

## **Отчет по лабораторной работе**

по дисциплине «Структуры и алгоритмы обработки данных»

на тему:

«Методы сортировки»

Выполнил:

Студент группы БСТ1902

Игнатов В.С.

Вариант №5

Москва 2020

## Оглавление

Задания на лабораторную работу .....	3
Ход работы .....	3
Задание 1 .....	3
Задание 2 .....	3
Задание 3 .....	5
Сортировка выбором .....	5
Сортировка вставками .....	5
Сортировка обменом .....	5
Сортировка Шелла .....	6
Быстрая сортировка .....	6
Пирамидальная сортировка .....	7
Встроенная сортировка .....	7
Турнирная сортировка .....	8
Результат выполнения сортировок .....	9

## Задания на лабораторную работу

1. Вывести в консоль «Hello World!»
2. Написать генератор случайных матриц(многомерных), который принимает опциональные параметры `m`, `n`, `min_limit`, `max_limit`, где `m` и `n` указывают размер матрицы, а `min_lim` и `max_lim` - минимальное и максимальное значение для генерируемого числа. По умолчанию при отсутствии параметров принимать следующие значения:
3. Реализовать методы сортировки строк числовой матрицы в соответствии с заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки. Испытания проводить на сгенерированных матрицах.
4. Создать публичный репозиторий на GitHub.

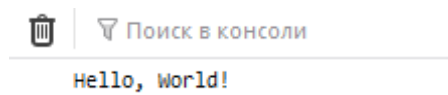
## Ход работы

### Задание 1

Выведем в консоль сообщение:

```
console.log("Hello, World!");
```

Результат команды:



### Задание 2

Реализуем функцию генерации матрицы:

```
function randomMatrix(m = 50, n = 50, minLim = -250, maxLim = 1005) {  
  let matrix = [];  
  for (let i = 1; i <= m; i++) {  
    let matrixInner = [];  
  
    for (let j = 1; j <= n; j++) {  
      let elem = Math.floor(minLim + Math.random() * (maxLim + 1 - minLim))  
    };  
    matrixInner.push(elem);  
  }  
}
```

```

        matrix.push(matrixInner);
    }

    return matrix
}

```

Вызов функции матрицы:

```

let matrix = randomMatrix();
console.log(matrix);

```

Результат вывода:

```

(50) [...]
▶ 0: Array(50) [ 415, -50, 984, ... ]
▶ 1: Array(50) [ -20, 485, 730, ... ]
▶ 2: Array(50) [ 883, 888, 563, ... ]
▶ 3: Array(50) [ 279, 522, -170, ... ]
▶ 4: Array(50) [ -240, 871, 161, ... ]
▶ 5: Array(50) [ 348, 112, -8, ... ]
▶ 6: Array(50) [ 544, 560, 31, ... ]
▶ 7: Array(50) [ 123, 896, -78, ... ]
▶ 8: Array(50) [ 723, 642, -154, ... ]
▶ 9: Array(50) [ 616, 314, -121, ... ]
▶ 10: Array(50) [ -90, 423, 282, ... ]
▶ 11: Array(50) [ -166, 631, 230, ... ]
▶ 12: Array(50) [ 866, 849, 775, ... ]
▶ 13: Array(50) [ 979, 14, 237, ... ]
▶ 14: Array(50) [ 13, 512, 634, ... ]
▶ 15: Array(50) [ 461, 878, 604, ... ]
▶ 16: Array(50) [ -18, 979, 133, ... ]
▶ 17: Array(50) [ 415, 573, 54, ... ]
▶ 18: Array(50) [ 920, 716, 599, ... ]
▶ 19: Array(50) [ 987, 251, 646, ... ]
▶ 20: Array(50) [ 97, 777, 403, ... ]
▶ 21: Array(50) [ 677, 322, 303, ... ]
▶ 22: Array(50) [ 732, 318, 918, ... ]
▶ 23: Array(50) [ 693, -187, 747, ... ]
▶ 24: Array(50) [ 393, 702, 711, ... ]
▶ 25: Array(50) [ 115, -172, 293, ... ]
▶ 26: Array(50) [ -197, 836, -10, ... ]
▶ 27: Array(50) [ 487, 650, -221, ... ]
▶ 28: Array(50) [ 681, 406, 292, ... ]
▶ 29: Array(50) [ 623, 526, 779, ... ]
▶ 30: Array(50) [ 689, 172, -211, ... ]
▶ 31: Array(50) [ 810, 300, -11, ... ]
▶ 32: Array(50) [ 957, 810, 420, ... ]
▶ 33: Array(50) [ 578, 641, 319, ... ]
▶ 34: Array(50) [ 565, 570, 374, ... ]
▶ 35: Array(50) [ 635, 224, 327, ... ]
▶ 36: Array(50) [ 707, 47, 763, ... ]
▶ 37: Array(50) [ 754, 779, 542, ... ]
▶ 38: Array(50) [ -146, 666, 442, ... ]
▶ 39: Array(50) [ 483, 158, -116, ... ]
▶ 40: Array(50) [ 35, 110, 475, ... ]
▶ 41: Array(50) [ 858, 1002, -80, ... ]
▶ 42: Array(50) [ 409, 749, -59, ... ]
▶ 43: Array(50) [ 109, 867, 763, ... ]
▶ 44: Array(50) [ 238, 727, -58, ... ]
▶ 45: Array(50) [ 940, 852, 972, ... ]
▶ 46: Array(50) [ -87, 736, 255, ... ]
▶ 47: Array(50) [ 495, 437, 93, ... ]
▶ 48: Array(50) [ 176, 297, 218, ... ]
▶ 49: Array(50) [ 731, 429, 825, ... ]
    length: 50
    <prototype>: Array []

```

### Задание 3

#### Сортировка выбором

```
function selectSort(arr) {
  for (let i = 0; i < arr.length - 1; i++) {
    let MinElemIndex = i;

    for (let j = i + 1; j < arr.length; j++) {
      if (arr[MinElemIndex] > arr[j]) {
        MinElemIndex = j;
      }
    }

    if (MinElemIndex !== i) {
      [arr[i], arr[MinElemIndex]] = [arr[MinElemIndex], arr[i]];
    }
  }
  return arr;
};
```

#### Сортировка вставками

```
function insertSort(arr) {
  for (let i = 1; i < arr.length; i++) {
    const current = arr[i];
    let j = i;

    while (j > 0 && arr[j - 1] > current) {
      arr[j] = arr[j - 1];
      j--;
    }

    arr[j] = current;
  }

  return arr;
};
```

#### Сортировка обменом

```
function swapSort(arr) {
  for (let i = 0; i < arr.length - 1; i++) {
    for (let j = 0; j < arr.length - 1 - i; j++) {
      if (arr[j] > arr[j + 1]) {
        let swap = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = swap;
      }
    }
  }
}
```

```

    }
  }
}

return arr;
}

```

## Сортировка Шелла

```

function ShellSort(arr) {
  let gap = Math.floor(arr.length / 2);

  while (gap >= 1) {
    for (let i = gap; i < arr.length; i++) {
      const current = arr[i];
      let j = i;

      while (j > 0 && arr[j - gap] > current) {
        arr[j] = arr[j - gap];
        j -= gap;
      }

      arr[j] = current;
    }

    gap = Math.floor(gap / 2);
  }

  return arr;
};

```

## Быстрая сортировка

```

function quickSort(arr) {
  if (arr.length < 2) return arr;

  let less = [],
      more = [],
      pivot = arr[0];

  for (let i = 1; i < arr.length; i++) {
    if (arr[i] < pivot) {
      less.push(arr[i])
    } else {
      more.push(arr[i]);
    }
  };

  return [...less, pivot, ...more];
}

```

```
    return quickSort(less).concat(pivot, quickSort(more));  
}
```

## Пирамидальная сортировка

```
function HeapSort(arr) {  
    let n = arr.length,  
        i = Math.floor(n / 2),  
        j, k, t;  
  
    while (true) {  
        if (i > 0) t = arr[--i];  
        else {  
            n--;  
  
            if (n == 0) return arr;  
            t = arr[n];  
            arr[n] = arr[0];  
        }  
  
        j = i;  
        k = j * 2 + 1;  
  
        while (k < n) {  
            if (k + 1 < n && arr[k + 1] > arr[k]) k++;  
            if (arr[k] > t) {  
                arr[j] = arr[k];  
                j = k;  
                k = j * 2 + 1;  
            }  
            else break;  
        }  
  
        arr[j] = t;  
    }  
}
```

## Встроенная сортировка

```
function includeSort(arr) {  
    return arr.sort((a, b) => a - b);  
}
```

## Турнирная сортировка

```
function TournamentSort(arr) {
  let three = Array(2 * (arr.length + arr.length % 2));
  let index = three.length - arr.length + arr.length % 2;

  for (let i = index; i < three.length; i++) {
    tree[i] = i - index, array[i - index];
  }

  for (let j = 0; j < arr.length; j++) {
    let n = arr.length;
    index = three.length - arr.length + arr.length % 2;

    while (index > -1) {
      n = (n + 1) / 2;

      for (let i = 0; i < n; i++) {
        let iCopy = Math.max(index + i * 2, 1);

        if (three[iCopy] != null && three[iCopy + 1] != null) {
          if (three[iCopy][1] < three[iCopy + 1][1]) {
            three[iCopy / 2] = three[iCopy];
          } else {
            three[iCopy / 2] = three[iCopy + 1]
          }
        } else {
          three[iCopy / 2] = (three[iCopy] != null) ? three[iCopy] : th
ree[iCopy + 1];
        }
      }

      index -= n;
    }

    index = three[0][0]
    let x = three[0][1]
    arr[j] = x;
    three[three.length - arr.length - arr.length % 2 + index] = null;
  }
}
```



## Результат выполнения сортировок

Сортировка выбором: 4мс - таймер закончился
Сортировка вставками: 2мс - таймер закончился
Сортировка обменом: 1мс - таймер закончился
Сортировка Шелла: 2мс - таймер закончился
Быстрая сортировка: 3мс - таймер закончился
Пирамидальная сортировка: 1мс - таймер закончился
Турнирная сортировка: 2мс - таймер закончился
Встроенная сортировка: 0мс - таймер закончился

## Вывод

Я изучил методы сортировки и реализовал их на практике.