

Короче, студент, я тебя обучаю (rofl) и в благородство играть не буду: выполнишь для меня пару заданий – и мы в расчёте. Заодно посмотрим, как быстро у тебя башка после выходных прояснится. А по твоей теме я постараюсь разузнать. **** его знает, на кой ляд тебе этот автомат по ТРПО сдался, но я в чужие дела не лезу, хочешь получить, значит есть за что.

Вводные данные:

```
struct exam {  
    string nameOfExam;  
    int mark;  
    int studentId;  
    double averageMarkLabs;  
};
```

Рисунок 1 – Структура для работы

TRPO	10	1	10.0
TRPO	9	2	9.0
TRPO	8	3	8.0
TRPO	7	4	7.0
TRPO	6	5	6.0
TRPO	5	6	5.0
TRPO	4	7	4.0

Рисунок 2 – Содержимое файла для работы

```
exam* readExamFromFile(string fileName, int& size);
```

Рисунок 3 – Функция, которая считывает из файла массив структур

1. Пишем функцию, которая будет открывать файл и читать из него. После открытия файла, проверяем открылся ли он.

```
ifstream file(fileName);  
if (!file.is_open()) {  
    return nullptr;  
}
```

ИМЯ ФАЙЛА

Проверка на открытие файла

Если файл не открылся, выход из функции

2. Необходимо определить куда мы будем записывать данные и счётчик, которые посчитает кол-во прочитанных строк.

```
size <= 0 ? size = 10 : size = size;  
exam* exams = new exam[size];  
int numberOfLines = 0;
```

Определяем размер буферного массива

Создаём новый массив

3. Читаем файл, пока не конец файла. Когда достигли конец файла, ходим из цикла.

```
while (!file.eof()) {  
    if (numberOfLines == size) { ... }  
    file >> exams[numberOfLines].nameOfExam;  
    file >> exams[numberOfLines].mark;  
    file >> exams[numberOfLines].studentId;  
    file >> exams[numberOfLines].averageMarkLabs;  
    numberOfLines++;  
}
```

Читаем до конца файла

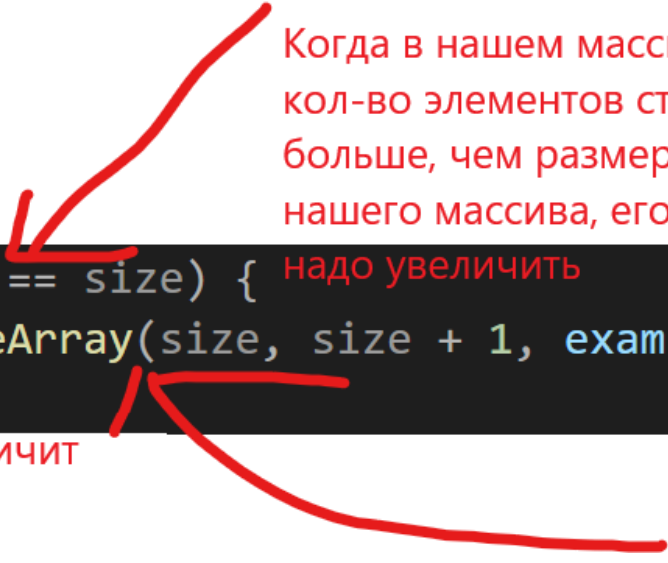
На следующем скрине)

Сам процесс считывания из файла

Увеличиваем кол-во прочитанных строк

3.5 Во время определения размера массива, мы точно не могли знать кол-во элементов массива. Поэтому надо будет увеличивать размер массива.

Когда кол-во прочитанных строк стало равно размеру массива (массив ведь индексируется с 0?), увеличим его размер на некоторую величину, может быть любое число > 0 .



```
if (numberOfLines == size) {  
    exams = resizeArray(size, size + 1, exams);  
}
```

Когда в нашем массиве кол-во элементов стало больше, чем размер нашего массива, его надо увеличить

Функция, которая увеличит наш массив

Функция ресайза массива, долго описывать. Она просто создаёт новый массив с необходимым количеством элементов (ещё переписывает элементы из старого в новый) и возвращает указатель на него. Главное не забыть очистить старый (утечки памяти плохо). Ниже реализация в лоб. Можно ещё было копирование в отдельную функцию (но сейчас ночь, а спать хочется)).

```
exam* resizeArray(int &oldSize, int newSize, exam* exams) {  
    if (oldSize == newSize) {  
        return exams;  
    }  
    exam* newArray = new exam[newSize];  
    oldSize = newSize < oldSize ? newSize : oldSize;  
    for (int i = 0; i < oldSize; i++) {  
        newArray[i] = exams[i];  
    }  
    oldSize = newSize;  
    delete[] exams;  
    return newArray;  
}
```

4. После того, как мы прочитали весь файл, необходимо сделать пункт 3.5. Ведь размер мог оказаться больше, фактического размера массива. И конечно же не забываем закрыть файл.

```
exams = resizeArray(size, numberOfLines - 1, exams);  
file.close();
```

5. Теперь надо записать файл то, что мы получили
Функция, которая это делает выглядит так.

```
void writeExamInFile(string fileName  
    , exam* exams  
    , int size  
    , int openMode);
```

Первый параметр имя файла. Второй параметр то, что мы туда будем записывать, в нашем случае массив структур. Третий параметр размер структуры. 4 необязательный, чтобы был(будем открывать для дозаписи или очищать старый файл).

Ну там, ещё код внутри есть.

```
ofstream file(fileName, openMode);  
if (!file.is_open()) {  
    return;  
}  
for (int i = 0; i < size; i++) {  
    file << exams[i].nameOfExam << " "  
        << exams[i].mark << " "  
        << exams[i].studentId << " "  
        << exams[i].averageMarkLabs << endl;  
}
```

Открыли файл, в цикле записали. Записываем поля в такой же последовательности, как потом будем читать.

Удивительный факт номер 2, если поменять file на cout в цикле, то получится вывод в консоль. Bay!!!



```
file.close();
```

6. Вызываем всё это.

```
int size = 1;  
exam* exams = readExamFromFile("text.txt", size);  
writeExamInFile("text1.txt", exams, size, ios_base::out);
```

Чтобы не писать заполнение файла какими-то данными, записал до этого в файл ручками. В итоге получилось два одинаковых файла.

Задание два вообще лёгкое, пишется в три строчки. Серьёзно в три. Не вру. Можно в 8, но и 3 хватит.

Просто меняем чтение функции, на чтение массива строк. Пару функций перегружаем. Над получен производим разные манипуляции и записываем в файл. Но вот примеры чтения строк из файлов, может быть поможет

```
ofstream file(fileName, openMode);
if (!file.is_open()) {
    return;
}
for (int i = 0; i < size; i++) {
    file << data[i]<<endl;
}
file.close();
```

```
ifstream file(fileName);
if (!file.is_open()) {
    return nullptr;
}
size <= 0 ? size = 10 : size = size;
string* data = new string[size];
int numberOfLines = 0;
while (!file.eof()) {
    if (numberOfLines == size) {
        data = resizeArray(size, size + 1, data);
    }
    (file,data[numberOfLines]);
    numberOfLines++;
}
data = resizeArray(size, numberOfLines - 1, data);
file.close();
return data;
```