Массивы (всего 10 баллов):

Эти задачи нужно оформить в соответствии с правилами оформления:
http://style.vdi.mipt.ru/CodeStyle.html и youtube.com/watch?v=NSNvfr_KpDc
и прислать мне на почту vladimir.biryukov@phystech.edu.
В некоторых задачах потребуются входные файлы, которые можно найти по адресу:
github.com/v-biryukov/cs_mipt_faki/tree/master/term1/seminar04_array/advanced_problems/inputs

1 Простая работа с файлами. fprintf и fscanf

Для простейшей работы с файлами мы будем использовать функции `fscanf` и `fprintf`, которые работают аналогично функциям `printf` и `scanf`. Подробней файлы будем проходить позднее. Библиотека для работы с файлами - `stdio.h` (та же самая, что и для `printf` и `scanf`).

Пример записи в файл:

```
#include <stdio.h>
int main()
{
        // Открываем файл под названием result.txt
        // "w" = write - открываем файл на запись
        // Так как файл открывается на запись, то необязательно чтобы он существовал
        FILE* fout = fopen("hello.txt", "w");

        fprintf(fout, "Hello world of files\n");

        // Закрываем файл
        fclose(fout);
}
```

Пример чтения чисел из файла:

```
#include <stdio.h>
int main()
{
        // Открываем файл под названием numbers.txt
        // "r" = read - открываем файл на чтение
        FILE* fin = fopen("numbers.txt", "r");

        int n;
        int a[100];
        fscanf(fin, "%d", &n);
        for (int i = 0; i < n; i++)
                fscanf(fin, "%d", &a[i]);

        fclose(fin);
}
```

- **Задача 1. Чтение/запись (0.5 балла):** Создайте файл `input.txt` в котором будут храниться входные числа в следующем виде:

  ```
  15
  54 32 53 64 1 21 77 4 6 81 34 10 92 17 42
  ```

  Сначала идёт число n - количевство чисел последовательности, а потом эти n чисел. Ваша задача - написать программу, которая будет считывать эти числа, возводить их в квадрат и записывать результат в файл `output.txt`.

2 Сортировки

```
#include <stdio.h>
void print_array(int lo, int hi, int* arr)
{
        for (int i = lo; i < hi; i++)
                printf("%d ", arr[i]);
        printf("\n");
}

// Сортировка выбором, которая сортирует числа массива с индексами от lo до hi-1
// Выбор таких индексов удобен для реализации реккурсивных алгоритмов сортировок
void selection_sort(int lo, int hi, int* arr)
{
        for (int j = lo; j < hi; j++)
        {
                // Находим индекс минимального элемента на отрезке [j:hi-1]
                int min_index = j;
                for (int i = j+1; i < hi; i++)
                        if (arr[i] < arr[min_index])
                                min_index = i;

                // Меняем местами элемент номер j и минимальный элемент
                int temp = arr[j];
                arr[j] = arr[min_index];
                arr[min_index] = temp;
        }
}

int main()
{
        int a[15] = {45, 73, 34, 82, 31, 1, 64, 54, 47, 73, 62, 11, 16, 7, 26};
        selection_sort(0, 15, a);
        print_array(0, 15, a);
}
```

- **Задача 2. Чтение/запись сортировки** (0.5 **балла):** Считайте числа из файла `input.txt` и запишите их в файл `output.txt` в отсортированом виде.
- **Задача 3. Сортировка пузырьком (1 балл):** Напишите функцию `void bubble_sort(int lo, int hi, int* arr)`, реализующую алгоритм сортировки пузырьком. О сортировке пузырьком можно посмотреть, например, тут:
  youtube.com/watch?v=oqpICiM165I
  youtube.com/watch?v=xli_FI7CuzA
- **Задача 4. Быстрая сортировка (2 балла):** Напишите функцию `void quick_sort(int lo, int hi, int* arr)`, реализующую алгоритм быстрой сортировки (советую использовать разбиение Ломуто, так как оно немного проще). О быстрой сортировке можно посмотреть, например, тут:
  Википедия
  www.youtube.com/watch?v=MZaf_9IZCrc
- **Задача 5. Сортировка большого количества чисел (1 балл):** В файле `numbers.txt` хранятся 300000 случайных чисел. Отсортируйте их с помощью 3-х различных методов сортировки (выбором, пузырьком и быстрой). Результат запишите в файл `sorted_numbers.txt`.

3 Двумерные массивы

```c
#include <stdio.h>
// Зададим константу MAX = 200 - максимальный возможный размер матрицы
#define MAX 200

// В отличии от одномерного массива, в двумерном массиве при передаче
// в функцию обязательно нужно указывать размер ( количество столбцов = MAX )
// %g - печатает вещественные числа также как и %f, но без нулей на конце
void print_array(int n, float arr[MAX][MAX])
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            printf("%5g ", arr[i][j]);
        printf("\n");
    }
}

// Суммируем 2 квадратные матрицы A и B размера n на n и записываем результат в C
void sum(int n, float A[MAX][MAX], float B[MAX][MAX], float C[MAX][MAX])
{
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            C[i][j] = A[i][j] + B[i][j];
}
int main()
{
    // Создаём массивы вещественных чисел ( с запасом )
    float a[MAX][MAX] = {{7, 7, 2}, {1, 8, 3}, {2, 1, 6}};
    float b[MAX][MAX] = {{5, 2, 9}, {-4, 2, 11}, {7, 1, -5}};

    // Мы создали матрицы с 200 на 200 ( с запасом ), но будем использовать только
    // маленькую часть 3 на 3. В будущем мы научимся как создавать матрицы нужного
    // размера во время выполнения программы

    printf("a = \n");
    print_array(3, a);

    printf("b = \n");
    print_array(3, b);

    float c[MAX][MAX];
    sum(3, a, b, c);
    printf("a + b = \n");
    print_array(3, c);
}
```

- **Задача 6. Умножение на число (0.5 балла):** Написать функцию `void multiply_by_number(int n, float A[MAX][MAX], float x)`, которая умножает квадратную матрицу A (n на n) на число x.
- **Задача 7. Присвоение (0.5 балла):** Написать функцию `void assign(int n, float A[MAX][MAX], float B[MAX][MAX])`, которая присваивает элементам матрицы A соответствующие элементы матрицы B (A = B).
- **Задача 8. Умножение матриц (1 балл):** Написать функцию `void multiply(int n, float A[MAX][MAX], float B[MAX][MAX], float C[MAX][MAX])`, которая перемножает матрицы A и B (строка на столбец), а результат записывает в матрицу C. Формула: $C_{ij} = \sum_k A_{ik} * B_{kj}$.

  Проверьте ваш код на следующих тестах:

$$\begin{pmatrix} 7 & 7 & 2 \\ 1 & 8 & 3 \\ 2 & 1 & 6 \end{pmatrix} * \begin{pmatrix} 5 & 2 & 9 \\ -4 & 2 & 11 \\ 7 & 1 & -5 \end{pmatrix} = \begin{pmatrix} 21 & 30 & 130 \\ -6 & 21 & 82 \\ 48 & 12 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 2 & 9 \\ -4 & 2 & 11 \\ 7 & 1 & -5 \end{pmatrix} * \begin{pmatrix} 7 & 7 & 2 \\ 1 & 8 & 3 \\ 2 & 1 & 6 \end{pmatrix} = \begin{pmatrix} 55 & 60 & 70 \\ -4 & -1 & 64 \\ 40 & 52 & -13 \end{pmatrix}$$

$$\begin{pmatrix} 7 & 7 & 2 \\ 1 & 8 & 3 \\ 2 & 1 & 6 \end{pmatrix} * \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 7 & 7 \\ 3 & 8 & 1 \\ 6 & 1 & 2 \end{pmatrix}$$

  В файлах `mat_A10.txt` и `mat_B10.txt` лежат матрицы 10 на 10. считайте эти матрицы с помощью `fscanf`, перемножьте (A на B) и запишите результат в другой файл с помощью `fprintf`. В результате должно получиться:

$$\begin{pmatrix} 259 & -15 & 237 & 257 & 231 & 67 & 237 & -64 & 152 & 363 \\ 555 & 233 & 539 & 188 & 356 & 325 & 423 & -47 & 123 & 387 \\ 497 & 512 & 572 & 95 & 619 & 155 & 414 & 207 & 203 & 217 \\ 455 & 280 & 675 & 354 & 664 & 346 & 483 & 177 & 168 & 404 \\ 264 & 182 & 272 & 290 & 474 & -33 & 234 & 99 & 379 & 156 \\ 272 & 180 & 469 & 286 & 326 & 282 & 325 & 215 & 195 & 231 \\ 421 & 363 & 475 & 506 & 359 & 481 & 468 & 101 & 325 & 328 \\ 384 & 218 & 567 & 395 & 475 & 488 & 361 & 168 & 291 & 298 \\ 387 & 297 & 480 & 170 & 318 & 423 & 483 & 10 & -17 & 406 \\ 193 & 241 & 486 & 38 & 403 & 146 & 286 & 326 & 212 & 172 \end{pmatrix}$$

- **Задача 9. Матрица в степени (1 балл):** Написать функцию `void power(int n, float A[MAX][MAX], float C[MAX][MAX], int k)`, которая вычисляет A^k, т.е. возводит матрицу A в k-ю степень, а результат записывает в матрицу C. Используйте функции `multiply` и `assign`. Псевдокод функции:

```
float B[MAX][MAX]
B = A
C = A
for ( k - 1 раз )
{
        C = A*B
        B = C
}
```

  Проверьте ваш код на следующих тестах:

$$\begin{pmatrix} 7 & 7 & 2 \\ 1 & 8 & 3 \\ 2 & 1 & 6 \end{pmatrix}^4 = \begin{pmatrix} 7116 & 15654 & 9549 \\ 4002 & 8955 & 6135 \\ 3369 & 6165 & 4350 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}^{50} = \begin{pmatrix} 525456 & 396655 & 299426 \\ 696081 & 525456 & 396655 \\ 396655 & 299426 & 226030 \end{pmatrix}$$

- **Задача 10. Метод Гаусса (2 балла):** Написать программу, которая бы решала линейную систему уравнений $Ax = b$ методом Гаусса. Главная функция этой программы должна иметь вид:
`void solve_linear_system(int n, float A[MAX][MAX], float b[], float x[])`.
Программа должна считывать матрицу A и столбец b из файла и записывать результат решения x в новый файл.

```c
#include <stdio.h>
#define MAX 200

// Возможно понадобится вспомогательная функция для перестановки строк матрицы A
void swap_rows(int n, float A[MAX][MAX], int k, int m)
{
    // Ваш код
}

void solve_linear_system(int n, float A[MAX][MAX], float b[], float x[])
{
    // Ваш код
}

int main()
{
    int n;
    float A[MAX][MAX];
    float b[MAX];
    float x[MAX];

    // Считываем n, A и b из файла

    solve_linear_system(n, A, b, x);

    // Записываем x в новый файл
}
```

Проверьте вашу программу на следующих тестах:

1. Следующая система:
$$\begin{cases} x_1 + x_2 - x_3 = 9 \\ x_2 + 3x_3 = 3 \\ -x_1 - 2x_3 = 2 \end{cases}$$
Файл для считывания должен выглядеть следующим образом:
```
 3
 1 1 -1
 0 1  3
-1 0 -2
9
3
2
```
Решение этой системы: $x = (\frac{2}{3}, 7, -\frac{4}{3}) \approx (0.67, 7, -1.33)$
2. Система из файла `system1.txt`. Решение в файле `x1.txt`.
3. Система из файла `system2.txt`. Решение в файле `x2.txt`.