Р. Задание к лабораторным работам по АиП

Касилов Василий

1 ноября 2024 г.

Версия 1.0

Задания к лабораторным работам 1-го семестра по дисциплине «Алгоритмизация и программирования» выполняются и принимаются по порядку. В рамках лабораторных работ не разрешается использовать стандартные контейнеры и `std::stringstream`. Допускается использование `std::pair` и `std::function`. Для реализации тестов допустимо использовать любые классы стандартной библиотеки.

Все реализуемые сущности должны быть расположены в отдельном пространстве имен. Имя этого пространства должно совпадать с фамилией студента в нижнем регистре (соответственно, оно совпадает с частью имени каталога с работами до точки), например, для Петрова Ивана каталог будет называться petrov.ivan, а имя пространства имен — `petrov`. Это пространство имен должно использоваться для всех работ.

0 Вступительная работа

Реализовать программу, которая выводит на стандартный вывод фамилию и имя студента, разделённые символом «.». Необходимо учесть:

1. Вывод должен быть на отдельной строке
2. Для вывода должны быть использованы ASCII-символы
3. Выводимая строка должна соответствовать имени каталога, в котором находятся работы студента
4. Работа должна быть выполнена в виде 1-го исполняемого файла, не обрабатывающего параметры командной строки:

```
$ ./lab
petrov.ivan
```

1 Обработка последовательностей I

Реализовать программу, которая рассчитывает характеристику последовательности целых чисел, введённых со стандартного ввода. Характеристика должна быть расчитана в соответствии с заданным преподавателем вариантом. Необходимо учесть:

1. Хранение элементов последовательности в динамическом массиве не допускается
2. Последовательность чисел содержит как отрицательные, так и положительные значения
3. Ввод последовательности завершается нулём. Ноль не считается членом последовательности
4. Последовательность может быть слишком короткой или слишком длинной, что потенциально не позволит обработать последовательность заданным образом
5. Программа должна завершаться с кодом возврата `1`, если входные данные нельзя идентифицировать как последовательность, и соответствующим сообщением об ошибке в стандартном потоке ошибок
6. Программа должна завершаться с кодом возврата `2` и соответствующим сообщением в станадртном потоке ошибок, если невозможно рассчитать хотя бы одну характеристику последовательности (например, последовательность слишком короткая). При этом, остальные характеристики должны быть рассчитаны
7. Каждая рассчитанная характеристика должна быть выведена на отдельной строке

Примеры входных данных, не являющихся последовательностью, для которых программа должна завершиться с кодом возврата `1`:

```
Numbers: 10 11 12 0
10000000000000000000000000000000 0
one two three zero
1.0 2.0 3.0 0
1 2 3 four 0
```

Примеры входных данных, являющихся последовательностью:

```
10 -5 15 0
-2 2 0
1 0
0
```

Студенту необходимо получить один (или более) вариантов у преподавателя. Далее приведён список вариантов и ожидаемого поведения программы, при вводе соответствующих последовательностей:

1. [INC-SEQ] Определить, сколько элементов последовательности больше предыдущего элемента

   ```
   1 2 3 -2 -1 0 //Expects output (return code 0): 3
   1 0 //Expects output (return code 0): 0
   0 //Expects output (return code 0): 0
   ```

2. [SUB-MAX] Определить значение второго по величине элемента последовательности. То есть элемента, который был бы наибольшим, если из последовательности удалить максимальный

   ```
   1 2 3 2 1 0 //Expects output (return code 0): 2
   1 0 //Expects error output (return code 2)
   0 //Expects error output (return code 2)
   ```

3. [CNT-MAX] Определить количество элементов последовательности равных максимальному элементу

   ```
   1 3 2 3 1 0 //Expects output (return code 0): 2
   1 0 //Expects output (return code 0): 1
   0 //Expects error output (return code 2)
   ```

4. [EQL-SEQ] Определить максимальное число подряд идущих равных элементов

   ```
   1 4 4 2 2 2 4 4 0 //Expects output (return code 0): 3
   1 0 //Expects output (return code 0): 1
   0 //Expects output (return code 0): 0
   ```

5. [MON-DEC] Определить наибольшую длину монотонно-убывающего фрагмента последовательности

   ```
   3 2 8 4 4 2 1 0 //Expects output (return code 0): 5
   1 0 //Expects output (return code 0): 1
   0 //Expects output (return code 0): 0
   ```

6. [MON-INC] Определить наибольшую длину монотонно-возрастающего фрагмента последовательности

   ```
   2 2 4 1 3 3 5 0 //Expects output (return code 0): 4
   1 0 //Expects output (return code 0): 1
   0 //Expects output (return code 0): 0
   ```

7. [LOC-MAX] Определить количество локальных максимумов в последовательности

   ```
   1 3 2 0 //Expects output (return code 0): 1
   1 3 3 2 4 1 0 //Expects output (return code 0): 1
   1 0 //Expects output (return code 0): 0
   0 //Expects error output (return code 2)
   ```

8. [LOC-MIN] Определить количество локальных минимумов в последовательности

   ```
   2 1 2 0 //Expects output (return code 0): 1
   2 1 2 4 3 0 //Expects output (return code 0): 1
   1 0 //Expects output (return code 0): 0
   0 //Expects error output (return code 2)
   ```

9. [SGN-CHG] Определить количество перемен знаков последовательности

   ```
   -2 1 -2 0 //Expects output (return code 0): 2
   -2 -3 -1 0 //Expects output (return code 0): 0
   1 0 //Expects output (return code 0): 0
   0 //Expects output (return code 0): 0
   ```

10. [GRT-LSS] Определить, сколько элементов последовательности меньше предыдущего элемента, но больше следующего

```
5 4 3 2 3 0 //Expects output (return code 0): 2
-2 -3 -4 0 //Expects output (return code 0): 1
1 0 //Expects output (return code 0): 0
0 //Expects output (return code 0): 0
```

11. [DIV-REM] Определить количество элементов последовательности делящихся на предыдущий без остатка

```
2 4 8 10 20 0 //Expects output (return code 0): 3
-2 4 -8 0 //Expects output (return code 0): 2
1 0 //Expects error output (return code 2)
0 //Expects error output (return code 2)
```

12. [AFT-MAX] Определить количество элементов последовательности расположенных после максимального

```
1 3 2 1 0 //Expects output (return code 0): 2
1 3 2 3 0 //Expects output (return code 0): 2
1 0 //Expects output (return code 0): 0
0 //Expects error output (return code 2)
```

13. [PTH-TRP] Определить количество пифагорофых троек из идущих подряд элементов последовательности

```
3 4 5 3 4 5 0 //Expects output (return code 0): 2
3 4 5 4 3 0 //Expects output (return code 0): 1
1 2 3 0 //Expects output (return code 0): 0
1 2 0 //Expects output (return code 0): 0
1 0 //Expects output (return code 0): 0
0 //Expects output (return code 0): 0
```

14. [EVN-CNT] Определить максимальное число подряд идущих чётных элементов

```
4 2 -1 2 4 6 0 //Expects output (return code 0): 3
1 2 0 //Expects output (return code 0): 1
1 0 //Expects output (return code 0): 0
0 //Expects output (return code 0): 0
```

15. [SUM-DUP] Определить количество элементов последовательности равных сумме двух предыдущих элементов

```
1 2 3 5 3 8 0 //Expects output (return code 0): 3
1 2 0 //Expects error output (return code 2)
1 0 //Expects error output (return code 2)
0 //Expects error output (return code 2)
```

16. [CNT-MIN] Определить количество элементов последовательности равных минимальному элементу

```
1 2 1 2 1 0 //Expects output (return code 0): 3
1 0 //Expects output (return code 0): 1
0 //Expects output (return code 0): 0
```

2 Обработка последовательностей II

Реализовать программу, строящую таблицу значений указанной функции, вычисленной с помощью ряда Тейлора. Программа не обрабатывает параметры командной строки. Кроме того, программа последовательно принимает со стандартного ввода:

1. начало и конец интервала вычисления,
2. максимальное число слагаемых,

Шаг вычислений и абсолютная погрешность пусть будут заданы внутри программы и выбираются студентом самостоятельно. Пусть программа реализована в соответствии с требованиями:

```
-0.5 0.5 10
```

Вычисления производятся на интервале от -0.5 до 0.5 с заданным в программе шагом. При этом количество слагаемых для каждого значения не должно превышать 10 с учётом выбранной точности.

Для вычисления значения с помощью ряда Тейлора должна быть реализована отдельная функция вида:

```
double f(double x, size_t k, double error);
```

Соответственно: x является аргументом функции, количество слагаемых не превышает k, абсолютная погрешность равна `error`. Вычисление каждого слагаемого должно быть выполнено на основе предыдущего, поэтому использовать, например, функции возведения в степень или факториала для расчета членов ряда не допускается. Функция должна генерировать исключение, если указанная точность вычислений не достигнута.

Программа должна вывести таблицу значений в стандартный вывод, в каждой строке которой последовательно указаны:

1. аргумент для которого вычисляется значение функции,
2. рассчитанное значение с помощью ряда Тейлора,
3. значение, рассчитанное с помощью стандартных математических функций из библиотеки `<cmath>`

Если указанная точность не достигнута при вычислении последнего слагаемого, то вместо рассчитанного значения вывести в таблице сообщение `<MATH ERROR>`. Первая и последняя строка таблицы должны содержать вычисления для левой границы интервала и для правой соответственно.

Программа должна завершаться с кодом возврата 1, если:

1. не удалось распознать ввод,
2. интервал задан некорректно,
3. интервал не находится целиком в области определения соответствующей функции

Во всех остальных случаях программа должна завершаться с кодом возврата 0.

Для реализации предлагается одна из указанных функций:

1. `[LN-X-SQRX2]` $\ln(x + \sqrt{1 + x^2})$
2. `[ARC-TG]` $\operatorname{arctg} x$
3. `[ARC-SIN]` $\arcsin x$
4. `[UNO-DIV-SQR]` $\frac{1}{\sqrt{1-x^2}}$
5. `[EXP-NEGX]` $\exp -x$
6. `[ARC-TGH]` $\operatorname{artanh} x$
7. `[SIN]` $\sin x$
8. `[COS]` $\cos x$
9. `[SINH]` $\sinh x$
10. `[COSH]` $\cosh x$
11. `[EXP-X]` $\exp x$
12. `[SINX-DIVX]` $\frac{\sin x}{x}$
13. `[SQR-UNOX]` $\sqrt{1 + x}$
14. `[EXP-POW2X]` $\exp -x^2$
15. `[UNO-DIV-CUBE]` $\frac{1}{1+x^2}$

3 Массивы

Реализовать программу, обрабатывающую параметры командной строки. Поведение программы должно меняться в зависимости от введённых параметров.

1. Программа должна считывать из файла двумерный массив, сохранять его в массив фиксированного размера или в динамических массив (в зависимости от параметра командной строки), выполнять его обработку в соответствии с заданными вариантами и выводить результаты обработки в другой файл

2. Программа принимает 3 параметра командной строки:

   ```
   ./lab num input output
   ```

   `num` — номер задания (`1` для массива фиксированного размера, `2` для динамического массива); `input` — имя файла, содержащего двумерный массив; `output` — имя файла, в который нужно вывести результаты обработки массива

3. Входной файл содержит двумерный целочисленный массив — матрицу: количество строк, количество столбцов, сами элементы. Размерности матрицы и элементы разделены друг от друга пробелом. Примеры описания двумерных массивов:

   ```
   2 3 -1 2 -3 4 -5 6
   1 1 -1
   0 0
   ```

4. Считанный массив должен быть сохранён в массиве фиксированного размера на стеке, если первый параметр командной строки `num` равен `1`. В этом случае гарантируется, что количество элементов матрицы не превышает 10000 элементов. Если первый параметр командной строки `num` равен `2`, то для хранения массива должен быть использован динамический массив, расположенный во free store.

5. После считывания массив должен быть обработан в соответствии с заданным вариантом и результат этой обработки должен быть выведен в файл. Программа должна завершиться с кодом возврата `0`

6. Если количество аргументов командной строки не соответствует описанию или первый параметр нельзя интерпретировать как указанные номера заданий, программа должна завершаться с кодом возврата `1` и сообщением об ошибке в стандартном потоке ошибок. Примеры запуска, в которых аргументы командной строки не соответствуют описанию:

   ```
   ./lab 1 //Not enough arguments
   ./lab 2 input.txt output.txt errors.txt //Too many arguments
   ./lab 3 input.txt output.txt //First parameter is out of range
   ./lab first input.txt output.txt //First parameter is not a number
   ./lab "2 second" input.txt output.txt //First parameter is not a number
   ```

7. Если содержимое файла нельзя интерпретировать как двумерный массив или файл пуст, программа должна завершаться с кодом возврата `2` и сообщением об ошибке в стандартном потоке ошибок. Примеры данных, не являющихся двумерным массивом:

   ```
   My matrix: 2 2 1 2 3 4
   2 3 10000000000000000000000000000000000 2 3 4
   a 2 3 1 2 3 4 5 6
   2 a 3 1 2 3 4 5 6
   2 3 1 2 3 4 5
   ```

Студенту необходимо получить один (или более) вариантов у преподавателя. Некоторые варианты требуют квадратную матрицу. В этом случае требуется «обрезать» ее: обрезанная матрица должна быть максимально возможного размера; если таких матриц несколько, то можно выбрать любую. Далее приведён список вариантов и ожидаемого поведения программы, для соответствующих матриц:

1. `[CNT-SDL-PNT]` Рассчитать количество седловых элементов матрицы. Элемент называется седловым, если он является минимальным в строке и максимальным в столбце

   ```
   2 3 1 2 3 5 4 6 //Expects output (return code 0): 1
   2 3 4 2 3 1 5 6 //Expects output (return code 0): 0
   0 0 //Expects output (return code 0): 0
   ```

2. [CNT-LOC-MIN] Рассчитать количество локальных минимумов, где локальным минимумом называется элемент, который строго меньше всех соседних элементов и не расположен на границе матрицы

```
3 3 3 2 3 2 1 2 2 3 2 //Expects output (return code 0): 1
5 3 3 2 3 2 1 2 3 2 3 2 1 2 3 2 3 //Expects output (return code 0): 2
2 2 1 2 2 2 //Expects output (return code 0): 0
0 0 //Expects output (return code 0): 0 0
```

3. [CNT-LOC-MAX] Рассчитать количество локальных максимумов, где локальным максимумом называется элемент, который строго больше всех соседних элементов и не расположен на границе матрицы

```
3 3 3 2 3 2 5 2 2 3 2 //Expects output (return code 0): 1
5 3 3 2 3 2 5 2 3 2 3 2 5 2 3 2 3 //Expects output (return code 0): 2
2 2 3 2 2 2 //Expects output (return code 0): 0
0 0 //Expects output (return code 0): 0 0
```

4. [LFT-BOT-CLK] Преобразовать матрицу следующим образом: уменьшить элемент, расположенный в левом нижнем углу на 1, далее, двигаясь по часовой стрелке (по спирали), уменьшать элементы на 2, на 3,и так далее, пока все элементы матрицы не будут изменены

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 3 3 -2 -2 -2 2 -4 0 6 0 2
2 2 1 2 3 4 //Expects output (return code 0): 2 2 -1 -1 2 0
0 0 //Expects output (return code 0): 0 0
```

5. [LFT-TOP-CLK] Преобразовать матрицу следующим образом: уменьшить элемент, расположенный в левом верхнем углу на 1, далее, двигаясь по часовой стрелке (по спирали), уменьшать элементы на 2, на 3, и так далее, пока все элементы матрицы не будут изменены

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 3 3 0 0 0 -4 -4 2 0 2 4
2 2 1 2 3 4 //Expects output (return code 0): 2 2 0 0 -1 1
0 0 //Expects output (return code 0): 0 0
```

6. [LFT-BOT-CNT] Преобразовать матрицу следующим образом: увеличить элемент, расположенный в левом нижнем углу на 1, далее, двигаясь против часовой стрелки (по спирали), увеличивать элементы на 2, на 3 и так далее, пока все элементы матрица не будут изменены

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 3 3 8 8 8 12 14 10 8 10 12
2 2 1 2 3 4 //Expects output (return code 0): 2 2 5 5 4 6
0 0 //Expects output (return code 0): 0 0
```

7. [LFT-TOP-CNT] Преобразовать матрицу следующим образом: увеличить элемент, расположенный в левом верхнем углу на 1, далее, двигаяь против часовой стрелки (по спирали), увеличивать элементы на 2, на 3 и так далее, пока все элементы матрицы не будут изменены

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 3 3 2 10 10 6 14 12 10 12 14
2 2 1 2 3 4 //Expects output (return code 0): 2 2 2 6 5 7
0 0 //Expects output (return code 0): 0 0
```

8. [FLL-INC-WAV] Преобразовать матрицу следующим образом: увеличить значения на переферии матрицы на 1, элементы оставшейся внутренней матрицы на 2, оставшейся - на 3 и так далее, пока не будут преобразованы все элементы матрицы

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 3 3 2 3 4 5 7 7 8 9 10
2 2 1 2 3 4 //Expects output (return code 0): 2 2 2 3 4 5
0 0 //Expects output (return code 0): 0 0
```

9. [CNT-COL-NSM] Количество столбцов, в которых нет подряд идущих одинаковых элементов

```
3 3 1 2 3 1 3 4 1 3 3 //Expects output (return code 0): 1
2 2 1 2 3 4 //Expects output (return code 0): 2
0 0 //Expects output (return code 0): 0
```

10. [CNT-ROW-NSM] Количество строк, в которых нет подряд идущих одинаковых элементов

```
3 3 1 2 3 4 5 6 7 8 8 //Expects output (return code 0): 2
2 2 1 2 3 4 //Expects output (return code 0): 2
0 0 //Expects output (return code 0): 0
```

11. [NUM-COL-LSR] Номер столбца, в котором находится самая длинная серия подряд идущих равных элементов

```
3 3 1 2 3 1 2 3 2 3 3 //Expects output (return code 0): 3
2 2 1 2 1 3 //Expects output (return code 0): 1
0 0 //Expects output (return code 0): 0
```

12. [NUM-ROW-LSR] Номер строки, в которой находится самая длинная серия подряд идущих равных элементов

```
3 3 1 1 2 3 3 3 4 5 5 //Expects output (return code 0): 2
2 2 1 1 2 3 //Expects output (return code 0): 1
0 0 //Expects output (return code 0): 0
```

13. [MAX-SUM-MDG] Найти максимальную сумму элементов среди диагоналей, параллельных побочной диагонали

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 14
2 2 1 2 3 4 //Expects output (return code 0): 3
0 0 //Expects output (return code 0): 0
```

14. [MAX-SUM-SDG] Найти максимальную сумму элементов среди диагоналей, параллельных главной диагонали

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 12
2 2 1 2 3 4 //Expects output (return code 0): 4
0 0 //Expects output (return code 0): 0
```

15. [MIN-SUM-MDG] Найти минимальную сумму элементов среди диагоналей, параллельных побочной диагонали

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 1
2 2 1 2 3 4 //Expects output (return code 0): 2
0 0 //Expects output (return code 0): 0
```

16. [MIN-SUM-SDG] Найти минимальную сумму элементов среди диагоналей, параллельных главной диагонали

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 1
2 2 1 2 3 4 //Expects output (return code 0): 1
0 0 //Expects output (return code 0): 0
```

17. [BLD-SMT-MTR] Построить сглаженную матрицу. Сглаженная матрица строится на основе имеющейся: каждый её элемент равен среднему арифметическому соседних элементов матрицы. Элементы матрицы выводить с точностью до десятых

```
3 3 1 2 3 4 5 6 7 8 9 //Expects output (return code 0): 3 3 3.7 3.8 4.3 3.8 5 4.5 5.7 5.2 6.3
2 2 1 2 3 4 //Expects output (return code 0): 2 2 3 2.7 3.3 2
0 0 //Expects output (return code 0): 0 0
```

18. [CNT-NZR-DIG] Посчитать количество диагоналей, не содержащих нулевых элементов и параллельных главной диагонали

```
3 3 0 1 0 3 4 5 0 7 8 //Expects output (return code 0): 2
2 2 1 2 0 4 //Expects output (return code 0): 1
0 0 //Expects output (return code 0): 0
```

19. [LWR-TRI-MTX] Проверить, является ли матрица нижней треугольной

```
3 3 1 0 0 4 5 0 7 8 9 //Expects output (return code 0): true
2 2 1 2 3 4 //Expects output (return code 0): false
0 0 //Expects output (return code 0): false
```

20. [UPP-TRI-MTX] Проверить, является ли матрица верхней треугольной

```
3 3 1 2 3 0 5 6 0 0 9 //Expects output (return code 0): true
2 2 1 2 3 4 //Expects output (return code 0): false
0 0 //Expects output (return code 0): false
```

4 Строки

Реализовать программу, которая считывает со стандартного ввода строку заранее неизвестного размера и преобразовывает её в соотвествии с заданным преподавателем вариантом или рассчитывает указанную характеристику. Результат преобразования выводится в стандартный вывод. Необходимо учесть:

1. Преобразование строки должно быть реализовано в виде отдельной функции, дизайн которой соответствует стандартным функциям из заголовка <cstring>, а именно:
   (a) Функция не должна генерировать исключений
   (b) Функция должна принимать и возвращать объекты только встроенных типов
   (c) Функция не должна содержать в сигнатуре ссылок и значений типа **bool**
2. Ввод строки завершается вводом символа конца строки. Если в соответствии с заданным вариантом в преобразовании участвует две и более строк, все прочие строки задаются внутри программы с помощью строковых литералов
3. Если считываемую строку или результат преобразования нельзя разместить в динамической памяти, программа должна завершаться с кодом возврата 1 и сообщением об ошибке в стандартном потоке ошибок
4. При выволнеии преобразования следует использовать функции из заголовка <cctype>. Например, функция std::isdigit возвращает 1, если переданный символ является цифрой. обратите внимание на другие функции внутри этого заголовочного файла (см. документацию)
5. Результаты каждого преобразования (или значения характеристики) должны быть выведены в стандартный вывод на отдельной строке и программа должна завершаться с кодом возврата 0

Обратите внимание, например, на функцию strncmp из заголовка <cstring>, которая имеет вид:

```
1 int strncmp(const char * str1, const char * str2, size_t num);
```

Функция сравнивает символы указанных строк, начиная с первой пары: если символы равны, выбирается следующая пара (но не более num пар). Функция возвращает 0, если строки равны, и значение отличное от 0 в противном случае (см. документацию на функцию).

Студенту необходимо получить один (или более) вариантов у преподавателя. Далее приведён список вариантов и описание ожидаемого поведения программы, для заданных строк (для наглядности примеров вместо пробелов использованы «_»)

1. [SPC-RMV] Сформировать новую строку, удалив из исходной строки идущие подряд пробелы (оставить только один), также пробелы в начале и в конце строки

   ```
   _abc___def___ //Expects output (return 0): abc_def
   ```

2. [DEC-RMV] Сформировать новую строку, удалив из исходной строки все десятичные цифры

   ```
   ab_c_1d //Expects output (return 0): ab_c_d
   ```

3. [LAT-RMV] Сформировать новую строку, удалив их исходной строки все буквы латинского алфавита

   ```
   2ab_c_1d //Expects output (return 0): 2_1
   ```

4. [CMN-SYM] Сформировать новую строку из символов, общих для двух исходный строк

   ```
   bc_f //Expects output (return 0) with second string "abc_ef": bc_f
   ```

5. [UNC-SYM] Сформировать новую строку из символов двух исходных строк, которые не являются для них общими

   `bc_fu //Expects output (return 0) with second string "abc_ef": aeu`

6. [REP-SYM] Сформировать новую строку из символов, которы в исходной строке повторяются более одного раза (в новой строке они должны встречать по одному разу)

   `abfc_af_eef //Expects output (return 0): af_e`

7. [DIF-LAT] Определить, сколько различных букв латинского алфавита содержится в исходной строке

   `1abfc11_a2f_eef //Expects output (return 0): 5`

8. [HAS-REP] Определить, есть ли повторяющиеся символы в заданной строке. Функция должна возвращать число, отличное от 0, если есть и 0 — в противном случае

   `_abc_ //Expects output (return 0): 1`

9. [SEQ-SYM] Определить, есть ли в заданном строке подряд стоящие одинаковые символы. Функция должна вовзращать число, отличное 0, если есть и 0 — в противном случае

   `aab_b_ //Expects output (return 0): 1`

10. [RPL-SYM] Сформировать новую строку, заменив в исходной строке все вхождения одного заданного символа на другой заданный символ

    `abc_abc //Expects output (return 0) with replacing 'c' to 'b': abb_abb`

11. [UPP-LOW] Сформировать новую строку, заменив в исходной строке все прописные латиснкие буквы на строчные

    `AbC_aBc //Expects output (return 0): abc_abc`

12. [UNI-TWO] Сформировать новую строку, объеденив две строки таким образом, чтобы символы на соответствующих позициях находились рядом друг с другом (первый символ одной строки должен быть соседним для первого символа другой строки). Если строки разной длины, то оставшиеся символы должны быть в конце строки с результатом

    `abc_abc //Expects output (return 0) with second string "def_": adbecf__abc`

13. [HAS-SAM] Проверить, есть ли в двух заданных строках одинаковые символы. Функция должна возвращать число отличное от 0, если есть и 0 — в противном случае

    ```
    ABS_ABC //Expects output (return 0) with second string "abs": 0
    aBS_ABC //Expects output (return 0) with second string "abs": 1
    ```

14. [SHR-SYM] Сформировать новую строку, содержащую все латинские буквы, отсутствующие в исходной строке. Заглавные и строчные буквы не различать. Буквы новой строки должны следовать в порядке возрастания их кодов в ASCII-таблице

    `defzabc //Expects output (return 0): ghijklmnopqrstuvwxy`

15. [EXC-SND] Сформировать новую строку из двух исходных, исключив из первой все символы, встречающиеся во второй

    `AbC_aBc //Expects output (return 0) with second string "abc": AC_B`

16. [DGT-SND] Сформировать новую строку из двух исходных, добавив в первую все символы, встречающиеся во второй и являющиеся десятичными цифрами

    `abc_def //Expects output (return 0) with second string "g1h2k": abc_def12`

17. [REP-DGT] Определить, есть ли повторяющиеся цифры в заданной строке. Функция должна возвращать число отличное от 0, если есть и 0 — в противном случае

`abc3_def2_3abc //Expects output (return 0): 1`

18. `[RMV-VOW]` Сформировать новую строку, удалив из исходной строки все гласные буквы латинского алфавита

    `abc_def_abc //Expects output (return 0): bc_df_bc`

19. `[FRQ-TOP]` Сформировать новую строку из трёх чаще всего встречающихся символов исходной строки. Символы новой строки должны следовать в порядке возрастания их кодов в ASCII таблице

    `abc_def_ab_de_a_e //Expects output (return 0): abe`

20. `[LAT-TWO]` Сформировать новую строку, содержащую все латинские буквы, присутствующие в двух заданных строках. Заглавные и строчные буквы не различать. Буквы новой строки должны следовать в порядке возрастания их кодов в ASCII

    `abc_def //Expects output (return 0) with second string "def_ghk": abcdefghk`