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-e 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

|  |
| --- |
|  |
| Рис. 1 Алгоритм сортировки вставкой (python) |

**Initial order 1:**

|  |
| --- |
|  |
| Рис. 2.1 Программа на IO 1 (1). |

|  |
| --- |
|  |
| 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.

|  |
| --- |
|  |
| Рис. 3.1 Программа на IO 1 (1). |

|  |
| --- |
|  |
| 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.