ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра прикладной математики

В.Н. Агеев

# ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

**Учебно-методическое пособие**
по выполнению комплексных домашних заданий

*для студентов I курса*
*направления 25.03.01*
*очной формы обучения*

Москва
ИД Академии Жуковского
2018

УДК 004(07)
ББК 6Ф6.5
А23

Рецензент:
*Егорова А.А.* – д-р техн. наук, проф. каф. прикладной математики

**Агеев В.Н.**

А23 Информатика и информационные технологии [Текст] : учебно-методическое пособие по выполнению комплексных домашних заданий / В.Н. Агеев. – М.: ИД Академии Жуковского, 2018. – 24 с.

Данное учебно-методическое пособие издается в соответствии с рабочей программой учебной дисциплины «Информатика и информационные технологии» по учебному плану для студентов I курса направления 25.03.01 очной формы обучения.

Рассмотрено и одобрено на заседании кафедры 09.02.2018 г. и методического совета 09.02.2018 г.

**УДК 004(07)**
**ББК 6Ф6.5**

*В авторской редакции*

Подписано в печать 30.05.2018 г.
Формат 60x84/16 Печ. л. 1,5 Усл. печ. л. 1,395
Заказ № 320/0514-УМП07 Тираж 60 экз.

Московский государственный технический университет ГА
125993, Москва, Кронштадтский бульвар, д. 20

Издательский дом Академии имени Н. Е. Жуковского
125167, Москва, 8-го Марта 4-я ул., д. 6А
Тел.: (495) 973-45-68
E-mail: zakaz@itsbook.ru

**Содержание**

## Введение

Данные методические указания предназначены для студентов 1-го курса дневной формы обучения по направлению 25.03.01 и содержат описания практических работ в 1-м и 2-м семестрах. Методические указания могут быть использованы в качестве учебно-методического материала по аналогичным дисциплинам других направлений подготовки.

Первая часть пособия содержит материал, изучаемый в 1-м семестре и включает следующие темы:

– системы счисления;
– арифметические основы ЭВМ;
– логические основы ЭВМ;
– алгоритмы и блок-схемы.

Вторая часть содержит материал, изучаемый во 2-м семестре и включает следующие темы:

– основы языка Visual Basic;
– программирование линейных вычислительных процессов;
– программирования разветвляющихся и циклических процессов;
– графические возможности языка Visual Basic;
– работа с файлами прямого и последовательного доступа.

В каждом разделе приведены краткие сведения по методам решения задач и приведены соответствующие примеры.

Во второй части рассматриваются вопросы разработки программ на языке Visual Basic. Особое внимание уделено вопросам организации ввода и вывода данных, программированию разветвляющихся вычислительных процессов, работе с числовыми массивами, построению графических изображений. Ко всем изучаемым темам прилагаются задания для самостоятельной работы. Выполнение практических работ предполагается после изучения лекционного материала по соответствующим темам.

Целью выполнения практических работ является:

– закрепление пройденного материала;
– выполнения комплексного индивидуального задания студентам;
– приобретения практических навыков программирования различных типов задач.

# 1. Арифметические основы ЭВМ

## 1.1. Прямой, обратный и дополнительный двоичные коды

Для записи двоичных чисел со знаком используются специальные коды, в которых старший разряд является знаковым. Туда записывается 0, если число положительное и 1, если отрицательное. В остальных разрядах записывается модуль числа. Это называется прямым кодом.

Кроме прямого, существуют также обратный и дополнительный коды числа со знаком. Для положительных чисел все три кода совпадают. Для отрицательных чисел обратный код получается из прямого инвертированием всех двоичных цифр модуля. Дополнительный код получается прибавлением 1 в младший разряд обратного кода.

Обратный и дополнительный коды используются для того, чтобы операцию вычитания заменить операцией поразрядного сложения.

Для этого поразрядно складываются обратные или дополнительные коды двоичных чисел. Знаковый разряд также учитывается при сложении. Если при сложении обратных кодов возникает единица переноса в старшем разряде, она прибавляется к полученному результату. При сложении дополнительных кодов такая единица переноса не прибавляется.

**Примечание.**

Во избежание так называемой «ошибки переполнения» при действии с обратными и дополнительными кодами при записи двоичного кода для модуля следует добавлять «незначащий» ноль слева, например:

$$193 = 11000001_2 = 01100001_2 \qquad 207 = 1100111_2 = 01100111_2$$

Дополнительные нули добавляются слева также для выравнивания количества разрядов в обоих числах.

## 1.2. Пример решения задачи

Даны два десятичных числа: A = 193 и B = -207.

Необходимо записать в двоичной системе обратный и дополнительный коды этих чисел и выполнить операцию поразрядного сложения этих кодов.

**Решение.**

Для получения двоичных кодов модулей заданных чисел воспользуемся методом деления «уголком»:

```
193 |2
  1  96 |2
     0  48 |2
        0  24 |2
           0  12 |2
              0  6 |2
                 0  3 |2
                    1  1
```

```
207 |2
  1 103 |2
     1  51 |2
        1  25 |2
           1  12 |2
              0  6 |2
                 0  3 |2
                    1  1
```

Для каждого числа двоичный код представляет собой последовательность остатков от деления, выписанные «снизу-вверх», как показано на рисунке. После добавления слева незначащих нулей, получим для модулей заданных чисел их двоичные коды:

$$193 = 011000001_2 \qquad 207 = 011001111_2$$

Добавляя знаковый разряд (0 для положительного числа, 1 – для отрицательного), получим прямые двоичные коды заданных чисел:

$193 = 0011111101_2$ $\quad\quad$ $-207 = 1011001111_2$

Обратный и дополнительный коды положительного числа совпадают с прямым кодом. Для отрицательного числа обратный код получается инвертированием всех разрядов, кроме знакового, дополнительный – получается из обратного прибавлением в младший разряд единицы.

| | Десятичное число | Двоичное число со знаком | Прямой код | Обратный код | Дополнительный код |
|---|---|---|---|---|---|
| A | 193 | 011000001 | 0011000001 | 0011000001 | 0011000001 |
| B | –207 | –011001111 | 1011001111 | 1100110000 | 1100110001 |

Результаты сложения обратных и дополнительных кодов показаны в следующей таблице.

| | Десятичное число | Прямой код | Обратный код | Дополнительный код |
|---|---|---|---|---|
| A | 193 | 0011000001 | 0011000001 | 0011000001 |
| B | –207 | 1011001111 | 1100110000 | 1100110001 |
| A+B | | | 1111110001 | 1111110010 |

Если в результате сложения обратных или дополнительных кодов в знаковом разряде окажется единица, это означает, что результат – число отрицательное. В этом случае для получения окончательного результата необходимо перейти к прямому коду. Для этого двоичные цифры обратного кода инвертируются (кроме знакового разряда), а в случае дополнительного кода после инвертирования нужно прибавить в младший разряд единицу.

Итак прямой двоичный код суммы заданных чисел:

– для случая сложения обратных кодов $A+B = 1000001110_2 = -1110_2$

– для случая сложения дополнительных кодов $A+B = 1000001101_2 + 1 = -1110_2$

Проверка: $A+B = 193 - 207 = -14 = -1110_2$.

### 1.3. Варианты задания

Выполнить сложение чисел A и B, представив их двоичными обратными и дополнительными кодами. Результат сложения перевести в десятичное число и проверить результат, выполнив операцию сложения с десятичными числами.

| Вариант | A B | Вариант | A B | Вариант | A B |
|---|---|---|---|---|---|
| 1 | 221 –118 | 11 | 313 –520 | 21 | –222 599 |
| 2 | 119 –303 | 12 | 455 –227 | 22 | –331 487 |
| 3 | –199 218 | 13 | 521 –333 | 23 | 554 –239 |
| 4 | 201 –222 | 14 | –259 399 | 24 | 298 –444 |
| 5 | –234 454 | 15 | –299 –101 | 25 | 301 –419 |
| 6 | –334 491 | 16 | 333 –555 | 26 | –292 –111 |
| 7 | 229 –355 | 17 | 291 –444 | 27 | –332 –109 |
| 8 | 360 –579 | 18 | –238 –322 | 28 | 313 –298 |
| 9 | 251 –222 | 19 | 211 –202 | 29 | 215 –245 |
| 10 | 121 –229 | 20 | 301 –299 | 30 | 112 –272 |

## 2. Логические основы ЭВМ

### 2.1. Логические переменные, логические операции

Логической переменной называют переменную, принимающую только одно из двух возможных значений: единица и ноль («истина» или «ложь»). Логической функцией называют функцию, принимающую на любом наборе определяющих ее переменных только одно из двух возможных значений – ноль или единица.

| Логические операции | | | | |
|---|---|---|---|---|
| инверсия | отрицание | НЕ (NOT) | ¬ A | $\overline{x}$ |
| конъюнкция | умножение | И (AND) | $x \wedge y$ | $x + y$ |
| дизъюнкция | сложение | ИЛИ (OR) | $x \vee y$ | $x \cdot y$ |

Результаты применения логических операций задаются следующими таблицами:

| НЕ | |
|---|---|
| x | $\overline{x}$ |
| 0 | 1 |
| 1 | 0 |

| И | | |
|---|---|---|
| x \ y | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| ИЛИ | | |
|---|---|---|
| x \ y | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 1 |

Логические операции подчиняются законам алгебры логики (алгебры Дж. Буля).

| *Закон* | *Операция ИЛИ* | **Операция И** |
|---|---|---|
| Переместительный | $x \vee y = y \vee x$ | $x \cdot y = y \cdot x$ |
| Сочетательный | $x \vee (y \vee z) = (x \vee y) \vee z$ | $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ |
| Распределительный | $x \cdot (y \vee z) = x \cdot y \vee x \cdot z$ | $x \vee (y \cdot z) = (x \vee y) \cdot (x \vee z)$ |
| Правила де Моргана | $\overline{x \vee y} = \overline{y} \cdot \overline{x}$ | $\overline{x \cdot y} = \overline{y} \vee \overline{x}$ |
| Поглощения | $x \vee (x \cdot y) = x$ | $x \cdot (x \vee y) = x$ |
| Склеивания | $(x \cdot y) \vee (\overline{x} \cdot y) = y$ | $(x \vee y) \cdot (\overline{x} \vee y) = y$ |
| Операция переменной с ее инверсией | $x \vee \overline{x} = 1$ | $x \cdot \overline{x} = 0$ |
| Операция с константами | $x \vee 0 = x;\ x \vee 1 = 1$ | $x \cdot 1 = x;\ x \cdot 0 = 0$ |
| **Двойного отрицания** | $\overline{\overline{x}} = x$ | |

**Пример.**

Упростить выражение: $x \cdot (y \cdot \overline{x} + x \cdot y + z)$

Решение:

$x \cdot (y \cdot \overline{x} + x \cdot y + z)) = x \cdot y \cdot \overline{x} + x \cdot x \cdot y + x \cdot z = x \cdot \overline{x} \cdot y + x \cdot x \cdot y + x \cdot z = x \cdot (y+z)$

Здесь использованы свойства конъюнкции (операции И): $x \cdot \overline{x} = 0$ и $x \cdot x = x$.

### 2.2. Способы задания логических функций

Чтобы задать логическую функцию, нужно указать, какие значения она принимает для всех комбинаций значений ее аргументов. Это можно сделать двумя способами: аналитическим, с помощью формулы и табличным (с помощью так называемой таблицы истинности). Например, логическую функцию четырех переменных можно задать формулой $F = x_2 \cdot \overline{x_1} + x_1 \cdot x_0$.

Эту же функцию можно описать с помощью таблицы истинности.

| $n$ | $x_2$ | $x_1$ | $x_0$ | $F$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 0 |
| 7 | 1 | 1 | 1 | 1 |

Для более компактной записи табличную форму представления можно заменить выражением типа $F(3,4,5,7)$, в котором указываются номера строк таблицы истинности, в которых функция $F$ имеет значения 1.

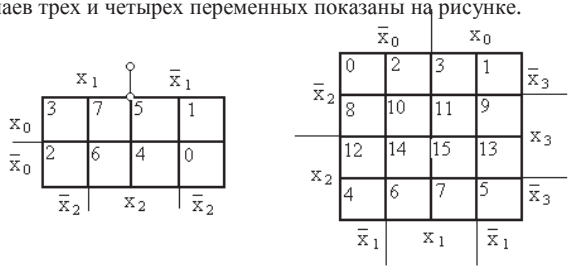
Функцию, заданную в табличной форме, можно представить в аналитическом виде. Для этого есть несколько способов, одним из них является представление функции в *совершенной нормальной дизъюнктивной форме* (СНДФ). Для получения СНДФ нужно выписать из таблицы наборы переменных, для которых значения функции равны единице, при этом переменные, которые в наборе имеют значение 0, инвертируются. Затем все эти наборы объединяются с помощью операцией сложения (дизъюнкцией). Для приведенного выше примера СНДФ имеет вид

$$F = \overline{x_2} \cdot x_1 \cdot x_0 + \cdot x_2 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 + \cdot x_2 \cdot x_1 \cdot x_0$$

После упрощения функция принимает вид $x_2 \cdot \overline{x_1} + x_1 \cdot x_0$.

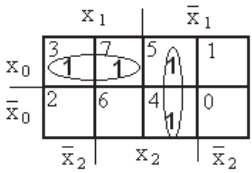
### 2.3. Построение карты Карно

Другим способом минимизации логических функций является способ минимизации с использованием карты Карно, которая представляет собой квадрат или прямоугольник, разбитый на квадраты по числу возможных комбинаций значений переменных. Для функции трех переменных прямоугольник разбивается на восемь, а для четырех – на 16 частей. Карты Карно для случаев трех и четырех переменных показаны на рисунке.



Цифры внутри ячеек соответствуют номерам строк таблицы истинности. Например, в приведенной выше таблице истинности функции трех переменных в 7-ой строке значения всех трех переменных равны единице. В СНДФ ей соответствует слагаемое, в котором все три переменные перемножаются прямо (без инверсии). Соответствующая ячейка карты Карно с цифрой 7 внутри отражает это свойство.

Карта Карно позволяет быстрее перейти от табличного способа задания логической функции к аналитическому. Если заполнить карту Карно данными из приведенной выше таблицы истинности для функции трех переменных, она примет следующий вид:




Единицы проставляются в ячейки, номера которых совпадают с номерами строк таблицы, в которых значение заданной функции равно 1.

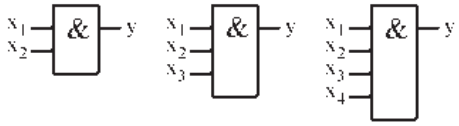
Смежные клетки могут образовывать группы по 2, 4 , 8 клеток. Для каждой такой группы в СНДФ будет одно слагаемое, в котором в качестве множителей входят те переменные (или их инверсии), которые являются общими для группы.

Для приведенного здесь случая СНДФ имеет вид $F= \overline{x_2}\cdot x_1+x_1\cdot x_0$ (результат, который был получен выше путем минимизации аналитического выражения для СНДФ, но гораздо более простым путем).

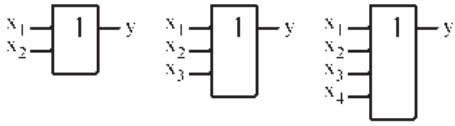
### 2.4. Комбинационные схемы

Логическим элементом называется электрическая схема, выполняющая логические операции над входными данными, заданными в виде уровней напряжения, и возвращающая результат операции в виде выходного уровня напряжения. Высокий уровень напряжения (напряжение логической единицы) символизирует истинное значение операнда, а низкий (напряжение логического нуля) – ложное.

К числу логических операций, выполняемых логическими элементами относятся конъюнкция (логическое умножение, И), дизъюнкция (логическое сложение, ИЛИ), отрицание (НЕ) и сложение по модулю 2 (исключающее ИЛИ).



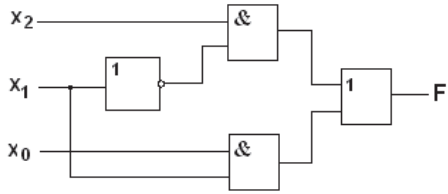

Графическое изображение элементов И с двумя, тремя и четырьмя входами.




Графическое изображение элементов ИЛИ с двумя, тремя и четырьмя входами.


Графическое изображение инвертора (элемента НЕ) с одним входом и одним выходом.

Пример комбинационной схемы (реализация логической функции трех переменных, рассмотренной в качестве примера в разделе 2.3).




**2.5. Пример выполнения задания**

Задана логическая функция четырех переменных $F(x_3, x_2, x_1, x_0) = F(3,7,11,12,13,14,15)$. Цифры в скобках означают номера строк таблицы истинности, в которых значение функции равно единицы.

Таблица истинности в этом случае имеет вид

| N | $x_3$ | $x_2$ | $x_1$ | $x_0$ | F |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 1 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 1 |

Совершенная нормальная дизъюнктивная форма СНДФ в этом случае имеет вид:

$$F = \bar{x}_3 \cdot \bar{x}_2 \cdot x_1 \cdot x_0 + \bar{x}_3 \cdot x_2 \cdot x_1 \cdot x_0 +$$
$$+ x_3 \cdot \bar{x}_2 \cdot x_1 \cdot x_0 + x_3 \cdot x_2 \cdot \bar{x}_1 \cdot \bar{x}_0 + x_3 \cdot x_2 \cdot \bar{x}_1 \cdot x_0 +$$
$$+ x_3 \cdot x_2 \cdot x_1 \cdot \bar{x}_0 + x_3 \cdot x_2 \cdot x_1 \cdot x_0.$$

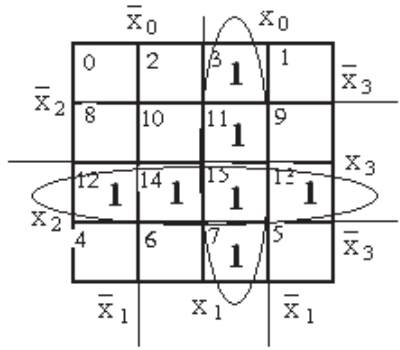
Выражение в правой части представляет собой сумму (дизъюнкцию) семи слагаемых (по числу единиц в колонке F таблицы истинности). Каждое слагаемое – произведение (конъюнкция) всех аргументов функции, при этом каждая переменная входит либо прямо, либо с отрицанием, в зависимости от того, какое значение, 1 или 0, она имеет в соответствующей строке.

Так, первое слагаемое соответствует второй строке таблицы, где переменные $x_3$ и $x_2$ имеют значение ноль, а $x_1$ и $x_0$ – единица. Поэтому в первое слагаемое переменные $x_1$ и $x_0$ входят прямо, а $x_3$ и $x_2$ – с отрицанием.

Используя свойства логических операций и законы алгебры логики можно упростить полученное выражение. Окончательный вид аналитического выражения:

$$F = x_3 \cdot x_2 + x_1 \cdot x_0$$

Этот же результат можно получить с помощью карты Карно. Запишем единицу в те клетки карты Карно, номера которых совпадают с номерами тех строк таблицы истинности, в которых поле F содержат единицу.



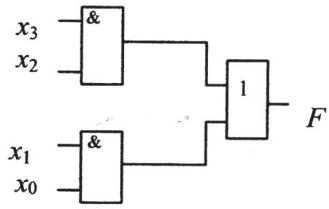

Из рисунка видно, то клетки с единицами образуют две группы смежных клеток. В каждую группу входит одинаковым образом только две переменные. Для первой (расположенной по горизонтали), такими являются переменные $x_3$ и $x_2$, а для второй (расположенной по вертикали) – переменные $x_1$ ·и $x_0$.

В соответствии со свойствами карты Карно аналитическое выражение для логической функции F можно представить в виде

$$F = x_3 \cdot x_2 + x_1 \cdot x_0$$

что совпадает с полученным ранее результатом.

Комбинационная схема, работа которой описывается данной логической функцией, имеет вид



### 2.6. Варианты задания

Логическая функция $F(x_3, x_2, x_1, x_0)$ задана в ци фровой форме (набором чисел, двоичный код которых определяет те значения аргументов, при которых F=1).

Порядок выполнения задания:

1. Составить таблицу истинности для логической функции.

2. Построить по этой таблице нормальную дизъюнктивную форму.

3. Минимизировать логическое выражение аналитическим способом.

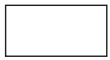
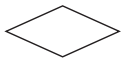
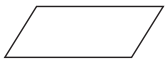
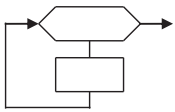
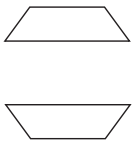
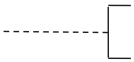
4. Построить логическое выражение для функции с помощью карты Карно.

5. Составить функциональную схему устройства с четырьмя входами и одним выходом с использованием логических элементов И, ИЛИ, НЕ.

| № | $F(x_3, x_2, x_1, x_0)$ | № | $F(x_3, x_2, x_1, x_0)$ |
|---|---|---|---|
| 1 | 0, 2, 7, 8, 10, 13, 15 | 16 | 0, 1, 2, 3, 4, 7, 11 |
| 2 | 2, 3, 5, 7, 10, 11, 15 | 17 | 3, 8, 9, 10, 11, 12, 15 |
| 3 | 2, 4, 6, 8, 10, 12, 14 | 18 | 1, 9, 10, 11, 12, 14, 15 |
| 4 | 4, 6, 7, 9, 11, 13, 15 | 19 | 2, 4, 6, 10, 13, 14, 15 |
| 5 | 6, 7, 8, 10, 12, 13, 14 | 20 | 0, 1, 4, 6, 8, 9, 14 |
| 6 | 5, 7, 8, 10, 13, 14, 15 | 21 | 3, 4, 6, 10, 11, 14, 15 |
| 7 | 1, 3, 5, 7, 10, 11, 14 | 22 | 0, 4, 6, 7, 10, 12, 14 |
| 8 | 0, 1, 4, 8, 12, 13, 14 | 23 | 1, 3, 4, 6, 9, 11, 12 |
| 9 | 1, 3, 7, 9, 11, 13, 15 | 24 | 4, 5, 6, 7, 9, 11, 15 |
| 10 | 0, 1, 4, 5, 11, 14, 15 | 25 | 4, 5, 7, 9, 11, 12, 13 |
| 11 | 2, 3, 6, 7, 8, 10, 15 | 26 | 0, 1, 4, 5, 12, 14, 15 |
| 12 | 2, 3, 6, 7, 12, 14, 15 | 27 | 3, 6, 7, 8, 10, 14, 15 |
| 13 | 3, 5, 7, 8, 9, 12, 13 | 28 | 1, 6, 9, 11, 13, 14, 15 |
| 14 | 4, 5, 7, 8, 10, 12, 14 | 29 | 4. 8, 11, 12, 13, 14, 15 |
| 15 | 1, 3, 6, 7, 9, 10, 15 | 30 | 2, 3, 6, 7, 11, 12, 15 |

## 3. Программирование на языке Visual Basic

### 3.1. Блок-схемы алгоритмов

| Наименование | Обозначение | Функция |
|---|---|---|
| Блок начало-конец |  | Начало и конец алгоритма. Внутри фигуры записывается соответствующее действие |
| Блок вычислений |  | Выполнение одной или нескольких операций, обработка данных любого вида |
| Проверка условия |  | Отображает функцию переключательного типа с одним входом и двумя или более альтернативными выходами, из которых только один может быть выбран после вычисления условий, определенных внутри этого элемента. |
| Предопределенный процесс |  | Процесс, состоящий из одной или нескольких операций, определенных в другом месте программы (в подпрограмме, модуле). |
| Ввод/вывод данных |  | Ввод или вывод данных, имена переменных и вид действия указываются внутри фигуры |
| Цикл типа пересчета |  | Цикл с заданным числом повторений. Внутри фигуры записывается параметр цикла для которого указываются начальное и конечное значения, а также шаг изменения. |
| Цикл с условием |  | Символ состоит из двух частей − начало и конец цикла − операции, выполняемые внутри цикла, размещаются между ними. Условия цикла и приращения записываются внутри символа начала или конца цикла − в зависимости от типа организации цикла. |
| Соединитель |  | Используется для обрыва линии и продолжения ее в другом месте |
| Комментарий |  | Используется для более подробного описания шага, процесса или группы процессов. Описание помещается со стороны квадратной скобки. |

### 3.2. Основные сведения о языке Visual Basic

Типы переменных:

| Тип | Возможные значения | Объем занимаемой памяти |
|---|---|---|
| Byte | целые числа от 0 до 255 | 1 байт |
| Integer | целые числа от -32768 до 32767 | 2 байта |
| Long | целые числа двойной длины | 4 байта |
| Single | вещественные числа | 4 байта |
| Double | вещественные числа двойной точности | 8 байт |
| Boolean | Логическое значение True или False | 2 байта |
| String | Строка символов | 1 байт на символ |

Математические функции в языке Visual Basic:

| Функция | Возвращаемое значение |
|---|---|
| Sin(A) | синус числа A |
| Cos (A) | косинус числа A |
| Tan (A) | тангенс числа A |
| Atn(A) | арктангенс числа A |
| Sqr (A) | квадратный корень из числа A |
| Log (A) | логарифм числа A |
| Exp(A) | показательная функция числа A |
| Int(A) | наибольшее целое число, не превышающее число A |
| Clnt(A) | целое число, ближайшее к числу A |
| Fix (A) | целое число, равное числу A без дробной части |
| Abs (A) | абсолютное значение числа A |
| Rnd | случайное число в интервале (0, 1) |

В таблице приведены примеры использования некоторых функций для работы с символьными строками.

| **Выражение** | **Комментарий** |
|---|---|
| N = Len (S) | Длина строки S присваивается числовой переменной N |
| S1 = Left (S, n) | Строковая переменная S1 получает n левых символов S |
| S1 = Right (S, n) | Строковая переменная S1 получает n правых символов S |
| S1 = Mid (S, n, m) | Строковая переменная S1 получает n символов S, начиная с m-го |
| N= InStr(S,c) | Определяется номер позиции символа c в строке S. Если такого символа в троке нет, переменной N присваивается значение 0. |
| N =Val (S) | Числовая переменная N получает значение числа из строки S |
| S = Str (N) | Строковая переменная S получает символьное представление числа N |

| | |
|---|---|
| N = Asc (C) | N получает значение ASCII-кода символа C |

Изменить последовательность действий можно с помощью операторов ветвления. Одним из них является условный оператор **If** … **Then**. Он имеет два форматов записи.

Однострочный формат:

**If** *условие* **Then** *Оператор1* [ **Else** *Оператор2*]

Блочный формат:

**If** *условие* **Then**
*Операторы1*
[ **Else**
*Операторы2* ]
**End If**

Для многократного выполнения одного или нескольких операторов применяются операторы цикла. Оператор цикла типа пересчета:

**For** *Счетчик* = *Начальное_значение* **To** *Конечное_значение* [ **Step** *Шаг* ]
*Операторы*
**Next** [*Счетчик*]

### 3.3. Графические возможности Visual Basic

На экранной форме или в графическом поле можно рисовать различные графические примитивы с использованием графических методов. Ниже приведены примеры использования этих методов. В качестве объекта *object*, куда выводятся графические примитивы, может служить сама форма (в этом случае имя объекта *Form* можно не указывать) или графическое окно *PictureBox*.

| Наименование | Синтаксис и комментарии |
|---|---|
| Точка | **object.Pset** (X,Y), C<br>X,Y – координаты точки, C – цвет. |
| Окружность | **object . Circle** (X, Y) , R, C<br>X, Y – координаты центра в выбранной системе координат, R – радиус, C – цвет. |
| Дуга окружности | **object . Circle** (X, Y) , R, C, A, B<br>X, Y – координаты центра, R – радиус, C – цвет. A, B – углы дуги в радианах. дуга строится против часовой стрелки от A к B. |
| Овал | **object . Circle** (X, Y) , R, C,,, K<br>K – коэффициент сжатия овала. При 0 < K< 1 сжатие по горизонтали, при K>1 – по вертикали. |
| Отрезок линии | **object.Line** (Xl,Y1) –(X2,Y2), C<br>XI, Yl – координаты точки начала отрезка, X2, Y2 – его конца, C – цвет. |
| Прямоугольник | **object.Line**(Xl,Y1) –(X2,Y2), C, B<br>XI, Yl — координаты левой верхней вершины прямоугольника, X2, Y2 — координаты правой нижней вершины, C – цвет. |
| Прямоугольник закрашенный | **object.Line** (Xl,Y1) –(X2,Y2), C, BF<br>XI, Yl — координаты левой верхней вершины прямоугольника, X2, Y2 — координаты правой нижней вершины, C – цвет. |
| Вывод строки символов | **object.Print** [output]<br>Вывод осуществляется от последней построенной точки изображения.<br>Для указания точки вывода можно использовать метод Pset(X,Y). |
| Масштабирование окна вывода | **object.Scale** (XI, Yl) – (X2,Y2)<br>(XI, Yl) и (X2,Y2) – «мировые» координаты выводимого изображения, левой верхней и правой нижней вершины окна соответственно |

### 3.4. Работа с внешними файлами

**Файлы последовательного доступа**

Для открытия файла последовательного доступа используется оператор

**Open** *имя_файла* **For** *тип_доступа* **As** *# номер*

Параметр *тип_доступа* задает тип доступа к данным: **Input** – для чтения, **Output** – для вывода, **Append** – для добавления, *номер* – порядковый номер от 1 до 255. В дальнейшем при работе с открытым файлом указывается не имя, а его порядковый номер.

Чтобы закрыть файл используется оператор **Close** (*номер).*

Для считывания одной строки из открытого для чтения текстового файла используется оператор **Input:**

**Input** *# номер, имя_переменной*

где *имя_переменной* – имя переменной типа **String.**

При открытии файла в его начало устанавливается так называемый «файловый указатель». После команды **Input** он перемещается к началу следующей строки. После считывания всех строк он устанавливается в конец файла. При попытке выполнить еще одну команду **Input** в этом случае будет выдано сообщение об ошибке (код 62).

Для того, чтобы избежать аварийного завершения программы можно использовать логическую функцию **EOF**(*номер*), где *номер* – номер открытого файла. Например, для считывания данных из файла в массив MAS(i) можно использовать цикл типа **Do Until:**

```
k=0
Do Until  EOF (1)
        k=k+1
        Input  # 1, MAS(k)
Loop
```

Вспомогательная переменная k используется для указания номера того элемента массива, которому присваивается считанная из файла строка.

Запись данных в файл, открытый для записи, осуществляется командой

**Print** *#номер, список_вывода*

Здесь *список_вывода* – одна или несколько переменных (или строки символов, взятые в кавычки), разделенных либо запятыми (,), либо точками с запятой (;). В первом случае выводимые значения будут записаны в виде одной строки и разделены пробелами, во втором – выведены без разделителя, слитно. Если поставить точку с запятой в конец списка вывода, то следующая порция данных, выводимая оператором будет дописана в ту же строку.

**Файлы произвольного доступа**

Произвольный (*Random*) доступ для ввода или вывода применяется в случае, когда записи в файле имеют фиксированную длину.

Открывается файл для произвольного доступа с помощью оператора

**Open** *имя_файла* **For Random As** *# номер* **Len** = *длина_записи*

Параметр **Len** задает длину одной записи в байтах.

Для чтения данных используется оператор **Get:**

**Get** *# номер , номер_записи, переменная*

Для записи – оператор **Put:**

**Put** *# номер , номер_записи, переменная*

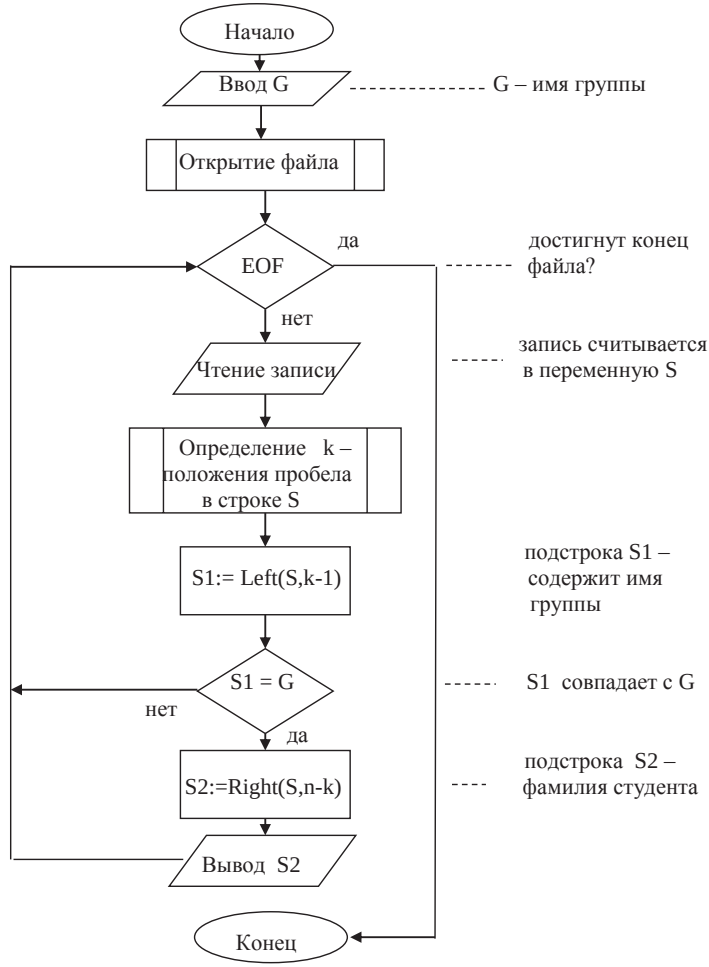
### 3.5. Пример выполнения задания.

**Задание**. Текстовый файл A.txt содержит список студентов. Каждая запись включает номер группы и через пробел фамилию студента и его инициалы. Разработать проект, в котором решается следующая задача:

В текстовое поле на экранной форме вводится название группы и при нажатии кнопки «ПУСК» из заданного файла считываются фамилии студентов этой группы и выводятся во второе текстовое поле.

**Решение**.

1. Блок-схема алгоритма:

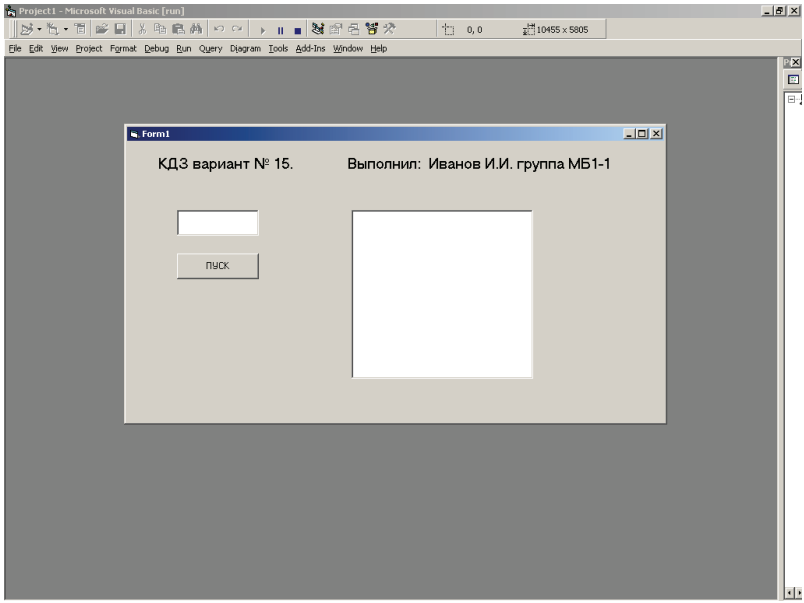


2. Текст программы.

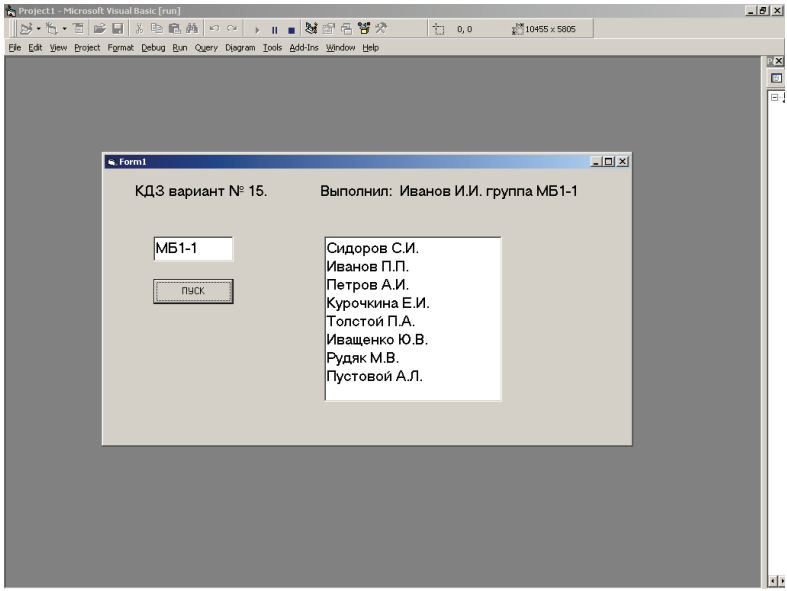
```
Private Sub Command1_Click ()
G = Text1.Text                                  ‘  G – имя группы
Open “A.txt” For Input As #1                    ‘ Файл A.txt открывается для чтения
Do Until  EOF(1)                                ‘  Повторять, пока EOF(1) = False
        Input #1, S                             ‘  Считывание очередной записи
        n=Len(S)                                 ‘    n – количество символов в строке S
        k = InStr (S,” “)                        ‘   k – номер позиции пробела в S
        S1 = Left (S, k-1)                      ‘    Подстрока с именем группы
        If  S1 = G Then                         ‘    Если она совпадает с Text1.Text
`       `       S2= Right (S,n-k)                ‘    Подстрока S2 – фамилия студента
                Text2.Text = S2 + Chr(13)+ Chr(10) ‘  Добавляется в поле Text2.Text
        Enf If
Loop
Close #1                                         ‘  Файл закрывается
End Sub
```

3. Экранная форма.

На экранной форме размещены два текстовых поля Text1 и Text2, а также Command1.




После ввода в первое текстовое поле названия группы и нажатия кнопки «ПУСК» во второе текстовое поле выводится список фамилий, считанных из файла A.txt.




**Примечания.**

1. Файл F.txt со списком фамилий должен быть расположен в той же папке, что и разрабатываемая программа. В противном случая в операторе Open кроме имени файла нужно указывать и путь к нему.

2. Для того, чтобы в текстовом окне каждая фамилия отображалась в отдельной строке, необходимо на этапе проектирования экранной формы свойству Multiline объекта Text2 следует присвоить значение True.

3. Отчет должен быть подготовлен в формате Word и включать:
   1) титульный лист с фамилией студента и номером группы
   2) задание, описание алгоритма решения задачи
   3) блок-схема алгоритма
   4) текст программного кода
   5) скрин-шоты экранных форм

Отчет должен быть представлен не позже последнего занятия в дисплейном классе. Преподаватель, ведущий лаб. работы в группе, делает отметку на титульном листе, что програма работает.

### 3.6. Варианты задания

**Вариант 1.**
В текстовом файле записана дата в формате *день.месяц* (типа 25.11).
Написать программу, которая считывает эту дату и выводит на экран ее словесное описание (типа «двадцать пятое ноября»).

*Указание.* Создать два массива с элементами типа String . Первый содержит названия числительных, второй - названия месяцев в нужном падеже.

**Вариант 2**.
Написать программу, которая выводит на экран изображение идущих часов, показывающих текущее московское время, имеющих движущуюся секундную стрелку.

*Указание.* Разместить на экранной форме проекта объект Timer.

**Вариант 3.**
Построить столбчатую диаграмму для семи положительных действительных чисел $a_1, \ldots a_7$. Их значения вывести в середину соответствующего прямоугольника. Эти числа считываются из текстового файла.

*Указание.* Масштабировать графическое окно с помощью метода Scale так, чтобы диаграмма с заданными числовыми значениями высоты столбцов полностью вписывалась в это окно.

**Вариант 4.**
Построить на экране розу ветров по заданным натуральным числа $к_1,\ldots к_8$, определяющим северное, северо-восточное, восточное, юго-восточное, южное, юго-западное, западное или северо-западное направление. Числа считываются из текстового файла.

*Указание.* Масштабировать графическое окно с помощью метода Scale и построить координатные линии, а также линии направлений на стороны света. Координаты вершин розы ветров определяются по формулам

$$X_i = k_i * \mathrm{Cos}(\varphi_i),\ Y_i = k_i * \mathrm{Sin}(\varphi_i),\quad i = 1,2, \ldots ,7,$$

где $\varphi_i$ - угол наклона i-го направления к горизонтальной оси в радианах.

**Вариант 5**.
В текстовом файле записаны m строк, каждая из которых содержит по три числа, разделенных пробелами. Каждая тройка чисел задает координаты центра квадрата и длину его стороны. Написать программу, которая считывает эти данные и строит на экране квадраты и раскрашивает их разными цветами.

*Указание.* Масштабировать графическое окно с помощью метода Scale так, чтобы опорные точки квадратов, данные для которых считываются из файла, оказались внутри графического окна. Для закрашивания квадратов нужным цветом необходимо установить для объекта Picture значения свойств FillStyle и FillColor ( см. конспект лекций [6]).

**Вариант 6.**
Текстовый файл содержит одну строку, в которой записаны два 16-ричных числа, разделенных пробелом. Написать программу, которая считывает эти числа и перемножает их. Оба сомножителя и результат выводится на экран.

*Указание.* Символьное представления 16-ричного числа A16 можно преобразовать в десятичное число A10 с помощью оператора A10 = Val ("&H"+A16).
Преобразовать десятичное число C10 в 16-ричное число C16 можно с помощью функции Hex: C16 = Hex (C10).

**Вариант 7.**
Построить круговую диаграмму для пяти положительных действительных чисел
$a_1, \ldots a_5$. Их значения вывести в середину соответствующего сектора. Эти числа считываются из текстового файла.
*Указание.* Для построения сектора круга используется метод Circle объекта Picture.

**Вариант 8.**
Написать программу, которая вычисляет число π методом Монте-Карло.
В квадрат со стороной 1 вписан круг радиуса 0,5. С помощью датчика случайных чисел генерируется N пар чисел (X,Y) – координаты точек, равномерно заполняющих квадрат. Если M – число точек, попавших внутрь круга, то приближенно число π равно 4·M / N.
Отобразить на экране точки, попадающие внутрь круга красным, а вне его – синим цветом. Вывести на экран приближенное значение **π** для заданного N.

*Указание.* Пример использования датчика случайных чисел см. в конспекте лекций [6].

**Вариант 9**.
Построить график функции $y = 2x^3 + 2x^2 + x$ на отрезке, координаты концов которого вводятся с клавиатуры. Построить оси координат и разместить на них деления, рядом с которыми расположить числа в соответствии с заданным масштабом.

*Указание.* О построении графиков функции см. конспект лекций [6].

**Вариант 10**.
В текстовом файле записано натуральное число N. Написать программу, которая считывает это число и выводит на экран N строк треугольника Паскаля (таблицы биноминальных коэффициентов).

*Указание.* Треугольник Паскаля строится по строкам. Строка с номером N содержит N чисел, которые подсчитываются по формуле

$$A(k) = N! / (k!*(N-k)!), \quad k = 0,1, \ldots ,N.$$

**Вариант 11.**
Написать программу, которая считывает из текстового файла число B и строит на экране график астроиды

$$X = B* \cos^3 t,$$
$$Y = B* \sin^3 t,$$

где $t \in [0,2\pi]$.
*Указание.* Масштабировать графическое окно с помощью метода Scale так, чтобы график полностью разместился в окне. Координаты точек графика X и Y изменяются в интервале [-B, B].
**Вариант 12**.
Текстовый файл содержит три строки, в каждой из которой содержится по 6 чисел, разделенных пробелами. Эти числа задают координаты вершин треугольника. Написать программу, которая считывают эти числа и строит на экране треугольники.

*Указание.* Масштабировать графическое окно с помощью метода Scale так, чтобы все треугольники разместились в в окне. Для этого надо определить наибольшие и наименьшие значения координат вершин треугольников.

**Вариант 13**.
Написать программу, которая выводит на экран изображение часов, показывающих текущее время. С помощью кнопки можно изменять вид часов: аналоговые или цифровые.

*Указание.* Разместить на экранной форме проекта объект Timer. Пример использования этого объекта можно найти в конспекте лекций [6] .

**Вариант 14.**
В текстовом файле несколько строк, каждая из которых содержит дату вида
*12 февраля 2016 г*. Написать программу, которая считывает эти записи и сортирует их в порядке возрастания даты. Результат выводится на экран.

*Указание*. Создать три массива. Первый типа String для записи прочитанных из файла дат. Второй также типа String для записи названий месяцев года. Третий - типа Long для записи числовых эквивалентов прочитанных дат.
Числовой эквивалент подсчитывается по формуле A=*год**10000+*месяц**100+*день*.
Пример: для даты *12 февраля 2016 г.* числовой эквивалент равен 2016*10000 + 2*100 + 12 = 20160212

**Вариант 15.**
В текстовом файле несколько строк, каждая из которых содержит одно слово с одинаковым для всех количеством букв. Написать программу, которая считывает эти слова и сортирует в порядке убывания суммы ASCII-кодов составляющих их букв. На экран выводится список этих слов с указанием соответствующих им сумм кодов.

*Указание.* Создать массив типа String для записи прочитанных из файла слов. Для сортировки этого массива использовать метод "пузырька".

**Вариант 16.**
Текстовый файл содержит строку с числами, разделенными запятыми.
Написать программу, которая считывает этот файл и создает новый, в котором записана строка с этими же числами, но отсортированными по убыванию.

*Указание*. Создать числовой массива, в который записать все числа из введенной строки. Для сортировки этого массива использовать метод "пузырька".

**Вариант 17.**
Число N вводится с клавиатуры (N<1000). Оно означает цену товара в копейках. Вывести цену в рублях и копейках с использованием слов «рубль» и «копейка» в соответствующих падежах.

*Указание.* Слова "рубли" и "копейки" необходимо выводить в соответствующем падеже. Например, 15 ***рублей***, 21 ***рубль*** , 13 ***копеек***, 22 ***копейки*** и т.д. Для этого нужны выявить закономерности, по которым для каждого числового значения выбирается нужное слово.

**Вариант 18.**
В текстовом файле записаны m строк, каждая из которых содержит по три числа, разделенных пробелами. Каждая тройка чисел задает координаты центра круга и его радиус. Написать программу, которая считывает эти данные и строит на экране круги и раскрашивает их разными цветами.

*Указание*. Масштабировать графическое окно с помощью метода Scale так, чтобы все круги оказались внутри графического окна.
Для закрашивания квадратов нужным цветом необходимо установить для объекта Picture значения свойств FillStyle и FillColor ( см. конспект лекций).

**Вариант 19.**
Текстовый файл содержит строку, в которой есть русские и английские слова. Написать программу, которая считывает этот файл и создает два новых, в одном из которых оказываются только русские, а в другом – только английские слова.

*Указание*. Использовать таблицу ASII-кодов. Русские буквы имеют коды от 192 до 255.

**Вариант 20.**
Два текстовых файла содержат списки фамилий. Программа проверяет, есть ли в них одинаковые фамилии. Если есть, то они выводятся на экран, в противном случае выводится сообщение «Повторяющихся фамилий нет».

*Указание*. Создать три массива типа String. В первые два записываются фамилии из файлов. Третий для повторяющихся фамилий.

**Вариант 21.**
Написать программу, которая выводит на экран изображение часов, показывающих текущее время для разных часовых поясов. Выбор осуществляется с помощью переключателей (элементы OptionButton) на экранной форме. Используется 4 переключателя, с надписями «Вашингтон», «Лондон», «Москва», «Токио».

*Указание*. Разместить на экранной форме проекта объект Timer. Пример использования этого объекта можно найти в конспекте лекций [6].

**Вариант 22.**
Текстовый файл содержит некоторый текст на русском языке. Написать программу, которая считывает этот текст и подсчитывает количество содержащихся в нем букв «а», «о», «е», «и». Результат выводится на экран.

*Указание*. Текст из файла заносится в текстовое окно. Для того, чтобы сохранить форматирование, на этапе проектирования экранной формы свойству **MultiLine** объекта **Text** необходимо присвоить значение **True.**

**Вариант 23.**
Текстовый файл содержит фамилии студентов и полученные ими оценки на экзамене. Написать программу, которая считывает этот файл и выводит на экран список студентов, получивших оценку «отлично».

*Указание.* Примеры использования функций для обработки символьных строк см. в конспекте лекций [6].

**Вариант 24.**
Построить график функции $y = e^x \cdot \cos(x)$ на отрезке [a, b], координаты концов которого a и b вводятся с клавиатуры. Построить оси координат и разместить на них деления, рядом с которыми расположить числа в соответствии с заданным масштабом.

*Указание.* О построении графиков функции см. конспект лекций [6].

**Вариант 25.**
В текстовом файле записано целое десятичное число N. Написать программу, которая считывает это число и выводит его на экран вместе с записями этого числа в двоичной, восьмеричной и 16-ричной формах.

*Указание*. Для пребразования деятичного числа в 8- и 16-ричное используютмя функции Oct и Hex: X8=Oct(X10), X16=Hex(X10). Двоичный код можно получить из десятичнго числа последавательным делением на 2 или из 8-ричного заменой каждого знака тройкой двоичных цифр.

**Вариант 26.**

В текстовом файле записана строка, содержащая N натуральных чисел ($2 < N < 10$), разделенных пробелами. Написать программу, которая считывает эти числа и определяет наименьшее и наибольшее из них. Результат выводится на экран.

*Указание.* Создать числовой массив для записи в него чесел из прочитанной строки. Алгоритм поиска наибольшего и наименьшего числа в массиве см. в конспекте лекций [6].

**Вариант 27.**
В текстовом файле записана цена товара в рублях и копейках. Написать программу, которая считывает этот файл и выводит на экран стоимость в долларах и центах. Обменный курс доллара вводится с клавиатуры.

*Указание*. Стоимость в долларах равна стоимости в рублях, деленной на обменный курс.

**Вариант 28.**
Написать программу, которая выводит на экран изображение работающего светофора, переключаемого с помощью клавиатуры.

*Указа*ние. Для закрашивания кругов нужным цветом необходимо установить для объекта **Picture** значения свойств **FillStyle** и **FillColo**r ( см. конспект лекций [6]).

**Вариант 29.**
В текстовом файле содержатся сведения о результатах экзаменационной сессии. Каждая запись содержит фамилию студента и четыре оценки, полученные им на экзаменах. Написать программу, которая подсчитывает средний балл и выводит на экран фамилии лучших студентов.

*Указание*. Для вывода в текстовое окно несколько строк, необходимо на этапе проектирования экранной формы **MultiLine** объекта **Text** следует присвоить значение **True.**

**Вариант 30.**
Текстовый файл содержит строку с числами, разделенными пробелами.
Написать программу, которая считывает этот файл и создает новый, в котором записана строка с этими же числами, но отсортированными в порядке возрастания.

## Литература

1. Симонович С.В. Информатика: базовый курс. Учебное пособие.–М., 2010
2. Гуда Н. и др. Информатика. Общий курс : Учебник.-4-е изд.– М., 2011
3. Аверьянов Г.П., Дмитриева В.В. Современная информатика: Учебное пособие.– М.: МИФИ, 2011.
4. Андреева Т.И., Пичугин А.А. (№1124) Информатика. Информатика и ИТ: пособие по выполнению практических занятий и комплексных домашних заданий для студ. 1 курса дневного обучения.– 2012.
5. Андреева Т.И.,Кишенский С.Ж.. Пичугин А.А. Информатика: пособие по изучению дисциплины, выполнению контрольных работ и варианты заданий. Эл.версия НТБ.– 2011.
6. Агеев В.Н. Программирование и основы алгоритмизации. Конспект лекций.– М., 2012.
7. Культин Н.Б. Visual Basic. Освой на примерах**.–М.,** 2012.