Выход на машины №4 *(заготовка* ***asm-файла*** *– прилагается отдельно)*

Условия задач никому не пересылать и нигде не выкладывать!

Требование: все используемые в программе процедуры и функции должны **соблюдать стандартные соглашения о связях** **stdcall**.
Пожелание: везде, где это уместно, **при работе с записями** рекомендуется использовать операторы **mask**, **width** (целесообразно также использовать **имена полей** при задании величины сдвигов).

Задача (единственная) выхода-4 *состоит из* ***четырёх обязательных этапов*** *(+* ***пятый и шестой этапы – дополнительные****), начало выполнения – обязательно* ***1-ый этап, далее этапы сдаются строго друг за другом (по отдельности)***.

Этап 1 (выполняется в первую очередь) (40 очков)

ОПИСАНИЕ ТИПА ЗАПИСИ:

```
Date_pack record D:5, M:4, Y:7  ;  упакованная дата в формате «день-месяц-год»
; (хранит значение даты в рамках некоторого столетия)
; в поле D хранится  номер дня (в месяце):  число от 1 до 31
; в поле M хранится  номер месяца:  число от 1 до 12
; в поле Y хранятся  две последние цифры года: число от 0 до 99
; считать, что  ДАТА ЗАДАНА КОРРЕКТНО (т.е. значения для D, M и Y - в рамках нужных  диапазонов)
```

ОПИСАНИЕ ДВУХ ПЕРЕМЕННЫХ-ЗАПИСЕЙ:

```
D1  Date_pack <>
D2  Date_pack <>
```

Ввести с клавиатуры значения для переменных-записей **D1** и **D2** типа **Date_pack** (*считать, что данные для ввода будут заданы корректно*). *Проверить*, предшествует ли дата **D1** дате **D2** (в рамках некоторого столетия), т.е. проверить условие **D1<D2** ?
Вывести ответ в виде: **«D1<D2 is true/false»**
(здесь вместо **D1** и **D2** надо напечатать значения введённых дат в формате **dd.mm.yy**).

Требования к решению на 1-ом этапе.

1) Описать *процедуру* **In_Rec(D)**, которая вводит (с клавиатуры) тройку чисел - значения полей *записи* типа **Date_pack** и присваивает введённую *запись* параметру **D** (*внимание*: очевидно, что передача параметра **D** должна осуществляться по ссылке !). (10 очков)
Замечание: Тройку чисел вводить путём последовательного выполнения трёх макрокоманд **inint**. Учесть при этом, что макрокоманда **inint** считает “концом” вводимого числа ***первую нецифру***, которая “проглатывается”; поэтому при вводе тройку чисел можно задавать либо в виде **25/4/23**, либо в виде **25.4.23** (либо отделять числа традиционно *пробелами* и т.п.).

2) Описать также *функцию* **Less(D1,D2)**, которая проверяет условие **D1<D2** (где **D1** и **D2** – записи типа **Date_pack**, переданные в функцию по значению) и возвращает (через **AL**) ответ **1** (*да*) или **0** (*нет*). Внимание! Для вывода символа “**<**” используйте **outchar 60** , где **60** – код символа “**<**” (иначе можете “запутать” макропроцессор, который трактует символ “**<**” как управляющий в записи любой макрокоманды – узнаете про это вскоре в теме «Макросредства»). (20 очков)

3) Описать *процедуру* **Out_Rec(D)** для вывода на экран значения даты-записи **D** в формате **dd.mm.yy** (параметр **D** передать в процедуру по значению). (10 очков)

НЕ ЗАНИМАЙТЕСЬ “САМОДЕЯТЕЛЬНОСТЬЮ”: по значению или по ссылке (как просят, так и делайте!)

СОВЕТ. Начинайте выполнять ***Этап 1*** с написания *процедуры* **In_Rec(D)**, примените её для ввода одной записи. Затем напишите процедуру **Out_Rec(D)** для вывода на экран ранее введённой записи. И только после отладки этих двух процедур приступайте к написанию *функции* **Less(D1,D2)**. Можно сдавать ***Этап 1*** по частям (ввод-вывод, сравнение), если сложно сдать всё сразу.

ЗАМЕЧАНИЕ ПРО ПЕРЕДАЧУ ЗАПИСИ ПО ЗНАЧЕНИЮ.
ПРИ ПЕРЕДАЧЕ ПО ЗНАЧЕНИЮ ОБРАТИТЕ ВНИМАНИЕ, ЧТО TYPE Date_pack = WORD => ВЕЛИЧИНУ, ХРАНЯЩУЮСЯ В ЗАПИСИ, ПЕРЕД ПОМЕЩЕНИЕМ (в качестве параметра-значения) В СТЕК СЛЕДУЕТ ПРЕДВАРИТЕЛЬНО РАСШИРИТЬ ДО 32-БИТОВОГО ФОРМАТА (СОГЛАСНО СТАНДАРТНЫМ СОГЛАШЕНИЯМ О СВЯЗЯХ). ЕСЛИ НАРУШИТЬ ЭТО ТРЕБОВАНИЕ, ТО ДАЛЕЕ ПРОГРАММА МОЖЕТ ВЕСТИ СЕБЯ НЕПРЕДСКАЗУЕМО (ТАК КАК ГРАНИЦЫ СТЕКА ПЕРЕСТАНУТЬ БЫТЬ ВЫРОВНЕНЫ ДО 4-Х БАЙТОВ).

ЗАМЕЧАНИЕ ПРО СРАВНЕНИЕ ДВУХ ДАТ. ДАТЫ МОЖНО СРАВНИВАТЬ ДВУМЯ СПОСОБАМИ.
СПОСОБ_1: СНАЧАЛА СРАВНИВАЕМ ЗНАЧЕНИЯ ПОЛЕЙ Y (ПРЕДВАРИТЕЛЬНО ПОГАСИВ ОСТАЛЬНЫЕ ПОЛЯ), ЕСЛИ ГОДЫ РАВНЫ, ТО СРАВНИВАЕМ ЗНАЧЕНИЯ ПОЛЕЙ M (ПРЕДВАРИТЕЛЬНО ПОГАСИВ ОСТАЛЬНЫЕ ПОЛЯ). ЕСЛИ ГОДЫ РАВНЫ И МЕСЯЦЫ РАВНЫ, ТО СРАВНИВАЕМ ЗНАЧЕНИЯ ПОЛЕЙ D (ПРЕДВАРИТЕЛЬНО ПОГАСИВ ОСТАЛЬНЫЕ ПОЛЯ). ПРИ ЭТОМ *СДВИГАТЬ СРАВНИВАЕМЫЕ ПОЛЯ НИКУДА НЕ НАДО* (ДОСТАТОЧНО ПОГАСИТЬ БИТЫ В ОСТАЛЬНЫХ ПОЛЯХ). ДЛЯ ТАКОГО ТРЁХ-ЭТАПНОГО СРАВНЕНИЯ НАДО ПЕРЕД КАЖДЫМ ЭТАПОМ ПОМЕЩАТЬ НА РЕГИСТРЫ КОПИИ СРАВНИВАЕМЫХ ЗАПИСЕЙ (НЕТ НИЧЕГО СТРАШНОГО, ЕСЛИ БУДЕТЕ ОБРАЩАТЬСЯ ПОВТОРНО ЗА ПАРАМЕТРАМИ В СТЕК, Т.К. В СОВРЕМЕННЫХ ПРОЦЕССОРАХ ОБЛАСТЬ, БЛИЗКАЯ К ВЕРШИНЕ СТЕКА, КЭШИРУЕТСЯ В БЫСТРОДЕЙСТВУЮЩЕЙ ПАМЯТИ, МОЖНО СЧИТАТЬ, ЧТО НАХОДИТСЯ НА РЕГИСТРАХ).
СПОСОБ_2: СРАВНЕНИЕ ЗАПИСЕЙ МОЖНО УСКОРИТЬ, ПЕРЕСТАВИВ МЕСТАМИ ПОЛЯ (В ПОРЯДКЕ Y,M,D), И ДАЛЕЕ СРАВНИТЬ ДВЕ ЗАПИСИ КАК ЕДИНЫЕ 16-БИТНЫЕ ОБЪЕКТЫ (ЦЕЛЫЕ Б/ЗН). НО ПЕРЕСТАВЛЯТЬ МЕСТАМИ ПОЛЯ МОЖНО ТОЛЬКО В ПРОЦЕССЕ РАБОТЫ ПРОЦЕДУРЫ Less(D1,D2), НИ В КОЕМ СЛУЧАЕ НЕ ИЗМЕНЯТЬ ПОРЯДОК СЛЕДОВАНИЯ ПОЛЕЙ ПРИ ОПИСАНИИ ТИПА Date_pack !
ВЫБИРАЙТЕ ТОТ СПОСОБ СРАВНЕНИЯ ДАТ, КОТОРЫЙ ВАМ БОЛЬШЕ НРАВИТСЯ.

Тесты для этапа 1:

номер теста	*ввод*	*вывод*
1	**10.2.20 13.2.20**	**10.2.20<13.2.20 is true**
2	**20.2.20 10.3.20**	**20.2.20<10.3.20 is true**
3	**15.2.20 10.2.21**	**15.2.20<10.2.21 is true**
4	**1.2.3 1.2.3**	**1.2.3<1.2.3 is false**
5	**10.2.20 15.1.20**	**10.2.20<15.1.20 is false**
6	**9.2.20 10.2.19**	**9.2.20<10.2.19 is false**

Этап 2 (выполняется только после сдачи ***этапа 1***) (50 очков)

ОПИСАНИЕ МАССИВА ИЗ ЗАПИСЕЙ:

```
N equ 6
Arr_of_Rec Date_pack N dup(<>); массив с именем Arr_of_Rec из N элементов-дат
                              ; типа Date_pack.
```

Этап_1 (который должен быть к этому моменту сдан) *оформить в виде комментария* (т.е. чтобы не было ввода двух дат, их сравнения и вывода ответа) следующим образом:

```
COMMENT ~
Фрагмент программы, относящийся к этапу_1
~
```

Ввести (с клавиатуры) N дат, сохранив их в последовательных элементах массива **Arr_of_Rec**. *Распечатать* получившийся массив дат. Далее *проверить* ***упорядочены ли введённые даты строго по возрастанию***, и выдать ответ «**sorted/not sorted**».

Требования к решению на 2-ом этапе.
Для *ввода*, *вывода* и *сравнения* дат использовать отлаженные ранее (на ***1-ом этапе***) процедуры **In_Rec(D)**, **Out_Rec(D)** и функцию **Less(D1,D2)**.

Других процедур и функций на ***2-ом этапе*** не использовать, то есть ***в основной программе*** следует выписать ***только три нижеследующих (независимых цикла)*** (циклы не совмещать!):

1) *цикл для ввода* **N** дат и сохранения их в элементах массива **Arr_of_Rec**; (15 очков)
2) *цикл для вывода* **N** дат, хранящихся в элементах массива **Arr_of_Rec**; (15 очков)
3) *цикл для проверки упорядочены ли введённые элементы-даты по возрастанию*? (20 очков)

Можно сдавать *Этап 2* по частям (ввод-вывод, проверка на упорядоченность), если сложно сдать всё сразу.

Тесты для этапа 2:

номер теста	тестовый массив						ответ
1	**10.2.19**	**13.2.19**	**13.3.19**	**14.3.19**	**14.3.20**	**15.4.20**	sorted
2	**10.2.19**	**13.2.19**	**13.3.19**	**13.3.19**	**14.3.20**	**15.4.2**	not sorted
3	**10.2.19**	**13.2.19**	**13.3.19**	**14.3.19**	**14.3.18**	**15.4.20**	not sorted
4	**10.2.19**	**13.2.19**	**13.3.19**	**14.3.19**	**14.3.20**	**13.3.20**	not sorted

---

Этап 3 (выполняется только после сдачи ***этапа 2***) (40 очков)

Этап_1 (который должен быть сдан) оформить в виде комментария (т.е. чтобы не было ввода двух дат, их сравнения и вывода ответа): COMMENT ~ *Этап_1* ~

Этап_2 должен остаться не закомментированным.

Найти среди элементов-дат массива **Arr_of_Rec** (описание массива **Arr_of_Rec** дано на ***этапе 2***) ***наименьшую дату*** (т.е. предшествующую всем остальным датам) и *вывести* её на экран в виде: **dd.mm.yy is minimum** .

Требования к решению на 3-ом этапе.

1) Для поиска в массиве **Arr_of_Rec** **наименьшей даты** реализовать функцию **Min_Date(Arr,Len)**, где **Arr** (по ссылке) - адрес массива из записей-дат, **Len** (по значению) – его длина. Функция *возвращает* через регистр **AX** ***минимальную дату***. В процессе своей работы функция для сравнения текущего минимума с очередной датой *обращается* к ранее (на ***этапе 1***) отлаженной функции **Less(D1,D2)**.
2) Для вывода наименьшей даты использовать ранее (на ***этапе 1***) отлаженную процедуру **Out_Rec(D)** .

Тесты для этапа 3:

номер теста	тестовый массив (красным – *минимальная дата*)					
1	**15.4.20**	**14.3.19**	**10.2.19**	**13.3.19**	**14.3.20**	**13.2.19**
2	**15.4.20**	**14.3.20**	**14.3.19**	**13.3.19**	**13.2.19**	**10.2.19**
3	**10.2.19**	**15.4.20**	**14.3.19**	**14.3.20**	**13.2.19**	**13.3.19**

Этап 4 (выполняется после сдачи ***этапа 3***) (60 очков)

Этап_1 (который должен быть сдан) оформить в виде комментария (т.е. чтобы не было ввода двух дат, их сравнения и вывода ответа): COMMENT ~ *Этап_1* ~

Этап_2 и *этап_3* должны остаться не закомментированными.

ОПИСАНИЕ ТИПА СТРУКТУРЫ:

```
Date_unpack struc ; распакованная дата
    D db ?        ; день  (число от 1 до 31)
    M db ?        ; месяц  (число от 1 до 12)
    Y db ?        ; год  (число от 0 до 99)
Date_unpack ends
```

ОПИСАНИЕ МАССИВА ИЗ СТРУКТУР:

```
; N equ 6 – эта константа должна быть описана выше (для 2-го этапа)
Arr_of_Struc Date_unpack N dup(<>)  ; массив с именем Arr_of_Struc из N элементов
                                    ; типа Date_unpack
```

Распаковать элементы-даты массива **Arr_of_Rec** (которые были введены на ***этапе 2***), сохранив их в массиве **Arr_of_Struc**. *Вывести* на экран получившийся массив **Arr_of_Struc** (каждую дату - в виде **dd.mm.yy**).

Требования к решению на 4-ом этапе.

Для распаковки даты описать процедуру **Rec_to_Struc(D_pack, D_unpack)**, которая по записи типа **Date_pack** формирует структуру типа **Date_unpack** (1-ый параметр **D_pack** – по значению, 2-ой параметр **D_unpack** – по ссылке).
Для вывода содержимого полей отдельной структуры описать процедуру **Out_Struc(D_unpack)** (параметр **D_unpack** в процедуру передать по ссылке).

Внимание! В теле процедур **Rec_to_Struc** и **Out_Struc** при обращении к **полям структуры** настоятельно рекомендуется использовать **имена этих полей.** Для этого следует применить к адресу структуры оператор **Date_unpack ptr**. *Например*, если в регистре **EBX** хранится адрес структуры типа **Date_unpack**, то для доступа к полю **M** этой структуры нужно использовать конструкцию **(Date_unpack ptr [EBX]).M** (такая конструкция задаёт адрес, тип которого совпадёт с типом поля **M**, то есть **равен байту**).

Других процедур и функций на ***4-ом этапе*** не использовать, то есть для *этапа 4* требуется в основной программе реализовать два последовательных цикла (циклы не совмещать!):
1) цикл движения по массиву **Arr_of_Rec** **с целью распаковки дат и сохранения их в массиве структур** **Arr_of_Struc**; (30 очков)
2) цикл движения по массиву **Arr_of_Struc** с целью **вывода дат на экран** (каждую дату - в виде **dd.mm.yy**). (30 очков)

Этап 5 (**дополнительный**, выполняется **ДОМА** только после сдачи ***всех четырёх этапов***) (60 очков) (сдать этот этап можно **до 17 мая время 22.00** *без потери очков*)

По мотивам функции **Min_Date(Arr,Len)** (см. *этап 3*), описать специально для *этапа 5* новую функцию **Ptr_to_Min_Date(Arr,Len)**. Эта функция должна возвращать через **EAX** **адрес (смещение)** минимального элемента массива **Arr** длины **Len** (элементы массива - типа **Date_pack**).

Воспользоваться новой функцией **Ptr_to_Min_Date** для сортировки элементов массива **Arr_of_Rec** **по неубыванию** методом ***выбора*** ***(догадаться самостоятельно - как нужно применить в решении новую функцию).*** Сортировку оформить в виде процедуры **Sort(Arr,Len)**. Вывод отсортированного массива выполнить с помощью процедуры **Out_Sort(Arr,Len)**, которая в процессе работы обращается к ранее отлаженной (на *этапе 1*) процедуре **Out_Rec(D)** .

Напоминание алгоритма сортировки методом выбора.
Найти минимальный (максимальный) элемент последовательности и переставить его с первым (последним) элементом. Далее аналогичный метод применить ко всем элементам, кроме первого (последнего), так как он находится на своём окончательном месте. И так далее.

Сдавать на ПРОВЕРКУ БЛОКОВ в виде ***архива*** из четырёх файлов: **asm, exe, lst** и **обязательного скриншота с результатами работы программы на ваших тестах**. Просьба перед **exit** ставить **pause** (ваши программы запускаются непосредственно из архива, важно до закрытия консольного окна успеть увидеть результаты работы программы; даты последнего сохранения asm, lst и exe-файлов должны совпадать !).

Тема письма: *выход-4 (этап_5)*

Этап 6 (**дополнительный,** выполняется **ДОМА** только после сдачи ***пяти этапов***) (10 очков) (сдать этот этап можно **до 17 мая время 22.00** *без потери очков*)

Дать решение задачи (*пяти этапов*) при следующих *условиях*: все процедуры и функции должны быть размещены в отдельном (*вспомогательном*) *модуле* и **соблюдать** стандартные соглашения о связях **stdcall**. Тема "Модули" будет изучаться сразу после **выхода-4**.

ЗАМЕЧАНИЕ. Если во *вспомогательном модуле* используется имя типа структуры или записи (или фигурируют имена полей), то описание соответствующего типа обязательно должно быть дано в этом модуле (т.к. модули транслируются отдельно друг от друга).

Требования к оформлению этапа 6.
Головной модуль следует назвать **main.asm.** *Вспомогательный модуль* назвать **unit.asm.**
Останов (завершение работы) – в модуле **main.asm** !
Сдавать на ПРОВЕРКУ БЛОКОВ в виде ***архива*** из шести файлов: main.**asm**, unit.asm, main.**exe**, main.**lst**, **unit.lst** и **обязательного скриншота с результатами прохождения этапа_6 на ваших тестах**. Просьба перед **exit** ставить **pause** (ваши программы запускаются непосредственно из архива, важно до закрытия консольного окна успеть увидеть результаты работы вашей программы; даты последнего сохранения asm, lst и exe-файлов должны совпадать !).
Тема письма: *выход-4 (этап_6)*
