Лабораторная работа 10.1. Системы счисления

Цель работы: Изучить алгоритмы перевода чисел между разными системами счисления (десятичной, двоичной, восьмеричной и шестнадцатеричной). Научиться пользоваться этими алгоритмами.

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ

1. Ознакомиться с теоретическими сведениями первой части лабораторной.
2. Ознакомиться с вариантом задания (стр. 5–6) – соответствует вашему номеру в списке группы (при нехватке заданий вариант задания вычисляется как *номер_в_списке_группы* - *количество_заданий*).
3. Выполнить задание.
4. Защитить лабораторную работу перед преподавателем.

1.1 Кодирование данных двоичным кодом

Для автоматизации работы с данными, относящимися к различным типам, очень важно унифицировать их форму представления – для этого обычно используется прием кодирования, то есть выражение данных одного типа через данные другого типа. Естественные человеческие *языки* – это не что иное, как системы кодирования понятий для выражения мыслей посредством речи. К языкам близко примыкают *азбуки* (системы кодирования компонентов языка с помощью графических символов). История знает интересные, хотя и безуспешные попытки создания «универсальных» языков и азбук.

Та же проблема универсального средства кодирования достаточно успешно реализуется в отдельных отраслях техники, науки и культуры. В качестве примеров можно привести систему записи математических выражений, телеграфную азбуку, морскую флажковую азбуку, систему Брайля для слепых и многое другое.

Своя система существует и в вычислительной технике – она называется *двоичным кодированием* и основана на представлении данных последовательностью всего двух знаков: 0 и 1. Эти знаки называются *двоичными цифрами*, по-английски – binary digit или bit (бит). Бит – двоичный разряд, принимающий значение 0 или 1.

Одним битом могут быть выражены два понятия: 0 или 1 (да или нет, черное или белое, истина или ложь и т.п.). Если количество битов увеличить до двух, то уже можно выразить четыре различных понятия:

00 01 10 11

Тремя битами можно закодировать восемь различных значений:

000 001 010 011 100 101 110 111

Увеличивая на единицу количество разрядов в системе двоичного кодирования, мы увеличиваем в два раза количество значений, которое может быть выражено в данной системе. Таким образом, если есть n бит, можно закодировать 2^n символов.

1.2 Системы счисления

Люди всегда считали и записывали числа, даже пять тысяч лет назад. Но записывали они их совершенно по-другому, по другим правилам.

Известно множество способов представления чисел. Число изображается символом или группой символов некоторого алфавита. Такие символы называются *цифрами*.

Числа складываются из цифр по особым правилам. На разных этапах развития человечества, у разных народов эти правила были различны. Сегодня мы их называем системами счисления.

Система счисления – это совокупность приемов и правил для обозначения и именования чисел.

Все системы счисления делятся на позиционные и непозиционные. Непозиционные системы счисления появились раньше позиционных. Последние являются результатом длительного исторического развития непозиционных систем счисления.

В *непозиционной системе* счисления цифры не меняют своего количественного значения при изменении их расположения в числе. Например, в римской непозиционной системе счисления для каждого числа используется некоторый набор базовых символов (I, V, X, L, C, D, M), соответствующих числам 1, 5, 10, 50, 100, 500, 1000. Остальные значения чисел получаются из базовых путем сложения (например, 700=DCC) или вычитания (например, 800=CCM).

Позиционные системы счисления – это системы, в которых количественные значения цифр, используемых для записи чисел, зависят от их положения.

Наиболее распространенными в настоящее время позиционными системами счисления являются десятичная, двоичная, восмеричная и шестнадцатиричная.

Основными характеристиками позиционной системы счисления являются алфавит цифр, основание и разряд.

Алфавит системы счисления – это совокупность всех цифр, используемых в системе счисления.

Основание системы счисления – количество цифр, используемое для представления чисел.

Основанием может быть любое натуральное число.

Разряд – позиция цифры в числе.

Система счисления		**Основание**	**Алфавит цифр**
Десятичная	DEC	10	0,1,2,3,4,5,6,7,8,9
Двоичная	BIN	2	0,1
Восьмеричная	OCT	8	0,1,2,3,4,5,6,7
Шестнадцатеричная	HEX	16	0,1,2,3,4,5,6,7,8,9,A(10),B(11),C(12),D(13),E(14),F(15)

Таблица 1. Соответствия чисел в разных системах счисления.

DEC	**BIN**	**OCT**	**HEX**
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Для перевода чисел из одной системы счисления в другую существуют определенные правила. Они различаются в зависимости от формата числа – целое или правильная дробь. Для вещественных чисел используется комбинация правил перевода для целого числа и правильной дроби.

1.3 Перевод чисел в десятичную систему счисления

В позиционной системе счисления любое число может быть представлено в развернутом виде. Возьмем число в десятичной системе счисления 247,32 и представим его в следующем виде:

$247{,}32_{10} = 2\cdot100 + 4\cdot10+7\cdot1+3/10+2/100 = 2\cdot10^2 + 4\cdot10^1 + 7\cdot10^0 + 3\cdot10^{-1} + 2\cdot10^{-2}$

Мы записали число в развернутой форме, в которой:

- 2,4,7,3,2 – цифры числа
- 10 – основание системы счисления
- показатели степени: 2, 1, 0, -1, -2 соответствуют номеру позиции цифры в числе.

Основанием системы счисления может служить любое натуральное число: 2, 3, 4 и т.д. Следовательно, возможно бесчисленное множество позиционных систем.

Пусть q – основание системы счисления,

n – число разрядов целой части числа,

m – число разрядов дробной части числа,

a_i – цифра числа,

A_q – само число,

тогда развернутую форму для числа представленного в любой системе счисления можно записать в общем виде следующим образом:

$A_q = a_{n-1}\cdot q^{n-1} + a_{n-2}\cdot q^{n-2} + ... + a_0\cdot q^0 + a_{-1}\cdot q^{-1} + a_{-2}\cdot q^{-2} + ... + a_{-m}\cdot q^{-m}$

q^i – называется весом цифры числа.

Вес цифры числа равен степени, где основание степени равно основанию системы счисления, а показатель – номеру позиции цифры в числе.

Развернутая форма записи числа равна сумме произведений цифры числа на ее вес.

Примеры развернутых записей чисел в различных системах счисления:

$423{,}312_5 = 4\cdot5^2 + 2\cdot5^1 + 3\cdot5^0 + 3\cdot5^{-1}+1\cdot5^{-2}+2\cdot5^{-3}$

$423{,}312_8 = 4\cdot8^2 + 2\cdot8^1 + 3\cdot8^0 + 3\cdot8^{-1}+1\cdot8^{-2}+2\cdot8^{-3}$

Развернутая форма служит для перевода чисел из любой системы счисления в десятичную.

Алгоритм перевода чисел из любой системы счисления в десятичную:

1. Представить число в развернутой форме. При этом основание системы счисления должно быть представлено в десятичной системе счисления.
2. Найти сумму ряда (выражения). Полученное число является значением числа в десятичной системе счисления.

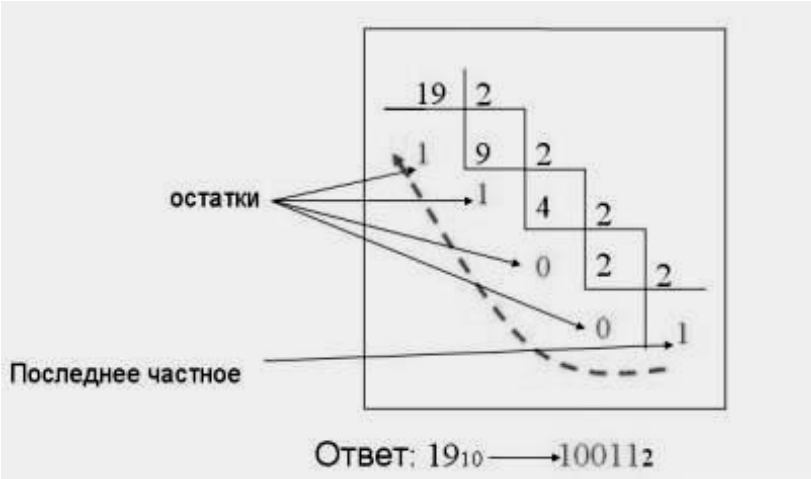
1.4 Перевод из десятичной системы счисления в любую другую

Перевод из десятичной системы счисления в любую другую более сложен, чем, наоборот, из любой в десятичную. При этом необходимо учитывать, что алгоритмы перевода целых чисел и правильных дробей различаются.

Алгоритм перевода целых чисел:

1. Разделить данное число на основание новой системы счисления. Зафиксировать целое частное и остаток от деления (остаток всегда меньше основания).
2. Если полученное частное больше основания, то разделить частное на основание и вновь зафиксировать новое частное и остаток от деления.
3. Повторять процесс до тех пор, пока частное не получится меньше делителя.
4. Записать последнее частное и полученные остатки в обратном порядке в ряд слева направо.

В качестве примера переведем 19_{10} в двоичную систему счисления согласно алгоритму.



Алгоритм перевода правильных десятичных дробей:

1. Последовательно выполнять умножение исходной десятичной дроби и получаемых дробей на основание системы до тех пор, пока не получим нулевую дробную часть или не будет достигнута требуемая точность вычислений.

2. Получить искомую дробную часть, записав полученные целые части произведения в прямой последовательности.

В качестве примера рассмотрим перевод десятичной дроби $0{,}75_{10}$ в двоичную систему, согласно алгоритму:

0,	**75**
	***2**
1	**50**
	***2**
1	**00**

Ответ: $0{,}75_{10} = 0{,}11_2$.

1.5 Перевод из двоичной системы счисления в шестнадцатеричную

1. Исходное число разбивается на тетрады (т. е. 4 цифры), начиная с младших разрядов. Если количество цифр исходного двоичного числа не кратно 4, оно дополняется слева незначащими нулями до достижения кратности 4.

2. Каждая тетрада заменятся соответствующей шестнадцатеричной цифрой в соответствии с таблицей 1.

Пример. Выполнить перевод числа 10011_2 в шестнадцатеричную систему счисления.

Поскольку в исходном двоичном числе количество цифр не кратно 4, дополняем его слева незначащими нулями до достижения кратности 4 числа цифр. Имеем:

$10011_2 = 00010011_2$

первая тетрада – младшая цифра числа

вторая тетрада – старшая цифра числа

В соответствии с таблицей 1:

$0011_2 = 11_2 = 3_{16}$ и $0001_2 = 1_2 = 1_{16}$.

Тогда $10011_2 = 13_{16}$.

1.6 Перевод из шестнадцатеричной системы счисления в двоичную

1. Каждая цифра исходного числа заменяется тетрадой двоичных цифр в соответствии с таблицей 2.1. Если в таблице двоичное число имеет менее 4 цифр, оно дополняется слева незначащими нулями до тетрады.

2. Незначащие нули в результирующем числе отбрасываются.

Пример. Выполнить перевод числа 13_{16} в двоичную систему счисления.

По таблице имеем:

$1_{16} = 1_2$ и после дополнения незначащими нулями двоичного числа $1_2 = 0001_2$;

$3_{16} = 11_2$ и после дополнения незначащими нулями двоичного числа $11_2 = 0011_2$.

Тогда $13_{16} = 00010011_2$. После удаления незначащих нулей имеем $13_{16} = 10011_2$.

Перевод **из двоичной системы в восьмеричную** выполняется аналогично правилу перевода из двоичной в шестнадцатеричную систему, за исключением того, что исходное число разбивается не по 4 цифры, а по 3.

Перевод **из восьмеричной ситемы в двоичную** выполняется аналогично правилу перевода из шестнадцатеричной системы счисления в двоичную, за исключением того, что исходное число разбивается не по 4 цифры, а по 3.

ВАРИАНТЫ ЗАДАНИЯ

Вариант 1. Найти числа под знаками «?»:
$1001000011_2 = ?_8 = ?_{10} = ?_{16}$
$1564_{10} = ?_2 = ?_8 = ?_{16}$
$A0B4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 2. Найти числа под знаками «?»:
$1101000011_2 = ?_8 = ?_{10} = ?_{16}$
$1464_{10} = ?_2 = ?_8 = ?_{16}$
$A1B4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 3. Найти числа под знаками «?»:
$1011000011_2 = ?_8 = ?_{10} = ?_{16}$
$1364_{10} = ?_2 = ?_8 = ?_{16}$
$A2B4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 4. Найти числа под знаками «?»:
$1001010011_2 = ?_8 = ?_{10} = ?_{16}$
$1264_{10} = ?_2 = ?_8 = ?_{16}$
$A3B4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 5. Найти числа под знаками «?»:
$1001001011_2 = ?_8 = ?_{10} = ?_{16}$
$1164_{10} = ?_2 = ?_8 = ?_{16}$
$A4B4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 6. Найти числа под знаками «?»:
$1001000111_2 = ?_8 = ?_{10} = ?_{16}$
$1064_{10} = ?_2 = ?_8 = ?_{16}$
$A5B4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 7. Найти числа под знаками «?»:
$1100000011_2 = ?_8 = ?_{10} = ?_{16}$
$1574_{10} = ?_2 = ?_8 = ?_{16}$
$A0C4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 8. Найти числа под знаками «?»:
$1010000011_2 = ?_8 = ?_{10} = ?_{16}$
$1584_{10} = ?_2 = ?_8 = ?_{16}$
$A0D4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 9. Найти числа под знаками «?»:
$1001100001_2 = ?_8 = ?_{10} = ?_{16}$
$1594_{10} = ?_2 = ?_8 = ?_{16}$
$A0E4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 10. Найти числа под знаками «?»:
$1000100011_2 = ?_8 = ?_{10} = ?_{16}$
$1565_{10} = ?_2 = ?_8 = ?_{16}$
$A0F4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 11. Найти числа под знаками «?»:
$1000010011_2 = ?_8 = ?_{10} = ?_{16}$
$1566_{10} = ?_2 = ?_8 = ?_{16}$
$A0A4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 12. Найти числа под знаками «?»:
$1001001011_2 = ?_8 = ?_{10} = ?_{16}$
$1567_{10} = ?_2 = ?_8 = ?_{16}$
$A094_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 13. Найти числа под знаками «?»:
$1111000011_2 = ?_8 = ?_{10} = ?_{16}$
$2564_{10} = ?_2 = ?_8 = ?_{16}$
$A1B7_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 14. Найти числа под знаками «?»:
$1011100011_2 = ?_8 = ?_{10} = ?_{16}$
$2464_{10} = ?_2 = ?_8 = ?_{16}$
$A2B6_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 15. Найти числа под знаками «?»:
$1001110011_2 = ?_8 = ?_{10} = ?_{16}$
$2364_{10} = ?_2 = ?_8 = ?_{16}$
$A2B5_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 16. Найти числа под знаками «?»:
$1001011011_2 = ?_8 = ?_{10} = ?_{16}$
$2264_{10} = ?_2 = ?_8 = ?_{16}$
$A2B4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 17. Найти числа под знаками «?»:
$1001001111_2 = ?_8 = ?_{10} = ?_{16}$
$2164_{10} = ?_2 = ?_8 = ?_{16}$
$A2B3_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 18. Найти числа под знаками «?»:
$1101001011_2 = ?_8 = ?_{10} = ?_{16}$
$1294_{10} = ?_2 = ?_8 = ?_{16}$
$AB04_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 19. Найти числа под знаками «?»:
$1011000111_2 = ?_8 = ?_{10} = ?_{16}$
$1374_{10} = ?_2 = ?_8 = ?_{16}$
$AB84_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 20. Найти числа под знаками «?»:
$1001100111_2 = ?_8 = ?_{10} = ?_{16}$
$1344_{10} = ?_2 = ?_8 = ?_{16}$
$AB94_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 21. Найти числа под знаками «?»:
$1011001011_2 = ?_8 = ?_{10} = ?_{16}$
$1234_{10} = ?_2 = ?_8 = ?_{16}$
$ABA4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 22. Найти числа под знаками «?»:
$1101000111_2 = ?_8 = ?_{10} = ?_{16}$
$1254_{10} = ?_2 = ?_8 = ?_{16}$
$ABB4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 23. Найти числа под знаками «?»:
$1000001011_2 = ?_8 = ?_{10} = ?_{16}$
$2124_{10} = ?_2 = ?_8 = ?_{16}$
$C484_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 24. Найти числа под знаками «?»:
$1000010101_2 = ?_8 = ?_{10} = ?_{16}$
$2215_{10} = ?_2 = ?_8 = ?_{16}$
$B8A4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 25. Найти числа под знаками «?»:
$1000101001_2 = ?_8 = ?_{10} = ?_{16}$
$3154_{10} = ?_2 = ?_8 = ?_{16}$
$F2B8_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 26. Найти числа под знаками «?»:
$1001010001_2 = ?_8 = ?_{10} = ?_{16}$
$3564_{10} = ?_2 = ?_8 = ?_{16}$
$AE43_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 27. Найти числа под знаками «?»:
$1010100001_2 = ?_8 = ?_{10} = ?_{16}$
$3156_{10} = ?_2 = ?_8 = ?_{16}$
$ECB1_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 28. Найти числа под знаками «?»:
$1101000001_2 = ?_8 = ?_{10} = ?_{16}$
$2159_{10} = ?_2 = ?_8 = ?_{16}$
$C5B4_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 29. Найти числа под знаками «?»:
$1000000001_2 = ?_8 = ?_{10} = ?_{16}$
$3654_{10} = ?_2 = ?_8 = ?_{16}$
$BAF7_{16} = ?_2 = ?_8 = ?_{10}$

Вариант 30. Найти числа под знаками «?»:
$1010010101_2 = ?_8 = ?_{10} = ?_{16}$
$3814_{10} = ?_2 = ?_8 = ?_{16}$
$B984_{16} = ?_2 = ?_8 = ?_{10}$

Лабораторная работа 10.2. Кодировки текста

Цель работы: Изучить особенности кодирования текста. Проверить создание текстовых документов в разных кодировках. Создать программу для чтения текстовых файлов.

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ

1. Ознакомиться с теоретическими сведениями.
2. Выполнить действия, описанные после теоретических сведений.
3. Защитить лабораторную работу перед преподавателем.

1.1 Кодирование текстовой информации в компьютере

Компьютерная обработка текстовой информации начала использоваться с середины 60-х годов. Помимо преимуществ, которые появляются при автоматическом внесении текстовых комментариев в результаты расчетных программ, создание, обработка и хранение текстовых документов в файловом виде представляет массу удобств.

При кодировании текста в память последовательно заносятся коды символов, составляющих текст, и команд, управляющих внешним видом и размещением этих символов. То есть если мы определяем числа 69 и 96 как текстовую информацию, коды этих чисел будут отличаться только порядком следования кодов цифр 6 и 9. Если же мы определяем их как числовую информацию, их коды будут совершенно различны, так как они представляют разные по величине числа.

Первоначально для кодирования одного символа использовался 1 байт. В байт можно записать в $2^8 = 256$ разных кодов (состояний). Эти состояния перенумерованы, и каждому сопоставляется какой-либо буквенный символ, графический элемент или команда, необходимая при оформлении текстовой информации. Такое соответствие называется ***кодовой таблицей.***

В настоящее время существуют и применяются разные варианты 8-битных кодовых таблиц. В России наиболее популярны следующие из низ:

- ***ASCII*** – American Standard Code for Information Interchange – американский стандартный код для обмена информацией.
- ***CP1251*** – (Code Page) – кодировка с кириллицей в Microsoft Windows.
- ***CP866*** – кодировка MS-DOS.
- ***КОИ8-Р*** – Код Обмена Информацией 8-битный с кириллицей.
- ***ISO 8859-5*** – International Standards Organization – Международная организация по стандартизации. Ещё один стандарт кодов для кириллицы.

Множество кодовых таблиц вызвано тем, что с учетом разнообразия естественных языков и фирм, выпускающих программное обеспечение, 256 состояний одного байта недостаточно для того, чтобы закодировать все встречающиеся символы и способы форматирования текста.

При разработке всех кодовых таблиц использовано следующее соглашение: первая половина таблицы – это коды с 0 по 127 – интернациональна, т.е. одинакова во всех вариантах кодировок. Первые 33 состояния (0–32) – это коды операций с текстом (перевод на новую строку, пробел, удаление последнего символа и т. п.). Затем состояния с 33 по 127 – это коды знаков препинания, арифметических действий, цифр, строчных и прописных букв ***латинского*** алфавита. Вторая половина кодовых таблиц отводится под знаки ***национальных и специальных*** алфавитов и ввода в текст графических элементов для оформления таблиц.

В конце 90-х годов появился новый международный стандарт ***Unicode***, который отводит под символ 2 байта. Каждый блок из 2-х байт может находиться в $2^{16} = 65536$

состояниях. Этого достаточно, чтобы в одной таблице собрать символы большинства алфавитов мира. Правда, длина текста удваивается, и скорость его обработки замедляется. Но, в связи с существенным увеличение памяти и быстродействия современных компьютеров, этот факт несущественен.

1.2 Текстовые файлы и текстовые редакторы

Для хранения текстовой информации в компьютере используются текстовые файлы. *Текстовым файлом* называется компьютерный файл, содержащий текстовые данные в виде последовательности символов. При этом каждый символ представляется в виде числа в определенной кодировке.

Для работы с текстовыми файлами существуют специальные программы – *текстовые редакторы*. Они могут быть отдельными самостоятельными компьютерными программами или являться компонентом программного комплекса.

а б

Рисунок 1. Пример текстового редактора как самостоятельной программы: а) «Блокнот»; б) «Notepad++»

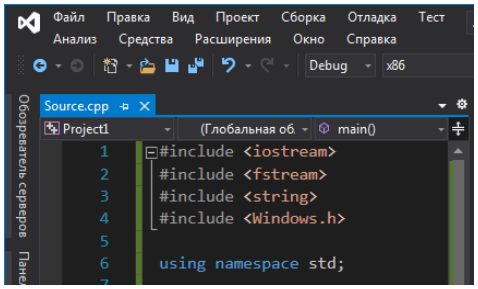


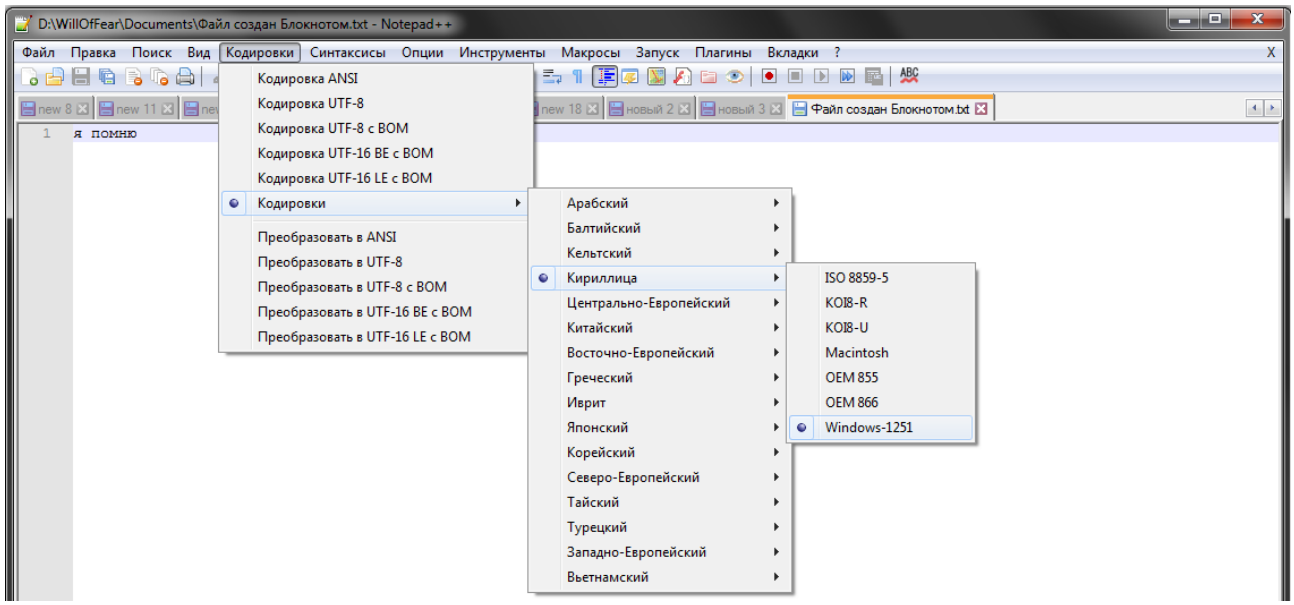
Рисунок 2. Пример текстового редактора как компоненты интегрированной среды разработки «Microsoft Visual Studio 2019»

Текстовые редакторы предназначены для работы с текстовыми файлами в интерактивном режиме. Они позволяют просматривать содержимое текстовых файлов и производить над ними различные действия: вставку, удаление и копирование текста, контекстный поиск и замену, и т.д.

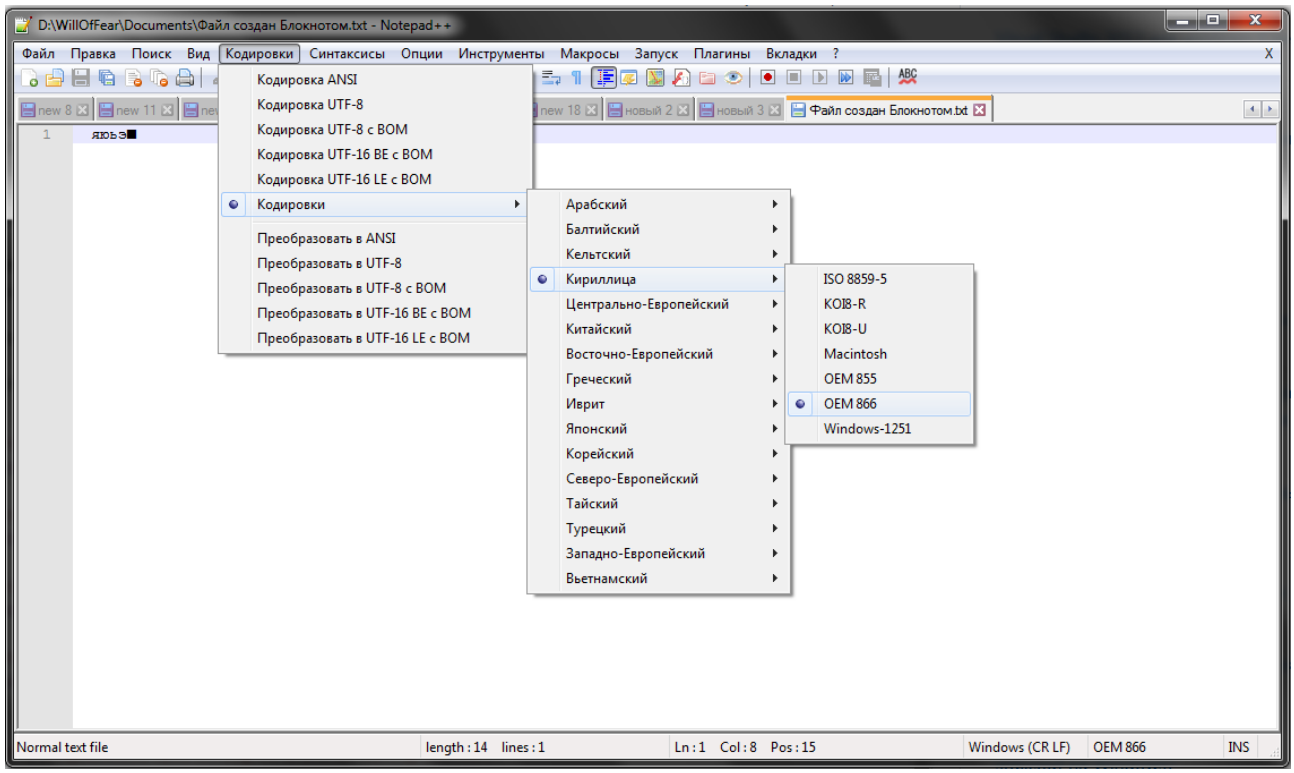
Часто интерактивные текстовые редакторы содержат дополнительную функциональность, призванную автоматизировать действия по редактированию (от записываемых последовательностей нажатий клавиш до полноценных встроенных языков программирования), отображать текстовые данные специальным образом (например, с подсветкой синтаксиса) или менять кодировку текста.

Кодировка, которая используется в текстовом файле по умолчанию, как правило, зависит от используемой операционной системы и установленного в ней языка. Так, если создать текстовый файл в русскоязычной версии операционной системы Windows 10 с помощью «Блокнота» или «Visual Studio», то будет использоваться кодировка CP1251.

Проверить это можно открыв файл из рисунка 1.а в «Notepad++» и перейдя в пункт меню «Кодировки»:

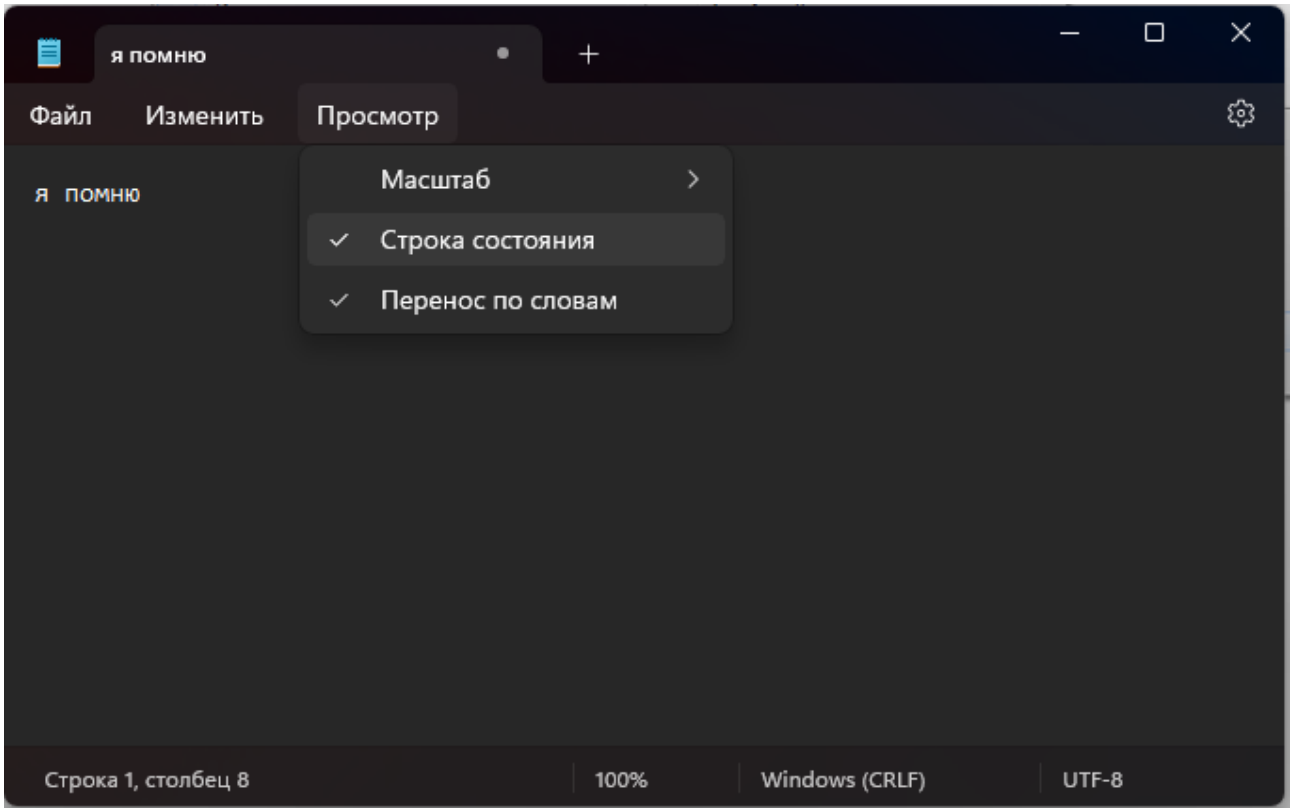


Чтобы корректно считать всю текстовую информацию, содержащуюся в текстовом файле, необходимо знать используемую кодировку. Если изменим кодировку отображения на CP866, то текст отобразится другой (так называемые «кракозябры»):

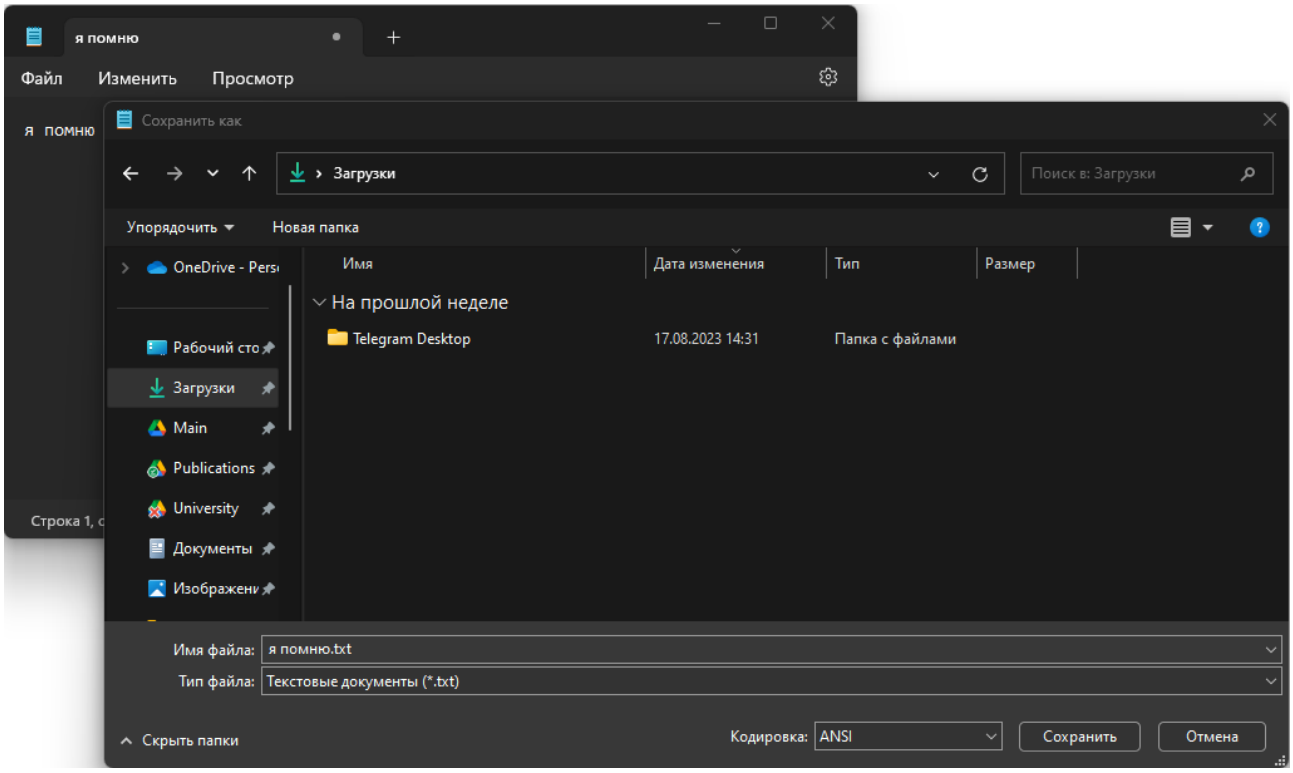


При этом содержание файла не изменилось (это всё тот же набор чисел), но поменялась таблица, по которой идёт сопоставление чисел и визуальных символов.

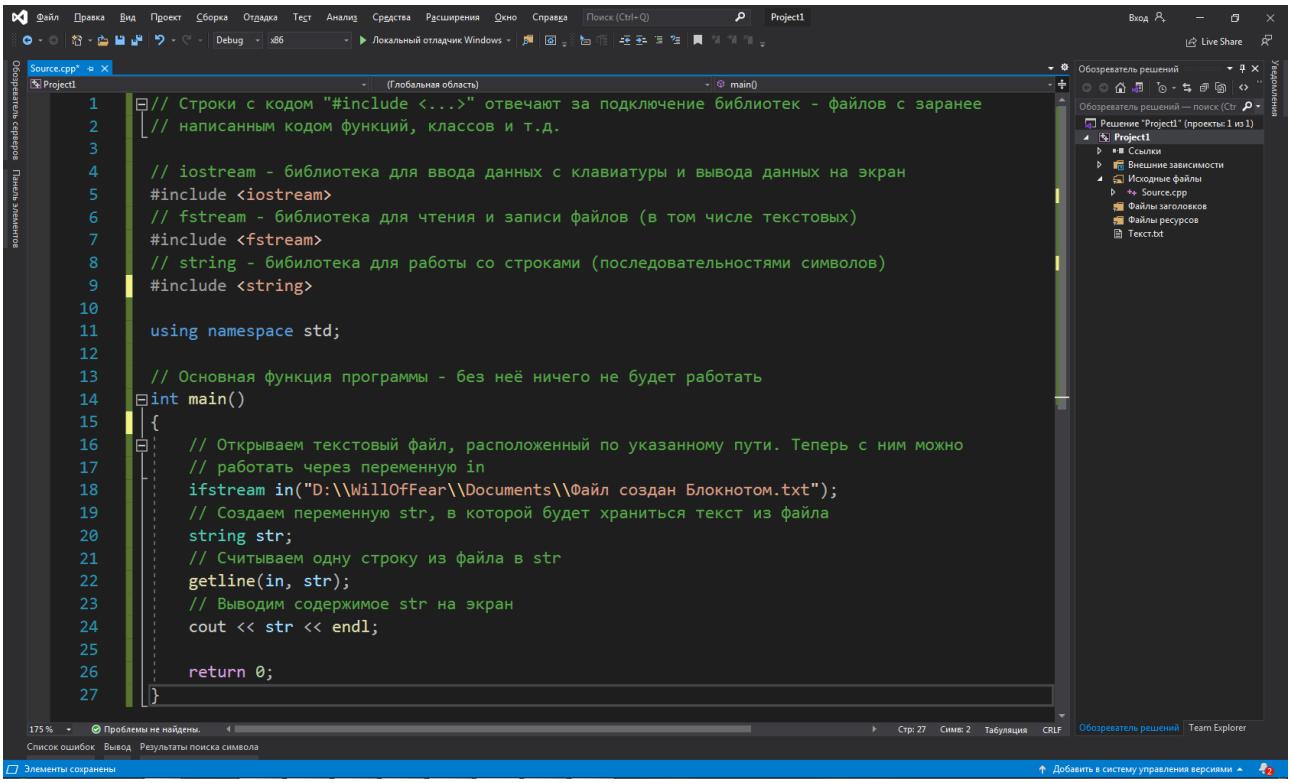
Проверить и поменять кодировку также можно в обычном «Блокноте». Для проверки убедитесь, что у вас стоит галочка напротив «Строки состояния» (пункт меню «Вид» или «Просмотр»). В нижней правой части будет отображаться кодировка.



Чтобы изменить UTF-8 на CP1251, необходимо выбрать пункт «Файл/Сохранить как» и в открывшемся диалоговом окне выбрать кодировку ANSI (нижняя правая часть окна).

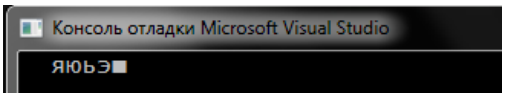


Теперь создадим проект в «Microsoft Visual Studio 2019» и попробуем прочитать данные из файла без указания конкретной кодировки. Для этого будем использовать следующий код:



```
// Строки с кодом "#include <...>" отвечают за подключение библиотек - файлов с заранее
// написанным кодом функций, классов и т.д.

// iostream - библиотека для ввода данных с клавиатуры и вывода данных на экран
#include <iostream>
// fstream - библиотека для чтения и записи файлов (в том числе текстовых)
#include <fstream>
// string - бибилотека для работы со строками (последовательностями символов)
#include <string>

using namespace std;

// Основная функция программы - без неё ничего не будет работать
int main()
{
    // Открываем текстовый файл, расположенный по указанному пути. Теперь с ним можно
    // работать через переменную in
    ifstream in("D:\\WillOfFear\\Documents\\Файл создан Блокнотом.txt");
    // Создаем переменную str, в которой будет храниться текст из файла
    string str;
    // Считываем одну строку из файла в str
    getline(in, str);
    // Выводим содержимое str на экран
    cout << str << endl;

    return 0;
}
```

Запустим код и получим следующий ответ:

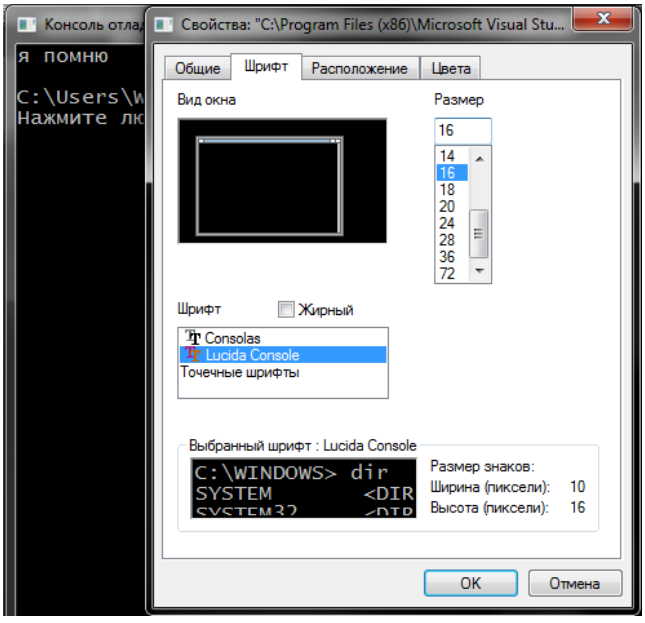


Вместо осмысленного текста «я помню» на экран выводятся кракозябры. Это связано с тем, что в текстовом файле текст набран в CP1251, а в консоли Windows используется кодировка CP866. Из-за этого целые числа, из которых состоит текстовый файл, преобразуется не в те символы. Чтобы это исправить, подключим библиотеку Windows.h и установим для консоли работу с кодировкой CP1251:

```
// iostream - библиотека для ввода данных с клавиатуры и вывода данных на экран
#include <iostream>
// fstream - библиотека для чтения и записи файлов (в том числе текстовых)
#include <fstream>
// string - бибилотека для работы со строками (последовательностями символов)
#include <string>
// Windows.h - библиотека с функциями характерными только для Windows.
// Нужна для указания кодировки
#include <Windows.h>
using namespace std;
// Основная функция программы - без неё ничего не будет работать
int main()
{
    SetConsoleCP(1251); // Кодировка для ввода
    SetConsoleOutputCP(1251); // Кодировка для вывода
    // Открываем текстовый файл, расположенный по указанному пути. Теперь с ним можно
    // работать через переменную in
    ifstream in("D:\\WillOfFear\\Documents\\Файл создан Блокнотом.txt");
    // Создаем переменную str, в которой будет храниться текст из файла
    string str;
    // Считываем одну строку из файла в str
    getline(in, str);
    // Выводим содержимое str на экран
    cout << str << endl;

    return 0;
}
```

Вместо строк с SetConsole… можно записать одну: `system("chcp1251");`

Дополнительно необходимо в свойствах консольного окна выбрать шрифт «Lucida Console» или «Consolas». После этого всё работает корректно, однако необходимо учитывать, что такой вариант подходит при обучении, но не реализации коммерческих программ.



Для более полного понимания проблемы кодировок рекомендуется прочитать следующую статью: https://zelserg.livejournal.com/2117.html.

ЗАДАНИЕ

Необходимо с помощью «Блокнота» создать текстовый файл с произвольным текстом, содержащим символы латинского и русского алфавита, а также числа. Узнать кодировку, в которой сохранились все эти символы, и проверить их отображение в других кодировках.

После этого необходимо создать проект в Visual Studio и, пользуясь теоретическими сведениями, корректно вывести на экран содержание текстового файла.