Лабораторные работы по курсу «Программирование»

Преподаватели:

Иван Григорьевич Корниенко

Алексей Константинович Федин

Санкт-Петербург
2021

1 Назначение курса

1 Лабораторные работы по курсу «Программирование» предназначены для практического закрепления навыков, полученных при прослушивании лекционного материала по соответствующему курсу.

2 Общие требования

2.1 Порядок выполнения лабораторной работы

1 Выполнение лабораторной работы начинается с получения задания.
2 Студент должен ознакомиться с заданием на лабораторную работу. В случае если задание непонятно, он может проконсультироваться у преподавателя.
3 Лабораторная работа выполняется индивидуально. Номер варианта определяется порядковым номером студента в списке, выложенном в папке курса 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 лабораторную работу студент приступает к выполнению теоретической части работы. В ходе теоретической части работы студент разрабатывает:
   – требования к программному продукту;
   – математические методы и алгоритмы;
   – структуру программы;
   – форматы представления данных.
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 сдать программу преподавателю, продемонстрировав её работу и исходный код.
7 По завершении работы студент оформляет 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 и модулей) должны отражать назначение именуемых объектов. Все называния 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 подробную информацию о своем назначении, авторе, решаемой задаче и предоставляемых результатах. В случае 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 отключить вывод 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 по умолчанию

1 Все целые литералы по умолчанию должны иметь тип `int`.
2 Литералы, представляющие действительные числа, по умолчанию должны иметь тип `double`.

2.4.2 Операции ввода-вывода

1 Для реализации ввода-вывода (экранного и файлового) необходимо использовать потоковые библиотеки C++.
2 Консольный ввод-вывод должен осуществляться с помощью объектов `cin` и `cout`.
3 Файловый ввод-вывод должен осуществляться с использованием 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 ввода-вывода (экран, клавиатура) необходимо предоставить возможность:
   – вводить исходные данные из файла;
   – сохранять исходные данные в файле;
   – сохранять результат работы программе в файле.

2 Полный путь к файлам в каждом случае задает пользователь.
3 В случае обнаружения файла с заданным именем по указанному пути, пользователю предлагается перезаписать существующий файл или указать новый. Данная проверка продолжается до тех пор, пока пользователь не укажет уникальное имя или не выберет перезапись.

2.4.5 Разделение на модули

1 Программа должна быть разделена (минимум) на три независимых *cpp*-файла.
   – основной модуль, осуществляющий запуск программы;
   – модуль, осуществляющий пользовательский интерфейс;
   – 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 (не менее пяти), позволяющих проверить её работоспособность на примерах с заранее известным результатом.
2 Модульные тесты должны запускаться автоматически или в случае выбора пользователем соответствующего пункта меню программы.
3 Модульные тесты должны содержать минимум пять 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 том, какой именно тест и на каком наборе данных не был пройден.

2.5 Содержание отчёта

1 В отчёт входит описание:
   – постановки задачи;
   – исходных данных;
   – особых ситуаций;
   – математических методов и алгоритмов решения задачи;
   – форматов представления данных в памяти и на внешних носителях;
   – структуры программы;
   – модулей, функций и переменных программы;
   – блок-схем алгоритмов 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 специальном разделе в форме и полноте, достаточной для восприятия другими разработчиками.

2.5.5 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.

2.5.6 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.

2.5.7 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».

2.5.8 Описание хода выполнения лабораторной работы

1 В отчёте должно содержаться подробное описание хода выполнения лабораторной работы.
2 Основное внимание должно быть уделено выполнению работы за компьютером.
3 Должны быть описаны все новые методы и приёмы, использованные в ходе выполнения лабораторной 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 лабораторная 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, распечатывается 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 лабораторной работы

1 В весеннем семестре выполняются четыре лабораторные работы из данного документа. График сдачи лабораторных работ:

Лабораторная работа №	Период защиты
1	24.02 – 05.03
2	29.03 – 09.04
3	26.04 – 07.05
4	31.05 – 11.06

2 Лабораторная работа защищается преподавателям, ведущим лекционные, лабораторные и практические занятия.
3 Для защиты необходимо иметь отчёт о проделанной работе, продемонстрировать работоспособность проекта на компьютере кафедры, ответить на вопросы.
4 Подпись за лабораторную работу выставляет преподаватель, которому работа была защищена.
5 Окончательная сдача лабораторных работ наряду с защитой курсового проекта является допуском к экзамену по предмету.

2.6.1 Окончательная сдача лабораторных работ

1 Для сдачи лабораторных работ, необходимо представить:
   – комплект отчётов по лабораторным 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 предмета, названия работ.

2.7 Пример выполнения задания (часть пунктов опущена)

При выполнении лабораторной 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 переменные:

Таблица 1 – Переменные, используемые в программе

Имя	Тип	Описание
`tries`	`int`	Число попыток в эксперименте
`in_circle`	`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 лабораторной работы

1 В ходе лабораторной 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
// Лабораторная работа № 0.
// Использование языка C++ для математических расчётов
// Расчёт числа пи методом Монте-Карло
// Студент группы NNN, Фамилия Имя Отчество. 2021 год

#include <iostream>
#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() / double(RAND_MAX);
  pt.y = rand() / double(RAND_MAX);
  return pt;
}

// Проверка нахождения точки внутри окружности
bool IsInsideCircle(const Point2D& pt) {
  const double circle_radius = 0.5;
  const Point2D circle = {0.5, 0.5};
  double dist_from_center =
    sqrt(pow(circle.x - pt.x, 2) + pow(circle.y - pt.y, 2));
  return dist_from_center < circle_radius;
}
```

```
// Обработка ввода пользователя
int ProcessInput() {
  const int lowerLimit = 100;
  const int upperLimit = 10000000;
  bool inputSuccess = false;
  int tries;
  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_ALL, "Russian");
  int in_circle = 0;
  int tries;
  tries = ProcessInput();

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

  return 0;
}
[--- Конец программы.]
```

3 Лабораторная работа № 1. Использование языка программирования C++ для математических расчётов

Первая лабораторная работа предназначена для приобретения практического опыта в создании простейшего приложения с использованием языка программирования C++, призвана:

– закрепить базовые навыки использования арифметических типов, операторов и инструкций языка программирования C++;
– рассмотреть основы ввода-вывода в консольном приложении.

Для каждого понятия (геометрические фигуры, другие объекты), использующегося в программе должна быть создана соответствующая структура данных.

При выполнении лабораторной работы нельзя использовать контейнеры и алгоритмы библиотеки STL или других сторонних библиотек.

3.1 Варианты заданий

1 Для заданной функции на заданном интервале найти требуемое значение методом половинного деления.
2 Для заданной функции на заданном интервале найти требуемое значение методом Ньютона (касательных).
3 Для заданной функции на заданном интервале найти требуемое значение методом хорд.
4 Для заданной окружности и луча в плоскости определить, пересекает ли луч окружность. Найти координаты точек пересечения.
5 Для заданных отрезков на плоскости определить, пересекаются ли они. Найти координаты точки пересечения.
6 Для заданной точки и треугольника на плоскости определить, принадлежит ли точка треугольнику.
7 В заданном интервале указать все числа, удовлетворяющие одновременно двум условиям:
   – это простые числа;
   – эти числа ряда Фибоначчи.
8 Для заданного луча и треугольника в трёхмерном пространстве определить, пересекает ли луч треугольник. Указать точку пересечения, если луч пересекает треугольник.
9 Для заданного прямоугольника и отрезка на плоскости определить, пересекаются ли они. Найти координаты точек пересечения.
10 Определить, имеют ли два заданных прямоугольника на плоскости общую область.
11 Для двух окружностей (в плоскости) определить, имеют ли они общую область. Найти площадь данной области.
12 Вычислить интеграл заданной функции методом трапеций и методом парабол.
13 Для заданной даты определить число дней, прошедших с начала года. Для заданных двух дат – число дней между ними. Предоставить возможность обработки даты как минимум в пяти различных форматах, например, 08-Feb-

2021; 08/02/21; February 8, 2021; 8/2/2021; 8 February. Программа должна самостоятельно определять, какой именно формат был использован.

4 Лабораторная работа № 2. Использование массивов

Необходимо выделить массив требуемого размера, запросив его у пользователя при запуске программы или считав из файла. В программе должны быть предусмотрены три варианта заполнения исходного массива: пользователем с клавиатуры, из файла и случайными числами.

В работе должны быть использованы методы вывода на экран с использованием различных цветов шрифта. Например, исходный и измененный массив (элементы массива) должны отличаться цветом.

При выполнении лабораторной работы нельзя использовать контейнеры и алгоритмы библиотеки STL или других сторонних библиотек.

Реализовать класс для представления массива, в деструкторе класса выполнять очистку памяти, перегрузить оператор доступа к элементу массива по индексу (`operator[]`)

4.1 Варианты заданий

1 Напишите программу, находящую «медиану» массива. То есть индекс ячейки массива, сумма элементов слева от которой минимально отличается от суммы элементов справа.
2 В трехмерном пространстве задан массив точек (тройками значений X, Y, Z) и сфера (центр и радиус). Напишите программу, выводящую точки (их координаты), которые попадают в заданную пользователем сферу.
3 Напишите программу для перестановки элементов массива так, чтобы все чётные элементы оказались в левой части массива, а все нечётные – в правой.
4 Напишите программу для перестановки элементов массива так, чтобы отрицательные элементы чередовались с положительными.
5 Напишите программу двоичного поиска заданного значения в упорядоченном массиве. Программа должна возвращать индекс искомого элемента в массиве или информировать об отсутствии искомого элемента в массиве.
6 Напишите программу, находящую в массиве вещественных чисел последовательность, имеющую максимальную сумму. Программа должна выводить начальный и конечный элемент.
7 Напишите программу, удаляющую из массива все повторяющиеся элементы.
8 Напишите программу, находящую в массиве две неубывающие последовательности максимальной длины.
9 Напишите программу, находящую в массиве значение, встречающееся чаще всего.
10 Напишите программу, находящую целочисленные значения из заданного интервала, отсутствующие в массиве.

5 Лабораторная работа № 3. Методы сортировки

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 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, 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, 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, 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, каждый столбец по убыванию.
21 Упорядочить каждую строку матрицы по возрастанию четных чисел.
22 Упорядочить каждый четный столбец по убыванию, каждую строку по убыванию.

23 В представленной матрице производить замену четных чисел по возрастанию по строкам, нечетных чисел по возрастанию по столбцам.
24 Представить шахматную доску. Упорядочить белые клетки по возрастанию по строкам, черные фигуры 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 Лабораторная работа № 4. Символьные строки

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

Цель лабораторной 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»). Придумать 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 Придумать три различных собственных 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».