По окончании выполнения цикла `for` пользователю снова предлагается ввести сообщение, и программа переходит к началу внешнего цикла `while`, где вновь выполняется проверка того, что сообщение введено. Если это так, пользователь снова может ввести количество желаемых повторений своего сообщения, и так до момента, пока не оставит вопрос о вводе сообщения без ответа. После этого тело цикла `while` будет пропущено, и программа завершит свое выполнение.

3.4. Упражнения

Следующие упражнения должны быть выполнены при помощи циклов. В некоторых из них будет указано, какой именно тип цикла необходимо использовать. В противном случае вы вольны сами выбирать тип применяемого цикла. Некоторые задачи могут быть решены как с применением цикла `for`, так и с использованием `while`. Есть в этом списке и упражнения на применение множественных циклов, которые должны быть вложены друг в друга. Тщательно выбирайте способ решения задачи в каждом отдельном случае.

Упражнение 63. Среднее значение

(*26 строк*)

В данном упражнении вы должны написать программу для подсчета среднего значения всех введенных пользователем чисел. Индикатором окончания ввода будет служить ноль. При этом программа должна выдавать соответствующее сообщение об ошибке, если первым же введенным пользователем значением будет ноль.

Подсказка. Поскольку ноль является индикатором окончания ввода, его не нужно учитывать при расчете среднего.

Упражнение 64. Таблица со скидками

(*18 строк*)

В магазине была объявлена скидка размером 60 % на ряд товаров, и для того чтобы покупатели лучше ориентировались, владелец торговой точки решил вывесить отдельную таблицу со скидками с указанием уцененных товаров и их оригинальных цен. Используйте цикл для создания подобной таблицы, в которой будут исходные цены, суммы скидок и новые цены для покупок на сумму $4,95, $9,95, $14,95, $19,95 и $24,95. Убедитесь в том, что суммы скидки и новые цены отображаются с двумя знаками после запятой.

Упражнение 65. Таблица соотношения температур

(*22 строки*)

Напишите программу для вывода таблицы соотношения температур, выраженных в градусах Цельсия и Фаренгейта. В таблице должны размещаться все температуры между 0 и 100 градусами Цельсия, кратные 10. Дополните таблицу подходящими заголовками. Формулу для перевода температуры из градусов Цельсия в градусы Фаренгейта можно легко найти на просторах интернета.

Упражнение 66. Никаких центов

(*Решено. 39 строк*)

4 февраля 2013 года Королевским канадским монетным двором была выпущена последняя монета номиналом в один цент. После вывода центов из обращения все магазины вынуждены были изменить цены на товары таким образом, чтобы они стали кратны пяти центам (расчеты по банковским картам по-прежнему ведутся с учетом центов). И хотя продавцы вольны сами определять политику преобразования цен, большинство из них просто округлили цены до ближайших пяти центов.

Напишите программу, запрашивающую у пользователя цены, пока не будет введена пустая строка. На первой строке выведите сумму всех введенных пользователем сумм, а на второй – сумму, которую покупатель должен заплатить наличными. Эта сумма должна быть округлена до ближайших пяти центов. Вычислить сумму для оплаты наличными можно, получив остаток от деления общей суммы в центах на 5. Если он будет меньше 2,5, следует округлить сумму вниз, а если больше – вверх.

Упражнение 67. Найти периметр многоугольника

(*Решено. 42 строки*)

Напишите программу для расчета периметра заданного многоугольника. Начните с запроса у пользователя координат *x* и *y* первой точки многоугольника. Продолжайте запрашивать координаты следующих точек фигуры, пока пользователь не оставит строку ввода координаты по оси *x* пустой. После ввода каждой пары значений вы должны вычислить длину очередной стороны многоугольника и прибавить полученное значение к будущему ответу. По окончании ввода необходимо вычислить расстояние от последней введенной точки до первой, чтобы замкнуть фигуру, и вывести итоговый результат. Пример ввода координат точек многоугольника и вывода периметра показан ниже. Введенные пользователем значения выделены жирным.

```
Введите первую координату X: 0
Введите первую координату Y: 0
```

```
Введите следующую координату X (Enter для окончания ввода): 1
Введите следующую координату Y: 0
Введите следующую координату X (Enter для окончания ввода): 0
Введите следующую координату Y: 1
Введите следующую координату X (Enter для окончания ввода):
Периметр многоугольника равен 3.414213562373095
```

Упражнение 68. Средняя оценка

(*62 строки*)

В задаче 52 мы уже создавали таблицу соответствий между оценками в буквенном и числовом выражении. Сейчас вам нужно будет рассчитать среднюю оценку по произвольному количеству введенных пользователем буквенных оценок. Для окончания ввода можно использовать индикатор в виде пустой строки. Например, если пользователь последовательно введет оценки A, затем C+, а после этого B и пустую строку, средний результат должен составить 3,1.

Для расчетов вам может пригодиться математика из упражнения 52. Никаких проверок на ошибки проводить не нужно. Предположим, что пользователь может вводить только корректные оценки или ноль.

Упражнение 69. Билеты в зоопарк

(*Решено. 38 строк*)

В зоопарке цена входного билета зависит от возраста посетителя. Дети до двух лет допускаются бесплатно. Дети в возрасте от трех до 12 лет могут посещать зоопарк за $14,00. Пенсионерам старше 65 лет вход обойдется в $18,00, а обычный взрослый билет стоит $23,00.

Напишите программу, которая будет запрашивать возраст всех посетителей в группе (по одному за раз) и выводить общую цену билетов для посещения зоопарка этой группой. В качестве индикатора окончания ввода можно по традиции использовать пустую строку. Общую цену билетов стоит выводить в формате с двумя знаками после запятой.

Упражнение 70. Биты четности

(*Решено. 27 строк*)

Определение бита четности – это простой механизм выявления ошибок при передаче данных в условиях низкой надежности соединения, например по телефонной линии. Идея заключается в том, что после передачи каждых восьми бит следом отправляется бит четности, позволяющий определить наличие ошибки при передаче одного бита из восьми.

При этом можно использовать как контроль четности, так и контроль нечетности. В первом случае бит четности, посылаемый следом за груп-

пой из восьми бит данных, выбирается таким образом, чтобы общее количество переданных единичек в числе восьми бит данных и проверочного бита было четным. Во втором случае их количество должно быть нечетным.

Напишите программу, вычисляющую значение бита четности для групп из восьми бит, введенных пользователем, с использованием контроля четности. Пользователь может вводить комбинации из восьми бит бесконечное количество раз, а индикатором окончания ввода пусть снова будет пустая строка. После каждой введенной группы из восьми бит программа должна выводить на экран сообщение о том, чему должен равняться бит четности: 0 или 1. Также осуществляйте контроль ввода и выводите соответствующее сообщение об ошибке, если пользователь ввел последовательность, отличную от восьми бит.

Подсказка. Пользователь должен вводить последовательность в виде строки. После ввода вы можете определить количество нулей и единиц во введенной последовательности при помощи метода `count`. Информацию о работе этого метода можно найти в онлайне.

Упражнение 71. Приблизительное число π

(*23 строки*)

Приблизительное значение числа π можно вычислить по следующей бесконечной формуле:

$$\pi \approx 3 + \frac{4}{2\times3\times4} - \frac{4}{4\times5\times6} + \frac{4}{6\times7\times8} - \frac{4}{8\times9\times10} + \frac{4}{10\times11\times12} - \cdots.$$

Напишите программу, выводящую на экран 15 приближений числа π. В первом приближении должно быть использовано только первое слагаемое приведенного бесконечного ряда. Каждое очередное приближение должно учитывать следующее слагаемое, тем самым увеличивая точность расчета.

Упражнение 72. Игра Fizz-Buzz

(*17 строк*)

Fizz-Buzz – это известная игра, помогающая детям освоить в игровой форме правила деления. Участники садятся в круг, чтобы игра теоретически могла продолжаться бесконечно. Первый игрок говорит «Один» и передает ход тому, кто слева. Каждый следующий игрок должен мысленно прибавить к предыдущему числу единицу и произнести либо его, либо одно из ключевых слов: Fizz, если число без остатка делится на три,

или Buzz, если на пять. Если соблюдаются оба этих условия, он произносит Fizz-Buzz. Игрок, не сумевший сказать правильное слово, выбывает из игры. Последний оставшийся игрок признается победителем.

Разработайте программу, реализующую алгоритм игры Fizz-Buzz применительно к первым 100 числам. Каждый следующий ответ должен отображаться на новой строке.

Упражнение 73. Код Цезаря

(*Решено. 35 строк*)

Одним из первых в истории примеров шифрования считаются закодированные послания Юлия Цезаря. Римскому полководцу необходимо было посылать письменные приказы своим генералам, но он не желал, чтобы в случае чего их прочитали недруги. В результате он стал шифровать свои послания довольно простым методом, который впоследствии стали называть кодом Цезаря.

Идея шифрования была совершенно тривиальной и заключалась в циклическом сдвиге букв на три позиции. В итоге буква A превращалась в D, B – в E, C – в F и т. д. Последние три буквы алфавита переносились на начало. Таким образом, буква X становилась A, Y – B, а Z – C. Цифры и другие символы не подвергались шифрованию.

Напишите программу, реализующую код Цезаря. Позвольте пользователю ввести фразу и количество символов для сдвига, после чего выведите результирующее сообщение. Убедитесь в том, что ваша программа шифрует как строчные, так и прописные буквы. Также должна быть возможность указывать отрицательный сдвиг, чтобы можно было использовать вашу программу для расшифровки фраз.

Упражнение 74. Квадратный корень

(*14 строк*)

Напишите программу, реализующую метод Ньютона для нахождения квадратного корня числа x, введенного пользователем. Алгоритм реализации метода Ньютона следующий:

```
Запрашиваем число x у пользователя
Присваиваем переменной guess значение x / 2
Пока значение переменной guess не будет обладать должной точностью
    Присваиваем переменной guess результат вычисления среднего между guess и x / guess
```

По завершении алгоритма в переменной *guess* будет находиться определенное приближение вычисления квадратного корня из *x*. Качество аппроксимации при этом будет зависеть только от вашего желания. В нашем случае расхождение между значениями *guess* * *guess* и *x* должно составлять не более 10^{-12}.

Упражнение 75. Палиндром или нет?

(*Решено. 26 строк*)

Строка называется палиндромом, если она пишется одинаково в обоих направлениях. Например, палиндромами в английском языке являются слова «anna», «civic», «level», «hannah». Напишите программу, запрашивающую у пользователя строку и при помощи цикла определяющую, является ли она палиндромом.

Примечание. Яибофобия (Aibohphobia) – это безрассудный страх палиндромов. Эти слова в русском и английском сами по себе являются палиндромами, что и привело к их образованию. Напротив, яилифилия (ailihphilia) характеризуется любовью к палиндромам. Объяснять образование этого слова нет нужды.

Упражнение 76. Многословные палиндромы

(*35 строк*)

Помимо слов, существуют целые фразы, являющиеся палиндромами, если не обращать внимания на пробелы. Вот лишь несколько примеров на английском: «go dog», «flee to me remote elf» and «some men interpret nine memos». Русские варианты есть следующие: «А кобыле цена дана, да не целы бока», «А Луна канула» и другие. Расширьте свое решение упражнения под номером 75, чтобы при вынесении решения о том, является ли строка палиндромом, игнорировались пробелы. Также можете поработать над тем, чтобы игнорировались знаки препинания, а заглавные и прописные буквы считались эквивалентными.

Упражнение 77. Таблица умножения

(*Решено. 18 строк*)

В данном упражнении вы создадите программу для отображения стандартной таблицы умножения от единицы до десяти. При этом ваша таблица умножения должна иметь заголовки над первой строкой и слева от первого столбца, как показано в представленном примере. Предполагаемый вывод таблицы умножения показан ниже.

```
    1   2   3   4   5   6   7   8   9  10
1   1   2   3   4   5   6   7   8   9  10
2   2   4   6   8  10  12  14  16  18  20
3   3   6   9  12  15  18  21  24  27  30
4   4   8  12  16  20  24  28  32  36  40
5   5  10  15  20  25  30  35  40  45  50
6   6  12  18  24  30  36  42  48  54  60
7   7  14  21  28  35  42  49  56  63  70
```

```
 8   8  16  24  32  40  48  56  64  72  80
 9   9  18  27  36  45  54  63  72  81  90
10  10  20  30  40  50  60  70  80  90 100
```

Возможно, для выполнения этого упражнения вам придется озаботиться тем, чтобы выводить значения на экран без принудительного перевода каретки на строку ниже. Этого можно добиться, если последним аргументом функции `print` передать `end=""`. Например, инструкция `print("A")` выведет на экран букву A, после чего автоматически перейдет на новую строку, тогда как `print("A", end="")` не станет переводить каретку, что позволит произвести следующий вывод в той же строке.

Упражнение 78. Гипотеза Коллатца

(*Решено. 18 строк*)

Представьте себе последовательность целых чисел, организованную следующим образом.

```
Начинаться последовательность должна с любого положительного числа
Пока последний элемент последовательности не равен единице, выполнять
    Если последний элемент последовательности четный, тогда
        Добавить новый элемент к последовательности путем деления последнего элемента
на два с округлением вниз
    Иначе
        Добавить новый элемент к последовательности путем умножения последнего элемента
на три с добавлением единицы.
```

Гипотеза Коллатца утверждает, что подобная последовательность при условии того, что начинается с положительного числа, рано или поздно завершится единицей. И хотя это так и не было доказано, все указывает на то, что это так и есть.

Напишите программу, которая будет запрашивать у пользователя целое число и выводить все числа, начиная с введенного числа и заканчивая единицей. После этого пользователь должен иметь возможность ввести другое число и снова получить ряд чисел, называемый сиракузской последовательностью. Условием выхода из программы должен быть ввод пользователем нуля или отрицательного числа.

Примечание. Гипотеза Коллатца по сей день остается одной из нерешенных проблем математики. Многие пытались представить доказательство, но никто не добился успеха.

Упражнение 79. Наибольший общий делитель

(*Решено. 17 строк*)

Наибольший общий делитель двух положительных чисел представляет собой наибольшее число, на которое без остатка делятся оба числа. Существует несколько алгоритмов, позволяющих определить наибольший общий делитель двух чисел, включая следующий.

```
Инициализируйте переменную d меньшим из чисел n и m
Пока n или m не делятся на d без остатка, выполнять
    Уменьшить d на единицу
Выведите на экран d, это и есть наибольший общий делитель для n и m
```

Напишите программу, запрашивающую у пользователя два положительных целых числа и выводящую для них наибольший общий делитель.

Упражнение 80. Простые множители

(*27 строк*)

Разложение целого числа *n* на простые множители может быть проведено по следующему алгоритму.

```
Инициализируйте переменную factor значением 2
Пока значение factor меньше или равно n, выполнять
    Если n без остатка делится на factor, тогда
        Сохранить factor как простой множитель числа n
        Разделить n на factor с округлением вниз
    Иначе
        Увеличить factor на единицу
```

Напишите программу, которая будет запрашивать целое число у пользователя. Если пользователь введет значение, меньшее двух, необходимо вывести соответствующее сообщение об ошибке. Иначе нужно перечислить в столбец список простых множителей заданного числа, которые при перемножении дали бы исходное число. Например:

```
Введите целое число (2 или больше): 72
Простые множители числа 72:
2
2
2
3
3
```

Упражнение 81. Двоичное число в десятичное

(*18 строк*)

Напишите программу, которая будет преобразовывать двоичные значения (по основанию 2) в десятичные (по основанию 10). Пользователь должен ввести число в двоичном виде как строку, а программа – преобразовать его посимвольно в десятичный вид и вывести на экран с соответствующим сообщением.

Упражнение 82. Десятичное число в двоичное

(*Решено. 27 строк*)

Напишите программу, которая будет преобразовывать десятичные значения (по основанию 10) в двоичные (по основанию 2). Запросите целое число у пользователя и, следуя алгоритму, приведенному ниже, преобразуйте его в двоичную запись. По завершении выполнения программы в переменной *result* должно оказаться двоичное представление исходного числа. Выведите результат на экран с соответствующим сообщением.

```
Инициализируйте переменную result пустой строкой
Пусть в переменной q хранится число, которое нужно преобразовать
Повторять
    Переменной r присвоить остаток от деления q на 2
    Преобразовать r в строку и добавить ее в начало переменной result
    Разделить q на 2 с отбрасыванием остатка и присвоить полученное значение переменной q
Пока q не станет равно нулю
```

Упражнение 83. Максимальное число в последовательности

(*Решено. 34 строки*)

Это упражнение преследует цель идентификации количества смен максимального значения в коллекции случайных чисел. Ряд должен быть заполнен числами от 1 до 100. При этом последовательность может содержать дубликаты, а некоторых чисел из диапазона от 1 до 100 в ней может не быть.

Сделайте паузу и подумайте с листочком в руках, как вы решили бы эту задачу. Многие стали бы сравнивать каждое очередное выпавшее число с текущим максимумом в последовательности и при необходимости обновлять максимум. Это вполне подходящий способ, который приведет к правильному результату при соблюдении алгоритма. А как вы думаете, сколько раз обновится максимум на протяжении генерирования всей последовательности?

На этот вопрос можно ответить при помощи теории вероятностей, но мы попробуем провести необходимые симуляции. Для начала в вашей

программе должно выбираться случайное число из диапазона от 1 до 100. Сразу сохраните это значение как максимальное. Далее сгенерируйте еще 99 случайных чисел в том же диапазоне. Для каждого значения выполняйте ту же проверку и при необходимости обновляйте максимум, попутно увеличивая переменную, хранящую количество «смен лидера», на единицу. Выводите каждое сгенерированное число на новой строке, добавляя примечание в случае, если на этом шаге обновлялся максимум.

После вывода всех значений на экран вы должны также оповестить пользователя о максимальном числе в ряду и количестве его обновлений. Частичный вывод программы приведен ниже. Запустите свою программу несколько раз. Оправдались ли ваши ожидания относительно количества смен максимального значения?

```
30
74 <== Обновление
58
17
40
37
13
34
46
52
80 <== Обновление
37
97 <== Обновление
45
55
73
…

Максимальное значение в ряду: 100
Количество смен максимального значения: 5
```

Упражнение 84. Подбрасываем монетку

(*47 строк*)

Какое минимальное количество раз вы должны подбросить монетку, чтобы три раза подряд выпал либо орел, либо решка? А какое максимальное количество попыток может для этого понадобиться? А в среднем? В данном упражнении мы выясним это, создав симулятор подбрасывания виртуальной монетки.

Напишите программу, использующую для подброса монетки генератор случайных чисел Python. Монетка при этом должна быть правильной формы, что означает равную вероятность выпадения орла и решки. Подбрасывать монетку необходимо до тех пор, пока три раза подряд не вы-

падет одно значение, вне зависимости от того, орел это будет или решка. Выводите на экран букву `O` всякий раз, когда выпадает орел, и `P` – когда выпадает решка. При этом для одной симуляции бросков все выпавшие значения необходимо размещать на одной строке. Также необходимо известить пользователя о том, сколько попыток потребовалось, чтобы получить нужный результат.

Программа должна выполнить десять симуляций и в конце представить среднее количество подбрасываний монетки, требуемое для достижения нужного нам результата. Пример вывода программы показан ниже:

```
O P P P (попыток: 4)
O O P P O P O P P O O P O P P O P P P (попыток: 19)
P P P (попыток: 3)
P O O O (попыток: 4)
O O O (попыток: 3)
P O P P O P O O P P O O P O P O O O (попыток: 18)
O P P O O O (попыток: 6)
P O P P P (попыток: 5)
P P O P P O P O P O O O (попыток: 12)
P O P P P (попыток: 5)
Среднее количество попыток: 7,9.
```