double>(RAND_MAX);
  return pt;
}

// Проверка нахождения точки внутри окружности
bool IsInsideCircle(const Point2D &pt) {
  const double circleRadius = 0.5;
  const Point2D circle = {0.5, 0.5};
```

```
  double distanceFromCenter =
    sqrt(pow(circle.x - pt.x, 2) + pow(circle.y - pt.y, 2));
  return distanceFromCenter < circleRadius;
}

// Обработка ввода пользователя
int ProcessInput() {
  const int lowerLimit = 100;
  const int upperLimit = 10000000;
  bool inputSuccess = false;
  int tries = 0;
  while (!inputSuccess) {
    cout << "Введите число экспериментов:";
    cin >> tries;
    if (tries <= 0) {
      cout <<
       "Число экспериментов должно быть положительным." << endl;
      continue;
    }
    inputSuccess = true;
  }
  if (tries < lowerLimit) {
    cout << "Результат может быть неточным." << endl;
  }
  if (tries > upperLimit) {
    cout << "Вычисления могут быть длительными." << endl;
  }
  return tries;
}

// Основной модуль
int main() {
  setlocale(LC_CTYPE, "Russian");
  int inCircle = 0;
  int tries = 0;
  tries = ProcessInput();

  // Инициализация генератора ПСЧ
  srand(static_cast<unsigned>(time(NULL)));
  // Генерация и проверка точек
  for (int i = 1; i <= tries; ++i) {
    Point2D pt = GeneratePoint();
    if (IsInsideCircle(pt))
      ++inCircle;
  }
  // Вычисление числа пи
  cout << "Число пи = ";
  cout << 4 * (inCircle / static_cast<double>(tries)) << endl;
}
[--- Конец программы]
```

3 Контрольная работа № 1. Инкапсуляция. Классы в C++ и средства их построения

Первая контрольная работа предназначена для приобретения практического опыта в создании простейших классов с использованием языка программирования C++. Необходимо разработать класс для указанной предметной области. Доступ к данным реализовать с помощью методов `Set`, `Get`, `Show`.

При выполнении контрольной работы нельзя использовать контейнеры и алгоритмы библиотеки STL или аналогичных сторонних библиотек.

3.1 Варианты заданий

1 Студент: фамилия, имя, отчество, дата рождения, адрес, телефон, факультет, курс. Создать массив объектов. Реализовать возможность получения:
   - списка студентов заданного факультета,
   - списков студентов для каждого факультета и курса,
   - списка студентов, родившихся после заданного года.

2 Абитуриент: фамилия, имя, отчество, адрес, баллы. Создать массив объектов. Реализовать возможность получения:
   - отсортированного списка всех абитуриентов,
   - списка абитуриентов, сумма баллов у которых не меньше заданной,
   - списка N абитуриентов, имеющих самую высокую сумму баллов.

3 Рейс: пункт назначения, номер рейса, тип самолета, время вылета, дни недели. Создать массив объектов. Реализовать возможность получения:
   - списка рейсов для заданного пункта назначения,
   - списка рейсов для заданного дня недели,
   - списка рейсов для заданного дня недели, время вылета для которых позже заданного.

4 Книга: автор, название, издательство, год, количество страниц. Создать массив объектов. Реализовать возможность получения:
   - списка книг заданного автора,
   - списка книг, выпущенных заданным издательством,
   - списка книг, выпущенных после заданного года.

5 Сотрудник: фамилия, имя, отчество, должность, год поступления на работу, зарплата. Создать массив объектов. Реализовать возможность получения:
   - списка работников, стаж работы которых на данном предприятии превышает заданное число лет,
   - списка работников, зарплата которых больше заданной,
   - списка работников, занимающих заданную должность.

6 Поезд: пункт назначения, номер поезда, время отправления, число общих мест, купейных, плацкартных. Создать массив объектов. Реализовать возможность получения:
   - списка поездов, следующих до заданного пункта назначения,
   - списка поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа,

– списка поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

7 Товар: наименование, производитель, цена, срок хранения, количество. Создать массив объектов. Реализовать возможность получения:
– списка товаров для заданного наименования,
– списка товаров для заданного наименования, цена которых не превышает указанной,
– списка товаров, срок хранения которых больше заданного.

8 Пациент: фамилия, имя, отчество, адрес, номер медицинской карты, диагноз. Создать массив объектов. Реализовать возможность получения:
– списка пациентов, имеющих данный диагноз,
– списка пациентов, номер медицинской карты которых находится в заданном интервале.

9 Автобус: фамилия и инициалы водителя, номер автобуса, номер маршрута, марка, год начала эксплуатации, пробег. Создать массив объектов. Реализовать возможность получения:
– списка автобусов для заданного номера маршрута,
– списка автобусов, которые эксплуатируются больше 10 лет,
– списка автобусов, пробег у которых больше 10 000 км.

10 Клиент: фамилия, имя, отчество, адрес, телефон, номер карты, номер банковского счета. Создать массив объектов. Реализовать возможность получения:
– списка клиентов в алфавитном порядке,
– списка клиентов, номер карты которых находится в заданном интервале.

11 Файл: имя файла, размер, дата создания, количество обращений. Создать массив объектов. Реализовать возможность получения:
– списка файлов, упорядоченного в алфавитном порядке,
– списка файлов, размер которых превышает заданный,
– списка файлов, число обращений к которым превышает заданное.

12 Квартира: адрес, этаж, количество комнат, площадь. Создать массив объектов. Реализовать возможность получения:
– списка квартир, имеющих заданное число комнат,
– списка квартир, имеющих заданное число комнат и расположенных на этаже, который находится в определенном промежутке,
– списка квартир, имеющих площадь, превосходящую заданную.

13 Абонент: фамилия, имя, отчество, адрес, номер телефона, время внутригородских разговоров, время междугородних разговоров. Создать массив объектов. Реализовать возможность получения:
– списка абонентов, время внутригородских разговоров которых превышает заданное,
– списка абонентов, воспользовавшихся междугородней связью,
– списка абонентов, выведенных в алфавитном порядке.

14 Гражданин: фамилия, имя, отчество, адрес, образование, год рождения. Создать массив объектов. Реализовать возможность получения:
– списка граждан, возраст которых превышает заданный,
– списка граждан с высшим образованием.

4 Контрольная работа № 2. Наследование

В контрольной работе задается базовый и производный класс. Необходимо разработать поля и методы, наследуемые из базового класса, а также собственные компоненты производных классов. Базовый класс может быть абстрактным. Реализовать возможность получения списка объектов в контейнере.

4.1 Варианты заданий

1 Базовый класс – средство передвижения. Поля в нем: вес, мощность мотора, скорость. Производный класс – автомобиль; производный класс второго поколения – спортивный автомобиль, грузовой автомобиль.
2 Базовый класс – книга. Производный класс – техническая литература.
3 Базовый класс – млекопитающие; поля – способ питания, вес, среда обитания. Производный класс – хищники.
4 Базовый класс – книга. Производный класс – художественная литература.
5 Создать базовый класс фигура, производные классы: круг, прямоугольник, трапеция. Определить виртуальные функции для вычисления площади и периметра.
6 Базовый класс – документ предприятия. Производный класс – приказ.
7 Базовый класс – средство передвижения. Поля в нем: вес, мощность мотора, скорость. Производный класс – самолет; производный класс второго поколения – военный самолет, транспортный самолет.
8 Базовый класс – личность; поля – фамилия, адрес. Производный класс – студент.
9 Создать базовый класс работник, производные классы: служащий с почасовой оплатой, служащий в штате. Определить функцию начисления зарплаты.
10 Базовый класс – товар; поля – наименование, количество, производитель, цена за единицу изделия, общая цена. Производный класс – товар на складе.
11 Базовый класс – личность; поля – фамилия, пол, адрес. Производный класс – сотрудник университета.
12 Базовый класс – млекопитающие; поля – способ питания, вес, среда обитания. Производный класс – травоядные.
13 Базовый класс – документ предприятия. Производный класс – письмо.

5 Контрольная работа № 3. Полиморфизм

Необходимо составить программу для сортировки массива данных методами: пузырьковой, отбора, вставки, Шелла и быстрой сортировки. Вывести на экран неупорядоченный (один раз) и упорядоченные (для каждого из методов) массивы данных. Составить сравнительную таблицу эффективности методов, в которой необходимо указать число сравнений и перестановок переменных в каждом методе сортировки.

Неупорядоченная матрица из N строк и M столбцов задается и заполняется один раз (с клавиатуры, из файла или случайными числами), далее она используется для каждого из методов сортировки.

Реализовать абстрактный базовый класс `ISort`, содержащий чистый виртуальный метод `Sort` и необходимые счетчики, от которого наследовать подклассы для реализации сортировок.

5.1 Варианты заданий

1 Упорядочить каждую строку матрицы по убыванию.
2 Упорядочить каждую четную строку по возрастанию, затем каждый нечетный столбец по возрастанию абсолютных величин.
3 Упорядочить каждый столбец матрицы по убыванию абсолютных величин
4 Упорядочить каждую нечетную строку по возрастанию абсолютных величин, затем каждый четный столбец по возрастанию.
5 Упорядочить каждую строку матрицы по возрастанию абсолютных величин
6 Упорядочить каждую строку матрицы по убыванию суммы значений цифр элементов матрицы.
7 Упорядочить каждый столбец матрицы по возрастанию суммы значений цифр элементов матрицы.
8 Упорядочить каждую строку матрицы по убыванию абсолютных величин.
9 Упорядочить диагональные элементы матрицы по возрастанию.
10 Упорядочить каждый столбец матрицы по возрастанию.
11 Упорядочить все нечетные элементы (значения элементов) строк по возрастанию.
12 Упорядочить все четные элементы (значения элементов) столбцов по убыванию.
13 Упорядочить каждый столбец матрицы по возрастанию абсолютных величин.
14 Упорядочить каждую четную строку по возрастанию, затем каждый четный столбец по возрастанию.
15 Упорядочить каждую строку матрицы по возрастанию.
16 Упорядочить каждую нечетную строку матрицы по возрастанию суммы значений цифр элементов матрицы.
17 Упорядочить каждый столбец матрицы по убыванию.
18 Упорядочить каждый четный столбец матрицы по убыванию суммы значений цифр элементов матрицы.
19 Упорядочить каждую строку матрицы по возрастанию отрицательных величин.
20 Упорядочить каждую строку по возрастанию, затем каждый столбец по убыванию.
21 Упорядочить каждую строку матрицы по возрастанию четных чисел.
22 Упорядочить каждый четный столбец по убыванию, затем каждую строку по убыванию.
23 В представленной матрице производить замену четных чисел по возрастанию по строкам, затем нечетных чисел по возрастанию по столбцам.
24 Представить матрицу, «окрашенную» наподобие шахматной доски. Упорядочить элементы на белых клетках по возрастанию по строкам, на черных – по убыванию по столбцам.
25 Упорядочить в каждом значении чисел матрицы цифры по возрастанию, затем упорядочить данные в столбцах по убыванию.

26 Упорядочить в каждом значении чисел матрицы цифры по убыванию, затем упорядочить данные в строках по возрастанию.
27 Упорядочить главную диагональ матрицы по возрастанию, данные сверху от главной диагонали упорядочить по убыванию, снизу от главной диагонали по возрастанию.

6 Контрольная работа № 4. Классы потоков ввода-вывода

Задание предназначено для приобретения практического опыта работы с классом `std::string` в языке программирования C++.

Цель контрольной работы состоит в формировании знаний и умений:

- по использованию различных способов описания и формирования символьных строк,
- по использованию методов чтения и записи строк в текстовых файлах,
- по использованию методов чтения и анализа потоковых данных, вводимых с клавиатуры.

Во всех программах необходимо предусмотреть возможность многострочного ввода с клавиатуры.

6.1 Варианты заданий

1 Подсчитать в заданном тексте количество символов, слов, строк, абзацев. Подсчитать количество слов в предложениях и вывести статистическую таблицу, в которой длине предложения в словах будет соответствовать количество таких предложений в анализируемом тексте.
2 Найти в заданном тексте все палиндромы (слова, которые одинаково читаются от начала к концу и от конца к началу: «оно», «шалаш»); слова, которые при прочтении от конца к началу дают другие существующие в этом тексте слова («кот» – «ток»); комбинации слов, которые при слиянии дают другие слова («подросток» – «подрос ток»); слова, при удалении буквы, дающие другие слова («глаз» – «газ») и слова, дающие другое слово при замене одной буквы («тон» – «ток»). Подобрать и реализовать два-три варианта других подобных правил.
3 Найти в тексте все последовательности идущих подряд одинаковых символов и заменить их сигнатурой {символ, количество}. Минимальная длинна последовательности, которая может подвергаться замене, задаётся пользователем. Предусмотреть режим восстановления оригинального текста. Пример: текст «длинношеее животное» должен быть заменён текстом «длиннош{е, 3} животное».
4 Выделить в заданном тексте все диалоги (начинающиеся с новой строки и символа «тире»). Сохранить диалоги в отдельных текстовых файлах. Использовать правила построения диалогов, применяющиеся в русскоязычных текстах.
5 Подсчитать в заданном тексте количество вхождений каждого слова. Представить результат в виде таблицы (слово, количество вхождений). Сохранять результат в несколько результирующих файлов: список по алфавиту, список по количеству вхождений. Перед выводом статистики в

файл предоставить пользователю информацию о количестве различных слов и предложить выбрать, какое количество слов должно быть записано в результирующие файлы. Обеспечить сохранение собираемой статистической информации во внешнем файле между сеансами работы программы с тем, чтобы накапливать базу данных по анализируемым файлам.

6 Найти в тексте все повторяющиеся подстроки длиннее заданной пользователем величины (например, длиннее пяти символов). Заменить все вхождения подстроки кроме первого специальной сигнатурой: {индекс первого символа оригинальной строки, длина цепочки}. Перед заменой убедиться, что в тексте нет комбинаций символов, которые могут ошибочно восприниматься как формируемая сигнатура. Предусмотреть режим восстановления оригинального текста. Пример: текст «тестовая строка» должен быть заменен на текст «тестовая {3, 2}рока» при условии, что рассматриваются подстроки, начиная с длинны в два символа.

7 Для двух заданных текстов найти самую длинную общую подстроку. Программа должна выводить позицию, с которой текст начинается в каждом из файлов, длину строки и текст самой строки.

8 Реализовать три различных механизма шифрования текста. Предусмотреть режим восстановления оригинального текста по зашифрованному. Программа должна самостоятельно определять, какой именно вариант был использован. Один из вариантов должен предоставлять пользователю ввести ключ (пароль), используемый при шифровании и расшифровке.

9 Найти в тексте все слова, встречающиеся в одинаковых контекстах (между одних и тех же слов). Подсчитать число вариантов перестановок таких слов (между участками с аналогичным контекстом) и вывести их. Пример: «напишите функцию вычисляющую ... напишите программу вычисляющую». Возможны три перестановки: «напишите программу вычисляющую ... напишите функцию вычисляющую» «напишите программу вычисляющую ... напишите программу вычисляющую» «напишите функцию вычисляющую ... напишите функцию вычисляющую».