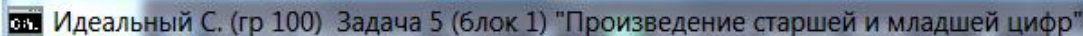


**Разобраться** предварительно с **вводом-выводом** по пособию В.Г.Баулы <http://arch32.cs.msu.su> (глава 6, раздел 6.5.1), и со **структурой программы на языке ассемблера** (глава 6, раздел 6.5.2). Только после этого приступить к написанию программ.

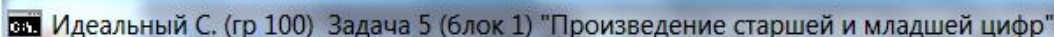
Обязательно усвоить следующие макрокоманды (из раздела 6.5.1): **ClrScr**, **ConsoleTitle**, **outstr[ln]**, **inchar**, **outchar**, **inint[ln]**, **outint[ln]**, **outword[ln]**, **newline**, **flush**, **pause**, **exit** (обратите при этом внимание, что **названия этих макрокоманд – имена пользователя**, следовательно, **малые и большие буквы в этих названиях - различаются**). Остальные макрокоманды осваивать не обязательно (но если есть желание и интерес, то их можно использовать, но не увлекаться...).

В программах в **заголовке консольного окна** выдавать **обязательно информацию**:

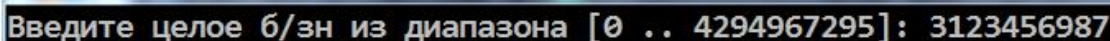
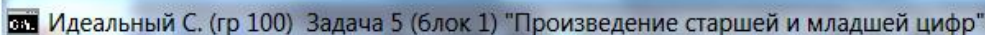
- фамилию и имя студента, номер учебной группы;
- номер решаемой задачи (из какого она блока) и название задачи, если оно имеется, *например*:



- приглашение к вводу данных (чего ожидаете от пользователя и в каком виде), *например*:



- объяснение (лаконично), что выдаётся вашей программой в качестве результатов её работы, *например*:



Следует комментировать исходный текст программы (лаконично, только наиболее важные этапы и тонкие места). Перед макрокомандой **exit** ставьте **обязательно (!!!)** макрокоманду **pause** (для удержания экрана с результатами работы программы – это нужно ТОЛЬКО при отправке программ на проверку по электронной почте), *например*:

```
include console.inc
.data
    K dd ?
.code
start:
    clrscr
    ConsoleTitle 'Идеальный С. (гр 100) Задача 5 (блок 1) "Произведение старшей и младшей цифр"'
    inint K, 'Введите целое б/зн из диапазона [0 .. 4294967295]: '

    mov EAX, K    ; можно было сразу inint EAX без использования K
    mov EDX, 0    ; (EDX,EAX) – делимое для сверхдлинного б/зн деления
    mov EBX, 10   ; делитель
    div EBX       ; (EDX,EAX)/EBX => EAX:=div, EDX:=mod
    ; ...
    ; другие команды программы
    ; ...
    pause
    exit
end start
```

Присылайте на ПРОВЕРКУ БЛОКОВ ([110.111.112@mail.ru](mailto:110.111.112@mail.ru)) свои программы (даже если она одна!) в **zip**-архиве, названном по фамилии автора, номеру учебной группы и номеру отправляемого блока задач. *Например*, **Ivanov-112-4.zip** или **Ivanov-112-4-add.zip** для обязательных или дополнительных (**add**) задач блока-4 студента Иванова из 112 группы. По каждой задаче присылайте **три файла** - с расширениями **asm**, **lst**, **exe**. Если в архиве несколько задач, то каждую задачу помещайте в **отдельную папку** с номером этой задачи. **В названиях** архива, папок и программ **русские буквы и пробелы - не использовать**. Проследите, чтобы в отправляемом **exe**-файле было предусмотрено **удержание экрана с результатами работы программы** (с помощью макрокоманды **pause**). Не забывайте также о нашей **договорённости**: если **обязательная задача** решена не самостоятельно, то необходимо об этом **заранее предупредить при отправке решения** (с кем консультировались, по какому вопросу; с кем совместно работали над задачей, какова доля вашего участия в решении), никаких санкций за эту информацию от меня не последует. **Дополнительные задачи** – только самостоятельно, совместные решения или консультации с коллегами - не приемлемы.

## Задачи (блок 1). Арифметика, переходы, циклы.

Обязательно нужно решить **три любых задачи** из десяти предлагаемых.

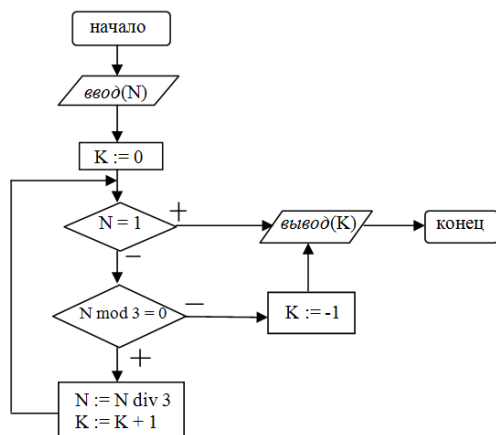
За каждую дополнительную задачу начисляется по 10 очков.

Решения обязательных задач нужно прислать до 21-го марта, дополнительных – до конца семестра

### Задача 1. “Степень тройки”

Ввести число  $\geq 1$  (считать, что ввод будет корректный и число укладывается в формат двойного слова). Определить, является ли это число **степенью тройки** (1, 3, 9, 27, ...). Если да, то вывести на экран показатель степени, иначе вывести число **-1**.

Решать согласно следующей **блок-схеме** (условные обозначения: **N** – исходное число, **K** – искомый показатель):



**РЕКОМЕНДАЦИИ:** Решать строго *методом деления на 3*: **делимое** представлять в формате учетверенного слова, **делитель** - в формате двойного слова, т.к. нет гарантии, что *результат деления на 3* укладывается в байт или в слово (например,  $900/3=300>255$ ,  $300000/3=100000>65535$ ). То есть требуется **сверхдлинное деление**. В решении не отклоняться от предлагаемой блок-схемы. Считывать (по **inint**) исходное число (N) выгодно сразу на регистр **EAX**; ответ (K) выгодно тоже формировать сразу на регистре (для повышения быстродействия программы). Добейтесь, чтобы в теле цикла не было обращений к ячейкам оперативной памяти (в этой задаче все данные лучше размещать на регистрах).

### Задача 2. “Простое число”

N dd ? ; N>1

Ввести число **N** (>1), считая, что ввод будет корректный и число укладывается в формат двойного слова. Проверить, является ли это число простым.

Ответ напечатать в виде одного из слов: ПРОСТОЕ или СОСТАВНОЕ.

**РЕКОМЕНДАЦИИ:** В цикле делить исходное **N** на числа от **2** до **N div 2** (с целью поиска первого делителя), т.е. максимальное число шагов цикла **N div 2 - 1**. Как только найдется первый делитель – досрочно выходить из цикла (число составное). Вышли по концу цикла – число простое.

Выбрать правильный вид деления (короткое, длинное или сверхдлинное).

Для облегчения вывода ответа сделать начальное предположение, что **N составное** (т.е. до начала проверки числа загрузить, например, в регистр **ESI** (или другой) адрес начала строки со словом “СОСТАВНОЕ”). Но если в конце выяснится, что делителей у числа не нашлось, то перед выводом ответа (по макрокоманде **outstrln ESI**) скорректировать значение в **ESI** (поместив в него адрес начала строки со словом “ПРОСТОЕ”). *Требование:* для вывода ответа иметь в программе только одну макрокоманду **outstrln ESI** (чтобы код программы не разрастался).

**Как поместить адрес начала строки S в регистр? Можно по-разному:**

1 способ (хороший)	2 способ (хороший)	3 способ (допустимый, но не рекомендуется)
<pre>.data S db 'СОСТАВНОЕ', 0, 'ПРОСТОЕ', 0 .code mov ESI, offset S ; offset S = смещение ячейки с адресом S ; от начала секции данных, где ; offset – одноместный оператор ; далее - решение, и если надо, корректировка содержимого ESI</pre>	<pre>.data S db 'СОСТАВНОЕ', 0, 'ПРОСТОЕ', 0 .code lea ESI, S; скоро изучим это! ; загрузка в регистр ESI ; исполнительного адреса ; второго операнда ; далее - решение, и если надо, корректировка содержимого ESI</pre>	<pre>.data S db 'СОСТАВНОЕ', 0, 'ПРОСТОЕ', 0 adr_S dd S ; это двойное слово ; хранит адрес начала строки S .code mov ESI, adr_S ; далее - решение, и если надо, корректировка содержимого ESI</pre>

### Задача 3. “Баланс скобок”

Ввести последовательность символов (отличных от точки), конец ввода - точка. Ввод осуществить за один сеанс, т.е. набрать всю последовательность символов (с точкой на конце) и затем нажать ENTER – как мы это делали в 1-ом семестре. Определить, сбалансирована ли эта последовательность по круглым скобкам. Ответ в виде одного из слов: ДА или НЕТ. (Требование: **текст не сохранять в памяти.**)

Пример сбалансированной последовательности:

```
a ( b ( ) c d ( e f ( s d f ) k ) s ) s ( s ) d .
```

Примеры несбалансированных последовательностей:

```
( a ) ( a s d .  
( a s ) ) s ( s d ) .  
) ( .
```

### РЕКОМЕНДАЦИИ:

Завести счётчик баланса скобок (увеличивается при появлении открывающей скобки, уменьшается при появлении закрывающей скобки). Обратить особое внимание на 2 случая: 1) появление непарной закрывающей скобки, т.е. счётчик стал отрицательным (далее нет смысла продолжать анализ текста, т.е. надо завершать цикл и выдавать ответ); 2) наличие непарных открывающих скобок (отлавливается в конце, по окончании ввода всего текста). Удачный исход: если дошли до точки и при этом счётчик баланса - нулевой.

Для облегчения вывода ответа сделать начальное предположение, что последовательность не сбалансирована (т.е. до начала чтения текста настроиться с помощью регистра **EDX**, например, на начало строки со словом “НЕТ”). Но если в конце выяснится, что имеет место баланс скобок, то перед выводом ответа (по макрокоманде **outstr EDX**) скорректировать значение в **EDX**. Как настраиваться на начало строки с выводимым текстом – написано в рекомендациях к **задаче 2** (см. *три способа*).

### Задача 4. “Дробь P/Q”

Описать в программе следующие переменные:

```
P dd ? ; P ≥ 0  
Q dd ? ; Q > 0
```

Ввести с клавиатуры значения для переменных P и Q.

Напечатать дробь P/Q в виде вещественного числа с 5 цифрами в дробной части.

### РЕКОМЕНДАЦИИ:

1) Сначала вывести *целую часть* от деления P на Q.  
2) Затем вывести *точку* (макрокоманда **outchar ‘.’**).  
3) Наконец, запустить цикл (**loop**) из пяти шагов. На каждом шаге будет выводиться очередная цифра дробной части: для её нахождения надо последний найденный остаток домножить на 10, а затем полученный результат разделить на Q. *Внимание:* команда **loop** реализует только короткий переход (то есть тело цикла не может содержать более 30-40 команд). Может так случиться, что придётся отказаться в этой задаче от использования **loop** (заменив её другими командами). Но сначала попробуйте воспользоваться **loop** (надеюсь, повезёт).

### Задача 5. “Произведение старшей и младшей десятичных цифр”

Ввести число без знака из диапазона  $[0..2^{32}-1]$  (по **inint**) и напечатать произведение **старшей** (значащей) и **младшей** цифр в десятичной записи этого числа (значащей называется цифра, удаление которой меняет величину числа).

Например, для числа **234567** напечатать **2\*7=14**, для числа **6** напечатать **6\*6=36** и т.п.

**РЕКОМЕНДАЦИИ:** Решать путём последовательного деления на **10**. Разобраться, какое деление (длинное, короткое или сверхдлинное) здесь необходимо. Не забыть про случай, когда в числе одна цифра (она является одновременно старшей и младшей).

### Задача 6. “Пятеричное число”

Ввести (по **inchar**) непустую последовательность цифр, оканчивающуюся пробелом, которая является правильной записью неотрицательного числа в пятеричной системе счисления. Напечатать это число (за одно обращение к команде **outword**) в десятичной системе. Считать, что искомое число уместится в формат двойного слова. *Например*, при вводе последовательности цифр **4321432** должно быть напечатано **73367**, а при вводе последовательности цифр **1000000000** должно быть напечатано **1953125**

**РЕКОМЕНДАЦИИ:** Ввод пятеричного числа осуществлять с использованием схемы Горнера. Разобраться, какой вид умножения (короткое, длинное или сверхдлинное) надо использовать.

### Задача 7. “Ближайшее число, кратное семи”

Ввести (по **inint**) число без знака из диапазона  $[0..2^{32}-1]$  и напечатать ближайшее к нему число, кратное семи. *Например*, для числа **4294967295** получится ответ **4294967292**

**РЕКОМЕНДАЦИИ:** Разобраться, какое деление (длинное, короткое или сверхдлинное) здесь необходимо. Проверить остаток от деления исходного числа на **7**. Если остаток равен **1, 2, 3**, то искомое число лежит (на числовой оси) *левее* заданного. Если остаток равен **4, 5, 6**, то искомое число лежит (на числовой оси) *правее* заданного. Если остаток **нулевой** – искомое число совпадает с заданным.

### Задача 8. “Алгебраическая сумма”

Дан текст следующего вида:

$$d_1 \pm d_2 \pm \dots \pm d_k .$$

где  $d_i$  - цифра от 0 до 9,  $k \geq 1$ , в конце текста - точка. Найти значение этой алгебраической суммы.

Вводить символы текста до появления точки. Считать, что в общем случае результат суммирования может не уместиться в формат байта. В этой задаче для ввода использовать только макрокоманду **inchar** (т.е. все числа и знаки вводятся **посимвольно**). Значение найденной суммы выводить по макрокоманде **outint**.

### Задача 9. “Первая и последняя буквы”

Дана непустая последовательность непустых слов из малых латинских букв; между соседними словами - запятая, за последним словом - точка. Подсчитать количество слов, которые **начинаются и оканчиваются одной и той же буквой**.

**РЕКОМЕНДАЦИИ:** Реализовать двойной цикл. *Внешний* цикл – по словам; *внутренний* – по символам текущего слова (до точки или запятой). Выйдя из внутреннего цикла – сравнить крайние буквы прочитанного слова. Текст набирать на клавиатуре следует сразу целиком (по аналогии с решением аналогичных задач на Паскале), ставить точку и нажимать **Enter** (предполагается, что текст не будет очень длинным и поэтому целиком “влезет” в буфер ввода). Символы текста вводить с помощью **inchar**. *Например*, при вводе текста **asd,w,dfgd.** должно быть напечатано **2**

### Задача 10. “Таблица умножения”

Напечатать таблицу умножения в десятичной системе счисления.

**Требование:** **умножение в решении не использовать!**

Постараться красиво оформить вывод, например, так (см. следующую страницу):

	!	0	1	2	3	4	5	6	7	8	9
0	!	0	0	0	0	0	0	0	0	0	0
1	!	0	1	2	3	4	5	6	7	8	9
2	!	0	2	4	6	8	10	12	14	16	18
3	!	0	3	6	9	12	15	18	21	24	27
4	!	0	4	8	12	16	20	24	28	32	36
5	!	0	5	10	15	20	25	30	35	40	45
6	!	0	6	12	18	24	30	36	42	48	54
7	!	0	7	14	21	28	35	42	49	56	63
8	!	0	8	16	24	32	40	48	56	64	72
9	!	0	9	18	27	36	45	54	63	72	81

или так:

Таблица умножения											
ХХ	0	1	2	3	4	5	6	7	8	9	
0	0	0	0	0	0	0	0	0	0	0	!
1	0	1	2	3	4	5	6	7	8	9	!
2	0	2	4	6	8	10	12	14	16	18	!
3	0	3	6	9	12	15	18	21	24	27	!
4	0	4	8	12	16	20	24	28	32	36	!
5	0	5	10	15	20	25	30	35	40	45	!
6	0	6	12	18	24	30	36	42	48	54	!
7	0	7	14	21	28	35	42	49	56	63	!
8	0	8	16	24	32	40	48	56	64	72	!
9	0	9	18	27	36	45	54	63	72	81	!

Или еще красивее...