**теоретические сведения**

**Массив** можно описать как упорядоченное множество элементов определенного типа. Упорядоченность массива определяется набором целых чисел, называемых индексами, которые связываются с каждым элементом массива и однозначно конкретизируют его расположение среди других элементов массива.

Под ***сортировкой***понимается процесс перестановки элементов некоторого множества в порядке убывания или возрастания. Если элементы множества — скаляры, то сортировка ведется относительно их значений. Если элементы множества — структуры, то каждая структура должна иметь поле (поля), относительно которого будет производиться упорядочивание местоположения элементов (то есть сортировка) относительно других элементов-структур множества.

Различают **два типа алгоритмов сортировки**: сортировку массивов и сортировку файлов. Другое их название — алгоритмы **внутренней** и **внешней сортировки**. Разница заключается в местонахождении упорядочиваемых объектов: для внутренней сортировки — это оперативная память компьютера, для внешней — файл.

Рассмотрим некоторые алгоритмы внутренней сортировки.

**Сортировка прямым включением**

Идея сортировки прямым включением заключается в том, что в упорядочиваемой последовательности masi длиной n *(i* = 0...n - 1) последовательно выбираются элементы, начиная со второго (i≤ 1) и ищутся их позиции в уже отсортированной левой части последовательности. При этом поиск места включения очередного элемента х в левой части последовательности mas может закончиться двумя ситуациями:

- найден элемент последовательности, для которого massi < х;

- достигнут левый конец отсортированной слева последовательности.

Первая ситуация разрешается тем, что последовательность mas, начиная с masi раздвигается в правую сторону и на место masi перемещается х. Во втором случае следует сместить всю последовательность вправо и на место mas0 переместить х.

Здесь для отслеживания условия окончания просмотра влево отсортированной последовательности используется прием «барьера». Суть его в том, что к исходной последовательности слева добавляется фиктивный элемент X. В начале каждого шага просмотра влево отсортированной части массива элемент X устанавливается в значение того элемента, для которого будет осуществляться поиск места вставки. Пример программы, реализующей алгоритм сортировки прямым включением, приведен ниже (Программа1).

***Программа1.***

***.data***

*;задаем массив*

*mas db 44h,55h,12h,42h,94h,18h,06h,67h*

*n=$-mas*

*X db 0 ;барьер*

***.stack 256***

***.486***

***.code***

*main:*

*;НАЧало\_ПРОГраммы*

*mov dx,@data*

*mov ds,dx*

*;ДЛЯ i:=1 ДО n-1 //i изменяется в диапазоне 0..n-1*

*mov cx,n-1 ;цикл по i*

*mov si,1 ;i=2*

*;НАЧало\_БЛОКа\_1*

*;присвоение исходных значений для очередного шага*

*;X:= mas[i]; mas[0]:=X; j:=i-1*

*cycl3: mov al,mas[si]*

*mov x,al*

*push si ;временно сохраним i, теперь j=i*

*;еще один цикл, который перебирает элементы до барьера x=mas[i]*

*;ПОКА (X< mas[j]) ДЕЛАТЬ*

*;НАЧало\_БЛОКа\_2*

*;mas[j+1]:= mas[j]; j:=j-1*

*cycl2: mov al,mas[si-1]*

*cmp x,al ;сравнить x < mas[j-1]*

*ja m1 ;если x > mas[j-1]*

*; если x < mas[j-1], то*

*mov al,mas[si-1]*

*mov mas[si],al*

*dec si*

*jnz cycl2*

*;КОНец\_БЛОКа\_2*

*; mas[j+1]:=X*

*m1: mov al,x*

*mov mas[si],al*

*pop si*

*inc si*

*loop cycl3*

*;КОНец\_БЛОКа\_1*

*;выход*

*;КОНец\_ПРОГраммы*

*mov ax,4c00h*

*int 21h*

*end main*

**Сортировка прямым выбором**

В алгоритме сортировки прямым выбором на каждом шаге просмотру подвергается неупорядоченная правая часть массива. В ней ищется очередной минимальный элемент. Если он найден, то производится его перемещение на место крайнего левого элемента несортированной правой части массива. Ниже приведен пример реализации сортировки прямым выбором (Программа 2).

***Программа 2.***

***.data***

*;задаем массив*

*mas db 44h,55h,12h,42h,94h,18h,06h,67h*

*n=$-mas*

*X db 0*

*K dw 0*

***.stack 256***

***.486***

***.code***

*;НАЧало\_ПРОГраммы*

*main:*

*mov dx,@data*

*mov ds,dx*

*;ДЛЯ i:=1 ДО n-1 //i изменяется в диапазоне 1..n-1*

*;внешний цикл - по i*

*mov cx,n-1*

*mov si,0*

*;НАЧало\_БЛОКа\_1*

*;//присвоение исходных значений для очередного шага*

*cycl1:*

*push cx*

*;k:=i; X:= mas[i]*

*mov k,si ;k:=i*

*mov al,mas[si]*

*mov x,al ;x:=mas[i]*

*push si ;временно сохраним i - теперь j=I+1*

*inc si ;j=I+1*

*;вложенный цикл - по j*

*mov al,n*

*sub ax,si*

*mov cx,ax ;количество повторений внутреннего цикла по j*

*;ДЛЯ j:=i+1 ДО n j изменяется в диапазоне i+1..n*

*cycl2:*

*;ЕСЛИ mas[j]<X ТО*

*;НАЧало\_БЛОКа\_2*

*mov al,mas[si]*

*cmp al,x*

*ja m1*

*;k:=j; x:= mas[j]*

*mov k,si ;k:=j*

*;КОНец\_БЛОКа\_2*

*mov al,mas[si]*

*mov x,al ;x:=mas[k]*

*m1: inc si ;j:=j+1*

*loop cycl2*

*;mas[k]:= mas[i]; mas[i]:=X*

*pop si*

*mov al,mas[si]*

*mov di,k*

*mov mas[di],al ;mas[k]:=mas[i]*

*mov al,x*

*mov mas[si],al ;mas[i]:=x*

*inc si*

*pop cx*

*;КОНец\_БЛОКа\_1*

*loop cycl1*

*;выход*

*mov ax,4c00h*

*int 21h*

*;КОНец\_ПРОГраммы*

*end main*

В приведенных примерах программ для организации циклов используется команда LOOP метка. Эта команда служит для организации и управления циклами. Число повторений цикла должно быть занесено в регистр СХ. В каждом цикле команда LOOP автоматически уменьшает содержимое регистра СХ на 1. Пока значение в СХ не равно нулю, управление передается по адресу (метке), указанному в операнде. Когда в СХ будет 0, управление перейдет на следующую после LOOP команду.

**Сортировка прямым обменом (метод пузырька)**

Этот метод основан на сравнении и обмене пар соседних элементов до их полного упорядочивания. Этот метод также называется методом пузырьковой сортировки (методом пузырька). Действительно, если упорядочиваемую последовательность расположить не слева направо, а сверху вниз то визуально каждый шаг сортировки будет напоминать всплытие легких (меньших по значению) пузырьков вверх.

Рассмотрим этот метод на следующем примере. Пусть задана некоторая несортированная последовательность Х с элементами х0, х1,..хn и служебная переменная а. Начнем сравнение крайне правого элемента хn со следующим элементом хn-1. Если хn ≤ хn-1 производим следующую замену: а= хn-1 , хn-1=хn, а хn=а. После этого переходим к сравнению элементов хn-1 и хn-2 и так далее до самого первого элемента. В результате этих проверок и замен самый маленький – «легкий» элемент «всплывет» - займет место первого в массиве элемента. Далее процедура повторяется начиная с элемента хn и до элемента х2 , потом от хn до х3 и так далее.

; фрагмент программы сортировки массива методом прямого обмена

*;-------------------------------------------------------------------------------*

*mas db 44,55,12,42,94,18,06,67 ; задаем массив*

*N=$-mas ; длина массива mas*

*X db 0*

*;-------------------------------------------------------------------------------*

*.code*

*;---------------------*

*; внешний цикл по i*

*mov cx,n-1 ; определим цикл по i*

*mov si,1*

*cykl1:*

*push cx*

*mov cx,N*

*sub cx,si ; количество повторений внутреннего цикла*

*push si ; временно сохраним i – теперь j:=N*

*mov si,N-1*

*cykl2: ; ЕСЛИ mas[j-1]<mas[j] ТО*

*mov al,mas[si-1]*

*cmp mas[si],al*

*ja m1*

*mov X,al ; X:=mas[j-1]*

*mov al,mas[si]*

*mov mas[si-1],al ;mas[j-1]:=mas[j]*

*mov al,X*

*mov mas[si],a l ; mas[j]:=X*

*m1: dec si ; цикл по j с декрементом N-1 раз*

*loop cykl2*

*pop si*

*inc si*

*pop cx*

*loop cykl1*

*;-------------------------------------------------------------------------------*

**Порядок выполнения работы.**

1. Загрузить программу эмулятор EMU8086
2. Набрать текст ***Программы 1***.
3. Выполнить программу в пошаговом режиме.
4. Получить у преподавателя последовательность для сортировки, выполнить с помощью программы сортировку и предъявить результат преподавателю.
5. Набрать текст ***Программы 2*** и повторить для нее предыдущие действия.
6. Используя предложенный фрагмент программы, реализующей метод сортировки прямым обменом, разработать программу полностью. Набрать и отладить эту программу. Выполнить сортировку массива данных, предложенного преподавателем.
7. Составить и зарисовать в отчете блок-схемы алгоритмов исследуемых программ.
8. Оценить скорость сортировки данных по исследуемым алгоритмам. Для выполнения оценки подсчитать количество команд от начала выполнения программы до ее завершения (выполнения полной сортировки). Объяснить полученные результаты.

**Содержание отчета.**

1. наименование, цель и программу работы;

2. блок-схемы алгоритмов сортировки, рассмотренных в лабораторной работе;

3. тексты и результаты работы программ сортировки методами прямого включения, прямого выбора и прямого обмена;

4. результаты сравнения эффективности работы всех трех алгоритмов;

5. выводы по работе.

**Контрольные вопросы**

1. Дать понятие массива данных.
2. Что такое сортировка данных?
3. Объяснить принцип сортировки прямым включением.
4. Объяснить принцип сортировки прямым выбором.
5. Дать понятие сортировки прямым обменом.
6. Какой из исследуемых алгоритмов сортировки дает наибольшую скорость в работе?

**ПРИЛОЖЕНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ №6**

***ЛИСТИНГ ПРОГРАММЫ ДЛЯ ЗАДАЧИ 1.***

**ЗАДАЧА 1.** Заданы два массива различной размерности, состоящие из целых положительных чисел. Упорядочить каждый из массивов по возрастанию. Буферный массив не использовать. Учесть частные случаи.





*Указание:* параметры, необходимые для обработки массива (его адрес и размерность) передаются в процедуру через регистры.

*Turbo Assembler Version 4.1 10/09/04 21:30:32 Page 1*

*l2.ASM*

*1 .286*

*2 0000s SEGMENT 'STACK'*

*3 0000 80\*(??) db 128 dup(?)*

*4 0080s ends*

*5*

*6 0000 dseg SEGMENT*

*7 0000 01\*(02 09 07 05 06 + mas1 db 1 dup (2, 9, 7, 5, 6, 1, 2)*

*8 01 02)*

*9 0007 07 n1 db 7*

*10 0008 01\*(04 02 07 01 00 + mas2 db 1 dup(4, 2, 7, 1, 0, 3, 9, 4, 4, 1)*

*11 03 09 04 04 01)*

*12 0012 0A n2 db 10*

*13 0013 0000 k dw 0*

*14 0015 00 x db 0*

*15 0016 dseg ends*

*16*

*17 0000 cseg SEGMENT*

*18 ASSUME ss:s, ds:dseg, cs:cseg*

*19*

*20 0000 p\_sort proc far*

*21 0000 60 pusha*

*22 0001 8B CA mov cx, dx*

*23 0003 83 F9 01 cmp cx,1*

*24 0006 74 3E je p\_sort\_end*

*25 0008 49 dec cx*

*26 0009 BE 0000 mov si,0*

*27 000C cycl1:*

*28 000C 51 push cx*

*29 000D 89 36 0013r mov k, si*

*30 0011 8A 00 mov al, [bx][si]*

*31 0013 A2 0015r mov x, al*

*32 0016 56 push si*

*33 0017 46 inc si*

*34 0018 33 C0 xor ax, ax*

*35 001A 8B C2 mov ax, dx*

*36 001C 2B C6 sub ax, si*

*37 001E 8B C8 mov cx, ax*

*38 0020 cycl2:*

*39 0020 8A 00 mov al, [bx][si]*

*40 0022 3A 06 0015r cmp al,x*

*41 0026 77 09 ja m1*

*42 0028 89 36 0013r mov k,si*

*43 002C 8A 00 mov al, [bx][si]*

*44 002E A2 0015r mov x, al*

*45 0031 46 m1: inc si*

*46 0032 E2 EC loop cycl2*

*47 0034 5E pop si*

*48 0035 8A 00 mov al,[bx][si]*

*49 0037 8B 3E 0013r mov di,k*

*50 003B 88 01 mov [bx][di],al*

*51 003D A0 0015r mov al,x*

*52 0040 88 00 mov [bx][si],al*

*53 0042 46 inc si*

*54 0043 59 pop cx*

*55 0044 E2 C6 loop cycl1*

*56 0046 p\_sort\_end:*

*57 0046 61 popa*

*58 0047 CB ret*

*59 0048 p\_sort endp*

*60*

*61 0048 base proc far*

*62 0048 1E push ds*

*63 0049 33 C0 xor ax,ax*

*64 004B 50 push ax*

*65 004C B8 0000s mov ax,dseg*

*66 004F 8E D8 mov ds,ax*

*67*

*68 0051 BB 0000r mov bx, offset mas1*

*69 0054 33 D2 xor dx, dx*

*70 0056 8A 16 0007r mov dl, n1*

*71 005A 0E E8 FFA2 call p\_sort*

*72*

*73 005E BB 0008r mov bx, offset mas2*

*74 0061 33 D2 xor dx, dx*

*75 0063 8A 16 0012r mov dl, n2*

*76 0067 0E E8 FF95 call p\_sort*

*77*

*78 006B CB retf*

*79 006C base endp*

*80*

*81 006C cseg ends*

*82 end base*

*Symbol Name Type Value Cref (defined at #)*

*??DATE Text "10/09/04"*

*??FILENAME Text "l2 "*

*??TIME Text "21:30:32"*

*??VERSION Number 040A*

*@CPU Text 0707H #1*

*@CURSEG Text CSEG #2 #6 #17*

*@FILENAME Text L2*

*@WORDSIZE Text 2 #1 #2 #6 #17*

*BASE Far CSEG:0048 #61 82*

*CYCL1 Near CSEG:000C #27 55*

*CYCL2 Near CSEG:0020 #38 46*

*K Word DSEG:0013 #13 29 42 49*

*M1 Near CSEG:0031 41 #45*

*MAS1 Byte DSEG:0000 #7 68*

*MAS2 Byte DSEG:0008 #10 73*

*N1 Byte DSEG:0007 #9 70*

*N2 Byte DSEG:0012 #12 75*

*P\_SORT Far CSEG:0000 #20 71 76*

*P\_SORT\_END Near CSEG:0046 24 #56*

*X Byte DSEG:0015 #14 31 40 44 51*

*Groups & Segments Bit Size Align Combine Class Cref (defined at #)*

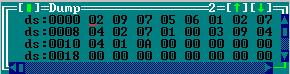
*CSEG 16 006C Para none #17 18*

*DSEG 16 0016 Para none #6 18 65*

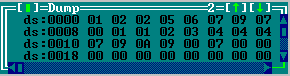
*S 16 0080 Para none STACK #2 18*

**Состояние памяти**

В начале выполнения программы:



В конце выполнения программы:



ВЫВОДЫ:

1. программа состоит из главной процедуры base, которая два раза осуществляет следующие действия: заполняет регистры параметрами и вызывает процедуру обработки массива.
2. С целью обеспечения стабильности программы процедура сохраняет регистры, которые она использует в своей работе. Это обеспечивает программную независимость и самодостаточность модуля программы.