Разное

Списывание :(

- Проверки на плагиат проводятся периодически.
- В проверке участвуют как работы этого года, так и прошлых лет.
- Списанные лабораторные работы оцениваются в 0 баллов.

- «Герои» первой проверки
  - Бурдунин, Ильин
  - Волков Г., Никулина
  - Исаев (списал практически один в один лабораторную прошлого года)
  - Динь Вьет Ань (списал один в один лабораторную прошлого года)

Сортировка

- В конце 60-х годов произошло … интенсивное развитие теории сортировки. Появившиеся позже алгоритмы во многом являлись вариациями уже известных методов. [wiki]

- Типичный вывод студента кафедры ИУ7 по седьмой лабораторной работе :)

  «При большом количестве чисел реализация mysort может увеличить производительность программы, или, как минимум, сравняться с быстротой функции qsort.»

Типичные ошибки в измерениях, 1

```
clock_t start = clock();
for (int j = 0; j < 1000; j++)
{
    rc = create_array(argv[1], &d_1, &de_1);
    if (rc != OK)
    {
        return rc;
    }

    mysort(d_1, de_1 - d_1, sizeof(int), compare);
    free(d_1);
}
clock_t finish = clock();
```

Типичные ошибки в измерениях, 2

```
for (int i = 1; i < 11; i++)
{
    printf("%d Elements\n", i * 100);
    rc = create_array(argv[i], &d_1, &de_1);
    if (rc != OK)
    {
        return rc;
    }
    clock_gettime(CLOCK_REALTIME, &ts);
    mysort(d_1, de_1 - d_1, sizeof(int), compare);
    clock_gettime(CLOCK_REALTIME, &ts2);
    free(d_1);
    printf("Nanosec for mysort: %lld\n", 1000000000 * (ts2.tv_sec - ts.tv_sec) + (ts2.tv_nsec - ts.tv_nsec));

    rc = create_array(argv[1], &d_12, &de_12);
    if (rc != OK)
    {
        return rc;
    }
    clock_gettime(CLOCK_REALTIME, &ts);
    qsort(d_12, de_12 - d_12, sizeof(int), compare);
    clock_gettime(CLOCK_REALTIME, &ts2);
    free(d_12);
    printf("Nanosec for qsort: %lld\n", 1000000000 * (ts2.tv_sec - ts.tv_sec) + (ts2.tv_nsec - ts.tv_nsec));
}
```

Динамические матрицы (способ 4), 1

```
// Найдите ошибки, если они есть
double** allocate_matrix(int n, int m)
{
    double **matrix = malloc(n * sizeof(double*) + m * sizeof(double));
    if (matrix == NULL)
        free(matrix);

    matrix[0] = matrix + n;
    for (int i = 1; i < n; i++)
        matrix[i] = matrix[0] + m * i;

    return matrix;
}
```

Динамические матрицы (способ 4), 2

```
// Ошибки выделены красным
double** allocate_matrix(int n, int m)
{
    double **matrix = malloc(n * sizeof(double*) + m * sizeof(double));
    if (matrix == NULL)
        free(matrix);

    matrix[0] = matrix + n;
    for (int i = 1; i < n; i++)
        matrix[i] = matrix[0] + m * i;

    return matrix;
}
```

Динамические матрицы (способ 4), 3

```
// Найдите ошибки, если они есть
double** allocate_matrix(int n, int m)
{
    double **matrix = malloc(n * sizeof(double*) + n * m * sizeof(double));
    if (!matrix)
        return NULL;

    for (int i = 0; i < n; i++)
        matrix[i] = matrix[n] + m * i;

    return matrix;
}
```

Динамические матрицы (способ 4), 4

```
// Ошибки выделены красным
double** allocate_matrix(int n, int m)
{
    double **matrix = malloc(n * sizeof(double*) + n * m * sizeof(double));
    if (!matrix)
        return NULL;

    for (int i = 0; i < n; i++)
        matrix[i] = matrix[n] + m * i;

    return matrix;
}
```