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 ASCII. Под текстом понимается непустая последовательность символов, в том числе пробельных и переводов строк, и в том числе с кодами 0 и 255. Последовательность символов не должна занимать в памяти более 512 байт.

Программа должна напечатать приветственное сообщение, затем вывести описение правил преобразования текстов и описание того, как будет определяться метрика длины текста, после этого напечать предложение ввести первый текст. Затем после успешного считывания первого текста приглашение ко вводу второго текста.

Признак конца ввода для каждого из вводимых текстов — последовательность символов *-:fin:-*. Данная последовательнось только метит конец текста и сама не должна входить в сохраняемый в памяти текст, кроме случаев, когда она экранирована симвлом обратный слеш '\'. Символ обратный слеш '\' является служебным символом, использующимся для ввода символов которые трудно напечатать, или которые не выводятся на экран. В рамках данного задания возможно 3 случая использования символа обратный слеш:

1. Указаните перед последовательностью символов конец ввода — приводит к включению в текст последовательности *-:fin:-*, при этом сам символ обратный слеш в память не сохраняется.

2. В случае если необходимо в текст поместить символ обратный слеш — это делается указанием 2-х обратных слеш подряд, примерно так. '\\text'. В текст внесётся один обратный слеш и будет получено следующее: '\text'.

3. В случае, когда необходимо указать символ по его коду: Для этого, после обратный слэш указывается 2 шестнадцатеричные цифры. например '\0a' означает перевод строки, а '\ff' символ с кодом 255.

Во всех остальных случаях во входной последовательности байт обратный слеш просто игнорируется и не сохраняется в памяти.

Если хотя бы одна из введённых последовательностей не является текстом (пустая, либо попытались ввести текст, так, что в памяти пришлось бы хранить более 512 символов), то требуется напечатать сообщение об ошибке ввода и завершить программу.

После успешного сохранения в памяти программы обоих текстов необходимо выяснить какой из них оказался длинее другого в некоторой специально введённой метрике длины. Конкретная метрика длины текстов является вариантом задания.

Далее оба текста преобразуются по 2-м возможным правилам преобразования текстов. После преобразования оба текста необходимо выести на печать в стандатный поток вывода в порядке определяемом значением метрики длины: короткий текст с точки зрения метрики выводится первым.

Правила преобразования выбираются следующим образом: короткий текст преобразуется по первому правилу, длинный по второму. сами правила преобразования текстов определяются вариантом задания.

При печати необходимо распечатать: правила по которым преобразовывались тексты с указанием номеров правил в задании, номер функции определения длины текста, затем длины исходных текстов, затем длины текстов после преобразования, далее сами преобразованные тексты. Каждый из преобразованных текстов необходимо напечатать внутри трёх идущих подряд двойных кавычек, примерно таких: """. Три двойные кавычки должны быть расположены в начале строки, далее

первод строки. После этого с новой строки соответственно следует переобразованный текст. Непосредственно после текста также как и до текста идут три подряд двойные кавычки. После них необходимо вставить перевод строки.

Три двойные кавычки подряд могут встретиться внутри текста. В этом случае, при выводе они должны быть защищены (экранированы) символом обратный слеш '\'.

Пример вывода:

```
"""
Text
  Text
text text text
\""" not end first text

-:fin:-- not end, not end  -:fin:-


now end"""
```

2. Требования к коду

- Функции преобразования текстов, считывание текста, печать текста, определение длины текста должны быть реализованы как ассемблерные функции, соблюдающие соглашение о передаче параметров stdcall.
- Значение 512 описать как константу.
- Для хранения текстов описать два массива не менее 512 байт, за концом текста всегда должен следовать байт со значением 0.

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

  Следующий байт в памяти 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. Запрещается выходить из программы внутри функций.
- Нельзя преобразовывать текст во время печати, требуется сначала преобразовать текст в массиве, а затем печатать.

3. Варианты задания

3.1. Первое правило преобразования

1. Заменить каждую ненулевую цифру соответствующей ей маленькой буквой латинского алфавита (1 → a, 2 → b и т.д.).
2. Заменить каждую строчную латинскую букву $N \mod 10$, где N – порядковый номер буквы в алфавите.
3. Заменить каждую Заглавную латинскую букву за ней по алфавиту: букву Z менять на A.
4. Заменить каждую заглавную латинскую букву соответствующей маленькой латинской буквой, а маленькую – заглавной.
5. Заменить каждую латинскую букву – буквой, симметричной ей в алфавите ($A \to Z, b \to y, ...,$).
6. Заменить на '*' все элементы, идущие за первым вхождением 'a' в текст.

7. Заменить на ‘#’ все элементы, идущие за последним вхождением ‘b’ в текст.
8. Заменить на ‘$’ все элементы, идущие перед первым вхождением ‘c’ в текст.
9. Заменить на ‘%’ все элементы, идущие перед последним вхождением ‘d’ в текст.

3.2. Второе правило преобразования

1. Написать текст задом наперёд, 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. Найти в тексте полиндромы слов длины k, состоящих из латинских символов, и заменить их на символы звёздочка. Здесь k – номер студента в списке группы плюс один. (Считаем, что студенты нумеруются с единицы).
5. Циклически сдвинуть текст на $K > 1$ (константа) позиций влево без использования дополнительной памяти большей чем на символов. Здесь k – номер студента в списке группы. (Считаем, что студенты нумеруются с единицы).
6. Циклически сдвинуть текст на $K > 1$ (константа) позиций вправо без использования дополнительной памяти большей чем на символов. Здесь k – номер студента в списке группы. (Считаем, что студенты нумеруются с единицы).
7. Удалить из текста все повторные вхождения его первого символа.
8. Переставить все цифры в начало текста, в обратном порядке относительно встречи в тексте.

3.3. Метрики длины

1. Количество заглавных латинских символов в тексте
2. Количество чисел в десятичной системе исчесления встречающихся в тексте
3. Количество пробельных символов в тексте (включая табуляцию, возврат каретки, перенос строки).
4. Количество признаков конца ввода, встречающихся в тексте.

4. Задание со звёздочкой

Специально для тех, кому кажется всё не интересно. Предлагается все действия проделать сняв ограничение ASCII на текст. Предполагаем, что нам на вход попадает текст в формате UTF-8, и все варианты функций предполагают работу не только с латинскими символами, но с символами произвольного языка поддерживаемого UTF-8. Например с русскими символами и с греческими.

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`.