Обработка текстов средсвами ASSEMBLER

Бордаченкова Е.А., Панфёров А.А.: модифицировано Сальниковым А.Н.

1. Постановка задачи

Написать программу, которая вводит два текста в кодировке UTF-8. Под текстом понимается непустая последовательность байт не длиннее 511 символов. Признак конца ввода каждого из текстов — последовательность символов _ @_ . Данная последовательнось не должна входить в сохраняемый в памяти текст. Тем не менее, необходимо пользователю программы предоставить возможность включить в текст последовательность символов _ @_ . В этом случае перед последовательностью должен идти символ '\'. Символ обратный слеш '\' является служебным символом. Как следствие, если необходимо в текст поместить символ обратный слеш, то это делается указанием 2-х обратных слеш подряд, примерно так. '\\text'. В текст внесётся один обратный слеш и будет получено следующее: '\text'.

Если хотя бы одна из введённых последовательностей не является текстом (пустая, либо длиннее 511 символов),то требуется напечатать сообщение об ошибке ввода и завершить программу.

Для текстов определяется какой из них длиннее другого. Длина определяется по максимальному количеству одинаковых UTF-8 символов в тексте. Например текст: 'Привет Привет' короче текста '33333', так как максимальная повторяемость символа в первом тексте – 2, а во втором 5.

Далее оба текста преобразуются по 2-м возможным правилам преобразования текстов. После преобразования оба текста необходимо выести на печать в стандатный поток вывода. Программа, после оценки длины текста, должна более длинный текст преобразовать по *правилу номер один*, а короткий текст – по *правилу номер два*.

При печати неабходимо распечатать длины исходных текстов, далее преобразованные тексты. Преобразованные тексты необходимо напечатать внутри трёх идущих подряд двойных кавычек, примерно таких: """. Три двойные кавычки должны быть расположены в начале строки, далее первод строки, на новой строке соответственно переобразованный текст. После текста также идут три подряд двойные кавычки. После конечных трёх двойных кавычек необходимо поставить перевод строки.

Если внутри текста встречаются три двойные кавычки подряд, то они при печати должны быть защищены символом '\'.

Пример вывода:

```
"""
Текст
  Текст
Текст Текст Текст
\""" это ещё не конец

_@_ да,да не конец _@_


Текст"""
```

Программа должна напечатать приветственное сообщение, должна печатать приглашение для ввода первого текста. Затем после ввода первого текста приглашение ко вводу второго текста. Далее, после определения длин текстов (по правилу определения длины описанному выше) напечатать сообщение с длинами текста. Затем напечатать по какому правилу преобразования текста: первому

или второму, будут изменены введённые тексты. Так же необходимо напечатать описание самих правил преобразования.

Правила преобразования текстов определяются вариантом задания.

При реализации в коде необходимо операции: определение длины, преобразование перед печатью, преобразование по правилу 1 и преобразование по правилу 2, реализовать как функции соблюдающие соглашение о передаче параметров stdcall.

гарантируется, что фактический размер символа не первосходит 4-х байт.

2. Варианты задания

2.1. Первое правило преобразования

1. Заменить каждую ненулевую цифру соответствующей ей строчной буквой русского алфавита ($1 \to$ а, $2 \to$ б и т.д.).
2. Заменить каждую строчную латинскую букву и русскую букву цифрой $N \mod 10$, где N – порядковый номер буквы в алфавите.
3. Заменить каждую прописную латинскую букву и русскую букву следующей за ней по алфавиту, букву Z менять на A, букву Я менять на А.
4. Заменить каждую строчную латинскую и русскую букву соответствующей прописной буквой, а прописную – строчной.
5. Заменить каждую латинскую букву и русскую букву – буквой, симметричной ей в алфавите ($A \to Z, b \to y, ...,$ А $\to$ Я, б $\to$ Ю).

2.2. Второе правило преобразования

1. Перевернуть текст, не используя дополнительную память большую, чем необходимо для хранения одного символа.
2. Удалить все знаки припинания из текста.
3. Удвоить каждую строчную латинскую букву текста.
4. Удвоить каждую цифру, входящую в текст.
5. Удвоить каждую прописную латинскую и прописную русскую буквы текста.
6. Удвоить каждую заглавную латинскую и заглавную русскую буквы текста.
7. Циклически сдвинуть текст на $K > 1$ (константа) позиций влево без использования дополнительной памяти большей чем на символов.
8. Циклически сдвинуть текст на $K > 1$ (константа) позиций вправо без использования дополнительной памяти большей чем на символов.
9. Удалить из текста все повторные вхождения его первого символа.
10. Переставить все цифры в начало текста, сохранив их взаимный порядок.

3. Требования к программе

- Значение 512 описать как константу. Для хранения текстов описать два массива на 512 байт, при этом последний байт в массиве всегда должен иметь значение 0.

  В случае, если вариант задания предполагает увеличение преобразованной строки необходимо описать массив такой длины, чтобы преобразованный текст в него гарантированно поместился.

  Следующий байт после преобразованного текста всегда должен иметь значение 0.

- Ввод текста описать в виде функции; функция возвращает значение *true* (1), если введённая последовательность является текстом, и *false* (0) в противном случае. Правила преобразования текста описать в виде процедур. Интерфейс процедур (соответствующий заголовок на Паскале) нужно написать в виде комментария перед кодом процедуры. Процедуры должны удовлетворять стандартным соглашениям о связях, параметры передавать в стеке.

- Нельзя преобразовывать текст во время печати, требуется сначала преобразовать текст в массиве, а затем печатать.

- Нельзя использовать дополнительную память (дополнительный массив или стек) для построения преобразованного текста. Если преобразование предполагает увеличение длины текста, нужно сразу описать массив с запасом.

Про кодирование текста в UTF-8 можно прочитать тут: `https://habr.com/ru/post/138173/` и тут `https://ru.wikipedia.org/wiki/UTF-8` и, посмотреть это видео: `https://www.youtube.com/watch?v=7v5ziFDlZ00`.