Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе № 2

Дисциплина: Низкоуровневое программирование

Тема: Программирование EDSAC

Вариант: 4

Выполнил студент гр. 3530901/90002 _____Е.К. Борисов

(подпись)

Принял старший преподаватель _____Д.С. Степанов

(подпись)

“_____” _____ 2021 г.

Санкт-Петербург

2021

Цель работы:

1. Разработать программу для EDSAC, реализующую определенную вариантом задания функциональность, и предполагающую загрузчик Initial Orders 1. Массив (массивы) данных и другие параметры (преобразуемое число, длина массива, параметр статистики и пр.) располагаются в памяти по фиксированным адресам.
2. Выделить определенную вариантом задания функциональность в замкнутую (closed) подпрограмму, разработать вызывающую ее тестовую программу. Использовать возможности загрузчика Initial Orders 2. Адрес обрабатываемого массива данных и другие параметры передавать через ячейки памяти с фиксированными адресами.

Вариант 4:

Сортировка вставкой массива чисел in-place.

Задание:

Необходимо смоделировать программу для EDSAC, которая реализует алгоритм сортировки массива вставкой. Отсортированный массив должен находится в тех же ячейках, где находился изначально.

Решение:

Для написания программы для EDSAC, был изначально написан алгоритм на языке python, комментарии в программах initial order 1 и initial order 2 опираются на написанный на python-е алгоритм (рис.7).

Алгоритм сортировки вставками - алгоритм, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов

```
def insertion_sort(nums):
    # Сортировку начинаем со второго элемента, т.к. считается, что первый элемент уже отсортирован
    for i in range(1, len(nums)):
        item_to_insert = nums[i]
        # Сохраняем ссылку на индекс предыдущего элемента
        j = i - 1
        # Элементы отсортированного сегмента перемещаем вперёд, если они больше
        # элемента для вставки
        while j >= 0 and nums[j] > item_to_insert:
            nums[j + 1] = nums[j]
            j -= 1
        # Вставляем элемент
        nums[j + 1] = item_to_insert
```

Рис. 1 Алгоритм сортировки вставкой (python)

Initial order 1:

```
[31]T 148 S [ end ]
[32]T 138 S [ acc clear ]
[33]A 140 S [ load j to put in j for while ]
[34]T 137 S [ j for while ]
[35]A 144 S [ load arr(i) ]
[36]T 136 S [ item_to_insert ]
[37]A 143 S [ load arr(j) to comparison]
[38]S 136 S [ subtraction item_to_insert to comparison ]
[39]G 99 S  [ start while / (arr(j) > item_to_insert) comprasion]
[40]T 138 S [ acc clear ]     [loop3]
[41]A 142 S [ end_arr(j) at the end of while ]
[42]T 135 S [ end_arr(j) at the end of while ]
[43]A 143 S [ arr(j) ]
[44]T 144 S [ arr(j + 1) = arr(j) ]
[45]A 43 S  [ load arr(j)-- ]
[46]S 134 S [ ... ]
[47]T 43 S  [ ... ]
[48]A 44 S  [ put arr(j+1)-- ]
[49]S 134 S [ ... ]
[50]T 44 S  [ ... ]
[51]A 41 S  [ end_arr(j)-- ]
[52]S 134 S [ ... ]
[53]T 41 S  [ ... ]
[54]A 141 S [ counter++ ]
[55]A 134 S [ ... ]
[56]T 141 S [ ... ]
[57]A 137 S [ j_for_while-- ]
[58]S 134 S [ ... ]
[59]U 137 S [ ... ]
[60]G 65 S  [ if j_for_while < 0 ]
[61]T 138 S [ acc clear ]
[62]A 135 S [ load end_arr(j) ]
[63]S 136 S [ subtraction item_to_insert to comprasion ]
[64]E 40 S  [ back to while / arr(j) > item to insert ]
[65]T 138 S [ loop2 , clear acc ]
[66]A 137 S [ there is we we get arr(j+1) after a while (66 - 75)]
[67]G 79 S  [ if j_for_while < 0 => arr(j+1) = arr(0) ]
[68]T 137 S [ acc clear ]
[69]A 81 S  [ ... ]
[70]A 134 S [ ... ]
[71]T 81 S  [ ... ]
[72]A 142 S [ ... ]
[73]A 134 S [ ... ]
[74]T 142 S [ ... ]
[75]A 137 S [ ... ]
[76]S 134 S [ ... ]
[77]T 137 S [ ... ]
[78]E 66 S  [ ... ]
[79]T 138 S [ acc clear ]    [loop4]
[80]A 136 S [ load item_to_insert ]
[81]T 143 S [ arr(j+1) = item_to_insert ]
[82]A 142 S [ back arr(j+1) to arr(0) ]
[83]G 92 S  [ ... ]
[84]T 138 S [ ... ]
[85]A 81 S  [ ... ]
```

Рис. 2.1 Программа на IO 1 (1).

```
[85]A 81 S  [ ... ]
[86]S 134 S [ ... ]
[87]T 81 S  [ ... ]
[88]A 142 S [ ... ]
[89]S 134 S [ ... ]
[90]U 142 S [ ... ]
[91]E 84 S  [ ... ]
[92]T 138 S [ ... ]
[93]A 142 S [ ... ]
[94]A 134 S [ ... ]
[95]T 142 S [ ... ]
[96]A 81 S  [ ... ]
[97]A 134 S [ ... ]
[98]T 81 S  [ ... ]
[99]T 138 S [ loop1 / acc clear ]
[100]A 141 S [ increase end_arr(j) , load arr(j), put arr(j + 1) in while ]
[101]G 116 S  [ ... ]
[102]T 138 S [ acc clear ]  [ loop7 ]
[103]A 41 S  [ ... ]
[104]A 134 S [ ... ]
[105]T 41 S  [ ... ]
[106]A 43 S  [ ... ]
[107]A 134 S [ ... ]
[108]T 43 S  [ ... ]
[109]A 44 S  [ ... ]
[110]A 134 S [ ... ]
[111]T 44 S  [ ... ]
[112]A 141 S [ ... ]
[113]S 134 S [ ... ]
[114]U 141 S [ ... ]
[115]E 102 S  [ ... ]
[116]T 138 S [ acc clear ]   [ loop6 ]
[117]A 141 S [ returning the counter to 0 ]
[118]A 134 S [ ... ]
[119]T 141 S [ ... ]
[120]T 138 S [ acc clear ] [ loop1 ]
[121]A 35 S  [ item_to_insert++ ]
[122]A 134 S [ ... ]
[123]T 35 S  [ ... ]
[124]A 37 S  [ arr(j)++ ]
[125]A 134 S [ ... ]
[126]T 37 S  [ ... ]
[127]A 140 S [ j++ ]
[128]A 134 S [ ... ]
[129]T 140 S [ ... ]
[130]A 139 S [ len(arr)-- ]
[131]S 134 S [ ... ]
[132]U 139 S [ ... ]
[133]E 32 S  [ end ]
[134]P 1 S   [ 1 ]
[135]P 0 S   [ end_arr(j) ]
[136]P 0 S   [ item_to_insert ]
[137]P 0 S   [ j_for_while ]
[138]P 0 S   [ acc clear ]
[139]P 3 S   [ len(arr) ]
[140]P 0 S   [ j ]
[141]P 0 S   [ counter ]
[142]P 0 S   [ counter_second ]
[143]P 15 S  [Array: ]
[144]P 5 S
[145]P 25 S
[146]P 20 S
[147]P 10 S
```

Рис. 2.2 Программа на IO 1 (2).

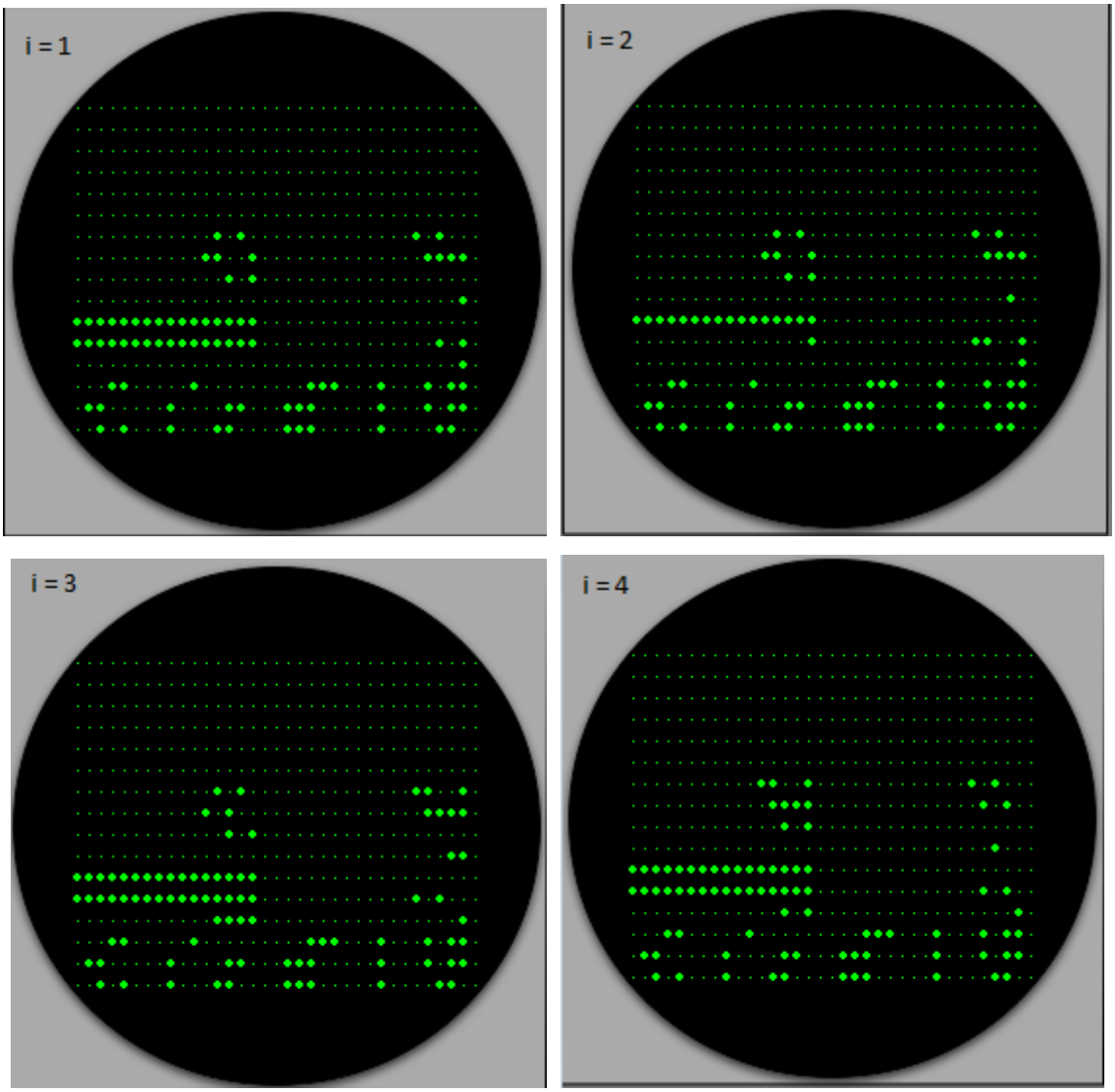

Рис 2.3 Результат работы программы на 1,2,3,4 итерации цикла For соответственно.

Входной массив изначально находится в ячейках 143-147.

На рис. 2.3 мы видим результат работы программы на каждой итерации цикла for. Отобразим изменения положения элементов массива в таблице 1 для наглядности работы алгоритма.

Табл.1 Результаты работы программы.

Ячейки	В начале работы	i = 1	i = 2	i = 3	i = 4
143	P 15 S	P 5 S	P 5 S	P 5 S	P 5 S
144	P 5 S	P 15 S	P 15 S	P 15 S	P 10 S
145	P 25 S	P 25 S	P 25 S	P 20 S	P 15 S
146	P 20 S	P 20 S	P 20 S	P 25 S	P 20 S
147	P 10 S	P 10 S	P 10 S	P 10 S	P 25 S

Как мы видим, по таблице 1 и результатам работы программы, алгоритм сортировки работает корректно. Чтобы добавить элементы массива достаточно в ячейки 148-… добавить новые значения и изменить константу, содержащую длину массива, находящуюся в ячейке 139.

Initial order 2:

Та же программа, используется как замкнутая подпрограмма с тестовой программой, вызывающей её.

```
[41]T 43 K  [start]
[42]G  K [ download address ]
[43] [0]A 3 F [ generating the return instruction code ]
[44] [1]T 104 @ [ recording the return instruction ]
[45] [2]T 109 @ [ acc clear ]
[46] [3]A 111 @ [ load j to put in j for while ]
[47] [4]T 108 @ [ j for while ]
[48] [5]A 115 @ [ load arr(i) ]
[49] [6]T 107 @ [ item_to_insert ]
[50] [7]A 114 @ [ load arr(j) to comparison]
[51] [8]S 107 @ [ subtraction item_to_insert to comparison ]
[52] [9]G 69 @  [ start while / (arr(j) > item_to_insert) comprasion]
[53] [10]T 109 @ [ acc clear ]     [loop3]
[54] [11]A 113 @ [ end_arr(j) at the end of while ]
[55] [12]T 106 @ [ end_arr(j) at the end of while ]
[56] [13]A 114 @ [ arr(j) ]
[57] [14]T 115 @ [ arr(j + 1) = arr(j) ]
[58] [15]A 13 @  [ load arr(j)-- ]
[59] [16]S 105 @ [ ... ]
[60] [17]T 13 @  [ ... ]
[61] [18]A 14 @  [ put arr(j+1)-- ]
[62] [19]S 105 @ [ ... ]
[63] [20]T 14 @  [ ... ]
[64] [21]A 11 @  [ end_arr(j)-- ]
[65] [22]S 105 @ [ ... ]
[66] [23]T 11 @  [ ... ]
[67] [24]A 112 @ [ counter++ ]
[68] [25]A 105 @ [ ... ]
[69] [26]T 112 @ [ ... ]
[70] [27]A 108 @ [ j_for_while-- ]
[71] [28]S 105 @ [ ... ]
[72] [29]U 108 @ [ ... ]
[73] [30]G 35 @  [ if j_for_while < 0 ]
[74] [31]T 109 @ [ acc clear ]
[75] [32]A 106 @ [ load end_arr(j) ]
[76] [33]S 107 @ [ subtraction item_to_insert to comprasion ]
[77] [34]E 10 @  [ back to while / arr(j) > item to insert ]
[78] [35]T 109 @ [ loop2 , clear acc ]
[79] [36]A 108 @ [ there is we we get arr(j+1) after a while (66 - 75)]
[80] [37]G 49 @  [ if j_for_while < 0 => arr(j+1) = arr(0) ]
[81] [38]T 109 @ [ acc clear ]
[82] [39]A 51 @  [ ... ]
[83] [40]A 105 @ [ ... ]
[84] [41]T 51 @  [ ... ]
[85] [42]A 113 @ [ ... ]
[86] [43]A 105 @ [ ... ]
[87] [44]T 113 @ [ ... ]
[88] [45]A 108 @ [ ... ]
[89] [46]S 105 @ [ ... ]
[90] [47]T 108 @ [ ... ]
[91] [48]E 36 @  [ ... ]
[92] [49]T 109 @ [ acc clear ]    [loop4]
[93] [50]A 107 @ [ load item_to_insert ]
[94] [51]T 114 @ [ arr(j+1) = item_to_insert ]
[95] [52]A 113 @ [ back arr(j+1) to arr(0) ]
[96] [53]G 62 @   [ ... ]
[97] [54]T 119 @  [ clear acc ]
[98] [55]A 51 @   [ ... ]
[99] [56]S 105 @  [ ... ]
[100] [57]T 51 @  [ ... ]
[101] [58]A 113 @ [ ... ]
[102] [59]S 105 @ [ ... ]
[103] [60]U 113 @ [ ... ]
```

Рис. 3.1 Программа на IO 1 (1).

```
[104] [61]E 54 @  [ ... ]
[105] [62]T 109 @ [ clear acc ]
[106] [63]A 113 @ [ ... ]
[107] [64]A 105 @ [ ... ]
[108] [65]T 113 @ [ ... ]
[109] [66]A 51 @  [ ... ]
[110] [67]A 105 @ [ ... ]
[111] [68]T 51 @  [ ... ]
[112] [69]T 109 @ [ loop1 / acc clear ]
[113] [70]A 112 @ [ increase end_arr(j) , load arr(j), put arr(j + 1) in while ]
[114] [71]G 86 @  [ ... ]
[115] [72]T 109 @ [ acc clear ]  [ loop7 ]
[116] [73]A 11 @  [ ... ]
[117] [74]A 105 @ [ ... ]
[118] [75]T 11 @  [ ... ]
[119] [76]A 13 @  [ ... ]
[120] [77]A 105 @ [ ... ]
[121] [78]T 13 @  [ ... ]
[122] [79]A 14 @  [ ... ]
[123] [80]A 105 @ [ ... ]
[124] [81]T 14 @  [ ... ]
[125] [82]A 112 @ [ ... ]
[126] [83]S 105 @ [ ... ]
[127] [84]U 112 @ [ ... ]
[128] [85]E 72 @  [ ... ]
[129] [86]T 109 @ [ acc clear ]   [ loop6 ]
[130] [87]A 112 @ [ returning the counter to 0 ]
[131] [88]A 105 @ [ ... ]
[132] [89]T 112 @ [ ... ]
[133] [90]T 109 @ [ acc clear ]
[134] [91]A 5 @  [ item_to_insert++ ]
[135] [92]A 105 @ [ ... ]
[136] [93]T 5 @  [ ... ]
[137] [94]A 7 @  [ arr(j)++ ]
[138] [95]A 105 @ [ ... ]
[139] [96]T 7 @  [ ... ]
[140] [97]A 111 @ [ j++ ]
[141] [98]A 105 @ [ ... ]
[142] [99]T 111 @ [ ... ]
[143] [100]A 110 @ [ len(arr)-- ]
[144] [101]S 105 @ [ ... ]
[145] [102]U 110 @ [ ... ]
[146] [103]E 2 @  [ end ]
[147] [104]E 0 F   [ returning from a subroutine ]
[148] [105]P 1 F   [ 1 ]
[149] [106]P 0 F   [ end_arr(j) ]
[150] [107]P 0 F   [ item_to_insert ]
[151] [108]P 0 F   [ j_for_while ]
[152] [109]P 0 F   [ acc clear ]
[153] [110]P 3 F   [ len(arr) ]
[154] [111]P 0 F   [ j ]
[155] [112]P 0 F   [ counter ]
[156] [113]P 0 F   [ counter_second ]
[157] [114]P 25 F  [Array: ]
[158] [115]P 20 F
[159] [116]P 15 F
[160] [117]P 10 F
[161] [118]P 5 F
[162] [119]G   K
[163] [0]  A 0 @   [ calling a subroutine ]
[164] [1]G 43 F
[165] [2]Z 0 F   [stop]
[166] [3]EZ PF   [end]
```

Рис. 3.2 Программа на IO 2 (2).

Рис. 3.3 Результат работы программы

В начале работы программы значения массива равны:

157 – 25

158 – 20

159 – 15

160 – 10

161 – 5

В конце работы программы по рис. 3.3 мы видим, что элементы массива находятся в нужном порядке:

157 – 5

158 – 10

159 – 15

160 – 20

161 – 25

Алгоритм работы программы:

1. Обновляем значение j для итерации в цикле while, загружаем число для вставки.
2. Проверяем arr[j] > item_to_insert , Если же число для вставки больше, то переходим к пункту **6**. Иначе заходим в цикл while и копируем arr[j] в arr[j+1] , далее уменьшаем j_for_while и повторяем эти действия пока не встретим элемент меньший числа для вставки или не дойдём до конца списка. Также в цикле накапливаем counter, для возвращения итерируемых значений в прежнее состояние.
3. Увеличиваем значение ячейки “T arr(0) S” до тех пор пока не дойдем до значения равному arr[j+1].
4. Устанавливаем число для вставки в arr[j+1].
5. Возвращаем значение ячейки “T arr(0) S” в прежнее состояние для последующих итераций.
6. Возвращаем значения итерируемых значений в цикле while до начального состояния +1 для следующей итерации цикла for. Обнуляем counter.
7. Передвигаем значение arr(i) и arr(j) на одну ячейку вперед.
8. Увеличиваем j и уменьшаем ячейку хранящую изначальный размер массива на 1.
9. Проверяем если значение ячейки размера массива стало меньше 0, то заканчиваем работу программы, иначе переходим к следующей итерации цикла for.