Министерство образования и науки Российской Федерации
Ярославский государственный университет им. П. Г. Демидова

Н. С. Лагутина, Ю. А. Ларина, А. М. Васильев

Разработка программных приложений

Практикум

Рекомендовано
Научно-методическим советом университета
для студентов, обучающихся по направлению
Фундаментальная информатика и информационные технологии

Ярославль
ЯрГУ
2014

УДК 519.68(076)
ББК З973.2-018я73
Л 14

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2014 года.

Рецензент
кафедра вычислительных и программных систем ЯрГУ

Лагутина, Надежда Станиславовна.

Л 14 Разработка программных приложений : практикум / Н. С. Лагутина, Ю. А. Ларина, А. М. Васильев ; Яросл. гос. ун-т им. П. Г. Демидова. — Ярославль : ЯрГУ, 2014. —72 с.

Практикум содержит набор задач по разработке программных приложений для изучения основных методов программирования.

Предназначен для студентов, обучающихся по направлению 010300.62 Фундаментальная информатика и информационные технологии (дисциплины «Объектно-ориентированное программирование», «Разработка приложений для мобильных платформ», циклы Б3, ФТД), очной формы обучения. Может быть полезен для преподавателей, ведущих лабораторные занятия по программированию.

Библиогр. : 9 назв.

УДК 519.68(076)
ББК З973.2-018я73

Оглавление

Введение **4**

1. Введение в программирование **5**
1.1. Комментарии к заданиям 5
1.2. Основные приёмы структурного программирования . . . 11
1.3. Одномерные массивы 15
1.4. Двумерные массивы 17
1.5. Обработка текста 19

2. Основы объектно-ориентированного программирования **22**
2.1. Классы 22
2.2. Контейнеры. Хранение и изменение данных 28

3. Графические пользовательские приложения **37**
3.1. Проектирование интерфейса приложения 37
3.2. Однооконные приложения 46
3.3. Хранение, изменение, отображение данных 49
3.4. Разработка проектов 56

Введение

Практикум разработан для ведения лабораторных и практических занятий в целях получения студентами практических навыков создания компьютерных приложений. Сборник содержит задачи разного уровня сложности, относящиеся как к структурному, так и к объектно-ориентированному программированию и предназначен для использования студентами при выполнении лабораторных, домашних и комплексных заданий.

Предлагаемые задачи могут быть полезны при изучении любого языка программирования. Большинство из них сформулированы таким образом, что требуют от учащихся предварительной работы над архитектурой приложения, способом хранения и представления данных, что позволяет выработать важные профессиональные компетенции разработчиков программного обеспечения.

Практикум состоит из трёх частей. Первая часть содержит задачи, для решения которых достаточно знания базовых структур языка программирования. Задачи подходят для первоначального знакомства с программированием, изучения и освоения основных конструкций языка, а также алгоритмов работы с отдельными данными и массивами данных.

Вторая часть предназначена для ознакомления с разработкой приложений с помощью объектно-ориентированного подхода. В ней собраны задачи для тренировки умений и навыков создания собственных классов, а также использования различных библиотечных классов, в частности для ввода и вывода данных, для хранения и обработки однотипных данных.

В третьей части практикума собраны задачи по созданию графических пользовательских приложений. Среди них есть очень простые, дающие материал для первоначального освоения выбранной графической библиотеки, средние по сложности, в которых требуется освоить заранее заданную архитектуру приложения, и сложные, где необходимо применить умения и навыки, полученные учащимся во время обучения.

1. Введение в программирование

1.1. Комментарии к заданиям. При написании программ необходимо соблюдать требования, изложенные в этом разделе. Они соответствуют хорошему стилю программирования и позволяют получить в результате правильное, удобное, красивое и профессиональное решение задачи.

Взаимодействие с приложением

Для решения предлагаемых в этой главе задач необходимо написать приложение, взаимодействующее с пользователем посредством текстового интерфейса. Пользователь передаёт данные приложению с помощью текста, и приложение, в свою очередь, отвечает некоторой строкой. Для запуска и взаимодействия с приложением в современных операционных системах используется эмулятор терминала.

Приложение должно быть самодостаточным и предоставлять пользователю, не знакомому с текстом задания, всю необходимую информацию. Например, перед считыванием очередной строки показать приглашение ввести данные вместе с их описанием.

Все данные, вводимые пользователем, должны быть проверены на корректность. Если были предоставлены некорректные данные, то приложение должно вывести соответствующее сообщение пользователю. После этого приложение может корректно завершить свою работу или предложить пользователю ввести данные повторно.

С точки зрения программиста, ввод данных будет осуществляться через стандартный интерфейс ввода, а результат работы программы будет выводиться через стандартный вывод. Стоит отметить, что данные на стандартный ввод приложения обычно поступают только лишь после нажатия на клавишу перевода строки («Ввод», «Enter») в окне терминала.

Требования к форматированию исходного кода

Исходный код программы предназначен в первую очередь для восприятия человеком и только лишь во вторую очередь для обработки сборочным окружением [1]. При написании кода следует придерживаться ряда правил, это в значительной степени упростит работу по созданию приложения и проверки кода преподавателем. Стоит отме-

тить, что вне зависимости от выбранного стиля его обязательно надо применять для всех файлов с исходным кодом.

Данные комментарии не претендуют на полноту освещения вопроса форматирования исходного кода приложения, так как языки программирования предоставляют различный функционал с помощью различного синтаксиса. Программисту обязательно следует ознакомится с официальными требованиями от разработчиков языка или правилами, сформулированными сообществом [2], [3], [4]. При работе также необходимо использовать средства форматирования исходного кода, предоставляемые интегрированными средами разработки или внешними инструментами.

Рассмотрим конкретные рекомендации, которые могут стать основой для формирования стиля программирования.

Отступы являются важной частью исходного кода, так как они позволяют показать принадлежность конструкций к функции, условному оператору, оператору цикла, а также параметров к конкретной функции. Предлагается следующая схема расстановки отступов:

- Тело блока (функции, условного оператора) должно отстоять на один отступ больше, чем определение функции:

```
int calculate() {
    ...
    result += 15 * 6;
    ...
}
```

- При переносе выражения на новую строку следует использовать два отступа на новой строке:

```
    preliminary = 15 * ... +
        50 * 68;
    result = preliminary * preliminary;
```

- При использовании множественного ветвления обязательно следует размещать конструкции else if строго друг под другом без дополнительных отступов.

Расстановка пробелов в значительной степени влияет на восприятие исходного кода, так как элементы, выделенные пробелами, читаются гораздо легче, нежели сплошная последовательность символов. Можно привести сравнение с обычными языками, где пробелы служат для разделения слов. Без пробелов между словами было бы труд-

но быстро разделять предложение на осмысленные смысловые конструкции. Предлагается следующая схема расстановки пробелов:

- Любые операторы (приравнивания, сравнения и т. д.) должны быть отделены пробелами:

```
result = 5;
radius < 15;
```

- Добавляйте пробел перед круглой скобкой и после круглой скобки, но не ставьте пробел между ними. Исключение составляет вызов функции, в этом случае скобки выделять пробелами не стоит:

```
if (length < 15) {
    result = sqrt(length);
}
```

- Добавляйте пробел после запятой или точки с запятой, отделяющих параметры операторов и функций друг от друга:

```
for(index = 0; index < 10; index++) {
    printf("%d\n", index);
}
```

Операторы составляют основу исходного кода приложения, образуя логическую последовательность действий, выполняющуюся внутри приложения. Поэтому стоит уделить особое внимание их написанию. Следуйте таким рекомендациям:

- Точка с запятой, завершающая пустой оператор, должна размещаться на отдельной строке.
- Запрещается размещение нескольких операторов в одной строке, кроме случаев, когда это не ухудшает удобочитаемости программы.
- Разрешается написание одной строкой условного оператора, в теле которого содержится всего один вызов return. Пример:

```
if (x > 0) return x;
```

Соглашения об именах

Имена переменных и функций составляют большую часть исходного кода приложения, поэтому им следует уделить больше внимания. Предлагаемая схема наименования может описывать элементы, не присутствующие в вашем языке программирования, а также конфликтовать с официальной схемой наименования.

- Имена должны быть осмысленными словами (или словосочетаниями в значении соответствующей части речи) английского или русского языка. Не допускается смешивание английских и русских имён в исходном коде приложения.
- Имена пакетов, модулей — существительные в единственном числе в нижнем регистре, слова разделяются подчёркиваниями (`program_installer`).
- Имена классов и интерфейсов — существительные или словосочетания в значении существительных в нижнем регистре, первые буквы слов — в верхнем регистре, разделители слов не используются (`ClientInfo`); имена классов-исключений заканчиваются на слово `Exception` (`InvalidUserException`).
- Рекомендуется для классов-наследников использовать имена, в которых содержится имя суперкласса (`ModalDialog` наследник класса `Dialog`); исключение — имена классов-наследников, из которых следует, что данный класс наследует суперкласс (`Triangle` наследник класса `Figure`).
- Имена полей и локальных переменных — существительные в нижнем регистре, первые буквы слов начиная со второго — в верхнем регистре, разделители слов не используются (`fileSize`).
- Имена полей и локальных переменных не должны вводить в заблуждение относительно их типов. Пример **неправильного** именования переменной:

  ```
  String currentPlayer = player.getName();
  ```

- Имена методов — глаголы в нижнем регистре (либо словосочетания, отражающие действия), первые буквы слов начиная со второго — в верхнем регистре, разделители слов не используются (`removeFile`).
- В названии методов нужно использовать глаголы, которые как можно более точно и полно описывают то, что выполняет метод.
- Имена методов, выполняющих чтение/изменение значений полей класса, должны начинаться на `get` и `set` (`getFileAttr`, `setFileAttr`) соответственно; исключение — методы, возвращающие значения полей логического типа, они начинаются на `is` (`isOk`).
- Имена методов, выполняющих преобразование к другому типу данных, должны начинаться на `to` (`toString`).

- Имена методов, которые создают и возвращают созданный объект, должны начинаться с `create` (`createRecord`).
- Имена методов, инициализирующие поля класса или элементы графического интерфейса, должны начинаться с `init` (`initWindow`) и вызываться только из конструктора класса. Использования таких методов следует избегать.
- Имена констант — существительные или словосочетания в верхнем регистре; слова разделены подчёркиваниями (`INVALID_RECORD_COLOR`).

Требования к документированию

Документационные комментарии в исходном коде приложения позволяют разработчикам быстрее воспринимать назначение классов и функций, предоставляемых приложением. В случае разработки программной библиотеки документация для публично доступных методов становится критически важной, так как пользователь библиотеки будет видеть только её.

- Обязательными являются документационные комментарии для всех классов, полей и методов.
- Если смысл хотя бы одного из принимаемых или возвращаемых методом значений или выбрасываемых исключений не является понятным с первого прочтения, то соответствующий документационный комментарий должен содержать описание всех указанных элементов.
- В случае использования меток в операторах `break` и `continue` обязательно наличие комментария в строке, содержащей один из указанных операторов.
- Комментарии не должны содержать орфографических и пунктуационных ошибок.
- Неочевидные фрагменты кода, нацеленные на исправление отдельных не явных ситуаций, нужно выделять в отдельные функции и подробно комментировать цель их использования. Даже если это всего лишь одна строка исходного кода.

Требования к отдельным деталям реализации

В данном разделе размещены некоторые общие правила организации исходного кода, которые помогут создавать понимаемый и поддерживаемый код. Основная мысль, стоящая за этими рекомендациями, призывает создавать приложения, состоящие из маленьких элементов, взаимодействующих друг с другом. Это обусловлено следую-

щим эмпирическим правилом: сложность работы с исходным кодом возрастает по экспоненте в зависимости от размера исходного кода.

- Все элементы программы должны иметь минимально возможную область видимости.
- Запрещается использование статических полей и методов при наличии возможности достичь результата другими средствами. Если метод не использует данные класса, то его следует сделать статическим, но только если невозможно достичь результата другим путём.
- Запрещается использование методов, выполняющих две или более самостоятельные операции. Каждый такой метод подлежит разбиению на более мелкие.
- Запрещается секционирование методов (разделение методов пустыми строками или комментариями). Секционирование означает, что метод можно разделить на несколько.
- Методы, изменяющие состояние объекта, не должны возвращать статус этой операции (например, метод `insert(object)` должен выполнять только операцию вставки и не должен возвращать статус этой операции).
- Не рекомендуется использование методов, занимающих более 10 строк. Исключение — методы, инициализирующие элементы пользовательского интерфейса (прежде чем реализовывать такие длинные методы, следует подумать, нельзя ли каким-либо образом избежать большой длины метода).
- Запрещается собственная реализация средств, имеющих аналоги в стандартной библиотеке. Во всех случаях, когда возможно использование библиотечных классов или функций стандартной библиотеки, они должны быть использованы.
- Рекомендуется использование сторонних библиотек в случаях, когда это возможно. Если в задании явно требуется создать структуру данных самостоятельно, то данной рекомендацией следует пренебречь.
- Запрещается посимвольная обработка строковых данных в стиле языка C в других языках программирования. Вместо этого необходимо использовать средства стандартной библиотеки или внешней.
- Запрещается использование средств библиотек, обозначенных в документации как deprecated (устаревшие).

- Нежелательно использование средств библиотек, обозначенных в документации как устаревшие и оставленных для обратной совместимости (legacy, deprecated).
- Метод `main()`, являющийся точкой входа в программу, не должен выбрасывать исключений.
- Запрещается импортирование всех элементов, содержащихся в пакете, например `import *`, а также подключение всего пространства имён, например `using namespace std;`
- Запрещается использование вложенных классов. Исключение — вложенные классы, заменяющие анонимные классы с несколькими методами.
- Тело метода анонимного класса должно состоять только из вызова метода объемлющего класса.

1.2. Основные приёмы структурного программирования. Задания этого раздела предназначены для первоначального освоения языка программирования. Задачи достаточно просты и могут быть выполнены в течение одного практического или лабораторного занятия.

Требования к организации исходного кода

При написании приложений достаточно реализовать лишь набор функций, выполняющих поставленную задачу. Запрещено использование глобальных переменных, так как это скрывает зависимости между функциями.

Логика по обработке данных (решение самой задачи) должна быть выделена в отдельную функцию или набор функций. То же самое следует сделать для функций, осуществляющих получение данных и вывод информации пользователю.

Задачи

1. Вводятся три числа a, b и c. Проверить, будут ли они сторонами треугольника, и вычислить площадь этого треугольника.
2. Вводятся длины трёх сторон треугольника. Определить является ли треугольник прямоугольным, и вычислить величины его углов.
3. Вводятся длины трёх сторон треугольника. Определить, можно ли разместить этот треугольник в круге радиуса r.
4. Вводятся координаты точки $T(x, y)$ и координаты трёх вершин треугольника $A(x_1, y_1)$, $B(x_2, y_2)$, и $C(x_3, y_3)$. Определить, принадлежит ли она треугольнику ABC.

5. Два прямоугольника со сторонами, параллельными осям координат, заданы координатами своих левого верхнего и правого нижнего углов (координаты вводит пользователь). Для первого прямоугольника это точки $A_1(x_1, y_1)$ и $A_2(x_2, y_2)$, для второго – $A_3(x_3, y_3)$, $A_4(x_4, y_4)$. Определить, пересекаются ли данные прямоугольники, и вычислить площадь общей части, если она существует.
6. Вводятся коэффициенты уравнений двух прямых $a_1x+b_1y+c_1 = 0$ и $a_2x+b_2y+c_2 = 0$. Написать программу, определяющую, угол между ними.
7. Вводятся коэффициенты уравнения прямой $y = kx + b$ и радиус R окружности с центром в начале координат. Найти координаты точек пересечения прямой и окружности. Если точек пересечения нет или прямая касается окружности, выдать соответствующее сообщение.
8. Вводятся координаты центра круга x и y, а также его радиус R. Написать программу, вычисляющую количество точек с целочисленными координатами, находящихся в этом круге.
9. Вводятся координаты четырёх точек $A_1(x_1, y_1)$, $A_2(x_2, y_2)$, $A_3(x_3, y_3)$, $A_4(x_4, y_4)$. Определить, будут ли они вершинами параллелограмма.
10. Вводятся координаты трёх точек $A_1(x_1, y_1)$, $A_2(x_2, y_2)$ и $A_3(x_3, y_3)$. Определить, будут ли они расположены на одной прямой.
11. Вводятся координаты трёх точек на оси OX: A, B, C. Определить, какая из точек B или C расположена ближе к A.
12. Вводятся четыре числа a, b, c и d. Найти $max\{min(a, b, min(c, d)\}$.
13. Напечатать в возрастающем порядке все трёхзначные числа, в десятичной записи которых нет одинаковых цифр.
14. Вводится целое число и основание системы счисления. Определить сумму цифр числа в этой системе счисления.
15. Вводится целое число и основание системы счисления. Определить произведение цифр числа в этой системе счисления.
16. Составить программу, которая печатает таблицу умножения и сложения однозначных и двузначных натуральных чисел в шестнадцатеричной системе счисления.

17. Составить программу, которая печатает таблицу умножения и сложения однозначных натуральных чисел в заданной системе счисления.
18. Вводятся два натуральных числа m и n. Проверить, есть ли в записи числа m цифры, совпадающие с цифрами в записи числа n, и напечатать их.
19. Вводятся два натуральных числа m и n ($m \leq n$). Найти на отрезке $[m, n]$ натуральное число, имеющее наибольшее количество делителей.
20. Вводится натуральное число N. Найти и вывести все числа в интервале от 1 до $N - 1$, у которых сумма всех цифр совпадает с суммой цифр данного числа. Если таких чисел нет, то вывести слово «нет». Пример: N = 44. Числа: 17, 26, 35.
21. Вводится натуральное число N. Представьте его в виде суммы степеней двойки. Результат напечатать.
22. Натуральное число M называется совершенным, если оно равно сумме всех своих делителей, включая 1, но исключая себя. Напечатать все совершенные числа, меньшие заданного числа N.
23. Натуральное число из n цифр является числом Армстронга, если сумма его цифр возведённых в степень n, равна самому числу ($153 = 1 * 1 * 1 + 5 * 5 * 5 + 3 * 3 * 3$). Вводится натуральное число K. Определить, является ли K числом Армстронга.
24. Вводится натуральное число N. Напечатать все простые числа из диапазона $[2, N]$.
25. Вводится натуральное число N. Напечатать все простые делители натурального числа N.
26. Вводится натуральное число N. Напечатать цифры этого числа в порядке возрастания.
27. Вводятся два натуральных числа m и n. Определить, в каком из данных двух чисел больше цифр.
28. Вводится натуральное число N. Напечатать число, которое получается из исходного записью его цифр в обратном порядке (например, дано число 156, нужно получить 651).
29. Вводится натуральное число N. Напечатать цифры этого числа, выбросив из него все единицы и пятёрки, порядок остальных цифр оставить прежним.
30. Составить программу, осуществляющую перевод величин из радианной меры в градусную и наоборот. Приложение должно за-

прашивать, какой перевод нужно осуществить, и выполнять указанное действие.

31. Написать программу решения уравнения $ax^3 + bx = 0$ для произвольных чисел a, b.
32. Вводятся действительные числа a, b, c. Решить биквадратное уравнение $ax^4 + bx^2 + c = 0$, если действительных корней нет, то должно быть выдано сообщение об этом, иначе вывести корни на экран, сообщив.
33. Вводятся размеры a и b прямоугольного отверстия и размеры x, y, z кирпича. Определить, пройдёт ли кирпич через отверстие.
34. В небоскрёбе N этажей и всего один подъезд; на каждом этаже по 3 квартиры; лифт может останавливаться только на нечётных этажах. Человек садится в лифт и набирает номер нужной ему квартиры M. Написать программу, которая по введённым N и M определяет, на какой этаж должен доставить лифт пассажира?
35. Вычислить число и месяц в не високосном году по номеру дня.
36. В старояпонском календаре был принят 12-летний цикл. Годы внутри цикла носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. Написать программу, которая вводит номер года и печатает его название по старояпонскому календарю. (Справка: 1996 г. — год Крысы — начало очередного цикла.)
37. Вводится возраст человека k. Приложение должно напечатать фразу «Мне k лет», учитывая, что при некоторых значениях k слово «лет» надо заменить на слово «год» или «года». Например, 11 лет, 22 года, 51 год.
38. Вводится шестизначное натуральное число — номер билета. Определить, является ли билет счастливым по Московской или Ленинградской системе. Московский счастливый билет — если в шестизначном числе сумма первых трёх цифр равна сумме последних трёх. Ленинградский — если сумма чётных цифр билета равна сумме нечётных цифр билета.
39. Имеется часть катушки с автобусными билетами. Номер билета шестизначный. Вводятся меньший номер билета — N и больший — M. Определить количество счастливых билетов на катушке. Билет является счастливым, если сумма первых трёх его цифр равна сумме последних трёх.

40. Вводится информация о банкомате: количество купюр достоинством 50, 100 и 1000 рублей. Вводится денежная сумма s, которую должен выдать автомат. Определить, какие купюры будут выданы и сколько. Банкомат старается выдать запрашиваемую сумму максимально крупными имеющимися купюрами. Если выдать сумму невозможно, об этом выводится сообщение.

1.3. Одномерные массивы. Массив — набор однотипных компонентов, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексу.

Взаимодействие с приложением

Приложение должно обеспечивать взаимодействие с пользователем с помощью командного интерфейса, а также получение данных и вывод информации в текстовые файлы. Если приложению в качестве первого аргумента передаётся имя файла, то данные считываются из него. Если приложению в качестве второго аргумента указывается имя файла, то вывод информации осуществляется в этот файл.

В случае отсутствия входного файла приложение должно сообщать об этом пользователю и корректно завершать свою работу. Вывод информации на консоль должен отличаться в зависимости от источника ввода данных. Например, нет необходимости выводить приглашение на ввод данных, если они получаются из файла.

Формат данных во входном файле может соответствовать следующей схеме. В первой строке указывается количество элементов массива. На следующих строках размещаются элементы массива. Приветствуется использование специализированных форматов сериализации данных в текстовом формате (YAML, XML, JSON и т. д.).

Требования к организации исходного кода

При выполнении заданий необходимо использовать структуру данных — массив или динамический массив, если он входит в стандартную библиотеку языка. Обработка входных потоков данных (со стандартного потока ввода или из файла) должна быть организована в отдельных функциях. Результаты работы программы должны быть выведены в стандартный поток вывода или в файл.

Задачи

1. В массиве поменять местами пары соседних элементов.
2. Определить, сколько раз в последовательности из целых чисел меняется знак, а также номера позиций, в которых происходит смена знака.

3. Напечатать те элементы массива, индексы которых являются степенями двойки.
4. Уплотнить массив чисел, удалив нули и сдвинув влево остальные элементы.
5. Упорядочить массив целых чисел так, чтобы в начале располагались все отрицательные, а затем все положительные элементы, сохранив первоначальный порядок следования.
6. Соединить два массива A и B в массив C так, чтобы элементы массива A чередовались с элементами массива B.
7. Соединить массивы A и B в массив C так, чтобы элементы массивов A и B чередовались по 10 штук.
8. В массиве действительных чисел размещены: в нечётных элементах — значения аргумента, в чётных — соответствующие им значения функции. Напечатать элементы этого массива в виде двух параллельных столбцов (аргумент и значения функции).
9. В массиве целых чисел найти наиболее часто встречающееся число. Если таких чисел несколько, то определить наименьшее из них.
10. Разбить неупорядоченный массив целых чисел на два массива с равным количеством элементов так, чтобы элементы одного массива были бы не больше наименьшего элемента другого.
11. Дан массив действительных чисел. Сжать массив, удалив из него каждый второй элемент.
12. В массиве действительных чисел есть нулевые элементы. Найти длину максимальной последовательности подряд идущих нулей и индекс первого элемента этой последовательности.
13. Дан массив натуральных чисел. Образовать новый массив, элементами которого будут элементы исходного массива, оканчивающиеся на цифру k.
14. Дан массив целых чисел. Не используя других массивов, переставить его элементы в обратном порядке.
15. Коэффициенты многочлена $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ хранятся в массиве $A[N]$, где N — натуральное число, степень многочлена. Вычислить значение этого многочлена в точке x. Вычислить значение его производной в той же точке.
16. Дан массив целых чисел. Найти количество различных чисел в этой последовательности.

17. Заданы два одномерных массива действительных чисел с различным количеством элементов и натуральное число k. Объединить их в один массив, включив второй массив между k-м и $(k+1)$-м элементами первого.
18. Даны два упорядоченных по возрастанию массива действительных чисел. Объединить их в один массив таким образом, чтобы он был упорядоченным по убыванию.
19. Дан массив четырёхзначных натуральных чисел. Вывести на экран только те, у которых сумма первых двух цифр равна сумме двух последних.
20. Дан массив целых чисел. Найти в этом массиве минимальный элемент min и максимальный элемент max. Получить в порядке возрастания все целые числа из интервала (min, max), которые не входят в данный массив.

1.4. Двумерные массивы. Двумерный массив представляет собой такой набор данных, в котором доступ к любому элементу осуществляется по двум индексам — номеру строки и номеру столбца. Аналогично могут быть определены и многомерные массивы. Двумерный массив часто называют матрицей или таблицей.

Взаимодействие с приложением

Так же как и в предыдущем разделе, приложение должно обеспечивать взаимодействие с пользователем с помощью командного интерфейса, а также получение данных и вывод информации в текстовые файлы. Требования к работе программы аналогичные.

Формат данных во входном файле может соответствовать следующей схеме. В первой строке указываются количество строк и количество столбцов матрицы. На следующих строках размещаются элементы массива.

Требования к организации исходного кода

При выполнении заданий необходимо использовать структуру данных — двумерный массив. Обработка входных потоков данных (со стандартного потока ввода или из файла) должна быть организована в отдельных функциях. Результаты работы программы должны быть выведены в стандартный поток вывода или в файл.

Задачи

1. Дана матрица размером $n \times m$. Переставляя её строки и столбцы, добиться того, чтобы наибольший элемент (или один из них) оказался в верхнем левом углу.

2. Сформировать матрицу, состоящую из 1 и 0, которые расположены в шахматном порядке, начиная с 1. Размерность матрицы задаётся пользователем.
3. Сформировать квадратную матрицу порядка n по заданному ниже образцу:

1	1	1	.	.	.	1	1	1
0	1	1	.	.	.	1	1	0
0	0	1	.	.	.	1	0	0
.	.	.	.	.	.	.	.	.
0	1	1	.	.	.	1	0	0
1	1	1	.	.	.	1	1	1

4. Составить программу, которая заполняет квадратную матрицу порядка n натуральными числами $1, 2, 3, \cdots, n^2$, записывая их в неё «по спирали». Например, для $n = 5$ получаем матрицу:

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

5. Дана целочисленная квадратная матрица. Найти в каждой строке наибольший элемент и поменять его местами с элементом в этой строке, стоящим на главной диагонали.
6. Дана матрица размером $n \times m$. Переставляя её строки и столбцы, добиться того, чтобы наибольший элемент (или один из них) оказался в верхнем левом углу.
7. Задана матрица размером $n \times m$. Найти максимальный по модулю элемент матрицы. Переставить строки и столбцы матрицы таким образом, чтобы максимальный по модулю элемент был расположен на пересечении k-й строки и k-го столбца.
8. Дана действительная квадратная матрица порядка N (N — нечётное), все элементы которой различны. Найти наибольший элемент среди элементов, стоящих на главной и побочной диагоналях, и поменять его местами с элементом, стоящим на пересечении этих диагоналей.
9. Квадратная матрица, симметричная относительно главной диагонали, задана верхним треугольником в виде одномерного массива, элементы идут по порядку строк. Восстановить исходную матрицу и напечатать её по строкам.

10. Дана прямоугольная матрица. Найти строку с наибольшей и наименьшей суммой элементов. Вывести на печать номера найденных строк и суммы их элементов.
11. Расположить столбцы матрицы $D[M, N]$ в порядке возрастания элементов k-й строки ($1 \leq k \leq$).
12. Среди тех строк целочисленной матрицы, которые содержат только нечётные элементы, найти строку с максимальной суммой модулей элементов.
13. Упорядочить строки целочисленной прямоугольной матрицы по возрастанию количества одинаковых элементов в каждой строке. Напечатать исходную и полученную матрицы.
14. Для квадратной матрицы найти такие k, что k-я строка матрицы совпадает с k-м столбцом.
15. Уплотнить матрицу, удаляя из неё строки и столбцы, заполненные нулями.
16. Элемент матрицы называется локальным минимумом, если он строго меньше всех соседних элементов. Посчитать количество локальных минимумов в матрице и вывести их на печать вместе с индексами.
17. Характеристикой столбца целочисленной матрицы назовём сумму модулей его отрицательных нечётных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик.
18. Даны квадратная таблица $A[n, m]$ и число $m \leq n$. Для каждого квадрата размером $m \times m$ в этой таблице вычислить сумму стоящих в нем чисел.
19. Дана действительная квадратная матрица порядка $2n$. Получить новую матрицу, переставив её блоки размера $n \times n$ по часовой стрелке, начиная с блока в левом верхнем углу.
20. Дана действительная квадратная матрица порядка 2^n. Получить новую матрицу, переставив её блоки размера $n \times n$ крест-накрест. Напечатать исходную и полученную матрицы.

1.5. Обработка текста. Обработка текстов — одно из первых по значимости приложениий компьютера. Важность этой задачи заключается не только в огромных возможностях, предоставляемых компьютером для набора и редактирования, но и в том, что и взаимодействие пользователя с системой, и хранение и обмен данными в системе зачастую происходят именно в текстовой форме.

Требования к организации исходного кода

При работе с текстом следует учитывать язык, установленный в операционной системе, а также используемую кодировку символов (во многих кодировках одному символу соответствует несколько байт). Стоит использовать библиотечные вызовы, способные определить принадлежность символов к символам алфавита языка пользователя. Если стандартная библиотека языка не предоставляет соответствующих средств, то стоит использовать внешние библиотеки.

Обработку строк следует осуществлять специализированными методами, а не рассматривать строку как массив символов.

Задачи

1. Вводится предложение, слова разделены пробелами. Расположить слова в нём в порядке возрастания числа букв в словах.
2. Дан текст, слова разделены пробелами. Отбросить повторяющиеся слова. Вывести повторяющиеся слова и их количество.
3. Вводятся две строки: a и c. Определить можно ли из символов входящих в строку a, составить строку c.
4. Вводится текст телеграммы и стоимость одного слова. Слова разделены пробелами, знаки препинания считаются отдельными словами. Определить стоимость телеграммы.
5. Дан текст, слова разделены пробелами. Слова зашифрованы — каждое из них записано наоборот. Расшифровать сообщение.
6. С клавиатуры вводится целое число в десятичной системе счисления. Написать программу, реализующую вывод его представления с разделением на триады цифр. Пример:

   Число: 100000

   Форматированный вывод: 100 000

   Число:1000000

   Форматированный вывод: 1 000 000
7. Проверить каким числом является введённая с клавиатуры строка: целым или действительным.
8. Из строки символов выделить все слова и составить массив слов. Словом является последовательность букв, все остальное — разделители.
9. Вводится строка, содержащая произвольный русский текст. Определить, какие буквы и сколько раз встречаются в этом тексте. Ответ должен приводиться в грамматически правильной форме, например: а — 25 раз, к — 3 раза и т. д.

10. Составить программу преобразования натуральных чисел, записанных в римской нумерации, в десятичную систему счисления.
11. Вводится текст, в котором встречаются структуры и . Заменить каждое вхождение на <курсив>, а каждое вхождение — на <конец курсива>. В программе следует учесть, что буква i может быть как строчной, так и прописной.

2. Основы объектно-ориентированного программирования

2.1. Классы. Задания этого раздела предназначены для изучения основ объектно-ориентированного программирования, в первую очередь – для получения навыков разработки классов.

При решении предлагаемых задач следует разработать подходящие классы. Для классов требуется выбрать поля, определить конструктор, методы для изменения и получения значений полей, методы, необходимые для решения поставленной задачи. Если используются действия соответствующие арифметическим, логическим или другим операциям, то такие операции должны быть переопределены.

Все данные, вводимые пользователем, должны быть проверены на корректность.

При разработке классов программисты сталкиваются с решением ряда одинаковых задач. Эффективные подходы к решению типовых задач были объединены в шаблоны программирования. Рекомендуется ознакомится с шаблонами, освещёнными в [5].

Задачи

1. Требуется хранить и обрабатывать информацию о студентах. Для студента определяются фамилия, имя, отчество, номер группы, название факультета, средний балл успеваемости. В файле хранится информация о нескольких студентах в виде строк и чисел, разделённых пробелами. Вывести на экран и в файл списки студентов по факультетам и группам, отсортированные по фамилии, имени и отчеству. Вывести список студентов, имеющих средний балл выше заданного уровня. Определить группу с самой высокой средней успеваемостью по группе в целом.
2. Требуется хранить и обрабатывать информацию о людях. Для человека определяются фамилия, имя, отчество, пол, день, месяц и год рождения, название города, где живет. В файле хранится информация о людях в виде строк и чисел, разделённых пробелами. Вывести на экран и в файл списки однофамильцев; лю-

дей, родившихся в одном месяце, начиная с января; пенсионеров (женщины выходят на пенсию с 55 лет, мужчины — с 60).

3. Требуется хранить и обрабатывать информацию о книгах. Для книги определяются фамилия автора, название, жанр и цена. В файле хранится информация о нескольких книгах в виде строк и чисел, разделённых пробелами. Вывести на экран и в файл списки книг каждого имеющегося жанра, список книг, отсортированный по автору и названию, пронумеровать этот список. Пользователь должен ввести номера книг, которые хочет приобрести, и сумму денег для оплаты покупки. Программа должна вывести стоимость покупки и сдачу или сообщение о том, что денег для покупки не хватает.
4. Требуется хранить и обрабатывать информацию о многоугольнике. Многоугольник задается координатами его вершин. В файле хранится информация о нескольких многоугольниках. Вывести список выпуклых многоугольников, отсортировав их по возрастанию площадей. Вывести многоугольник с наименьшим периметром. Вывести список всех четырехугольников, которые являются квадратами, прямоугольниками или ромбами.
5. Требуется хранить и обрабатывать информацию о многочлене вида
$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$
Многочлен задается набором действительных коэффициентов. В файле хранится информация о двух многочленах в виде набора чисел: первое — степень многочлена, далее — коэффициенты, начиная со старшего. Вывести сумму, разность, произведение многочленов, таблицу значений каждого многочлена для аргументов в заданном диапазоне от a до b с шагом e.
6. Требуется хранить и обрабатывать информацию о многочлене вида
$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$
Многочлен задается набором действительных коэффициентов. В файле хранится информация о двух многочленах в виде набора чисел: первое — степень многочлена, далее — коэффициенты, начиная со старшего. Вывести многочлен, являющийся суммой всех многочленов. Продифференцировать все многочлены и вывести список результатов, отсортированный по убыванию степени многочленов и их коэффициентам, в виде таблицы из двух

колонок: в первой колонке — многочлен, во второй — его производная. Вывести многочлен, принимающий максимальное значение в заданной точке x.

7. Требуется хранить и обрабатывать информацию о дробях. Дробь задается числителем и знаменателем (целые числа). В файле хранится информация о дробях (пары целых чисел). Вывести попарные суммы, разности, произведения и деления всех дробей, максимальную и минимальную дроби, список дробей, отсортированный по убыванию с выделением целой части дроби, если это возможно.
8. Требуется хранить и обрабатывать информацию о комплексных числах. Комплексное число задается парой: действительной и мнимой частью. В файле хранится информация о нескольких комплексных числах — пары действительных чисел. Вывести сумму и произведение всех комплексных чисел. Вывести для всех чисел в виде таблицы из трёх колонок само число, его модуль, тригонометрическую форму числа.
9. Требуется хранить и обрабатывать информацию об n-мерном векторе целых чисел. В файле хранится информация о размерности векторов (первое число), а также координаты нескольких векторов, количество векторов произвольно. Вывести список векторов на экран, отсортировав по величине модуля. Вывести сумму всех векторов, скалярное произведение первого и последнего векторов, определить, является ли данный набор векторов линейно зависимым.
10. Требуется хранить и обрабатывать информацию о векторе в трёхмерном пространстве, задаваемом координатами двух точек — началом и концом вектора. В файле хранится информация о векторах (координаты точек), количество векторов произвольно. Вывести список векторов на экран, отсортировав по величине модуля. Информацию о векторе выводить в виде набора его трёх координат — (1,-2,3). Вывести попарные суммы, скалярные произведения векторов и углы между ними. Вывести тройки компланарных векторов.
11. Требуется хранить и обрабатывать информацию о прямых на плоскости. В файле хранится информация о прямых в виде коэффициентов общего уравнения прямой $ax + by + c = 0$. Вывести список уравнений прямых в виде $y = kx + b$ или $x = a$. Вы-

вести группы параллельных прямых, пары перпендикулярных прямых. Ввести координаты точки на плоскости и определить, каким прямым принадлежит эта точка.

12. Требуется хранить и обрабатывать информацию о прямых в трёхмерном пространстве. В файле хранится информация о прямых в виде координат двух точек, через которые проходит прямая. Вывести список уравнений прямых в виде параметрических уравнений:

$$x = x_0 + lt, y = y_0 + mt, z = z_0 + nt.$$

Здесь (x_0, y_0, z_0) — координаты точки, лежащей на прямой, а (l, m, n) — направляющий вектор. Вывести группы параллельных прямых, пары перпендикулярных прямых. Ввести координаты точки в пространстве и определить, каким прямым принадлежит эта точка.

13. Требуется хранить и обрабатывать информацию о прямой на плоскости. Вводятся коэффициенты a, b, c общего уравнения прямой $ax + by + c = 0$ и координаты произвольной точки на плоскости. Вывести уравнение прямой, параллельной данной, проходящей через заданную точку, а также уравнение прямой перпендикулярной данной, проходящей через эту же точку. В файле хранится информация о нескольких точках на плоскости. Вывести сначала координаты тех точек, которые расположены на прямой, затем тех, которые находятся над прямой, а в конце тех, которые лежат под прямой.

14. Требуется хранить и обрабатывать информацию об окружностях. В файле хранится информация о координатах центра окружностей и их радиусах (тройки действительных чисел). Вывести список окружностей, отсортированный по возрастанию радиуса; информацию об окружности выводить в виде уравнения $(x-x_0)^2+(y-y_0)^2 = r^2$, где (x_0, y_0) — координаты центра окружности, а r – её радиус. Вывести пары пересекающихся окружностей, пары касающихся окружностей, группы концентрических окружностей.

15. Требуется хранить и обрабатывать информацию о сфере. Вводятся координаты центра сферы и координаты точки, лежащей на сфере. Вывести уравнение сферы в виде $(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2$, где (x_0, y_0, z_0) — координаты центра сферы, а

r — её радиус. В файле хранится информация о нескольких точках в пространстве. Вывести сначала координаты тех точек, которые расположены на сфере, затем тех, которые находятся внутри сферы, а в конце тех, которые лежат вне сферы. Определите, как изменится эта информация, если радиус сферы увеличить или уменьшить в заданное число раз.

16. Требуется хранить и обрабатывать информацию о треугольниках. В файле хранится информация о длинах сторон треугольников — тройки действительных чисел. Выведите список треугольников, отсортированный в порядке возрастания площадей. Выведите отдельно списки прямоугольных, равнобедренных, тупоугольных треугольников, выведите информацию о парах подобных треугольников.
17. Требуется хранить и обрабатывать информацию о четырёхугольниках. В файле хранится информация о координатах четырёх вершин четырёхугольников. Выведите список четырёхугольников отсортированный в порядке возрастания периметров. Выведите отдельно списки прямоугольников и ромбов, выведите информацию о четырёхугольниках, в которые можно вписать окружность.
18. Требуется хранить и обрабатывать информацию о дате — число, месяц и год XXI века. По введённой дате определить соответствующий день недели, определить дату, наступающую через заданное число дней (вводит пользователь) и наступившую заданное число дней назад. В файле хранится информация о датах. Выведите списки дат, которые наступили раньше введённой даты и позднее.
19. Требуется хранить и обрабатывать информацию о датах — число, месяц и год XXI века. В файле хранится информация о датах. Выведите список дат, отсортированный по возрастанию, информация о дате выводится в виде 05.06.07. Выведите календарь месяца для даты, расположенной в середине этого списка, если количество элементов списка четное, то для двух средних дат.
20. Входной файл содержит размер и элементы квадратной вещественной матрицы — числа, разделённые пробелами. Программа должна читать описание матрицы из входного файла и выводить в выходной файл квадрат, куб исходной матрицы, её определитель, а также обратную матрицу, если такая существует.

21. Входной файл содержит размер и элементы квадратной целочисленной матрицы — числа, разделённые пробелами. Программа должна читать описание матрицы из входного файла и выводить в выходной файл матрицы, полученные из исходной путём поворота на 90, 180 и 270 градусов против часовой стрелки, а также определять, является ли исходная матрица магическим и латинским квадратом.
22. Неориентированный граф задан матрицей смежности: если в графе существует ребро, соединяющее i-ю и j-ю вершины, то элементы i-й строки j-го столбца и j-й строки i-го столбца матрицы содержат положительное число – длину ребра, иначе – 0. В файле хранится информация о графе: в первой строке задаётся количество вершин в графе, в последующих – элементы матрицы смежности, разделённые произвольным количеством пробелов. Программа должна читать описание графа из входного файла и выводить в выходной файл количество рёбер графа, все рёбра наибольшего и наименьшего веса, а также вектор, содержащий номера компонент связности, которым принадлежат соответствующие вершины графа.
23. Неориентированный граф задан матрицей смежности: если в графе существует ребро, соединяющее i-ю и j-ю вершины, то элементы i-й строки j-го столбца и j-й строки i-го столбца матрицы содержат положительное число – длину ребра, иначе – 0. В файле хранится информация о графе: в первой строке задаётся количество вершин в графе, в последующих – элементы матрицы смежности, разделённые произвольным количеством пробелов. Программа должна читать описание графа из входного файла и выводить в выходной файл количество дуг графа, количество дуг, входящих в каждую вершину графа, количество дуг, выходящих из них, кратчайшие пути от каждой вершины до всех остальных достижимых из неё.
24. Ориентированный граф задан набором рёбер. Каждое ребро имеет длину и вес, задаваемые целыми положительными числами. В файле хранится информация о графе: в первой строке задаётся количество вершин в графе, в последующих – четвёрки чисел: номера вершин, образующих ребро, длину и вес ребра соответственно. Числа могут быть разделены произвольным количеством пробелов. Программа должна читать описание графа

из входного файла и выводить в выходной файл матрицу длин кратчайших путей в графе; количество дуг, исходящих из каждой вершины графа; рёбра, имеющие наибольший и наименьший вес.

25. Входной файл содержит информацию о лабиринте в виде матрицы, в которой разными символами обозначены пустое пространство и стены (например, 0 и 1), а также информацию о начальной и конечной позициях (двумя другими символами). Программа должна найти в лабиринте путь, если это возможно, от начальной позиции до конечной и вывести результат в файл в виде матрицы аналогично входным данным, где найденный путь отмечен любым удобным символом. Путь не обязательно должен быть кратчайшим. Если найти путь невозможно, в файл выводится соответствующее сообщение. Формат входного файла: размеры матрицы, описывающей лабиринт, матрица.
26. Входной файл содержит информацию о количестве городов, которые должен объехать коммивояжер, и о расстояниях между ними. Информация хранится в виде матрицы смежности соответствующего графа (города – вершины графа, расстояния – веса ребер). Программа должна найти в графе гамильтонов цикл (обход всех вершин графа) наименьшей длины, если это возможно, и вывести результат в файл в виде последовательности обхода городов. Если найти путь невозможно, в файл выводится соответствующее сообщение. Формат входного файла: количество вершин (городов), матрица смежности.
27. Требуется разработать калькулятор для бесконечно больших целых чисел. Входной файл содержит информацию о паре больших чисел и операции, которую необходимо над ними произвести. Операции: сложение, вычитание, умножение, целочисленное деление, нахождение наибольшего общего делителя, наименьшего общего кратного. Программа должна выполнять указанную операцию и выводить результат в файл. Формат входного файла: число, операция, число.

2.2. Контейнеры. Хранение и изменение данных. Для представления данных следует создать необходимые классы и использовать подходящие стандартные контейнерные классы. Предполагается, что приложение взаимодействует с пользователем посредством текстового интерфейса.

Задачи

1. Написать программу, моделирующую работу автобусного парка. Сведения о каждом маршруте содержат: номер, длину, количество автобусов, необходимых для обеспечения движения по маршруту. Сведения о каждом автобусе содержат: номер автобуса, расход бензина (в литрах на километр), фамилию и имя водителя, номер маршрута. Программа должна создавать список маршрутов, список автобусов, находящихся в парке. Начальное формирование данных осуществляется из файла (или из двух файлов). С помощью меню необходимо обеспечить следующие функции:
    a) добавление автобуса из парка в список выехавших на маршрут (при этом автобус из соответствующего списка исключается);
    b) исключение автобуса из списка выехавших на маршрут и возвращение его в парк;
    c) просмотр содержимого любого списка;
    d) добавление маршрута;
    e) добавление автобуса;
    f) достаточность автобусов на заданном маршруте;
    g) поиск по фамилии водителя соответствующего ему автобуса и маршрута;
    h) определение затрат бензина на всех маршрутах и отдельно для каждого.
2. Написать программу, моделирующую учет заявок на авиабилеты. Сведения о каждом рейсе содержат: номер, пункт отправления, пункт назначения, дату и время вылета, тип самолета, стоимость билета. Сведения о каждой заявке содержат: пункт отправления, пункт назначения, дату и время вылета фамилию и имя пассажира. Программа должна создавать список рейсов, список заявок. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:
    a) добавление рейса;
    b) удаление рейса;

c) добавление заявки;

d) для каждой заявки подбор подходящего рейса;

e) вывод всех заявок по заданным дате и времени вылета;

f) вывод всех рейсов по заданным пунктам отправления и назначения;

g) удаление всех заявок по заданной дате.

3. Написать программу, моделирующую информационную систему на железнодорожном вокзале. Сведения о каждом поезде содержат: номер, пункт отправления, пункт назначения, время отправления, время прибытия, стоимость билета. Программа должна создавать список поездов. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

   a) добавление поезда;

   b) удаление поезда;

   c) по пунктам отправления и назначения разработать маршрут поездки, возможно с пересадками;

   d) вывод всех поездов по заданному пункту отправления и времени прибытия;

   e) вывод всех поездов по заданному пункту отправления и назначения.

4. Написать программу, моделирующую учет книг в библиотеке. Сведения о каждой книге содержат: фамилию и инициалы автора, название, номер, жанр, возраст читателя, количество книг в библиотеке. Сведения о каждом читателе содержат: фамилию, имя, отчество, возраст, список книг с указанием даты возврата. Программа должна создавать список книг и список читателей. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

   a) добавление книги или читателя;

   b) удаление книги или читателя;

   c) подбор книги для данного читателя по заданному жанру или автору (с учетом возраста читателя);

d) выдачу книги читателю, при этом должно учитываться количество экземпляров книги, оставшихся в библиотеке;

e) возврат книги, за не вовремя сданную книгу рассчитывается штраф (например, 1 рубль за каждый день);

f) вывод списка книг, упорядоченного по фамилии автора;

g) вывод списка книг, упорядоченного по названию.

5. Написать программу, моделирующую процесс купли-продажи квартир. Сведения о каждой квартире содержат: метраж, количество комнат, адрес (район, улицу, дом), этаж, вид дома (панельный, кирпичный) и количество этажей, стоимость. Сведения о каждой заявке на покупку квартиры содержат: количество комнат, район, вид дома. Программа должна создавать список квартир и заявок. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

   a) добавление квартиры или заявки;

   b) удаление квартиры или заявки;

   c) поиск по каждой заявке подходящей квартиры, при этом сведения о подобранных парах выводятся на экран и в отдельный файл, а соответствующие элементы из списков удаляются;

   d) вывод на экран всех продаваемых квартир, отсортированных по районам;

   e) вывод на экран всех заявок, отсортированных по количеству комнат;

   f) вывод на экран всех квартир по заданному диапазону стоимости.

6. Написать программу перевода слов с русского языка на английский и наоборот. Каждому слову должен соответствовать его перевод с возможными синонимами. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

   a) выбор режима работы: перевод с русского языка на английский или с английского на русский;

b) по данному слову выводить его перевод с синонимами;

c) выводить список слов с переводом на заданную букву или буквосочетание;

d) по последовательности слов вывести ее перевод.

7. Написать программу, моделирующую процесс обмена квартир. Сведения о каждой квартире содержат: метраж, количество комнат, адрес (район, улицу, дом), этаж, вид дома (панельный, кирпичный) и количество этажей, стоимость, параметры обмена (район, улица, этаж). Программа должна создавать список квартир. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

   a) добавление квартиры;

   b) удаление квартиры;

   c) поиск варианта обмена для заданной квартиры с расчетом доплаты, при этом сведения о подобранных вариантах выводятся на экран, а вариант, выбранный пользователем, записывается в отдельный файл, соответствующие элементы из списков удаляются;

   d) вывод на экран всех квартир, отсортированных по районам;

   e) вывод на экран всех квартир, отсортированных по количеству комнат;

   f) вывод на экран всех квартир по заданному диапазону стоимости.

8. Написать программу, играющую роль записной книжки. Сведения о каждой записи содержат: имя, фамилию, отчество человека, его сотовый и домашний телефоны, адрес, статус (друг, коллега по работе, парикмахер и т. п.). Программа должна создавать список знакомых. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

   a) добавление знакомого;

   b) удаление знакомого;

   c) изменение адреса или телефона;

d) по заданному статусу и событию сформировать список приглашенных лиц, для каждого создать отдельный файл с обращением и текстом приглашения или поздравления, по желанию пользователя в начале письма вставляется домашний адрес;

e) вывести на экран список знакомых, отсортированных по фамилии или по статусу.

9. Написать программу, моделирующую работу с вкладами в банке. Сведения о каждом вкладе содержат: название, годовой процент, возможность пополнения счета. Сведения о каждом вкладчике содержат: фамилию, имя, отчество, номер счета, вид вклада, сумму, дату открытия. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

   a) добавление вклада или вкладчика;

   b) удаление вклада или вкладчика;

   c) пополнение счета, если это возможно, по имени вкладчика и номеру счета;

   d) снятие процентов с вклада;

   e) закрытие счета с выплатой процентов;

   f) вывод на экран списка вкладчиков, отсортированных по фамилии;

   g) вывод на экран списка вкладчиков по заданному диапазону размера вклада.

10. Написать программу, моделирующую работу отдела кадров. Сведения о каждой должности содержат: название, отдел, оклад, требования к должности (возраст, образование, специальность), количество рабочих мест. Сведения о каждом человеке содержат: фамилию, имя, отчество, возраст, образование, специальность. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

    a) добавление должности или человека;

    b) удаление должности или человека;

c) подбор для каждого человека подходящей должности, если это возможно, то включение его в список сотрудников;

d) вывод на экран списка сотрудников, отсортированных по фамилии или должности;

e) вывод на экран списка вакансий (не занятых должностей);

f) расчет ежемесячной заработной платы и вывод соответствующей ведомости в файл.

11. Написать программу, моделирующую работу с коммунальными платежами. Сведения о каждом счете содержат: фамилию, имя, отчество человека, осуществляющего платеж, сумму платежа, внесенную сумму оплаты, вид платежа (квартплата, плата за электроэнергию, плата за телефон). Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

    a) добавление платежа;

    b) удаление платежа;

    c) вывод на экран всех счетов по имени человека с подсчетом общей суммы платежей;

    d) вывод на экран списка должников, отсортированного по алфавиту;

    e) вывод на экран списка должников с заданным диапазоном размера долга;

    f) возможность оплаты выбранного счета.

12. Написать программу, моделирующую работу туристической фирмы. Сведения о каждом туре содержат: название страны или города, список достопримечательностей, количество дней, вид проезда (автобус, поезд, самолет, теплоход), стоимость, количество туристов. Сведения о каждом туристе содержат: фамилию, имя, отчество, пожелания (страна, достопримечательность, вид проезда, диапазон приемлемой стоимости). Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

    a) добавление тура или туриста;

    b) удаление тура или туриста;

c) формирование туристических групп по имеющимся данным;

d) вывод на экран сведений о подобранном туре для заданного туриста;

e) вывод на экран туров, отобранных по стране или по достопримечательности, или по стоимости;

f) вывод на экран списка туристов, сформированного для заданного тура, список должен быть отсортирован по фамилии.

13. Написать программу, моделирующую работу с расписанием занятий студентов. Сведения о каждом элементе расписания содержат: день недели, время или номер пары, название предмета, номер аудитории, название группы студентов. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

    a) добавление элемента расписания, с проверкой корректности (можно ли конкретный предмет поставить для данной группы в заданное время в указанной аудитории);

    b) удаление элемента;

    c) изменение элемента;

    d) вывод на экран расписания на заданный день недели;

    e) вывод на экран расписания для заданной группы;

    f) вывод на экран всего расписания, отсортированного по дням недели и времени (номеру пары).

14. Написать программу, моделирующую работу книжного магазина. Сведения о каждой книге содержат: фамилию автора, название, жанр, цену, количество книг в магазине. Кроме книг, в магазине продаются канцелярские товары. Информация о каждом товаре следующая: название, цена, количество единиц товара в магазине. Программа должна сообщать о товарах в магазине и создавать список покупок. Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

    a) добавление книги или канцелярской принадлежности;

b) удаление книги или канцелярской принадлежности;

c) поиск книги по заданному жанру или названию;

d) поиск товара по названию;

e) вывод списка всех книг и товаров, каждый список упорядочен по названию;

f) выбор товара для покупки;

g) вывод списка покупок с информацией о цене каждого товара и об итоговой стоимости покупки.

15. Написать программу, моделирующую работу отделения телекоммуникационной компании, обслуживающей домашние телефоны. Сведения об абоненте содержат: фамилию, имя, шестизначный номер телефона, тарифный план, общее количество минут телефонных разговоров за последний месяц.

Тарифы	Безлимитный	Комбинированный	Повременный
Объем минут, включенных в ежемесячную плату	не ограничен	350	0
Ежемесячная плата, руб.	420,00	300,00	180,00
Стоимость минуты сверх включенного объема за 1 минуту, руб.	0,00	0,34	0,38

Начальное формирование данных осуществляется из файла (или файлов). С помощью меню необходимо обеспечить следующие функции:

a) добавление абонента с предоставлением ему индивидуального номера;

b) удаление абонента;

c) поиск абонента по имени или номеру телефона;

d) вывод списка всех абонентов;

e) вывод извещения на оплату за последний месяц для указанного абонента;

f) вывод списка абонентов с заданным тарифным планом.

3. Графические пользовательские приложения

3.1. Проектирование интерфейса приложения. Цель любого приложения – удобная автоматизация некоторого набора процессов, выполняемых пользователем. Для создания удобного приложения необходимо тщательно подойти к проектированию взаимодействия между пользователем и приложением.

Разработчики операционных систем, а также оконных окружений предоставляют руководства по проектированию взаимодействия пользователей и приложений для данной системы (Windows [6], OS X [7], KDE [8], GNOME [9]). Если приложение удовлетворяет требованиям руководств, тогда пользователю, знакомому с другими приложениями в операционной системе, будет легче научиться взаимодействовать с вашим приложением.

Стоит отметить, что некоторые мобильные операционные системы не допускают установку приложений, чей интерфейс не согласуется с официальными руководствами. Такая практика может в дальнейшем распространиться и на настольные операционные системы.

В пособии мы постараемся уделить внимание базовым подходам к проектированию пользовательского интерфейса, а также осветим основные элементы пользовательского интерфейса.

Модель решения задачи

При разработке приложения всегда стоит учитывать то, что пользователь уже представляет, как решить конкретную задачу, которую решает разрабатываемое приложение. Данное представление формируется на основе предыдущего опыта пользователя, возникшего в результате взаимодействия с реальными объектами, работой с приложениями на данной платформе, а также компьютерами в целом. Например, пользователи имеют представление о написании и отправке писем в реальной жизни, а также использовали различное программное обеспечение для обмена электронными письмами. Поэтому у пользователя накапливаются некоторые ожидания относительно функций приложения: возможность составления писем, выбора адресата и от-

правки письма. Приложение для отправки писем, конфликтующее с моделью поведения пользователя будет, как минимум, неудобным для работы с ним, а, скорее всего, просто не будет использовано.

Перед началом решения задачи постарайтесь выяснить модель решения задачи, сформированную у вашей целевой аудитории. В модель могут входить следующие компоненты:

- Метафоры или объекты, которыми оперирует пользователь при решении задачи. Например, для редактора фотографий к объектам могут быть отнесены фотографии, альбомы, фотоаппараты.
- Конкретные задачи, которые должно и не должно решать ваше приложение. Например, почтовый клиент должен отправлять и принимать почтовые сообщения, но не должен конвертировать изображения из одного формата в другой.
- Ключевые сценарии использования приложения, описывающие наиболее важные аспекты взаимодействия пользователя и приложения.

Чётко сформулированная модель решения задачи позволит создать действительно удобный пользовательский интерфейс.

Явные и неявные действия

Каждое действие в приложении манипулирует некоторым объектом. Сначала пользователь видит интересующий его объект на экране. Затем он выбирает или обозначает объект. В конце пользователь выполняет некоторое действие, или выбирая команду из меню приложения, или посредством прямого действия на объект с помощью мыши или другого устройства. Такая схема взаимодействия с объектом порождает две парадигмы взаимодействия с объектом: явные и неявные действия.

Явные действия чётко обозначают результат модификации объекта. Например, меню отображают список команд, которые можно выполнить на выделенном объекте. Название команды меню чётко показывает, какое действие надо выполнить, а также саму возможность его выполнения (пункт меню активен или обесцвечен) в данном контексте. Явные действия не требуют от пользователя запоминать список команд, которые могут быть выполнены на объекте.

Неявные действия сообщают результат с помощью визуальных подсказок или контекста. Примером такого действия является операция перетаскивания (drag-and-drop). Перетаскивание одного объекта на другой указывает на тип взаимоотношения объектов и обозначает

действие, которое будет выполнено. К примеру, перетаскивание значка файла на значок корзины идентифицирует действие удаления файла. Для того чтобы пользователь мог выполнить неявное действие, он должен легко опознавать вовлечённые объекты, необходимые для выполнения операции, а также последствия своего действия.

При проектировании интерфейса пользовательского приложения используйте эти парадигмы взаимодействия с объектами в соответствии с моделью решения задачи. Если вовлечённые объекты предоставляют достаточно контекстной информации, тогда можете использовать неявные действия для обозначения операции. В большинстве же случаев объект не обладает данным свойством, поэтому лучше использовать явные действия.

Контроль со стороны пользователя

Дайте возможность пользователю, а не приложению, инициировать и контролировать действия. Некоторые приложения пытаются помочь, предоставляя только те альтернативы, которые кажутся разработчику подходящими для пользователя или защищающими его от слишком сложных действий. Такое решение ставит приложение, а не пользователя, ответственным за выполнение действий. Подобное поведение может быть полезно только для неопытных пользователей или во время обучения использованию приложения. Обязательно согласуйте поведение приложения с опытом целевой аудитории приложения.

Ключевая задача – предоставить пользователям необходимые возможности, при этом помогая им избежать опасных, разрушительных действий. Если пользователь может по неосторожности уничтожить данные, приложение должно показывать предупреждающее сообщение, но предложить пользователю продолжить, если он пожелает.

Обратная связь и взаимодействие

Обратная связь приложения включает в себя гораздо больше, чем отображение сообщений в случае ошибки или неправильного действия. Наоборот, приложение должно информировать пользователя о том, что происходит внутри приложения и активно содействовать пользователю.

Когда пользователь инициирует действие, приложение должно показать, что команда была получена и она начала выполняться. Пользователю необходимо знать, что действие выполняется. Если действие не может быть выполнено, то необходимо сообщить причину и воз-

можные шаги по достижению нужного результата. Отклик на действие может быть выполнен в виде сообщения в статусной строке или в виде анимации.

Обратная связь должна быть организована простым и понятным для пользователя способом. Например, сообщение об ошибке должно чётко отражать причину возникновения ошибки («Недостаточно места на диске для сохранения документа»), а также возможное действие для решения проблемы («Попытайтесь сохранить документ на другом диске»).

Согласованность интерфейса

Согласованность интерфейса позволяет пользователю переносить знания и умения с одного приложения на другое. Используйте стандартные элементы интерфейса, доступные в операционной системе, и получите дополнительное преимущество в виде простоты использования приложения. Используйте следующие проверочные вопросы, чтобы улучшить согласованность вашего интерфейса:

- Интерфейс согласован с требованиями операционной системы? Например, используются ли сочетания клавиш, принятые в операционной системе? Использует ли оно стандартные решения для типичных задач?
- Согласован ли интерфейс внутри приложения? Используется ли единая терминология в надписях внутри приложения? Означают ли иконки одно и то же действие во всех местах, в которых оно используется? Используются ли одинаковые компоненты интерфейса и одинаковая компоновка в окнах и диалогах?
- Согласован ли интерфейс с предыдущими версиями приложения? Используется ли одинаковая терминология с предыдущей версией? Изменились ли фундаментальные концепции взаимодействия с приложением?
- Согласован ли интерфейс с ожиданиями пользователя? Реализует ли он требования пользователя без использования странных последовательностей действий? Согласуется ли интерфейс с моделью решения задачи?

Удовлетворить потребности всех пользователей и сохранить интерфейс согласованным является сверхсложной задачей, особенно если ваше приложение будет использовано широкой аудиторией. Данную проблему можно решить путём тщательного анализа проблем со-

Рис. 3.1. Набросок главного окна приложения с панелью меню

гласования интерфейса в контексте требований целевой аудитории и их потребностей.

Стандартные компоненты интерфейса настольного приложения

Для проектирования интерфейса приложения необходимо ознакомиться со всеми компонентами, предоставляемыми графической библиотекой, используемой для разработки. Это позволит выбирать компоненты наиболее подходящие для решения поставленной перед вами задачи.

Далее рассмотрим ряд компонент и их назначение в интерфейсе приложения.

- Панель меню. Панель располагается в верхней части главного окна приложения и предоставляет доступ ко всем командам и к большинству настроек приложения. Она содержит в себе списки функций, а также подменю, которые открываются по нажатию на элемент меню. Пример главного окна с панелью меню изображён на рис. 3.1.
- Контекстное меню. Меню состоит из списка действий, которые можно выполнить над объектом приложения. Контекстное меню обычно скрыто от пользователя и появляется при нажатии на правую клавишу «мыши». Пример окна с контекстным меню изображен на рис. 3.2.

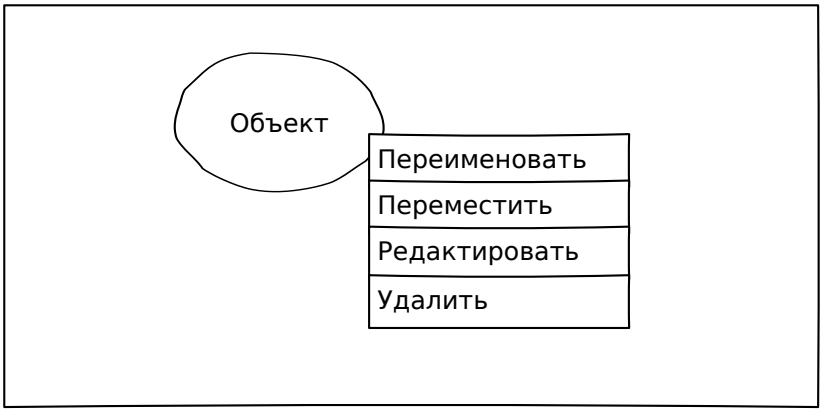

Рис. 3.2. Набросок контекстного меню в приложении

- Панель инструментов является графическим представлением команд, что позволяет оптимизировать быстрый доступ к этим командам. Обычно элементы панели инструментов содержат в себе действия из меню приложения. Пример главного окна с панелью инструментов изображен на рис. 3.3.
- Кнопка инициализирует действие, когда пользователь нажимает на неё. Кнопки являются самым простым элементом, но в то же самое время пользователю понятно, как её использовать.
- Кнопка-переключатель выглядит как обычная кнопка, но используется для показа или изменения состояния, а не для инициирования действия. Переключатель имеет два состояния: установлен и снят. Обычно состояние отображается с помощью различных изображений. Кнопка и кнопка-переключатель изображены на рис. 3.4.
- Group Box представляет из себя прямоугольную рамку с меткой, которая окружает набор родственных компонент приложения. Пример части приложения, использующего группировку элементов, изображён на рис. 3.5.
- Список элементов служит для показа набора однородных элементов. Он позволяет ориентироваться среди набора элементов без дополнительных элементов. Также список может быть использован для выделения одного или нескольких элементов.

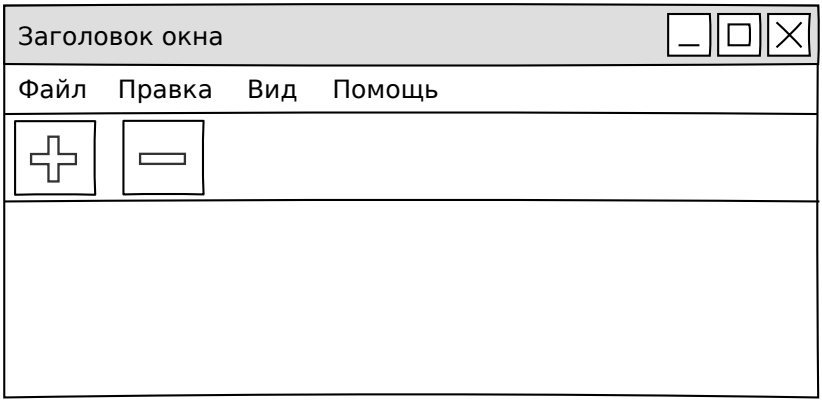

Рис. 3.3. Набросок главного окна приложения с панелью инструментов

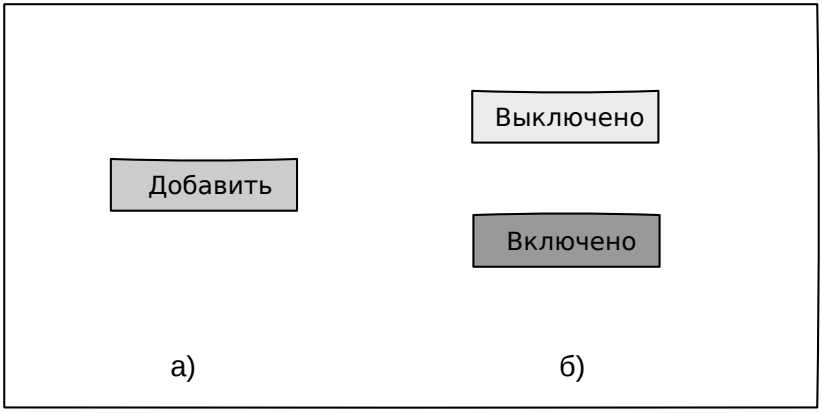

Рис. 3.4. Набросок кнопки (а) и кнопки-переключателя (б)

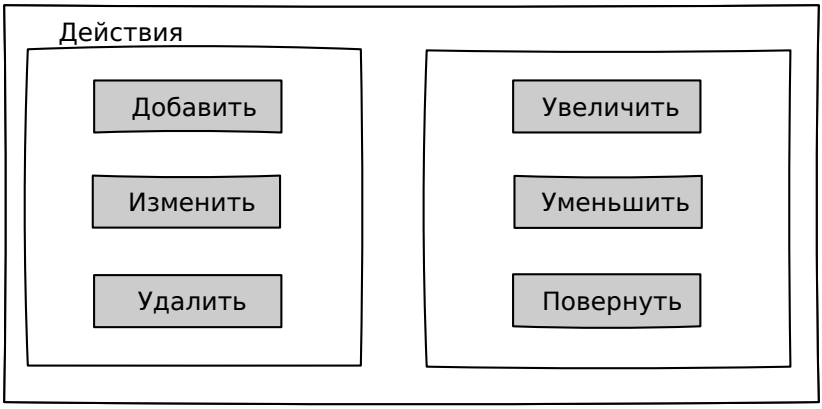

Рис. 3.5. Набросок группировок элементов с меткой и без неё

Пример интерфейса приложения, использующего список элементов, изображён на рис. 3.6.

Рекомендации по проектированию диалогов

Диалоговые окна являются вторичными окнами приложения, предоставляя возможности по выполнению команды, запроса данных от пользователя или предоставления информации пользователю. Диалоги могут быть сделаны модальными, в этом случае пользователь сначала обязан завершить действие внутри диалога, только после этого он сможет вернуться к главному окну приложения.

Используйте следующие рекомендации для использования диалогов во время проектирования интерфейса приложения.

- Используйте диалоги для структурирования взаимодействия с пользователем. Например, такие действия, как открытие, закрытие и сохранение файла, подразумевают ввод пользователем имени файла или подтверждение действия. В частности, пользователь использует диалоговые окна для конфигурации действия.
- Не используйте диалоги, если процесс не должен прерываться. В этом случае используйте компоненты, встроенные в основное окно.
- Сопоставьте альтернативные пути взаимодействия с пользователем, такие как всплывающие подсказки.
- Используйте стандартный диалог, если это возможно.

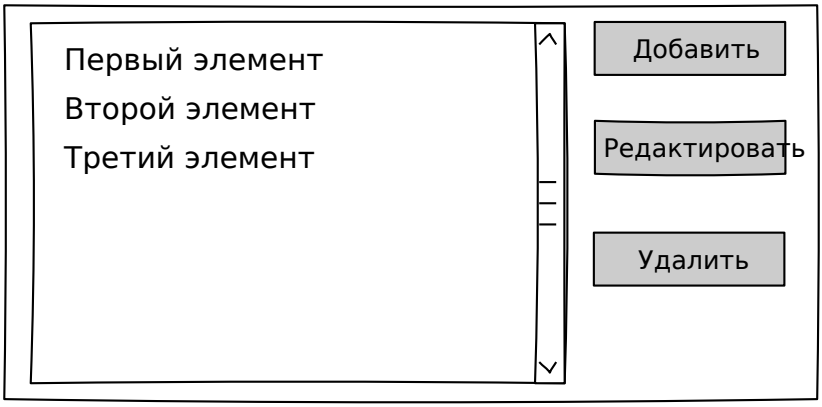

Рис. 3.6. Набросок элемента интерфейса со списком элементов

Следуйте следующим рекомендациям при проектировании поведения диалога:

- Не применяйте диалогов, которые требуют прокрутки для навигации внутри них.
- Не добавляйте панель меню и панель статуса в диалоги.
- Не показывайте больше одного диалогового окна и одного главного окна.
- Устанавливайте фокус на кнопку подтверждения для диалогов ввода информации и подтверждения действия. Для диалогов, подтверждающих разрушительное действие, фокус следует устанавливать на кнопку отмены действия.
- Не стоит вызывать вложенные диалоги из других диалогов, особенно в случае модальных диалогов.
- Не стоит создавать диалогов, состоящих только лишь из одной или двух опций. Следует внедрять соответствующие элементы в основное окно.
- Используйте модальные диалоги только лишь в том случае, если одновременная работа с основным приложением может привести к потере данных или к другим плохим последствиям.
- Предоставьте понятный способ закрытия диалога, к примеру используйте стандартную кнопку «Отмена».

3.2. Однооконные приложения. В этом разделе предлагается разработать простое графическое приложение, состоящее только из одного окна, содержащего виджеты для ввода и вывода данных.

Задачи

1. Определить знак зодиака пользователя. Окно приложения содержит поле для ввода даты рождения и кнопку «OK». После нажатия на кнопку появляется сообщение о знаке зодиака пользователя.
2. Определить синонимы слова. Информация о синонимах хранится в программе в любом удобном виде, количество слов не менее двадцати, каждому соответствует 1-2 синонима. Окно приложения содержит поле для выбора или ввода слова и кнопку «OK». После нажатия на кнопку появляется сообщение о синонимах слова или о том, что слово не знакомо.
3. Определить название года по японскому календарю. В этом календаре был принят 12-летний цикл. Годы внутри цикла носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. Окно приложения содержит поле для ввода года и кнопку «OK». После нажатия на кнопку появляется сообщение о названии года по японскому календарю.
4. Определить столицу страны. Окно приложения содержит поле для выбора или ввода названия страны (не менее 30 вариантов) и кнопку «OK». После нажатия на кнопку появляется сообщение о названии столицы этой страны или о том, что информации об этом нет.
5. Определить по названию продукта, является ли он фруктом, овощем или ягодой. Окно приложения содержит поле для выбора или ввода названия продукта (не менее 30 вариантов) и кнопку «OK». После нажатия на кнопку в виджете появляется сообщение: «фрукт», «овощ» или «ягода», либо выдаётся сообщение о том, что информации нет.
6. Определить стоимость весового товара. Для каждого товара известна цена одного килограмма в рублях. Окно приложения содержит поле для выбора товара (не менее 20 вариантов), поле для ввода веса в граммах и кнопку «OK». После нажатия на кнопку появляется сообщение о стоимости товара.

7. Определить флаг заданной страны. Окно приложения содержит поле для выбора или ввода названия страны (не менее 30 вариантов), виджет для отображения картинки с флагом и кнопку «OK». После нажатия на кнопку в виджете появляется соответствующий рисунок или выдаётся сообщение о том, что информации о флаге страны нет.
8. Вывести таблицу умножения для заданного однозначного числа. Окно приложения содержит поле для ввода числа, виджет для отображения таблицы умножения и кнопку «OK». После нажатия на кнопку в виджете появляется соответствующая информация.
9. Определить эмблему автомобиля. Окно приложения содержит поле для выбора или ввода названия марки автомобиля (не менее 20 вариантов), виджет для отображения картинки с эмблемой и кнопку «OK». После нажатия на кнопку в виджете появляется соответствующий рисунок или выдаётся сообщение о том, что информации нет.
10. Вывести картинку с изображением фрукта. Окно приложения содержит поле для выбора или ввода названия фрукта (не менее 20 вариантов), виджет для отображения картинки и кнопку «OK». После нажатия на кнопку в виджете появляется соответствующий рисунок или выдаётся сообщение о том, что информации нет.
11. Вывести таблицу истинности для заданной логической функции. Окно приложения содержит поле для выбора логической функции (не менее 10 вариантов), виджет для отображения таблицы истинности и кнопку «OK». После нажатия на кнопку в виджете появляется соответствующая таблица.
12. Определить принадлежность животного к одному из классов (млекопитающие, птицы, рыбы, земноводные, пресмыкающиеся, насекомые). Окно приложения содержит поле для выбора или ввода названия животного (не менее 30 вариантов) и кнопку «OK». После нажатия на кнопку появляется сообщение о классе, к которому принадлежит животное или сообщение о том, что информации об этом нет.
13. Найти определение введённого математического термина. Окно приложения содержит поле для выбора или ввода термина (не менее 30 вариантов), виджет для отображения текста опреде-

ления и кнопку «OK». После нажатия на кнопку в виджете появляется соответствующая информация или выдаётся сообщение о том, что сведений о термине нет.

14. Решить квадратное уравнение. Окно приложения содержит три поля для ввода коэффициентов уравнения, виджет в котором выводится значение дискриминанта и корней, если они есть и кнопку «OK». После нажатия на кнопку в виджете появляется информация о решении уравнения.
15. Составить список блюд для завтрака, обеда, полдника или ужина. Имеется список блюд (не менее 30), каждое из которых относится к одной из следующих категорий: первое, второе, салат, десерт, напиток. Завтрак должен состоять из второго или салата и напитка, обед – из первого, второго, десерта и напитка, полдник – из десерта и напитка, ужин – из салата, второго и напитка. Окно приложения содержит поле для ввода одного из четырёх вариантов: завтрак, обед, полдник или ужин и кнопку «OK». После нажатия на кнопку предлагается меню, случайно выбранное из имеющихся блюд, соответствующее описанным условиям.
16. Отображать время в виде двух чисел: часов и минут. Главное окно приложения содержит два поля для корректировки времени, виджет, отображающий время и картинку, соответствующую времени суток, в зависимости от текущего часа: утро, день, вечер, ночь; кнопку «OK». После корректировки часов и минут и нажатия на кнопку изменяется содержимое виджета.
17. Отображать текущую дату в виде числа и названия месяца. Главное окно приложения содержит два поля для корректировки числа и месяца, виджет, отображающий дату и картинку, соответствующую времени года, в зависимости от текущего месяца: зима, весна, лето, осень; кнопку «OK». После корректировки даты и нажатия на кнопку изменяется содержимое виджета.
18. Отображать точку с заданными координатами на координатной плоскости. Главное окно приложения содержит два поля для ввода координат точки (числа от -20 до 20), виджет, отображающий точку на координатной плоскости, и кнопку «OK». После ввода данных и нажатия на кнопку рисунок в виджете изменяется в соответствии с введёнными координатами.
19. Нарисовать окружность. Окно приложения содержит поле для ввода радиуса окружности, виджет, где будет нарисована окруж-

ность, и кнопку «OK». После нажатия на кнопку окружность отображается в виджете.

20. Нарисовать многоэтажный дом в виде прямоугольника, на каждом этаже по пять окон, на первом вместо среднего окна дверь подъезда. Окно приложения содержит поле для ввода количества этажей, виджет, где будет нарисован дом, и кнопку «OK». После нажатия на кнопку дом отображается в виджете.
21. Нарисовать заданное количество окружностей. Главное окно приложения содержит поле для ввода количества окружностей, виджет, отображающий эти окружности, и кнопку «OK». После ввода данных и нажатия на кнопку в виджете рисуется заданное количество окружностей.
22. Нарисовать схему купейного вагона поезда (девять купе, по четыре места в каждом), места пронумерованы. Разным цветом отметить свободные и занятые места вагона, в начале работы программы все места свободны. Главное окно приложения содержит поле для ввода номера места, которое нужно занять, виджет, отображающий схему вагона, и кнопку «OK». После ввода номера и нажатия на кнопку соответствующее место на схеме вагона перекрашивается, если место уже занято, выдаётся сообщение.
23. Нарисовать пешку на шахматном поле. Главное окно приложения содержит два поля для ввода координат пешки (букву от A до H и цифру от 1 до 8), виджет, отображающий пешку на шахматной доске, и кнопку «OK». После ввода данных и нажатия на кнопку рисунок в виджете изменяется в соответствии с введёнными координатами фигуры.
24. Нарисовать светофор в виде трёх вертикально расположенных кругов. Программа должна имитировать работу светофора: в цикле 20 секунд «горит» красный свет (верхний круг закрашен красным), 5 секунд «горит» желтый (средний круг закрашен красным), 20 секунд «горит» зелёный свет (верхний круг закрашен зелёным).

3.3. Хранение, изменение, отображение данных. В задачах этого раздела требуется использовать архитектуру «Модель-Вид-Контроллер». Приложение должно работать с данными о любом количестве объектов, для этого требуется организовать возможность добавления данных пользователем, изменения имеющихся данных, отображения всех введённых данных, реализовать возможность чтения дан-

ных из текстового файла. Все вводимые данные должны быть проверены на правильность.

Главное окно приложения должно содержать меню и центральный виджет, отображающий список объектов. В меню приложения должны присутствовать опции для выхода из приложения, открытия файла с данными, записи данных в файл, добавления элемента данных в общий список, редактирования, удаления выделенного элемента, а также отображения информации о выделенном элементе в отдельном окне или в нескольких окнах, если задание предусматривает несколько видов для отображения элемента. Редактирование и ввод новых элементов осуществляется с помощью отдельного диалогового окна. Обязательной частью меню является пункт «О программе».

В случае некорректных действий пользователя или ошибок в данных программа должна выводить об этом информацию в отдельном окне.

Задачи

1. Разработать приложение, отображающее данные о треугольнике на координатной плоскости.

   Модель – класс, хранящий и вычисляющий информацию о треугольнике.

   Вид (1) – виджет, в котором нарисован треугольник.

   Вид (2) – виджет, отображающий информацию в текстовом виде (параметры треугольника, его площадь и периметр).

   Контроллер – главное окно приложения.

   Дополнительно к параметрам треугольника добавляются цвет и толщина линий, для класса «вид (1)» реализуется возможность поворота треугольника на заданный угол и перетаскивание треугольника кнопкой «мыши» внутри виджета.
2. Разработать приложение, отображающее данные о квадратном трёхчлене.

   Модель – класс, хранящий и вычисляющий информацию о квадратном трёхчлене.

   Вид (1) – виджет, в котором нарисована соответствующая парабола и оси координат.

   Вид (2) – виджет, отображающий информацию в текстовом виде (параметры квадратного трёхчлена, его точки пересечения с осями координат).

   Контроллер – главное окно приложения.

Дополнительно к параметрам модели добавляются цвет и толщина линий, для класса «вид (1)» реализуется возможность перемещения параболы кнопкой «мыши» внутри виджета.

3. Разработать приложение, отображающее данные о прямой, заданной уравнением вида $ax + by = c$.

   Модель – класс, хранящий и вычисляющий информацию о прямой.

   Вид (1) – виджет, в котором нарисована прямая на координатной плоскости.

   Вид (2) – виджет, отображающий информацию в текстовом виде (уравнение прямой, координаты точек пересечения с осями Ox и Oy).

   Контроллер – главное окно приложения.

   Дополнительно к параметрам прямых добавляются цвет и толщина линий, для класса «вид (1)» реализуется возможность перетаскивания прямой кнопкой «мыши» внутри виджета.

4. Разработать приложение, отображающее данные об окружности на координатной плоскости.

   Модель – класс, хранящий и вычисляющий информацию об окружности.

   Вид (1) – виджет, в котором нарисована окружность.

   Вид (2) – виджет, отображающий информацию в текстовом виде (координаты центра и радиус окружности).

   Контроллер – главное окно приложения.

   Дополнительно к параметрам окружностей добавляются цвет и толщина линий.

5. Разработать приложение, отображающее данные о четырёхугольнике, в который можно вписать окружность.

   Модель – класс, хранящий и вычисляющий информацию о четырёхугольнике.

   Вид (1) – виджет, в котором нарисован четырёхугольник и вписанная в него окружность.

   Вид (2) – виджет, отображающий информацию в текстовом виде (параметры четырёхугольника и радиус вписанной окружности).

   Контроллер – главное окно приложения.

   Дополнительно к параметрам четырёхугольника добавляются цвет и толщина линий, для класса «вид (1)» реализуется воз-

можность поворота четырёхугольника на заданный угол относительно центра вписанной окружности.

6. Разработать приложение, отображающее данные о неориентированном графе.

   Модель – класс, хранящий (в виде матрицы смежности) и вычисляющий информацию о графе.

   Вид (1) – виджет, в котором нарисован граф.

   Вид (2) – виджет, отображающий информацию в текстовом виде (матрицу смежности, количество вершин и рёбер графа, степень вершин).

   Контроллер – главное окно приложения.

   Дополнительно к параметрам рёбер добавляются цвет и толщина линий, для класса «вид (1)» реализуется возможность перетаскивания графа кнопкой «мыши» внутри виджета.

7. Разработать приложение, отображающее данные о последовательности чисел.

   Модель – класс, хранящий и вычисляющий информацию о последовательности.

   Вид (1) – виджет, в котором нарисована диаграмма (круг с секторами), отражающая частоту появления каждого числа в последовательности.

   Вид (2) – виджет, отображающий информацию в текстовом виде (сколько раз каждое число встречается в последовательности, максимальное число, минимальное, среднее арифметическое).

   Вид (3) – виджет, в котором нарисована гистограмма, отражающая частоту появления каждого числа в последовательности.

   Контроллер – главное окно приложения.

   Дополнительно к параметрам последовательности добавляются цвет для каждого числа, для класса «вид (1)» реализуется возможность перетаскивания диаграммы кнопкой «мыши» внутри виджета.

8. Разработать приложение, отображающее данные об ориентированном графе, ребра которого имеют вес.

   Модель – класс, хранящий (в виде матрицы смежности) и вычисляющий информацию о графе.

   Вид (1) – виджет, в котором нарисован граф.

Вид (2) – виджет, отображающий информацию в текстовом виде (матрицу смежности, количество вершин, максимальное и минимальное по весу ребра графа).

Контроллер – главное окно приложения.

Дополнительно: к параметрам рёбер добавляются цвет и толщина (в зависимости от веса) линий, для класса «вид (1)» реализуется возможность перетаскивания графа кнопкой «мыши» внутри виджета.

9. Разработать приложение, строящее графики функций с заданными параметрами:
   - $ax + by = c$,
   - $y = ax^2 + bx + c$,
   - $y = \frac{1}{ax+b}$,
   - $y = sin(ax + b)$.

   Модель – классы, хранящие информацию о функциях.

   Вид – виджеты, в котором отображаются графики функций.

   Контроллер – главное окно приложения.

   Дополнительно к параметрам функций добавляются цвет и толщина линий, для класса «вид» реализуется возможность перетаскивания графика кнопкой «мыши» внутри виджета.
10. Разработать приложение, строящее графики функций с заданными параметрами и их производных:
    - $y = ax^2 + bx + c$,
    - $y = sin(ax + b)$.

    Модель – классы, хранящие информацию о функциях.

    Вид – виджеты, в котором отображаются графики функций.

    Контроллер – главное окно приложения.

    Дополнительно к параметрам функций добавляются цвет и толщина линий, для класса «вид» реализуется возможность перетаскивания графика кнопкой «мыши» внутри виджета.
11. Разработать приложение, позволяющее работать с данными о человеке. Данные содержат:
    - фамилию, имя, отчество;
    - пол;
    - дату рождения, возраст;

- образование;
- семейное положение.

Модель – класс, хранящий информацию о человеке.

Вид – виджет, в котором отображаются сведения о человеке в текстовых строках, выровненных по центру.

Контроллер – главное окно приложения.

12. Разработать приложение, позволяющее работать с данными о геометрической трёхмерной фигуре (призма или пирамида). Данные содержат:
    - вид фигуры;
    - вид основания;
    - длины сторон основания;
    - длину высоты.

    Модель – класс, хранящий информацию о фигуре.

    Вид – виджет, в котором отображаются сведения о фигуре в текстовых строках, выровненных по центру.

    Контроллер – главное окно приложения.
13. Разработать приложение, позволяющее работать с данными о геометрической фигуре (круг, треугольник, четырёхугольник). Данные содержат:
    - вид фигуры;
    - в зависимости от вида фигуры радиус или длины сторон

    Модель – класс, хранящий информацию о фигуре и ее рисунок.

    Вид – виджет, в котором отображаются сведения о фигуре.

    Контроллер – главное окно приложения.
14. Разработать приложение, позволяющее работать с данными о товаре. Данные содержат:
    - название товара;
    - название поставщика;
    - оптовую цену;
    - название магазина, где товар продаётся;
    - наценку магазина в процентах.

Модель – класс, хранящий информацию о товаре.

Вид – виджет, в котором отображаются сведения о товаре в текстовых строках, выровненных по центру.

Контроллер – главное окно приложения.

15. Разработать приложение, позволяющее работать с данными о студенте. Данные содержат:
    - фамилию, имя, отчество;
    - пол;
    - возраст;
    - номер группы;
    - предмет;
    - сведения о зачёте или оценке.

    Модель – класс, хранящий информацию о студенте.

    Вид – виджет, в котором отображаются сведения о студенте в текстовых строках, выровненных по центру.

    Контроллер – главное окно приложения.
16. Разработать приложение, позволяющее работать с данными о счёте. Данные содержат:
    - название или имя плательщика;
    - название или имя получателя;
    - вид платежа;
    - размер платежа;
    - срок уплаты.

    Модель – класс, хранящий информацию о товаре.

    Вид – виджет, в котором отображаются сведения о счете.

    Контроллер – главное окно приложения.
17. Разработать приложение, позволяющее работать с данными о квартире. Данные содержат:
    - адрес (город, улица и т. д.);
    - количество комнат;
    - метраж;
    - стоимость;
    - примечания.

Модель – класс, хранящий информацию о квартире.
Вид – виджет, в котором отображаются сведения о квартире.
Контроллер – главное окно приложения.

18. Разработать приложение, позволяющее работать с данными о железнодорожном билете. Данные содержат:
    - пункт отправления;
    - пункт назначения;
    - дата и время отправления;
    - стоимость;
    - фамилию, имя, отчество человека.

    Модель – класс, хранящий информацию о билете.
    Вид – виджет, в котором отображаются сведения о товаре.
    Контроллер – главное окно приложения, в котором находится меню, содержащее пункты, необходимые для ввода и обработки данных.
19. Разработать приложение, позволяющее работать с данными о книге. Данные содержат:
    - название;
    - имя автора;
    - жанр;
    - издательство;
    - год издания;
    - количество страниц;
    - цена.

    Модель – класс, хранящий информацию о товаре.
    Вид – виджет, в котором отображаются сведения о товаре.
    Контроллер – главное окно приложения.

3.4. Разработка проектов. В этом разделе собраны задачи разного уровня сложности. В процессе написания программы следует разработать структуру хранения данных, продумать интерфейс приложения, главное окно приложения, подходящие диалоговые окна.

Задачи

1. Написать программу, реализующую шифрование и дешифрование, применяя шифр Цезаря. Шифр Цезаря — это вид шифра подстановки, в котором каждый символ в тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите.

   Меню «Файл» предполагает возможность извлечения текста для шифрования/дешифрования из файла, сохранение результатов шифрования/дешифрования в файл, выход.

   Меню «Опции» предполагает изменение цвета фона окна приложения и стиля кнопок.

   Ограничения на алгоритм шифрования: шифровать только буквы латинского алфавита (остальные символы, например знаки препинания или кириллицу, не изменять).
2. Написать программу «Словарь». Программа предоставляет возможность перевода слов с русского на английский язык, и с английского — на русский.

   Главное окно приложения содержит панель инструментов с тремя кнопками:

   - En -> Ru;
   - Ru -> En;
   - Exit.

   Также в главном окне приложения отображаются виджеты для ввода слова для перевода и результата перевода.

   Англо-русский и русско-английский словари хранятся в файлах, например EnRuDict.txt и RuEnDict.txt.

   Пример заполнения словарей:

   - EnRuDict.txt
     - car машина
     - dog собака
     - girl девушка
     - book книга
   - RuEnDict.txt
     - машина car
     - собака dog

– девушка girl
– книга book

3. Написать программу «Расширение словаря», увеличивающую словарный запас программы «Словарь». Так же как и в предыдущей задаче, русско-английский словарь находится в файле RuEnDict.txt, англо-русский – в EnRuDict.txt.

   В главном окне приложения отображается содержание словарей и виджеты для ввода новых слов с переводом.

   В многострочных полях ввода «Текущий En->Ru» и «Текущий Рус -> Англ», доступных только для чтения, отображаются содержания файлов EnRuDict.dat и RuEnDict.dat соответственно.

   Приложение содержит две кнопки «Add» и «Добавить». При нажатии на кнопку «Add» словарь пополняется парой слов из полей ввода «Английский:» и «Русский:». При нажатии на кнопку «Добавить» словарь пополняется парой слов из полей «Русский:» и «Английский:».

   Меню «Файл» содержит только пункт меню «Выход», при нажатии на которое программа завершает свою работу.

   Меню «Помощь» содержит пункт меню «О приложении», при нажатии на которое в отдельном окне отображается информация о назначении приложения и её разработчике.
4. Написать программу-блокнот, позволяющую открывать, редактировать и сохранять текстовые документы.

   Меню «Файл» предполагает возможность извлечения текста для редактирования из файла, сохранения результатов в файл, выход из программы.

   Меню «Помощь» содержит подменю «О приложении», при нажатии на которое в отдельном окне отображается информация о разработчике приложения.

   Строка состояния содержит две панели:

   1. Правая панель – состояние документа «Сохранен/Изменён» (текст на панели = «Сохранён», если документ только что открыли или сохранили, либо «Изменён», если содержимое документа изменено).

   2. Левая панель – состояние блокнота (если пользователь сохраняет документ, текст панели = «Сохраняю…», открывает = «Открываю…», программа находится в режиме ожидания = «Готов»).

5. Написать программу, предоставляющую возможность составить заказ на приобретение некоторого товара.

   В главном окне приложения слева отображается список всех товаров и информация о текущем товаре. Справа выводится состояние заказа: список выбранных товаров и их общая стоимость. Для управления работой приложения используются три кнопки: «Добавить товар в заказ», «Новый заказ» и «Сохранить заказ».

   В комбинированном списке «Товар:» содержатся наименования товаров, доступных для покупки. При выборе товара текст метки «Стоимость:...», текст метки «Изготовитель:...» и текст поля ввода «Описание:» меняют свои значения на значения, соответствующие выбранному товару.

   Комбинированный список «Товар:» заполняется при запуске программы. Значения для списка хранятся в исходном файле «PriceList.txt» в произвольном формате. В этом же файле хранятся стоимость, изготовитель и описание товара.

   В таблице описаны реакции программы при нажатии на кнопки.

Нажатие на кнопку	Действие
«Добавить товар в заказ»	В поле ввода «Заказ:» добавляется запись о выбранном товаре и пересчитывается общая стоимость в поле ввода «ИТОГО:»
«Новый заказ»	Очищаются поля ввода «Заказ:» и «ИТОГО:»
«Сохранить заказ»	Заказ и общая стоимость сохраняются в файле

   Пользователь выбирает товар из списка, нажимает на кнопку «Добавить товар в заказ», затем выбирает другой товар, снова нажимает на кнопку «Добавить товар в заказ» и т. д. После того, как добавлены все товары для приобретения, пользователь нажимает на кнопку «Сохранить заказ» и выбирает файл для сохранения.
6. Написать программу проведения тестирования. Пользователю предлагается ответить на 10 вопросов. Тематика вопросов может быть выбрана произвольно. На каждый вопрос предусматривается по четыре варианта ответов. Информация о вопросах и ответах содержится в файле «Questions.txt».

   Интерфейс программы должен состоять из трех окон:

   a) окно приветствия;

   b) окно тестирования;

c) окно результатов.

При запуске программы появляется окно приветствия. Пользователь вводит своё имя и нажимает кнопку «Готово». После этого появляется окно тестирования. В нём текст метки «Участник:» содержит имя тестируемого. Ниже расположена группа виджетов, содержащая вопрос и кнопки для выбора ответа. Текст заголовка группы содержит номер вопроса. В виджете отображается текст вопроса и четыре варианта ответов под буквами «а», «б», «в» и «г».

При нажатии на одну из кнопок с буквами «а», «б», «в» или «г» в отдельном окне выводится сообщение о правильности результата.

Строка состояния, расположенная внизу окна тестирования, содержит три панели:

a) Левая панель – «Всего ответов».

b) Средняя панель – «Правильных», отображает число верных ответов.

c) Правая панель – «Ошибочных:», отображает количество неправильных ответов.

Панель инструментов содержит кнопки переключения вопросов и кнопку «Выход».

Если пользователь ответил на все вопросы, выводится окно результатов, содержащее имя тестируемого, и сообщение, что тест пройден.

7. Написать программу «Учет пользователей». Программа хранит информацию о пользователях: имя, фамилию и адрес электронной почты. Изменять информацию пользователи могут только о себе. Вход в программу осуществляется по логину и паролю. Информация об учетных записях хранится в файле «Accounts.txt» в произвольном формате.

   Интерфейс программы должен состоять из трёх окон:

   a) окно авторизации, где вводятся логин и пароль;

   b) окно создания новой учетной записи, где вводится новый логин и пароль, ввод пароля дублируется;

c) главное окно программы, в котором отображаются имя, фамилия, адрес электронной почты пользователя, а также присутствуют кнопки для редактирования данных.

При запуске программы появляется окно авторизации. После проверки введённых данных если пользователя нет в списке авторизированных пользователей, то при нажатии на кнопку «OK» появляется сообщение об ошибке. При нажатии на кнопку «Create User» появляется окно создания новой учетной записи.

При создании новой учетной записи происходит проверка на уникальность логина и сложность пароля. Алгоритм определения сложности пароля предлагается выбрать самостоятельно. После ввода данных пользователь нажимает кнопку «Create». Если вновь введённый логин уже существует или пароль недостаточно сложен, появляется соответствующее сообщение и предлагается повторить ввод или отказаться от работы. Если проверка пройдена успешно, появляется главное окно программы.

Если же пользователь есть в списке авторизированных пользователей, при нажатии на кнопку «OK» в окне авторизации появляется главное окно программы. Поля ввода в окне доступны только для чтения. Чтобы изменить значение в поле ввода, введены три кнопки «<- Edit». Эти кнопки изменяют состояния полей ввода с «Только для чтения» на «Чтение / запись» и обратно.

Главное окно программы содержит также кнопки для выхода из приложения (Exit) и смены пользователя (Logout).

8. Написать программу, моделирующую работу регистратуры поликлиники.

Главное окно приложения содержит расписание работы врачей с информацией о фамилии, имени, отчестве и специальности врача, дне недели, времени приёма (с 8 до 13 часов или с 14 до 19), номере кабинета.

Главное меню приложения должно содержать пункты для создания и изменения данных: чтение данных из файла, сохранение данных в файл, редактирование информации о враче, добавление информации о враче, поиск врача с заданной специальностью (результат поиска выводится в отдельном окне), а также вывод информации о самой программе (пункт меню «О программе»).

Все данные при добавлении и изменении должны проверяться на корректность: один и тот же врач может работать только пять дней в неделю и только в один из возможных интервалов времени, два врача не могут одновременно работать в одном и том же кабинете. При попытке ввода некорректных данных должно выдаваться соответствующее сообщение.

9. Написать программу, работающую с информацией о служащих учреждения. Данные об отдельном работнике состоят из имени, фамилии, отчества, даты рождения, образования, должности и названия отдела.

   Главное окно приложения должно содержать список людей, отвечающий условиям, которые определяются в меню.

   Главное меню должно содержать пункты для создания, изменения и отображения данных: чтение данных из файла, сохранение данных в файл, редактирование информации о служащем, добавление информации о служащем, вывод списка работников с заданным образованием, вывод списка работников заданного возраста (от a до b лет), вывод списка работников заданного отдела, а также вывод информации о самой программе (пункт меню «О программе»).

10. Написать программу, позволяющую компьютеру и человеку играть в слова. Предварительно программа объясняет правила игры и позволяет уточнить их в любой момент.

    Тематикой игры могут быть по выбору города, животные, растения и т. д.(не менее 5 тем). Тему из предложенных компьютером вариантов выбирает человек. Для игры компьютер использует собственную базу данных (для каждой тематики свою), хранящуюся в виде текстового файла. Если названное человеком слово отсутствует в базе, уточняется, правильно ли оно названо, и в случае правильности заносится в базу. Правила игры: первый игрок называет слово, затем второй должен предложить другое, начинающееся с той буквы, на которую оканчивается слово, названное первым. Повторять слова в течение одной игры нельзя.

    Компьютер должен реализовывать стратегию, отличающуюся от случайного выбора подходящего слова, например выбирать слово, заканчивающееся на редко встречающуюся букву.

11. Написать программу для следующей игры: имеется кучка из случайного количества камней (от 10 до 100). Каждый игрок по очереди берёт 1, 2, 3 или 4 камня. Выигрывает тот, кто

    a) забирает последний камень;

    b) оставляет противнику последний камень.

    Вариант выигрыша определяется в настройках игры.
    В настройках также определяется кто играет:

    a) игрок с компьютером;

    b) два игрока.

    Программа должна реализовывать следующие функции:

    - рисовать в главном окне приложения кучку камней в текущем состоянии;
    - с помощью «мыши» или клавиш выбирать камни, которые следует взять при очередном ходе;
    - реализовать выигрышную стратегию для компьютера в случае игры человека с компьютером;
    - сохранять текущее состояние игры и восстанавливать его при желании пользователя;
    - выводить сообщение о выигрыше или проигрыше.

12. Написать программу, позволяющую играть на бесконечном поле в «крестики-нолики»:

    a) игроку с компьютером;

    b) двум игрокам.

    Если в качестве игрока выступает компьютер, программа делает первый ход. Делая очередной ход, программа анализирует ситуацию, рассчитывая возможные ходы противника вперед на 1—2 хода, и в результате проведенного анализа поступает оптимальным образом.

13. Написать программу, позволяющую играть в «Быки и коровы»:

    a) игроку с компьютером;

    b) двум игрокам.

Каждый из противников задумывает четырехзначное число, все цифры которого различны (первая цифра числа отлична от нуля). Необходимо разгадать задуманное число. Выигрывает тот, кто отгадает первый. Противники по очереди называют друг другу числа и сообщают о количестве «быков» и «коров» в названном числе («бык» — цифра есть в записи задуманного числа и стоит в той же позиции, что и в задуманном числе; «корова» — цифра есть в записи задуманного числа, но не стоит в той же позиции, что и в задуманном числе).

Например, если задумано число 3275 и названо число 1234, получаем в названном числе одного «быка» и одну «корову». Число отгадано в том случае, если получилось 4 «быка».

14. Написать программу для игры человека в головоломки со спичками. На игровом поле находятся несколько спичек, сложенных определенным образом в определенные фигуры. Задача пользователя — убрать определенное количество спичек так, чтобы решить некоторую задачу, например из шести сложенных квадратов сделать три, убрав три спички. В процессе решения головоломок сложность увеличивается.

    При разработке проекта следует подобрать несколько головоломок разной сложности (не менее 15). В главном окне приложения отображается фигура из спичек, пользователь с помощью «мыши» или клавиш со стрелками выбирает спичку, которую хочет убрать, при нажатии на клавишу «Enter» или при двойном щелчке мыши спичка убирается. На игровом поле должна быть кнопка, позволяющая вернуть спичку, которую убрали. Если заданное количество спичек удалено, программа сообщает о правильности решения. Переход к следующей головоломке осуществляется после правильного решения текущей задачи или по желанию пользователя.

15. Написать программу, обучающую учащихся арифметическим действиям с отрицательными числами, а также предлагающую серию заданий различной сложности для закрепления навыков действий над такими числами.

    Программа должна содержать справочный материал, который описывает правила выполнения действий и приводит к каждому правилу подходящий пример.

В режиме закрепления навыков предлагается набор заданий по возрастанию сложности (не менее 25 примеров), где учащийся вводит ответ к очередному примеру и получает информацию о его правильности. Если ответ не верен, то предлагается решить аналогичный пример, но не более трёх раз для текущего уровня сложности. После завершения тренировки выдаётся информация о количестве и соотношении правильных и неправильных ответов, и анализ результатов (на какие правила допущено больше всего ошибок).

Кроме основного режима работы, должна быть реализована возможность отработки навыков для заданного правила или арифметического действия.

16. Написать программу, автоматизирующую процесс построения фигур на плоскости с помощью циркуля и линейки. Программа должна уметь выполнять следующие команды:

    - отметить произвольную точку и обозначить ее;
    - отметить произвольную точку и обозначить ее;
    - построить произвольную прямую;
    - построить окружность с заданным центром данного радиуса;
    - построить и обозначить точку пересечения двух линий.

    Программа должна содержать 10—15 стандартных задач на построение школьного курса геометрии, предлагать их для решения и контролировать процесс построения и полученный результат.

17. Написать программу, моделирующую экологическую модель. Остров размером 20x20 заселен дикими кроликами, волками и волчицами. Имеется по нескольку представителей каждого вида. Кролики довольно глупы: в каждый момент времени они с одинаковой вероятностью 1/9 передвигаются в один из восьми соседних квадратов (за исключением участков, ограниченных береговой линией) или просто сидят неподвижно. Каждый кролик с вероятностью 0,2 превращается в двух кроликов, новый кролик занимает случайно выбранную соседнюю клетку.

    Волки и волчицы передвигаются случайным образом, пока в одном из соседних восьми квадратов не окажется кролик, за которым они охотятся. Если волк и кролик оказываются в одном

квадрате, волк съедает кролика и получает одно очко. В противном случае он теряет 0,1 очка. Волки и волчицы с нулевым количеством очков умирают.

В начальный момент времени все волки и волчицы имеют 1 очко. Если волк и волчица окажутся в соседних квадратах и рядом нет кроликов, которых можно съесть, они производят потомство случайного пола.

Запрограммировать описанную модель и понаблюдать за изменением популяции в течение заданного периода времени.

18. Написать программу для игры с головоломкой Пятнашки. Игровое поле представляет собой набор одинаковых квадратных фишек с числами в квадратной коробке. Длина стороны коробки в четыре раза больше длины стороны костяшек. В коробке находится набор из 15 пронумерованных от 1 до 15 костяшек, соответственно остаётся незаполненным одна квадратная ячейка. Цель игры — перемещая костяшки по коробке, добиться упорядочивания их по номерам, сделав как можно меньше шагов.

    Программа должна реализовывать следующие функции:

    - в начале новой игры создавать игровое поле генератором случайных чисел;
    - рисовать в главном окне приложения игровое поле в текущем состоянии;
    - с помощью «мыши» или клавиш со стрелками выбирать костяшку для перемещения;
    - при нажатии на клавишу «Enter» или при двойном щелчке «мыши» перемещать выбранную костяшку в свободную ячейку, если это возможно;
    - сохранять текущее состояние игрового поля и восстанавливать его при желании пользователя;
    - если костяшки упорядочены, то выводить сообщение о выигрыше и количестве сделанных перемещений.

19. Написать программу, моделирующую аквариум. Аквариум содержит камни и рыб. Он представляет собой область главного окна приложения, наполненную водой. Рыбы живут в аквариуме. Рыбка имеет координаты, скорость, размер, цвет, направление движения, поле зрения (небольшой отрезок по направле-

нию движения). Нарисовать её можно в виде стрелки, направленной острием по ходу движения. Рыбка перемещается в текущем направлении на расстояние, зависящее от скорости, иногда случайным образом меняет направление движения. Если рыба видит препятствие, направление движения меняется, пока препятствие не исчезнет из поля зрения.

Приложение содержит три кнопки:

- Init — включает графический режим, заполняет аквариум водой, камнями и рыбами;
- Run — организует бесконечный цикл, в котором движутся все обитатели аквариума;
- Done — выключает графический режим.

20. Написать программу, моделирующую Солнечную систему. Необходимо изобразить на экране компьютера Солнце и восемь планет Солнечной системы (от Меркурия до Нептуна), а также основные их спутники (например, для Земли – Луну, для Марса – Фобос и Деймос), в их движении по орбитам. Можно считать, что вращение планет вокруг Солнца происходит в одной плоскости (поскольку плоскости орбит планет близки к плоскости земной орбиты), к этой плоскости можно отнести и орбиты спутников планет. Вращение планет вокруг своей оси можно не учитывать.

    При визуализации Солнечной системы на экране компьютера должно быть соблюдено правильное соотношение размеров орбит планет и скоростей их движения (то есть они должны быть пропорциональны их реальным значениям). Соотношение размеров изображений самих планет также должно соответствовать действительности, но при этом для наглядности масштаб их изображения должен быть больше масштаба показа их орбит, иначе некоторые планеты отображаются на экране точками.

    Кроме того, поскольку при показе на экране сразу всех восьми планет Солнечной системы изображения ближайших к Солнцу планет (и их спутников) получаются слишком мелкими и сливаются друг с другом, следует либо использовать крупный масштаб и предоставить скроллинг изображения, либо предусмотреть визуализацию Солнечной системы в двух масштабах (для всех планет и для ближайших к Солнцу). Пользователь дол-

жен иметь возможность включать и отключать показ названий планет и спутников, их орбит и траекторий движения других тел, а также ускорять или замедлять движение тел в Солнечной системе.

21. Написать программу, позволяющую конструировать электрические схемы с помощью графических инструментов, представляющих отдельные элементы электрической цепи. Пользователь системы должен иметь возможность в рабочем окне:

    - задавать контур электрической цепи, выбирая его элементы из нескольких возможных;
    - располагать в нужных точках заданного контура необходимые элементы;
    - изменять расположение отдельных элементов заданной цепи;
    - замыкать и размыкать входящие в цепь выключатели (замыкающие ключи);
    - запоминать построенную схему в файле и считывать ее из файла в рабочее окно.

    Необходимо, чтобы указанные действия пользователь мог производить в произвольном, удобном для него порядке. Визуализация электрической схемы должна включать изображение цепи и всех входящих в нее элементов, а также показ текущего состояния выключателей (замыкающих ключей) и горящих лампочек, сигнализирующих о наличии тока на соответствующем участке цепи.

    Дополнительно можно задавать внутренние характеристики отдельных элементов электрической цепи, величину тока, напряжения и сопротивления в определенных точках/участках построенной схемы, осуществлять расчет для заданной электрической схемы основных ее характеристик, то есть величины тока в каждой ее точке и величины напряжения между двумя произвольными точками.

22. Написать программу, моделирующую систему регулирования домашнего отопления. Система отопления представляет собой водный обогреватель, работающий на газе и нагревающий воду в батареях отопления, установленных в комнатах дома (вклю-

чая гостиную, рабочий кабинет, кухню, ванную комнату и другие помещения). В каждой комнате есть специальный клапан, регулирующий поступление горячей воды в батареи этой комнаты и тем самым температуру в этой комнате. Возможные положения клапана (полностью открыт/закрыт/полуоткрыт) могут быть установлены дистанционно. В каждой комнате находится также датчик текущей температуры.

Основное назначение системы автоматического регулирования отопления – поддержание в каждой из комнат дома нужной температуры путем установки соответствующих положений клапанов обогревателя. Пользователь системы может включать и выключать систему регулирования, а также задавать рабочую температуру в каждой комнате, т. е. температуру, которая должна быть в комнате в случае присутствия в ней людей. В случае же их отсутствия в целях экономии должна поддерживаться температура ожидания – она определяется на M градусов ($1 \leq M \leq 5$) ниже рабочей в этой комнате.

Системе известно недельное расписание пребывания людей в комнатах дома – для того, чтобы заранее, к моменту ожидаемого появления в конкретной комнате людей, начать прогревать ее до рабочей температуры. Регулирование температуры системой основано на показаниях датчиков температуры, заданных величинах рабочей температуры в каждой комнате и недельного графика пребывания людей в доме. Эмуляция течения времени осуществляется с помощью таймера, например одна секунда работы модели соответствует часу реального времени работы системы.

Для проведения экспериментов необходимо программно эмулировать показания датчиков текущей температуры в комнатах. Следует считать, что при закрытом клапане обогревателя температура в каждой комнате медленно падает (линейно по времени, коэффициент линейной зависимости определяется временем суток), при открытом – растет (по аналогичному закону), при полуоткрытом – сохраняется постоянной.

Если температура в каком-либо помещении опускается или поднимается на N градусов ($1 \leq N \leq 5$) ниже или выше требуемой, то система формирует команду на изменение положения клапана в этой комнате.

Модельная система регулирования отопления подсчитывает также общий расход топлива на обогрев (в условных единицах), при условии, что на обогрев 1 кв. м комнаты за 1 минуту расходуется величина $P = C \times K$, где значение K соответствует положению клапана обогревателя в этой комнате: $K = 0$ – закрыт, $K = 2$ – полуоткрыт, $K = 5$ – открыт полностью; а C – некоторая заранее заданная для каждой комнаты константа (зависит от площади батарей в этой комнате).

Цель моделирования – изучение зависимости величины расхода топлива от параметров M и N и недельного расписания занятости комнат дома.

В ходе моделирования должны быть изображены: план дома, наличие в каждой комнате людей, положение клапанов обогревателя, а также указаны температура в каждой комнате, время суток и день недели, расход топлива на текущий момент.

Литература

1. Хант, Эндрю. Программист-прагматик. Путь от подмастерья к мастеру / Эндрю Хант, Дэвид Томас. — [Б. м.] : Лори, Питер, 2007.
2. Босуэлл, Д. Читаемый код, или Программирование как искусство / Д. Босуэлл, Фаучер Т. — [Б. м.] : Питер, 2012.
3. Code Conventions for the Java Programming Language — [S. l. : s. n.], 2014. — Jan. — URL: http://www.oracle.com/technetwork/java/codeconv-138413.html.
4. C++ Programming Style Guidelines — [S. l. : s. n.], 2012. — Jan. — URL: http://geosoft.no/development/cppstyle.html.
5. Приёмы объектно-ориентированного проектирования: Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Библиотека программиста. — [Б. м.] : Питер, 2001.
6. Windows® Desktop Design Guidelines — [S. l. : s. n.], 2014. — Jan. — URL: http://msdn.microsoft.com/en-us/library/windows/desktop/aa511440.aspx.
7. OS X Human Interface Guidelines — [S. l. : s. n.], 2014. — Jan. — URL: https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/AppleHIGuidelines/Intro/Intro.html.
8. KDE4 Human Interface Guidelines — [S. l. : s. n.], 2014. — Jan. — URL: http://techbase.kde.org/Projects/Usability/HIG.
9. GNOME Human Interface Guidelines 2.2.3 — [S. l. : s. n.], 2014. — Jan. — URL: https://developer.gnome.org/hig-book/stable/.

Учебное издание

Лагутина Надежда Станиславовна
Ларина Юлия Александровна
Васильев Андрей Михайлович

Разработка программных приложений

Практикум

Редактор, корректор М. В. Никулина
Компьютерный набор, вёрстка А. М. Васильев

Подписано в печать 24.03.2014. Формат 60×84/16.
Усл. печ. л. 4,18. Уч.-изд. л. 3,0.
Тираж 75 экз. Заказ .

Оригинал-макет подготовлен
в редакционно-издательском отделе ЯрГУ.

Ярославский государственный университет
им. П. Г. Демидова
150000, Ярославль, ул. Советская, 14.