ПРАКТИЧЕСКАЯ РАБОТА № 8.

**Тема: *«РАЗРАБОТКА ПРОГРАММ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ C ИСПОЛЬЗОВАНИЕМ ОДНОМЕРНЫХ И ДВУМЕРНЫХ МАССИВОВ».***

**Цель:**

* Изучить особенности выполнения команд, организующих циклы (LOOPx), и команд загрузки исполнительного адреса (LEA).
* Научиться использовать указанные команды при составлении программ циклической структуры на языке ассемблера.

**ХОД РАБОТЫ:**

1. Повторить приведенный ниже теоретический материал.
2. Разобрать примеры приведенных программ, реализующих обработку массивов.
3. Внимательно изучить задание своего варианта на обработку элементов массива.
4. Разработать блок-схему программы обработки элементов массива.
5. Задать одномерный массив целочисленных данных MAS в одном из заданных форматов (DB, DW, DD) и реализовать обработку массива в соответствии с заданием. Длина массива N. Исходные данные задать самостоятельно, учитывая формат элементов массива MAS. Тип результата определяется из контекста задачи.
6. Произвести тестовые проверки, отметить нормальные и аномальные результаты, сделать анализ результатов.
7. Сравнить результаты, полученные в результате работы программы, с рассчитанными вручную.
8. Ответить на контрольные вопросы.
9. Оформить отчет по практической работе, включающий тему и цель работы, листинги программ, тестовые примеры, ответы на контрольные вопросы.

**ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ.**

Для того чтобы обрабатывать массив в ассемблере, нужно знать, где он хранится (его начальный адрес), и длину его элемента. Как и в языке C/C++, **имя массива в ассемблере является и его начальным адресом**.

Архитектура CPU предоставляет достаточно удобные программно-аппаратные средства для работы с массивами. К ним относятся базовые и индексные регистры, позволяющие реализовывать несколько режимов адресации данных. Используя базовый, индексный, базово-индексный и базово-индексный со смещением режимы адресации можно организовать эффективную работу с массивами в памяти.

Например, индексный режим адресации вида **имя\_массива [регистр\_индекса]**позволяет обрабатывать ***каждый*** элемент массива, изменяя значение индексного регистра SI, DI.

При необходимости использования массива в программе его описывают одним из следующих способов:

**1.** Перечислением элементов массива в поле операндов одной из директив описания данных. При перечислении элементы разделяются запятыми.

**Например:**

**; массив из 4-х элементов. Размер каждого элемента – машинное слово**

**mas dw 0d50h, 77FFh, 5080h, 0ah**

**2.** Используя оператор повторения **dup**. Его используют обычно для резервирования памяти с целью размещения и инициализации элементов массива.

**Например:**

**;массив из 25 букв "а"**

**mas db 25 dup ("a")**

**3.** Используя директивы **label** и **rept**. Директива **rept** относится к макросредставм языка ассемблера и вызывает повторение указанное число раз строк, заключенных между директивой и строкой **endm.** Достоинство директивы **label** в том, что она не резервирует память, а лишь определяет характеристики ячейки памяти. Использование нескольких директив label, записанных одна за другой, дает возможность присвоить одной и той же области памяти разные имена и типы.

**Например:**

**mas1 label byte**

**mas2 label word**

**rept 8**

**db 0Ch**

**endm**

В результате в памяти будет создана последовательность из 8 байт 0Ch. Эту последовательность можно трактовать как массив байт с именем mas1 или как массив слов с именем mas2.

**4.** Использованием цикла для инициализации значениями области памяти, которую можно будет впоследствии трактовать как массив.

**Например:**

**mas dw 10 dup (?)**

**Организация циклов.**

Организовать циклическое выполнение некоторого участка программы можно, например, используя команды условной передачи управления или команду безусловного перехода **jmp.**

Пример программы, подсчитывающей количество нулевых элементов в массиве байтов:

***.model small***

***.stack 100h***

***.data***

***len equ 10 ; количество элементов в mas***

***mas db 1, 0, 9, 8, 0, 7, 8, 0, 2, 0***

***.code***

***start:***

***mov ax, @data***

***mov ds, ax***

***mov cx, len ; кол-во элементов в mas записали в cx***

***xor ax, ax***

***xor si, si***

***cycl:***

***jcxz exit ; проверка cx на 0, если cx=0, то выход***

***cmp mas[si], 0 ; сравнение очередного элемента массива mas с 0***

***jne ml ; если не равно 0, то переход на метку ml***

***inc al ; в регистре al организуем счетчик нулевых элементов***

***ml:***

***inc si ; увеличиваем si на 1, для перехода к след. элементу mas***

***dec cx ; уменьшаем значение счетчика цикла на 1***

***jmp cycl***

***exit:***

***mov ax, 4c00h***

***int 21h***

***end start***

Цикл в данной программе организован тремя командами **jcxz, dec** и **jmp.** Команда **jcxz** выполняет две функции: предотвращает выполнение "пустого" цикла (когда счетчик в cx равен 0) и отслеживает окончание цикла после обработки всех элементов поля mas. Команда **dec** после каждой итерации (прохода) цикла уменьшает значение счетчика в регистре **cx** на 1. При такой организации цикла все операции по его функционированию предусматриваются программистом. Учитывая важность циклов при создании программ, в систему команд ввели группу из трех команд, облегчающую программирование циклов. Эти команды также используют регистр **cx** в качестве счетчика цикла. Далее приводится описание этих команд.

**КОМАНДЫ, ОРГАНИЗУЮЩИЕ ЦИКЛЫ**

**LOOP *метка\_перехода*** - повторить цикл. Команда позволяет организовать циклы. подобные циклам for в языках высокого уровня с автоматическим уменьшением счетчика цикла. Работа команды заключается в выполнении следующих действий:

* декремента регистра **cx**;
* сравнения регистра **cx** с нулем;

если **cx > 0,** то управление передается на метку перехода;

если **cx = 0,** то управление передается на следующую после loop команду программы.

**LOOPE/ LOOPZ *метка\_перехода*** - повторить цикл пока cx <> 0 или флаг zf = 0. Обе эти команды являются полными синонимами, работайте с той, с которой удобно. Работа этих команд заключатся в выполнении следующих действий:

* декремента регистра **cx;**
* сравнения регистра **cx** с нулем;
* анализа состояния флага нуля **zf;**

если **cx > 0 и zf = 1**, то управление передается на метку перехода;

если **cx = 0 или zf = 0,** то управление передается на следующую после loop команду программы.

**LOOPNE/ LOOPNZ *метка\_перехода*** - повторить цикл пока **cx <> 0** **или флаг zf = 1**. Обе эти команды также являются абсолютными синонимами, работайте с той, с которой удобно. Работа этих команд заключатся в выполнении следующих действий:

* декремента регистра **cx;**
* сравнения регистра **cx** с нулем;
* анализа состояния флага нуля **zf;**

если **cx > 0 и zf = 0,** то управление передается на метку перехода;

если **cx = 0 или zf = 1,** то управление передается на следующую после loop команду программы.

Команды **LOOPE/LOOPZ** и **LOOPNE/LOOPNZ** по принципу своей работы являются взаимообратными. Они расширяют действие команды **LOOP** тем, что дополнительно анализируют флаг **zf,** что дает возможность организовать досрочный выход цикла, используя этот флаг в качестве индикатора. **Типичное использование данных команд связано с операцией поиска определенного значения в последовательности или со сравнением двух чисел**.

Недостаток всех перечисленных команд организации цикла в том, что они реализуют только короткие переходы (от -128 до +127 байтов). Для работы с длинными циклами придется использовать команды условного перехода и команду **jmp**, поэтому необходимо освоить оба способа организации циклов.

**Организация одномерных массивов**

Все элементы массива располагаются в памяти последовательно

**Описание элементов массива**

mas db 1,2,3,4,5

mas dw 5 dup (0)

#### Доступ к элементам массива

mov ax,mas[si] ; в si номер элемента в массиве

mov mas[si], ax ; в di номер элемента в массиве

**ПРИМЕРЫ ПРОГРАММ**

***ПРИМЕР1.*** Программа, подсчитывающая количество нулевых элементов в массиве байтов:

**.model small**

**.stack 100h**

**.data**

**len equ 10 ; количество элементов в mas**

**mas db 1, 0, 9, 8, 0, 7, 8, 0, 2, 0**

**.code**

**start:**

**mov ax, @data**

**mov ds, ax**

**mov cx, len ; кол-во элементов в mas записали в cx**

**xor ax, ax**

**xor si, si**

**jcxz exit ; проверка cx на 0, если cx=0, то выход**

**cycl:**

**cmp mas[si], 0 ; сравнение очередного элемента массива mas с 0**

**jne ml ; если не равно 0, то переход на метку ml**

**inc al ; в регистре al организуем счетчик нулевых элементов**

**ml:**

**inc si ; увеличиваем si на 1, для перехода к след. элементу mas**

**loop cycl ; уменьшение значения счетчика цикла на 1 и сравнение его**

**; значения с 0**

**exit:**

**mov ax, 4c00h**

**int 21h**

**end start**

У команды **jcxz** осталась только одна функция – не допустить выполнения "пустого" цикла, поэтому изменилось ее место в теле программы, теперь она стоит до начала цикла **cycl**.

***ПРИМЕР 2*** Программа, осуществляющая поиск первого нулевого элемента в массиве байтов. Используем команду **LOOPNZ**

**.model small**

**.stack 100h**

**.data**

**len equ 10 ; количество элементов в mas**

**mas db 1, 0, 9, 8, 0, 7, 8, 0, 2, 0**

**message db "В поле mas нет элементов, равных 0, $"**

**.code**

**start:**

**mov ax, @data**

**mov ds, ax**

**mov cx, len ; кол-во элементов в mas записали в cx**

**xor ax, ax**

**xor si, si**

**jcxz exit ; проверка cx на 0, если cx=0, то выход**

**mov si, -1 ; готовим si к адресации элементов массива mas**

**cycl:**

**inc si ; увеличиваем si на 1, для перехода к след. элементу mas**

**cmp mas[si], 0 ; сравнение очередного элемента массива mas с 0**

**loopnz cycl**

**jz exit ; если zf равен 0, то выход из цикла**

**; вывод сообщения, если нет нулевых элементов в mas**

**mov ah, 9**

**mov dx, offset message**

**int 21h**

**exit:**

**mov ax, 4c00h**

**int 21h**

**end start**

*ПРИМЕР 3.* Найти в строке хотя бы один нулевой элемент (еще один возможный вариант реализации)

**model small**

**.stack 100h**

**.data**

**bufer dw 25 ;формирую размер буфера для ввода строки**

**mas db 25 dup (' ') ;формирую буфер**

**subj1 db ‘в строке найден нулевой элемент', '$'**

**subj2 db ‘в строке не найден нулевой элемент', '$'**

**.code**

**main:**

**mov ax,@data**

**mov ds,ax**

**; ввод строки с клавиатуры**

**mov ah,0ah**

**mov dx, offset bufer**

**int 21h**

**;поиск нулевого элемента**

**xor si, si**

**mov cl, mas[si] ;загружаем в сl количество элементов в строке**

**mov al, 030h ;в al загружаем ASCII код нуля**

**m1: inc si**

**cmp al, mas[si]**

**je m2**

**;если в строке найдем нулевой элемент, то выходим из цикла на вывод subj1**

**loop m1**

**;нормальный выход из цикла означает, что в строке нет нулевых элементов**

**lea dx, subj2**

**jmp m3**

**m2: lea dx,subj1**

**m3: mov ah, 09h**

**int 21h**

**mov ax,4c00h**

**int 21h**

**end main**

*ПРИМЕР 4.* Найти максимальный элементы в каждой строке массива 5\*7

**model small**

**.stack 100h**

**.data**

**mas dw 5 dup( 7 dup(0))**

**max dw 0**

**subj db ‘введите строку',13,10,'$'**

**.code**

**main:**

**mov ax, @data**

**mov ds, ax**

**;заполнение массива**

**xor si, si**

**mov cx, 05h**

**incykl: push cx**

**mov ah, 09h**

**lea dx, subj**

**int 21h ;вывод информационной строки**

**mov cx, 07h**

**mov ah, 01h**

**outcykl: int 21h ;ввод элементов массива**

**mov mas[si], ax ;размещение элементов на месте**

**inc si**

**inc si**

**loop outcykl**

**pop cx**

**loop incykl**

**;поиск максимального/ минимального в строках**

**xor si,si**

**mov cx, 05h**

**s1t: push cx**

**mov cx, 06h**

**mov dx, mas[si]**

**maxi: add si, 2**

**cmp dx, mas[si]**

**ja min1 ;если меньше, то переходим**

**mov dx, mas[si]**

**min1: loop maxi**

**;вывод максимального**

**mov ah, 02h**

**int 21h**

**pop cx**

**loop s1t**

**mov ax, 04c00h**

**int 21h**

**end main**

**ОСТОРОЖНО ОШИБКИ!**

**Начало цикла с нулевым счетчиком.** Обычной ошибкой начинающих программистов является то, что они устанавливают значение счетчика CX в нуль. Результатом этой ошибки является повторение цикла 65 535 раз, т.к. после декремента нуля в CX получается значение FFFFh.

**Изменение счетчика циклов.** Будьте внимательны и не модифицируйте регистр CX во время исполнения цикла. В следующем прмере CX инкрементируется внутри повторяющегося блока команд. В результате значение счетчика никогда не станет равным нулю, и цикл никогда не закончится.

**mov ah, 2 ; функция DOS – отображение символа**

**mov cx, 10 ; счетчик циклов**

**mov dl, '\*' ; отображение символа**

**top:**

**int 21h ; вызов DOS**

**inc cx ; добавляем 1 к cx (!)**

**loop top ; повторение до cx = 0**

**При выполнении работы следует учитывать**, что каждая из создаваемых программ должна включать очистку экрана, выдавать необходимые текстовые сообщения и запросы, результат выводить в удобной для восприятия форме в выделенном окне. При формировании массива желательно использовать процедуру **Random\_range** для генерации псевдослучайных чисел в заданном диапазоне.

***Фрагмент листинга*  программы, демонстрирующей отображение случайных чисел.**

Программа, отображает список из двадцати 32-разрядных случайных чисел между 0 и 999. Сначала вызывается **Randomize** для получения базового числа случайной последовательности. Оно зависит от времени суток. Затем вызывается **Random\_range** для генерации чисел и помещения их в EAX.

.code

**extrn Randomize:proc, Random\_range:proc, Writeint:proc, Crlf:proc**

call Randomize

mov cx,20

L1: move ax,1000

call Random\_range ; EAX=случайное целое число

mov bx,10 ; задается десятичная система счисления

call Writeint

call Crlf

Loop L1

***УКАЗАНИЕ:***

Вспомним также, как можно задать **длину массива Len\_mas**:

mas dw 45Ah, 73DCh, 5 dup (0), 0AF4h

Len\_mas = $ - mas

В этом случае команды

mov si, Len\_mas

mov ax, mas[si]

засылают в регистр AX **последний** элемент массива. Уменьшая содержимое индексного регистра SI на два, будем обращаться к элементам массива в обратном порядке !

**ПРАКТИЧЕСКИЕ ЗАДАНИЯ**

**Задание 1. (общее для *всех* вариантов). Отображение символа в случайных местах.**

Напишите программу, которая отображает символ 500 раз в различных местах экрана. Используйте процедуры **Writechar, Random\_range** и **Gotoxy** из библиотеки загрузочных модулей.

# ВАРИАНТЫ ЗАДАНИЙ

**1**. 1)Найти максимум массива;

2)удалить из массива максимум;

3) определить среднее арифметическое до и после преобразования.

4)найти минимальный по модулю элемент массива;

5)удалить такие элементы из массива;

6)перед каждым отрицательным элементом вставить нулевой.

**2**. 1)Найти минимальный четный элемент массива;

2)подсчитать количество отрицательных элементов.

3)удалить все четные отрицательные элементы;

4) найти среднее арифметическое четных элементов массива;

5) найти максимальный по модулю четный элемент;

6) расположить сначала положительные элементы в той же последовательности, что в исходном массиве, затем остальные в обратной последовательности.

**3**. 1)переставить положительные элементы в обратном порядке, не меняя положения остальных элементов массива;

2)определить положение максимума до и после преобразования;

3) определить сумму положительных элементов и затем удалить их.

4)найти минимальный четный элемент в массиве;

5) удалить этот элемент;

6) удалить все четные элементы, кроме первого такого.

**4**. 1)отсортировать массив в порядке увеличения значений элементов;

2)найти максимальный отрицательный элемент;

3)удалить его.

4) найти сумму положительных и отрицательных элементов;

5) удалить первый четный элемент;

6) если сумма положительных больше суммы отрицательных, то удалить положительные элементы из массива, иначе – отрицательные.

**5**. 1) найти среднее арифметическое элементов массива;

2) определить количество элементов, больших среднего арифметического;

3) удалить такие элементы.

4) найти первую серию нулевых элементов;

5) заменить элементы этой серии на значения их индексов;

6) поставить сначала положительные элементы в обратной последовательности, по сравнению с исходной в массиве, а затем остальные в той же последовательности.

**6**. 1) найти первую серию отрицательных элементов, состоящую из 3 и более элементов;

2) определить наибольшую по длине серию отрицательных элементов;

3) удалить ее.

4) найти минимальный по модулю элемент;

5) отсортировать элементы массива, расположенные после максимального по модулю элемента по возрастанию;

6) удалить нулевые элементы.

**7**. 1) подсчитать количество четных элементов массива и определить индекс последнего четного.

2) подсчитать сумму элементов массива до и после последнего четного;

3) удалить из массива отрицательные элементы.

4) найти первый положительный элемент;

5) заменить первый положительный элемент на среднее арифметическое отрицательных элементов;

6) расположить сначала отрицательные элементы в обратной последовательности, а затем положительные в прямой, в конце – нулевые.

**8.** 1) удалить из массива нулевые элементы;

2) оставшиеся элементы расположить в обратном порядке;

3) определить как изменилось положение максимума и минимума при этом.

4) обменять максимальный элемент с первым нулевым;

5) подсчитать количество отрицательных четных элементов;

6) удалить максимальный элемент.

**9**. 1) найти за один проход два максимума в массиве;

2) переставить в обратном порядке элементы массива, находящиеся между этими максимумами;

3) удалить эти максимумы.

4) найти максимальный элемент, делящийся на 3;

5) подсчитать количество чисел, больших найденного элемента;

6) удалить такие числа.

**10**. 1) из массива удалить все нечетные числа, кроме первого;

2) определить минимум массива;

3) отсортировать массив в порядке убывания.

4) найти среднее арифметическое чисел, больших заданного числа а;

5) обменять максимум и минимум;

6) переставить максимум и минимум в начало массива.

**11**. 1) определить максимум и минимум массива;

2) удалить элементы массива, находящиеся между максимумом и минимумом

3) расположить сначала отрицательные, а затем неотрицательные элементы.

4)обменять максимальный отрицательный и минимальный положительный;

5) найти два минимума в массиве;

6) удалить их.

**12**. 1) определить количество четных элементов в массиве и индекс первого четного;

2) удалить первый четный элемент;

3) между двумя смежными отрицательными элементами вставить нулевой элемент.

4) отсортировать массив в порядке уменьшения модулей элементов;

5) найти все отрицательные элементы в левой половине массива ;

6) удалить ее.

**13**. 1) найти максимальный отрицательный четный элемент в массиве;

2) найти сумму элементов до этого элемента;

3) удалить такие элементы.

4) подсчитать количество пар элементов, где первый элемент больше второго;

5) найти среднее арифметическое положительных элементов;

6) удалить из массива повторяющиеся элементы.

**14.** 1) удалить нечетные элементы массива, кроме первого такого элемента;

2) найти среднее арифметическое положительных элементов;

3) подсчитать количество повторяющихся элементов массива.

4) найти первый и последний нулевой элементы;

5) поставить в обратном порядке элементы, расположенные между этими нулевыми;

6) удалить первый и последний нулевой элементы.

**15**. 1) найти отдельно сумму четных и сумму нечетных элементов;

2) удалить четные элементы массива;

3) удалить повторяющиеся элементы массива.

4) подсчитать количество элементов, меньших среднего арифметического;

5) удалить элементы, меньшие среднего арифметического;

6) расположить сначала отрицательные, а затем неотрицательные элементы, изменив в каждой из групп относительный порядок элементов на обратный по сравнению с исходным.

**16**. 1) найти последнюю серию отрицательных элементов (серия – 3 и больше элемента);

2) удалить ее;

3) отсортировать массив в порядке увеличения значений элементов.

4) найти минимальный элемент массива;

5) определить сумму элементов массива, за исключением минимальных;

6) удалить минимальные элементы из массива.

**17**. 1) найти элемент массива, наименее отличающийся от среднего арифметического;

2) подсчитать среднее арифметическое элементов, расположенных после такого элемента;

3) удалить такой элемент.

4) подсчитать количество пар соседних элементов Xi и Xi+1, для которых Xi-- Xi+1=1;

5) найти первый и последний нулевые элементы;

6) удалить из массива элементы, находящиеся между первым и последним нулевыми.

**18**. 1) определить число и сумму элементов, расположенных внутри отрезка [а,в] (а,в заданы);

2) найти максимум элементов, расположенных внутри этого отрезка;

3) удалить элементы, расположенные внутри этого отрезка;

4) найти максимальный элемент, который делится на 5;

5) удалить такой элемент;

6) найти элемент, повторяющийся наиболее часто в массиве;

**19**. 1) найти сумму нечетных элементов массива;

2) подсчитать количество отрицательных элементов;

3) удалить все нечетные элементы.

4) найти максимальный элемент массива, который находится на четной позиции;

5) удалить такой элемент;

6) выполнить циклический сдвиг массива на к чисел влево (к- задано).

**20**. 1) найти максимальный повторяющийся элемент;

2) подсчитать среднее арифметическое четных элементов;

3) удалить максимальные повторяющиеся элементы.

4) найти сумму отдельно элементов с четными и с нечетными индексами;

5) если сумма элементов с четными индексами больше, чем с нечетными, заменить все элементы с четными индексами нулями, иначе обнулить элементы на нечетных позициях;

6) поставить нули в конец массива.

**21**. 1) найти минимальный нечетный элемент;

2) удалить все такие элементы;

3) подсчитать количество повторяющихся элементов массива.

4) найти среднее арифметическое нечетных элементов;

5) удалить элемент, наиболее удаленный от среднего арифметического;

6) найти максимальный неповторяющийся элемент.

**22** 1) переставить в обратном порядке отрицательные элементы массива, не меняя положения остальных элементов;

2) подсчитать количество нулевых элементов;

3) расположить сначала нулевые, затем положительные, затем отрицательные элементы массива.

4) обменять местами первый и второй отрицательные элементы;

5) найти среднее арифметическое положительных четных элементов;

6) удалить из массива повторяющиеся элементы.

**23**. 1) определить индексы первого и последнего отрицательных элементов;

2) поставить в обратном порядке элементы, расположенные между первым и последним отрицательными элементами;

3)удалить элементы, расположенные между первым и последним отрицательными элементами.

4) отсортировать массив по возрастанию;

5) вставить в отсортированный массив заданное число а, чтобы не нарушить упорядоченность.

6) преобразовать массив, чтобы четные и нечетные элементы в нем были расположены попарно (четный – нечетный – четный – нечетный …)

**24**. 1) найти первую серию четных чисел (серия – 3 элемента);

2) найти максимальный элемент этой серии;

3) удалить эту серию.

4) найти в массиве первую тройку подряд идущих чисел, расположенных по убыванию;

5) удалить ее.

6) удалить неповторяющиеся элементы из массива.

**25**. 1)найти последнюю серию отрицательных чисел (серия – 4 элемента);

2) найти среднее арифметическое элементов этой серии;

3) переставить в этой серии элементы в обратном порядке.

4) найти максимальный нечетный элемент;

5) переставить в обратном порядке элементы, расположенные до максимального нечетного;

6) удалить отрицательные элементы, кроме первого такого.

**26**. 1) отсортировать массив по убыванию модулей значений;

2) найти максимум и минимум массива;

3) удалить максимум и минимум массива.

4) определить первую серию отрицательных элементов (серия 4 элемента);

5) в этой серии переставить элементы в обратном порядке;

6) удалить нечетные элементы.

**27**. 1) обменять первый четный с последним нечетным элементом в массиве;

2)отсортировать массив по возрастанию.

3) расположить сначала положительные, затем нулевые, затем отрицательные элементы.

4) найдите максимум в массиве байт, и замените элементы массива со значением 0 на максимум;

5) вычислите сумму (в формате слова) значений массива байт. По найденной сумме вычислите среднее значение в формате байта;

6) задан массив знаковых байтов. Сформируйте массив, содержащий индексы тех элементов исходного массива, значения которых находятся в заданном диапазоне (Отсчет индексов от нуля).

**КОНТРОЛЬНЫЕ ВОПРОСЫ.**

1. Какая структура представления данных является массивом?
2. Что определяет упорядоченность элементов массива?
3. Как располагается в памяти одномерный массив? Двумерный?
4. Какая структура хранения соответствует одномерному массиву?
5. Какие локализовать i-тый элемент одномерного массива (общая формула)?
6. Какие локализовать i-тый элемент двумерного массива (общая формула)?
7. Какие режимы адресации можно использовать для обращения к i-му элементу массива?
8. В чем отличие размещения в памяти одномерного и двумерного массивов?
9. Что определяет начальный адрес массива в памяти?
10. Какие 4 способа описания массивов Вы знаете?
11. Какие способы описания массивов просто резервирует место в памяти под элементы массива? Какие инициализируют элементы массив?
12. Какой участок программы отвечает за моделирование обработки массива ?
13. Напишите команды, составляющие тело цикла для инкремента массива слов ***array1*** с использованием косвенной адресации без смещения.
14. Напишите команды, образующие тело цикла для декремента каждого четвертого байта последовательности данных ***array2,*** с использованием косвенной адресации со смещением и без смещения.
15. Напишите команды, составляющие тело цикла для инкремента слов, начиная с адреса array, с использованием косвенной адресации со смещением.
16. Напишите команды, составляющие тело цикла для увеличения на 20 каждого второго слова последовательности данных с адреса ***array***, с использованием косвенной адресации со смещением.
17. Данные заданы следующей директивой: **buf dw 12 dup (1, 3, -1), -3.** Напишите команды, образующие тело цикла для инкремента слов массива ***buf***, с использованием косвенной адресации со смещением.
18. Данные заданы следующей директивой: **buf dw 12 dup (1, 3, -1), -3.** Напишите команды, образующие тело цикла для инкремента байт (!), начиная от адреса ***buf***, с использованием косвенной адресации со смещением.

**ПРИЛОЖЕНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ №8**

***П. 1 Задача 1.***

Задан массив NxN элементов. Переместить элементы симметрично относительно побочной диагонали.

**Блок – схема алгоритма:**

Начало

M, N

J=0

I=0

A=M[I][J].

M[I][J]=

M[N-1-J][N-1-I].

M[N-1-J][N-1-I]=A

I=N-1-J

I=I+1

нет

J=N-1

J=J+1

нет

Конец

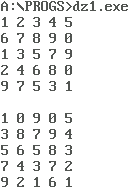
M

да

да

**Текст программы с комментариями:**

**Результат работы программы:**



data segment para public 'data'

m db 1,2,3,4,5

db 6,7,8,9,0

db 1,3,5,7,9

db 2,4,6,8,0

db 9,7,5,3,1

n dw 5

data ends

stk segment stack

db 256 dup (?)

stk ends

code segment para public 'code'

main proc

assume cs:code, ds:data, ss:stk

mov ax, data

mov ds, ax

call output\_m ; Выводим исходный массив

xor dx, dx ; В этом регистре будет переменная j

external: xor cx, cx ; А в этом - переменная i

internal: ; Сначала вычислим адреса переменных:

push dx ; Сохр. DX (его изменит ком-да MUL)

mov ax, dx ; ax = j

mov bx, n ; bx = n

mul bx ; ax = j \* n

add ax, cx ; ax = j \* n + i

mov di, ax ; В DI находится адрес 1-й пер-й

mov ax, n ; ax = n

dec ax ; ax = n - 1

sub ax, cx ; ax = n - 1 - i

mul bx ; ax = (n - 1 - i) \* n

pop dx ; Восст. DX (теперь нам нужен j)

mov bx, n ; bx = n

dec bx ; bx = n - 1

sub bx, dx ; bx = n - 1 - j

add bx, ax ; bx = (n - 1 - i) \* n + n - 1 - j

mov al, m[bx] ; Считываем первое и второе

mov ah, m[di] ; числа из массива

mov m[bx], ah ; Меняем их местами

mov m[di], al

inc cx ; i = i + 1

mov ax, n ; ax = n

dec ax ; ax = n - 1

sub ax, dx ; ax = n - 1 - j

cmp cx, ax ; Сравн. i и n - 1 - j

jne internal ; Если не равно продолж. внут. цикл

inc dx ; j = j + 1

mov ax, n ; ax = n

dec ax ; ax = n - 1

cmp dx, ax ; Сравн. j и n - 1

jne external ; Если не равно продолж. внеш. цикл

call output\_m ; Выводим полученный массив

mov ax, 4c00h

int 21h

main endp

output\_m proc ; Небольшая подпрограмма для вывода

mov ah, 02h ; Готовим прерывание 21 для вывода

mov cx, n ; Внешний цикл на n проходов

xor di, di ; Очищаем смещение массива

out\_extrn: push cx ; Запоминаем счетчик в стек

mov cx, n ; Внутренний цикл на n проходов

out\_intrn: mov dl, m[di] ; Считываем число

add dl, 30h ; Переводим его в ASCII-код

int 21h ; Выводим на экран

mov dl, ' ' ; Выводим пробел

int 21h

inc di ; Увеличиваем смещение массива

loop out\_intrn ; Завершаем внутренний цикл

pop cx ; Восстанавливаем счетчик

mov dl, 0ah ; Переходим на новую строчку

int 21h

mov dl, 0dh

int 21h

loop out\_extrn ; Завершаем внешний цикл

mov dl, 0ah ; Переходим на новую строчку

int 21h

mov dl, 0dh

int 21h

ret ; Завершаем подпрограмму

output\_m endp

code ends

end main

***П.2. Задача 2.***

Ввести 2 одномерных массива, первый из N целых чисел, второй из 5 различных целых чисел. Удалить из первого массива числа, содержащиеся во втором массиве.

**Блок – схема алгоритма:**

Начало

A, N

С=1 до 5

I=0

J=0

D=N

A[J]=B[I]

K=J

E=N-J-1 до 0

A[K]=A[K+1]

K=K+1

N=N-1

D=D-1

J=J-1

J=J+1

D=D-1

A

A

B

B

D=0

I=I+1

Конец

A, B, N

да

нет

**Текст программы с комментариями:**

data segment para public 'data'

Am db 3,5,1,8,7,6,4,9,2,0

Bm db 1,3,5,7,9

n dw 10

data ends

stk segment stack

db 256 dup (?)

stk ends

code segment para public 'code'

main proc

assume cs:code, ds:data, ss:stk

mov ax, data

mov ds, ax

call out\_m ; Вывести массив на экран

mov cx, 5 ; Цикл на 5 проходов (c = 5)

xor bx, bx ; i = 0

cmp\_ext: xor di, di ; Начало цикла; j = 0

mov dx, n ; d = n

cmp\_int: mov al, Am[di]

cmp al, Bm[bx] ; Сравнить A[j] и B[i]

jne cmp\_all ; Если не равны - не удалять число

push cx ; Сохранить счетчик в стеке

mov cx, n ; e = n

sub cx, di ; e = n - j

dec cx ; e = n - j - 1 (счетчик цикла)

push di ; k = j

cycl\_del: mov al, Am[di+1] ; A[k] = A[k+1]

mov Am[di], al

inc di ; k = k + 1

loop cycl\_del ; Завершить цикл

pop di ; Восстановить j

pop cx ; Восстановить счетчик

dec n ; n = n - 1

dec dx ; d = d - 1

dec di ; j = j - 1

cmp\_all: inc di ; j = j + 1

dec dx ; d = d - 1

test dx, dx ; d = 0 ?

jnz cmp\_int ; Если нет - продолж. внутр. цикл

inc bx ; i = i + 1

loop cmp\_ext ; Завершить внешний цикл

call out\_m ; Вывести массив на экран

mov ax, 4c00h

int 21h

main endp

out\_m proc ; Подпрограмма для вывода массива

mov cx, n ; Счетчик на n элементов массива

mov ah, 02h ; Подгот. 21-е прер. к выв. числа

xor bx, bx ; Очистить смещение адреса

out\_cycle: mov dl, Am[bx] ; Прочитать число из массива

inc bx ; Увеличить смещение адреса

add dl, 30h ; Перевести число в ASCII - код

int 21h ; Вывести число на экран

mov dl, ' ' ; Вывести пробел

int 21h

loop out\_cycle ; Завершить цикл

mov dl, 0ah ; Перейти на следующ. строчку

int 21h

mov dl, 0dh

int 21h

ret

out\_m endp

code ends

end main

**Результат работы программы:**



***П.3. Задача 3.***

В одномерный массив ввести N произвольных чисел, переместить его элементы так, чтобы в конце были отрицательные числа, и сохранился порядок следования.

**Блок – схема алгоритма:**

Начало

I = 0

C = N до 0

M, N

M[I] < 0

J = I

A = M[I]

D=1 до N – 1 – I

M[J] = M[J+1]

J = J + 1

M[N-1] = A

I = I – 1

C = C - 1

I = I + 1

M, N

Конец

нет

да

**Текст программы с комментариями:**

data segment para public 'data'

m db 2, -3, 8, -4, 6, 5, -1, 3, 2, 1

n dw 10

data ends

stk segment stack

db 256 dup (?)

stk ends

code segment para public 'code'

main proc

assume cs:code, ds:data, ss:stk

mov ax, data

mov ds, ax

call output\_m ; Вывести исходный массив

mov cx, n ; Цикл на n проходов

xor bx, bx ; i = 0

work\_cycle: mov al, m[bx] ; Начало цикла

cmp al, 0 ; m[i] < 0 ?

jnl work\_next ; Если нет - ничего не меняем

push cx ; Сохраняем счетчик в стеке

push bx ; j = i

mov cx, n ; cx = n

sub cx, bx ; cx = n - i

dec cx ; cx = n - i - 1

shift\_cycle: mov ah, m[bx+1] ; Начало цикла со счет. n - 1 - i

mov m[bx], ah ; m[j] = m[j+1]

inc bx ; j = j + 1

loop shift\_cycle ; Завершить цикл

mov bx, n

dec bx ; bx = n - 1

mov m[bx], al ; Записать удал. число в m[n - 1]

pop bx ; Восстановить i

dec bx ; i = i - 1

pop cx ; Восстановить счетчик из стека

dec cx ; Уменьшить его на 1

work\_next: inc bx ; i = i + 1

loop work\_cycle ; Завершить внешний цикл

call output\_m ; Вывести полученный массив

mov ax, 4c00h

int 21h

main endp

output\_m proc ; Подпрограмма для вывода массива

mov ah, 02h ; Подготовить прер. 21 к выв. числа

mov cx, n ; Пройти все числа в массиве

xor bx, bx ; Очистить индексный регистр

out\_cycle: mov dl, m[bx] ; Прочитать число из массива

inc bx ; Увеличить смещение адреса

cmp dl, 0 ; Сравнить это число с нулем

jnl out\_dl ; Если оно не ниже - вывод как есть

not dl ; Если ниже, поменять

inc dl ; знак числа,

push dx

mov dl, '-' ; Вывести знак минуса,

int 21h

pop dx

out\_dl: add dl, 30h ; Перевести число в ASCII - код

int 21h ; И вывести его на экран

mov dl, ' ' ; Затем вывести пробел

int 21h

loop out\_cycle ; Завершить цикл

mov dl, 0ah ; И перейти на новую строку

int 21h

mov dl, 0dh

int 21h

ret ; Завершить подпрограмму

output\_m endp

code ends

end main

**Результат работы программы:**

****