БЛОК-4 (часть 1)
«Процедуры нерекурсивные»

Задача 1. Частоты вхождений букв (обязательная)

Описать в программе следующие данные:

```
.data
N equ 100
S db N dup (?)
```

Дан непустой текст (последовательность из не более **N** символов, среди которых возможны малые латинские буквы), признак конца - точка. Требуется ввести с клавиатуры данный текст и записать его в начало рабочего массива **S (этот этап – обязателен!)**. Далее для введённого текста ответить на вопрос: **какая из малых латинских букв, входящих в состав текста, чаще всего встречается в нём (если таких букв несколько – напечатать любую из них).**

Требования.

Описать функцию **ReadText(S)**, которая вводит текст (до точки) и размещает его в начале рабочего массива **S** (параметр функции – адрес рабочего массива). Функция возвращает (через **EAX**) длину введённого текста (без точки).

Для ответа на основной вопрос описать функцию **MaxLet(S,N)**, где **S** – адрес символьного массива, **N** – его размерность. Функция определяет, какая из малых латинских букв чаще всего встречается в этом массиве (если таких букв несколько, то выдать любую). Найденную букву возвратить через регистр AL.

Обязательное требование: в своей работе функция **MaxLet(S,N)** должна использовать вспомогательный (локальный) массив, отведя ему место в стеке (это массив из 26 байтов, в каждом байте которого хранится информация по числу вхождений соответствующей буквы латинского алфавита). Так как количество байтов, выделенное под область локальных переменных, должно быть кратно четырём, то размер этой области должен быть расширен до 28 байтов, адрес начала области [EBP-28].
Потребуются *три этапа*: 1) инициализация нулями элементов локального массива (ведь содержимое стека выше текущего положения ESP – не определено, там может находиться что угодно); 2) просмотр массива символов (**S**) и корректировка элементов вспомогательного (локального) массива; 3) просмотр вспомогательного (локального) массива с целью поиска индекса максимального элемента. Искомая буква – это найденный индекс плюс код буквы **'a'**. Считать, что в используемой кодировке малые латинские буквы следуют строго друг за другом (в соответствии с алфавитом и без пропусков).

Обе функции (**ReadText** и **MaxLet**) должны выполнять стандартные соглашения о связях stdcall.

Задача 2. Матрицы (прямоугольные и квадратные)
(обязательно надо решить только пункты 1-5, пункт 6 – за дополнительные очки)

Описать в программе следующие данные:

```
.data
MaxSize equ 150       ; предельный размер рабочего массива (для хранения матрицы).
n db ?                ; количество строк матрицы (б/зн)
m db ?                ; количество столбцов матрицы (б/зн)
X dw MaxSize dup(?) ; рабочий массив для хранения матрицы размером n x m (где n*m ≤ MaxSize)
                      ; элементы матрицы рассматриваются как числа со знаком
```

Ввод элементов матрицы в память:

Программа запрашивает у пользователя параметры матрицы, с которой он желает работать: **n** - количество её строк, и **m** - количество её столбцов.
Ввод осуществляется в переменные **n** и **m** по макрокомандам **inint n** и **inint m**. Если для введённых пользователем значений **n** и **m** не выполняется требование **n*m ≤ MaxSize** (или какая-либо размерность равна нулю или единице), то программа выдаёт на экран соответствующее сообщение и далее спрашивает пользователя, желает ли он ввести другие параметры (**Y/N**). Если *нет* – то программа на этом завершает свою работу. Если *да* – программа ожидает ввода новых значений для **n** и **m** (т.е. новый шаг цикла). В случае задания корректных значений для **n** и **m** начинается *ввод элементов матрицы* (при тестировании программы рекомендуется вводить элементы по строкам, разделяя их друг от друга пробелами, нажимая ENTER после ввода каждой строки – это нормально, т.к. по правилам макрокоманды **inint op** вводимые числа отделяются друг от друга либо "пробелами", либо "концами строк"). В случае ввода лишних элементов – они должны отбрасываются (обеспечить это с помощью макрокоманды **flush**). Вводимые с клавиатуры элементы сохраняются в памяти в рабочем массиве **X**. *Эту часть задачи сделать без использования каких-либо процедур.*

После ввода элементов матрицы в память:

Программа *распечатывает* введённую матрицу *по строкам* (для реализации этой подзадачи предусмотреть соответствующую процедуру, параметры в которую следует передавать через стек – в соответствии со стандартными соглашениями о связях stdcall). *Требование*: в теле процедуры нельзя обращаться по имени к объектам, описанным вне процедуры (т.е. к именам **X**, **n** и **m**) ⇒ в качестве *параметров* в процедуру следует передавать адрес обрабатываемого массива и две его размерности.

Далее программа *анализирует* введённую матрицу и выдаёт на экран следующую информацию:

1) номера (через пробелы) тех строк, элементы которых упорядочены *по неубыванию* (≤);
2) номера (через пробелы) всех *симметричных* строк матрицы;
3) номера (через пробелы) тех столбцов, все элементы в которых *одинаковые*;

Замечание (важное !!!).
Считать, что пользователь нумерует строки и столбцы, начиная с единицы, – это *логическая* (т.е. с точки зрения пользователя) нумерация строк и столбцов. Согласно такой нумерации и следует выдавать ответ на экран. Совет: основная работа должна происходить с *внутренней* нумерацией (от нуля); переход же к *логической* нумерации – на последнем этапе перед выдачей ответа.

Если матрица оказалась **квадратной** (т.е. **n=m**), то далее программа распечатывает в отдельных строках (через пробелы) следующую информацию:

4) все элементы, находящиеся на главной диагонали (начиная с элемента верхней строки);
5) все элементы, находящиеся на побочной диагонали (начиная с элемента верхней строки);
6) (**дополнительный пункт на 15 очков**) проверить, является ли данная матрица симметричной

(Для реализации каждого из пунктов 1)-6) следует предусмотреть свою процедуру (*требования* ко всем процедурам – выполнять стандартные соглашения о связях stdcall). При выводе ответа указывать номер пункта (чтобы было понятно при проверке, к чему относится та или иная выводимая информация).

После вывода требуемой (*см. пункты* 1)-6) *выше*) информации программа запрашивает у пользователя, желает ли он работать с новой матрицей (**Y/N**)? Если нет, то на этом программа завершает свою работу. Если да, то экран очищается, и описанный выше процесс повторяется для следующей матрицы (запрашиваются новые значения для **n** и **m** и т.п.).

Внимание! При сдаче программы выдавайте на экран информацию о том, какие пункты реализованы в вашей программе (для облегчения проверки). И в письме дополнительно сообщайте эту же информацию.

Задача 3. Переворот порядка слов (дополнительная на 30 очков)

Описать в программе следующие данные*:*

```
.data
N equ 100
S db N dup (?)
```

Дан непустой текст из не более чем **N** символов (малых латинских букв и пробелов), признак конца – точка (к тексту не относится). Текст структурно состоит из непустого количества слов, которые отделяются друг от друга пробелами (слева от первого слова и справа от последнего слова – возможны тоже пробелы). Требуется ввести с клавиатуры такой текст и сохранить его в начале рабочего массива **S (1-ый этап)**. Далее введённый текст следует преобразовать, переставив в нём порядок слов **(2-ый этап)**. Вывести на экран преобразованный текст **(3-ый этап)**.
Например, при вводе текста **’_ _ _ abc _ _ defg _ f _ _ _ qwerty _ _ .’** должно быть напечатано: **_ _ qwerty _ _ _ f _ defg _ _ abc _ _ _**

Требование.

При решении задачи **не использовать дополнительную память**.
Следует описать процедуру **Back(left,right)**, которая переворачивает заданный текст, начинающийся с адреса **left** и кончающийся адресом **right** (параметры **left** и **right** – полные адреса, **left⩽right**). Способ передачи параметров в процедуру (через регистры или стек) можно выбрать по своему усмотрению.

Используя эту процедуру, сначала перевернуть исходный текст. А затем перевернуть отдельно каждое его слово.

Например, для указанного выше текста получим после переворота всего этого текста:
_ _ ytrewq _ _ _ f _ gfed _ _ cba _ _ _

После переворота каждого слова в этом промежуточном тексте получим текст требуемого вида:
_ _ qwerty _ _ _ f _ defg _ _ abc _ _ _

При сдаче задания прислать *скриншоты* для нескольких *тестов*, на которых проверялась работа вашей программы.