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: 14

Выполнил студент гр. 3530901/90002 _____С.А. Федоров
(подпись)

Принял преподаватель _____ Д.С. Степанов
(подпись)

“_____” _____ 2021 г.

Санкт-Петербург
2021

Цели работы:

1. Разработать программу для EDSAC, реализующую определенную вариантом задания функциональность, и предполагающую загрузчик Initial Orders 1. Массив (массивы) данных и другие параметры (количество итераций, счетчик результата 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.

Начальные данные для 14 варианта

Определение наиболее часто встречающегося в массиве значения.

1. Постановка задачи и алгоритм решения

Требуется смоделировать программу для EDSAC, которая определит наиболее часто встречающееся значение в массиве (по заданию надо найти одно значение, поэтому если таких значений будет несколько, то возьмем наибольшее по величине).

Для реализации воспользуемся сортировкой методом пузырька, а затем в отсортированным массиве пройдемся по всем элементам и найдем наиболее часто встречающееся значение.

2. Initial Orders 1

Программа для EDSAC, которая реализует поиск наиболее часто встречающегося значения в массиве, использующая загрузчик Initial Orders 1

```
[31] T 176 [указываем конец программы] S
[32] E 40[startOfCycle] S
[33] [save_first] A 162[x1] S
[34] [save_second] S 163[x2] S
[35] T 0 S
[36] [save_copy_x1] A 162[x1] S
[37] [save_paste_x1] T 162[x1] S
[38] [save_copy_x2] A 163[x2] S
[39] [save_paste_x2] T 163[x2] S
[40] [startOfCycle]T 0 S [очистим аккумулятор]
[41] A 170[totalIter] S     [проверим, закончились ли все круги итераций]
[42] G 100[endOfSorting] S  [если закончились, то выходим из программы]
[43] T 0 S
[44] A 169[iter] S [осуществим проверку текущего индекса]
[45] E 65[currentIter] S [если индекс >= 0, то продолжим текущий круг итераций]
[46] [newIter]T 0 S           [\Совершим переход на новый круг итераций]
[47] A 33[save_first] S       [|]
[48] T 66[first] S            [|]
[49] A 34[save_second] S      [|]
[50] T 67[second] S           [|]
[51] A 36[save_copy_x1] S     [|]
[52] T 70[copy_x1] S          [|]
[53] A 38[save_copy_x2] S     [|]
[54] T 72[copy_x2] S          [|]
[55] A 37[save_paste_x1] S    [|]
[56] T 73[paste_x1] S         [|]
[57] A 39[save_paste_x2] S    [|]
[58] T 75[paste_x2] S         [|]
```

Рис.1. Программа при IO 1 часть 1.

```
[59] A 170[totalIter] S       [Меняем значение текущего этапа цикла]
[60] S 168[index] S           [|]
[61] U 170[totalIter] S       [|]
[62] T 169[iter] S            [Меняем значение текущей итерации]
[63] A 168[index] S           [Возвращаемся в начало цикла]
[64] E 40[startOfCycle] S     [/]
[65] [currentIter]T 0 S [Очищаем аккумулятор]
[66] [first]A 162[x1] S [запишем в аккумулятор значение x(i)]
[67] [second]S 163[x2] S  [осуществляем вычитание x(i) - x(i+1)]
[68] G 76[noChange] S [if (x(i) - x(i+1)) < 0, то перейдем в ячейку noChange]
[69] [change]T 0 S           [\поменяем элементы местами]
[70] [copy x1]A 162[x1] S    [|]
[71] T 167[save] S           [|]
[72] [copy x2]A 163[x2] S    [|]
[73] [paste x1]T 162[x1] S   [|]
[74] A 167[save] S           [|]
[75] [paste x2]T 163[x2] S   [/]
[76] [noChange]T 0 S [Очищаем аккумулятор]
[77] A 66[first] S [помещаем значение start ячейки(команду)]
[78] A 168[index] S [добавляем к команде 1 -> индекс изменяется на +1]
[79] T 66[first] S [заменяем значение в start на значение след элемента массива]
[80] A 67[second] S   [\Изменяем значение в ячейке second ]
[81] A 168[index] S   [|]
[82] T 67[second] S   [/]
[83] A 70[copy x1] S     [\Изменим значения ячеек 69 - 74]
[84] A 168[index] S      [|]
[85] T 70[copy x1] S     [|]
[86] A 72[copy x2] S     [|]
[87] A 168[index] S      [|]
[88] T 72[copy x2] S     [|]
[89] A 73[paste x1] S    [|]
[90] A 168[index] S      [|]
[91] T 73[paste x1] S    [|]
[92] A 75[paste x2] S    [|]
[93] A 168[index] S      [|]
[94] T 75[paste x2] S    [/]
[95] A 169[iter] S            [\Изменяем значение iter]
[96] S 168[index] S           [|]
[97] T 169[iter] S            [/]
[98] A 168[index] S               [\Возвращаемся в начало цикла]
[99] E 40[startOfCycle] S         [/]
[100] [endOfSorting]T 0 S    [конец работы сортировки]
[101] A 162[x1] S   [запишем в результат значение x1 ]
[102] T 173[result] S
[103] [newResIter] T 0 S [новая итерация при вычислении результата]
[104] [element]A 162[x1] S [для хранения ссылки на следующее значение]
[105] T 175[saveElement] S
[106] A 174[resIter] S
[107] G 154[endCycle] S [если индекс < 0, то завершаем работу цикла поиска]
[108] T 0 S
[109] [first]A 162[x1] S [запишем в аккумулятор значение x(i)]
[110] [second]S 163[x2] S  [осуществляем вычитание x(i) - x(i+1)]
[111] G 129[newNumber] S [(x(i) - x(i+1)) < 0 --> число изменилось ]
[112] [noNewNumber]T 0 S          [если число не менятеся, то ]
[113] A 172[currentCounter] S     [|]
[114] A 168[index] S              [|]
```

Рис.2. Программа при IO 1 часть 2.

```
[114] A 168[index] S              [|]
[115] T 172[currentCounter] S     [изменяем значение currentCounter]
[116] A 109[first] S              [|]
[117] A 168[index] S              [|]
[118] T 109[first] S              [изменяем first]
[119] A 104[element] S            [|]
[120] A 168[index] S              [|]
[121] T 104[element] S            [изменяем element]
[122] A 110[second] S             [|]
[123] A 168[index] S              [|]
[124] T 110[second] S             [изменяем second]
[125] A 174[resIter] S            [|]
[126] S 168[index] S              [|]
[127] T 174[resIter] S            [изменим значение resIter]
[128] E 103[newResIter] S         [/возвращаемся на новую итерацию]
[129] [newNumber] T 0 S        [число изменилось, значит]
[130] A 172[currentCounter] S
[131] S 171[resCounter] S
[132] G 138[noChangeRes] S [не появилось значение, которое встречается чаще]
[133] T 0 S
[134] A 175[saveElement] S       [\появилось -> изменим значение в result]
[135] T 173[result] S            [|]
[136] A 172[currentCounter] S    [измение значение resCounter]
[137] T 171[resCounter] S        [/]
[138] [noChangeRes] T 0 S
[139] A 168[index] S                   [\изменим значение currentCounter на 1]
[140] T 172[currentCounter] S          [|]
[141] A 174[resIter] S                 [|]
[142] S 168[index] S                   [|]
[143] T 174[resIter] S                 [изменим значение resIter]
[144] A 109[first] S                   [|]
[145] A 168[index] S                   [|]
[146] T 109[first] S                   [изменяем first]
[147] A 104[element] S                 [|]
[148] A 168[index] S                   [|]
[149] T 104[element] S                 [изменяем element]
[150] A 110[second] S                  [|]
[151] A 168[index] S                   [|]
[152] T 110[second] S                  [изменяем second]
[153] E 103[newResIter] S              [/возвращаемся на новую итерацию]
[154] [endCycle]T 0 S  [конец поиска, сделаем проверку краевого случая]
[155] A 172[currentCounter] S
[156] S 171[resCounter] S
[157] G 161[end] S  [если текущее значение счетчика <= resCounter --> end]
[158] T 0 S         [иначе делаем переприсвоение]
[159] A 175[saveElement] S
[160] T 173[result] S
[161] [end] T 0 S   [конец работы программы, result в 173 ячейке]
[162] [x1]P 15 S             [\ Проинициализируем массив ]
[163] [x2]P 40 S             [|]
[164] [x3]P 10 S             [|]
[165] [x4]P 15 S             [|]
[166] [x5]P 50 S             [/]
[167] [save]P 0 S [переменная для перемещения элементов]
[168] [index]P 1 S [переменная для изменения индексов]
[169] [iter]P 3 S [переменная для хранения текущей итерации]
[170] [totalIter]P 3 S [переменная для хранения количества итераций]
[171] [resCounter]P 1 S [счетчик для результата]
[172] [currentCounter]P 1 S [счетчик для текущего количества значений]
[173] [result]P 0 S [переменная для храрения результата]
[174] [resIter]P 3 S [для хранения количества итераций при вычислении result]
[175] [saveElement]P 0 S [Храним текущий элемент]
```

Рис.3. Программа при IO 1 часть 3.

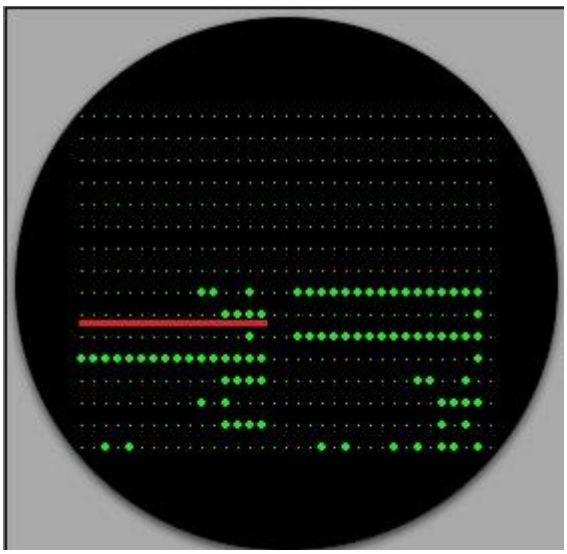

Рис.4. Результат работы программы при IO 1.

```
WORD 173   Order = P 15 S   Integer 173S = 30   Fraction 172L = 0.00045776379
```

Рис.5. Значение в 173 ячейке после завершения работы.

Исходя из рис.4 – рис.5 можно сделать вывод, что программы работает правильно, так как в изначальном массиве данных все значения встречаются по одному разу, кроме значения P 15 S, которое и вывелось в результат.

Входной массив данных хранится в 162 – 166 ячейках (132 – 136 строки), а в последующих ячейках хранятся промежуточные переменные. Программа универсальна, так как работает для массива любого размера, но из-за добавления новых элементов в массив все промежуточные значения изменят свои адреса ячеек (потребуется менять значения номеров этих ячеек во всей программе).

Результат хранится в 173 ячейке (143 строка), куда он будет перезаписан по окончанию работы программы, то есть для повторного запуска программы потребуется совершить сброс (clear) EDSAC.

Сортировка методом пузырька находится в 31-100 ячейках (1 – 70 строки). В 33- 39 ячейках хранятся команды для реализации перехода на новый круг итераций (внешний цикл). В ячейках 69 – 75 происходит перестановка, остальные строки нужны для изменения индексов (внутренний цикл).

В 101-161 ячейках происходит поиск наиболее часто встречающегося в массиве значения при помощи итерации по отсортированному массиву и сравнения соседних элементов. В 134-137 ячейках происходит присвоение результату нового

значения, если значение будет встречаться чаще или же будет больше по величине, чем предыдущее.

3. Initial Orders 2

Такая же программа, как и в пункте IO 1, но уже используется как замкнутая подпрограмма с тестовой программой, которая вызывает её.

```
[41] T 43[start] K
[42] G  K [фиксируем адрес]
[43] [0] A 3 F [инструкция возврата]
[44] [1] T 132 @ [запись инструкции возврата]
[45] [2] E 10[startOfCycle] @
[46] [3] [save_first] A 133[x1] @
[47] [4] [save_second] S 134[x2] @
[48] [5] T 0 F
[49] [6] [save_copy_x1] A 133[x1] @
[50] [7] [save_paste_x1] T 133[x1] @
[51] [8] [save_copy_x2] A 134[x2] @
[52] [9] [save_paste_x2] T 134[x2] @
[53] [10] [startOfCycle]T 0 F [очистим аккумулятор]
[54] [11] A 141[totalIter] @     [проверим, закончились ли все круги итераций]
[55] [12] G 70[endOfSorting] @   [если закончились, то выходим из программы]
[56] [13] T 0 F
[57] [14] A 140[iter] @ [осуществим проверку текущего индекса]
[58] [15] E 35[currentIter] @ [если индекс >= 0, то продолжим текущий круг итераций]
[59] [16] [newIter]T 0 F           [\Совершим переход на новый круг итераций]
[60] [17] A 3[save_first] @        [|]
[61] [18] T 36[first] @            [|]
[62] [19] A 4[save_second] @       [|]
[63] [20] T 37[second] @           [|]
[64] [21] A 6[save_copy_x1] @      [|]
[65] [22] T 40[copy_x1] @          [|]
[66] [23] A 8[save_copy_x2] @      [|]
[67] [24] T 42[copy_x2] @          [|]
[68] [25] A 7[save_paste_x1] @     [|]
[69] [26] T 43[paste_x1] @         [|]
[70] [27] A 9[save_paste_x2] @     [|]
[71] [28] T 45[paste_x2] @         [|]
[72] [29] A 141[totalIter] @       [Меняем значение текущего этапа цикла]
[73] [30] S 139[index] @           [|]
[74] [31] U 141[totalIter] @       [|]
[75] [32] T 140[iter] @            [Меняем значение текущей итерации]
[76] [33] A 139[index] @           [Возвращаемся в начало цикла]
[77] [34] E 10[startOfCycle] @     [/]
[78] [35] [currentIter]T 0 F [Очищаем аккумулятор]
[79] [36] [first]A 133[x1] @ [запишем в аккумулятор значение x(i)]
[80] [37] [second]S 134[x2] @  [осуществляем вычитание x(i) - x(i+1)]
[81] [38] G 46[noChange] @ [if (x(i) - x(i+1)) < 0, то перейдем в ячейку noChange]
[82] [39] [change]T 0 F          [\поменяем элементы местами]
[83] [40] [copy_x1]A 133[x1] @   [|]
[84] [41] T 138[save] @          [|]
[85] [42] [copy_x2]A 134[x2] @   [|]
[86] [43] [paste_x1]T 133[x1] @  [|]
[87] [44] A 138[save] @          [|]
[88] [45] [paste_x2]T 134[x2] @  [/]
[89] [46] [noChange]T 0 F [Очищаем аккумулятор]
```

Рис.5. Программа при IO 2 часть 1.

```
50  [90] [47] A 36[first] @ [помещаем значение start ячейки(команду)]
51  [91] [48] A 139[index] @ [добавляем к команде 1 -> индекс изменяется на +1]
52  [92] [49] T 36[first] @ [заменяем значение в start на значение след элемента массива]
53  [93] [50] A 37[second] @   [\Изменяем значение в ячейке second ]
54  [94] [51] A 139[index] @   [|]
55  [95] [52] T 37[second] @   [/]
56  [96] [53] A 40[copy x1] @      [\Изменим значения ячеек 69 - 74]
57  [97] [54] A 139[index] @       [|]
58  [98] [55] T 40[copy x1] @      [|]
59  [99] [56] A 42[copy x2] @      [|]
60  [100] [57] A 139[index] @      [|]
61  [101] [58] T 42[copy x2] @     [|]
62  [102] [59] A 43[paste x1] @    [|]
63  [103] [60] A 139[index] @      [|]
64  [104] [61] T 43[paste x1] @    [|]
65  [105] [62] A 45[paste x2] @    [|]
66  [106] [63] A 139[index] @      [|]
67  [107] [64] T 45[paste x2] @    [/]
68  [108] [65] A 140[iter] @           [\Изменяем значение iter]
69  [109] [66] S 139[index] @          [|]
70  [110] [67] T 140[iter] @           [/]
71  [111] [68] A 139[index] @              [\Возвращаемся в начало цикла]
72  [112] [69] E 10[startOfCycle] @        [/]
73  [113] [70] [endOfSorting]T 0 F    [конец работы сортировки]
74  [114] [71] A 133[x1] @   [запишем в результат значение x1 ]
75  [115] [72] T 144[result] @
76  [116] [73] [newResIter] T 0 F [новая итерация при вычислении результата]
77  [117] [74] [element]A 133[x1] @ [для хранения ссылки на следующее значение]
78  [118] [75] T 146[saveElement] @
79  [119] [76] A 145[resIter] @
80  [120] [77] G 124[endCycle] @ [если индекс < 0, то завершаем работу цикла поиска]
81  [121] [78] T 0 F
82  [122] [79] [first]A 133[x1] @ [запишем в аккумулятор значение x(i)]
83  [123] [80] [second]S 134[x2] @  [осуществляем вычитание x(i) - x(i+1)]
84  [124] [81] G 99[newNumber] @ [(x(i) - x(i+1)) < 0 --> число изменилось ]
85  [125] [82] [noNewNumber]T 0 F          [если число не менятеся, то ]
86  [126] [83] A 143[currentCounter] @     [|]
87  [127] [84] A 139[index] @              [|]
88  [128] [85] T 143[currentCounter] @     [изменяем значение currentCounter]
89  [129] [86] A 79[first] @               [|]
90  [130] [87] A 139[index] @              [|]
91  [131] [88] T 79[first] @               [изменяем first]
92  [132] [89] A 74[element] @             [|]
93  [133] [90] A 139[index] @              [|]
94  [134] [91] T 74[element] @             [изменяем element]
95  [135] [92] A 80[second] @              [|]
96  [136] [93] A 139[index] @              [|]
97  [137] [94] T 80[second] @              [изменяем second]
98  [138] [95] A 145[resIter] @            [|]
99  [139] [96] S 139[index] @              [|]
100 [140] [97] T 145[resIter] @            [изменим значение resIter]
101 [141] [98] E 73[newResIter] @          [/возвращаемся на новую итерацию]
```

Рис.6. Программа при IO 2 часть 2.

```
[142] [99] [newNumber] T 0 F        [число изменилось, значит]
[143] [100] A 143[currentCounter] @
[144] [101] S 142[resCounter] @
[145] [102] G 138[noChangeRes] @ [не появилось значение, которое встречается чаще]
[146] [103] T 0 F
[147] [104] A 146[saveElement] @       [\появилось -> изменим значение в result]
[148] [105] T 144[result] @            [|]
[149] [106] A 143[currentCounter] @    [измение значение resCounter]
[150] [107] T 142[resCounter] @        [/]
[151] [108] [noChangeRes] T 0 F
[152] [109] A 139[index] @                  [\изменим значение currentCounter на 1]
[153] [110] T 143[currentCounter] @         [|]
[154] [111] A 145[resIter] @                [|]
[155] [112] S 139[index] @                  [|]
[156] [113] T 145[resIter] @                [изменим значение resIter]
[157] [114] A 79[first] @                   [|]
[158] [115] A 139[index] @                  [|]
[159] [116] T 79[first] @                   [изменяем first]
[160] [117] A 74[element] @                 [|]
[161] [118] A 139[index] @                  [|]
[162] [119] T 74[element] @                 [изменяем element]
[163] [120] A 80[second] @                  [|]
[164] [121] A 139[index] @                  [|]
[165] [122] T 80[second] @                  [изменяем second]
[166] [123] E 73[newResIter] @              [/возвращаемся на новую итерацию]
[167] [124] [endCycle]T 0 F  [конец поиска, сделаем проверку краевого случая]
[168] [125] A 143[currentCounter] @
[169] [126] S 142[resCounter] @
[170] [127] G 131[end] @  [если текущее значение счечтика <= resCounter --> end]
[171] [128] T 0 F         [иначе делаем переприсвоение]
[172] [129] A 146[saveElement] @
[173] [130] T 144[result] @
[174] [131] [end] T 0 F [конец работы программы, результат в 187 ячейке]
[175] [132] E 0 F [возврат из подпрограммы]
[176] [133] [x1]P 15 F             [\ Проинициализируем массив ]
[177] [134] [x2]P 40 F             [|]
[178] [135] [x3]P 10 F             [|]
[179] [136] [x4]P 15 F             [|]
[180] [137] [x5]P 50 F             [/]
[181] [138] [save]P 0 F [переменная для перемещения элементов]
[182] [139] [index]P 1 F [переменная для изменения индексов]
[183] [140] [iter]P 3 F [переменная для хранения текущей итерации]
[184] [141] [totalIter]P 3 F [переменная для хранения количества итераций]
[185] [142] [resCounter]P 1 F [счетчик для результата]
[186] [143] [currentCounter]P 1 F [счетчик для текущего количества значений]
[187] [144] [result]P 0 F [переменная для хрaрения результата]
[188] [145] [resIter]P 3 F [для хранения количества итераций при вычислении result]
[189] [146] [saveElement]P 0 F [Храним текущий элемент]
[190] [147] G   K [фиксируем адрес]
[191] [1] A 0 @  [\вызов подпрограммы]
[192] [2] G 43 F [/]
[193] [3] Z 0 F  [останов]
[194] [4] EZ PF  [завершение]
```

Рис.7. Программа при IO 2 часть 3.

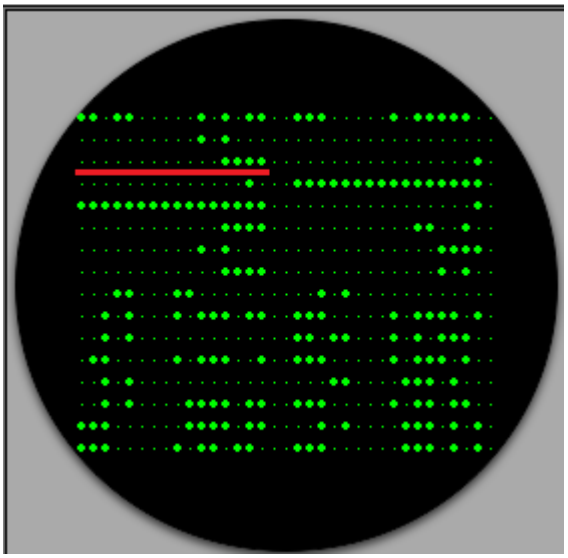

Рис.8. Результат работы программы при IO 2.

WORD 187 Order = P 15 F Integer 187F = 30 Fraction 186D = 0.00045776379

Рис.9. Значение в 187 ячейке после завершения работы.

Исходя из рис.8 – рис.9 можно сделать вывод, что в 187 ячейке оказалось значение P 15 F, что соответствует правильному ответу при заданному массиве данных.

Входной массив данных хранится в 176 – 180 ячейках (136 – 140 строки), а в последующих ячейках хранятся промежуточные переменные. Программа универсальна, так как работает для массива любого размера, но из-за добавления новых элементов в массив все промежуточные значения изменят свои адреса ячеек (потребуется менять значения номеров этих ячеек во всей программе).

Результат хранится в 187 ячейке (147 строка), куда он будет перезаписан по окончанию работы программы, то есть для повторного запуска программы потребуется совершить сброс (clear) EDSAC.

4. Подробный алгоритм для решения поставленной задачи

1) Проверка индекса внешнего цикла сортировки (если он будет меньше 0, значит цикл завершился – завершилась сортировка)
2) Проверка индекса внутреннего цикла, если он будет меньше 0, то надо осуществить переход на новый круг итераций (уменьшить значение индекса внешнего цикла и обновить команды для возвращения к начальными

индексами)

3) Если индекс внутреннего цикла оказался не меньше нуля, тогда осуществим проверку текущих элементов массива(x[i] и x[i+1]):
    - Если x[i] - x[i+1] окажется меньше нуля, значит элементы уже отсортированы и их не надо менять местами, то есть уменьшаем индекс внутреннего цикла и меняем индексы для используемых команд
    - Если окажется больше или равен нулю, значит надо осуществить перестановку элементов местами и осуществить все операции для случая, когда разность меньше нуля
4) Начинаем итерироваться по отсортированному массиву:
    - Если разность x[i] и x[i+1] оказывается меньше нуля, значит значения отличаются и надо осуществить проверку на обновление результата (сравнить cuurentCounter и resCounter – счетчики для количества текущих значений и значений результата) и осуществить обновление результата, если потребуется (cuurentCounter>= resCounter). Также надо изменить значение cuurentCounter на 1 и обновить индексы команд.
    - Если разность больше или равна нулю, тогда к cuurentCounter прибавляется 1 и обновляются индексы для команд
5) Осуществить проверку краевого случая (если cuurentCounter> = resCounter. То следует обновить результат)

5. Вывод

В данной лабораторной работе я познакомился с принципом работы EDSAC с использованием различных видов загрузки данных (IO 1 и IO 2), в частности была написана программы для поиска наиболее часто встречающегося в массиве значения.

Список использованных источников

http://kspt.icc.spbstu.ru/media/files/2020/lowlevelprog/lab2.pdf

https://www.dcs.warwick.ac.uk/~edsac/

https://www.dcs.warwick.ac.uk/~edsac/Software/EdsacTG.pdf