МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет ИТМО

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Дополнительная работа №2

По дисциплине «Аппаратное обеспечение вычислительных систем»

Выполнил студент группы №M3113

Балакирева Виктория Валерьевна

Проверил

Шевчик Софья Владимировна

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург

2024

Условие

Создать программу на языке C, которая считывает из файла целые числа.

Необходимо написать ассемблерную вставку, осуществляющую поиск одинаковых чисел в массиве. Вставка должна быть вынесена в отдельную функцию. Использовать глобальные переменные для передачи данных в указанную функию запрещено.

Найденные значения необходимо сохранить в файле вывода.

Входные данные

Первая строка входного файла INPUT.TXT содержит одно целое число N ($1 \leq N \leq 100$).

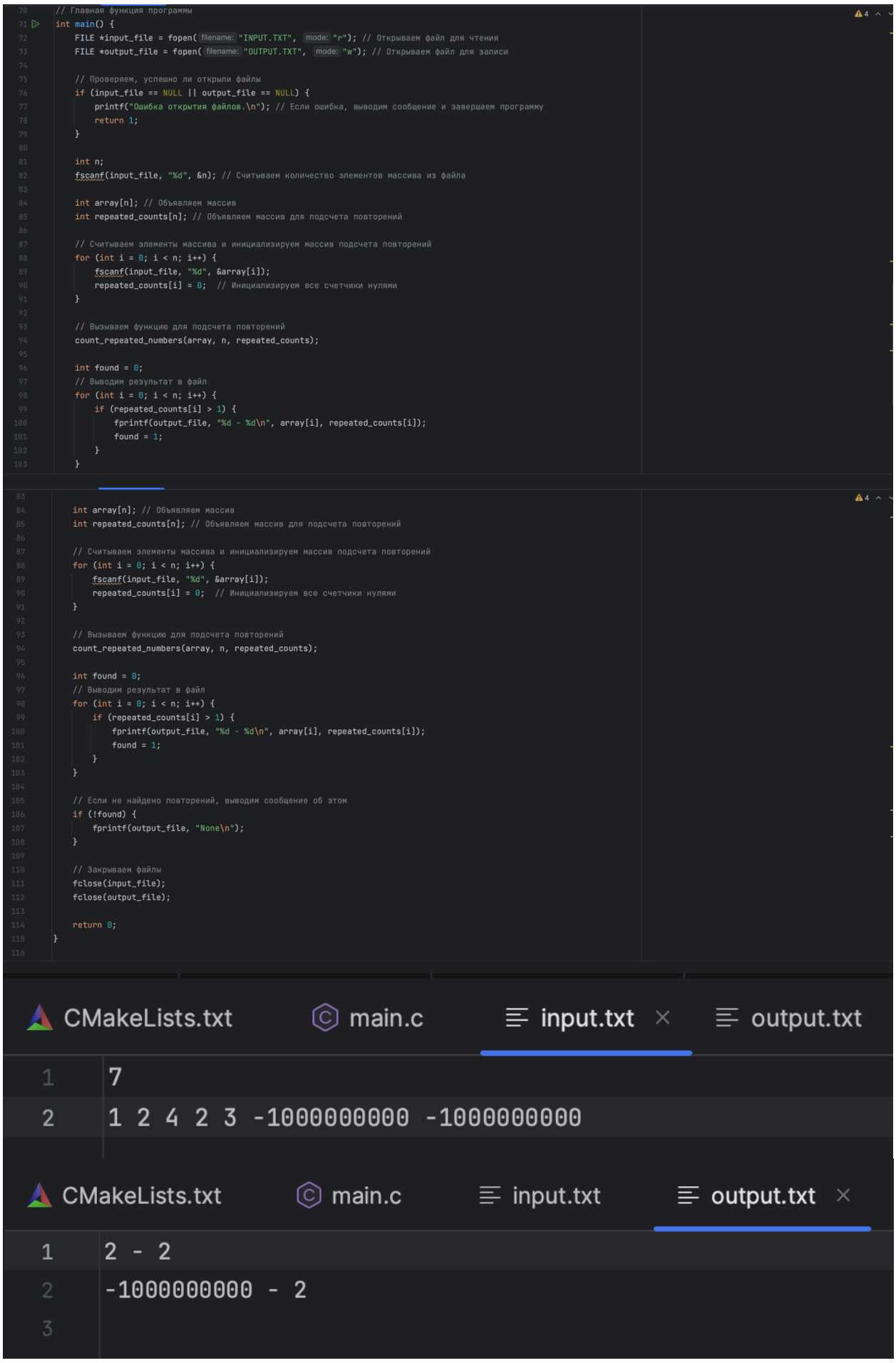
В следующих N строках содержатся числа, по модулю не превышающие 10^9.

Выходные данные

Вывести строки вида A - B, где - A - повторяющееся число, а B - количество повторений этого числа.

Если повторяющихся чисел нет, вывести None.

```c
#include <stdio.h>

// Функция для подсчета повторяющихся чисел в массиве
void count_repeated_numbers(int *array, int n, int *repeated_counts) {
    __asm__(
            "mov x0, #0\n"            // Устанавливаем i = 0

            "1:\n"                     // Внешний цикл
            "cmp x0, %2\n"            // Проверяем, если (i >= n)
            "bge 4f\n"                 // Если да, выходим из цикла

            "ldr w1, [%0, x0, lsl #2]\n" // Текущее значение = array[i]
            "ldr w2, [%1, x0, lsl #2]\n" // Количество повторений = repeated_counts[i]
            "cmp w2, #0\n"            // Проверяем, если (repeated_counts[i] != 0)
            "bne 3f\n"                 // Если уже посчитано, переходим к следующему i

            "mov w3, #1\n"            // Устанавливаем счетчик в 1
            "add x4, x0, #1\n"       // Устанавливаем j = i + 1

            "2:\n"                     // Внутренний цикл
            "cmp x4, %2\n"            // Проверяем, если (j >= n)
            "bge 3f\n"                 // Если да, выходим из цикла

            "ldr w5, [%0, x4, lsl #2]\n" // Значение для сравнения = array[j]
            "cmp w1, w5\n"           // Проверяем, если (текущее значение == значение для сравнения)
            "bne 5f\n"                 // Если не равны, продолжаем

            "add w3, w3, #1\n"       // Увеличиваем счетчик
            "mov w5, #-1\n"             //помечаем элементы -1 чтобы потом их не учитывать
            "str w5, [%1, x4, lsl #2]\n" // Помечаем повторение как посчитанное, присваивая -1
            "5:\n"
            "add x4, x4, #1\n"       // Увеличиваем j
            "b 2b\n"

            "3:\n"
            "cmp w3, #1\n"            // Проверяем, если (счетчик > 1)
            "ble 6f\n"                 // Если меньше или равно 1, пропускаем

            "str w3, [%1, x0, lsl #2]\n" // Записываем количество повторений в repeated_counts[i]

            // Отмечаем остальные дубликаты как 0, чтобы избежать повторного вывода
            "cmp w3, #1\n"
            "ble 6f\n"
            "mov x4, x0\n"            // Устанавливаем j = i
            "add x4, x4, #1\n"        // Увеличиваем j на 1

            "7:\n"
            "cmp x4, %2\n"            // Проверяем, если (j >= n)
            "bge 6f\n"                 // Если да, выходим из цикла
            "ldr w5, [%0, x4, lsl #2]\n" // Значение для сравнения = array[j]
            "cmp w1, w5\n"            // Проверяем, если (текущее значение == значение для сравнения)
            "bne 8f\n"                 // Если не равны, пропускаем
            "mov w5, #0\n"
            "str w5, [%1, x4, lsl #2]\n" // Помечаем как выведенное, присваивая 0
            "8:\n"
            "add x4, x4, #1\n"        // Увеличиваем j
            "b 7b\n"

            "6:\n"
            "add x0, x0, #1\n"       // Увеличиваем i
            "b 1b\n"

            "4:\n"                     // Конец
            :
            : "r"(array), "r"(repeated_counts), "r"(n)
            : "x0", "x2", "x4"
            );
}
```



```c
// Главная функция программы
int main() {
    FILE *input_file = fopen( filename: "INPUT.TXT",  mode: "r"); // Открываем файл для чтения
    FILE *output_file = fopen( filename: "OUTPUT.TXT",  mode: "w"); // Открываем файл для записи

    // Проверяем, успешно ли открыли файлы
    if (input_file == NULL || output_file == NULL) {
        printf("Ошибка открытия файлов.\n"); // Если ошибка, выводим сообщение и завершаем программу
        return 1;
    }

    int n;
    fscanf(input_file, "%d", &n); // Считываем количество элементов массива из файла

    int array[n]; // Объявляем массив
    int repeated_counts[n]; // Объявляем массив для подсчета повторений

    // Считываем элементы массива и инициализируем массив подсчета повторений
    for (int i = 0; i < n; i++) {
        fscanf(input_file, "%d", &array[i]);
        repeated_counts[i] = 0;  // Инициализируем все счетчики нулями
    }

    // Вызываем функцию для подсчета повторений
    count_repeated_numbers(array, n, repeated_counts);

    int found = 0;
    // Выводим результат в файл
    for (int i = 0; i < n; i++) {
        if (repeated_counts[i] > 1) {
            fprintf(output_file, "%d - %d\n", array[i], repeated_counts[i]);
            found = 1;
        }
    }
```

```c
    int array[n]; // Объявляем массив
    int repeated_counts[n]; // Объявляем массив для подсчета повторений

    // Считываем элементы массива и инициализируем массив подсчета повторений
    for (int i = 0; i < n; i++) {
        fscanf(input_file, "%d", &array[i]);
        repeated_counts[i] = 0;  // Инициализируем все счетчики нулями
    }

    // Вызываем функцию для подсчета повторений
    count_repeated_numbers(array, n, repeated_counts);

    int found = 0;
    // Выводим результат в файл
    for (int i = 0; i < n; i++) {
        if (repeated_counts[i] > 1) {
            fprintf(output_file, "%d - %d\n", array[i], repeated_counts[i]);
            found = 1;
        }
    }

    // Если не найдено повторений, выводим сообщение об этом
    if (!found) {
        fprintf(output_file, "None\n");
    }

    // Закрываем файлы
    fclose(input_file);
    fclose(output_file);

    return 0;
}
```

CMakeLists.txt | main.c | input.txt | output.txt

```
1  7
2  1 2 4 2 3 -1000000000 -1000000000
```

CMakeLists.txt | main.c | input.txt | output.txt

```
1  2 - 2
2  -1000000000 - 2
3
```

Функция count_repeated_numbers написана на языке C с использованием ассемблерной вставки, которая выполняет поиск повторяющихся чисел в массиве. Результаты записываются в массив repeated_counts, где каждый элемент хранит количество повторений соответствующего числа из массива array.

Описание работы ассемблерного кода:

1. **Инициализация:**
   - mov x0, #0 устанавливает начальный индекс i равным 0.
2. **Внешний цикл (итерация по массиву):**
   - Метка 1: обозначает начало внешнего цикла.
   - cmp x0, %2 сравнивает текущий индекс i с размером массива n.
   - bge 4f если i больше или равно n, переход к метке 4, что означает конец обработки.
3. **Проверка повторяющихся элементов:**
   - ldr w1, [%0, x0, lsl #2] загружает текущее значение массива array[i] в регистр w1.
   - ldr w2, [%1, x0, lsl #2] загружает значение из repeated_counts[i] в регистр w2.
   - cmp w2, #0 проверяет, если repeated_counts[i] не равно 0.
   - bne 3f если repeated_counts[i] уже посчитано, переход к следующему элементу массива (метка 3).
4. **Подсчет повторений текущего элемента:**
   - mov w3, #1 устанавливает начальный счетчик повторений w3 равным 1.
   - add x4, x0, #1 устанавливает начальный индекс внутреннего цикла j = i + 1.
5. **Внутренний цикл (поиск повторяющихся элементов):**
   - Метка 2: обозначает начало внутреннего цикла.
   - cmp x4, %2 сравнивает текущий индекс j с размером массива n.
   - bge 3f если j больше или равно n, переход к метке 3, что означает конец внутреннего цикла.
   - ldr w5, [%0, x4, lsl #2] загружает значение array[j] в регистр w5.
   - cmp w1, w5 сравнивает значения array[i] и array[j].
   - bne 5f если значения не равны, переход к метке 5.
   - add w3, w3, #1 увеличивает счетчик повторений.
   - mov w5, #-1 устанавливает значение -1 для пометки повторяющегося элемента.
   - str w5, [%1, x4, lsl #2] записывает -1 в repeated_counts[j], помечая элемент как обработанный.
   - Метка 5: увеличивает индекс j на 1 и продолжает внутренний цикл.
6. **Запись результата повторений:**
   - Метка 3: проверяет значение счетчика w3.
   - cmp w3, #1 сравнивает счетчик с 1.
   - ble 6f если счетчик меньше или равен 1, переход к метке 6, пропуск записи.
   - str w3, [%1, x0, lsl #2] записывает значение счетчика в repeated_counts[i].
7. **Пометка оставшихся дубликатов:**
   - Метка 7: используется для пометки всех повторяющихся элементов, чтобы избежать повторного вывода.
   - cmp x4, %2 проверяет, если j больше или равно n.
   - bge 6f если да, переход к метке 6.
   - ldr w5, [%0, x4, lsl #2] загружает значение array[j] в регистр w5.
   - cmp w1, w5 сравнивает значения array[i] и array[j].
   - bne 8f если значения не равны, переход к метке 8.
   - mov w5, #0 устанавливает значение 0 для пометки элемента как выведенного.
   - str w5, [%1, x4, lsl #2] записывает 0 в repeated_counts[j].
   - Метка 8: увеличивает индекс j на 1 и продолжает пометку дубликатов.
8. **Переход к следующему элементу:**
   - Метка 6: увеличивает индекс i на 1.
   - Переход к началу внешнего цикла (метка 1:).
9. **Завершение:**

- Метка 4: обозначает конец обработки массива.

Таким образом, ассемблерная вставка последовательно обрабатывает элементы массива, находит повторяющиеся значения и записывает их количество повторений в массив repeated_counts, помечая обработанные элементы для предотвращения повторного подсчета

Также, проверим корректность работы данной ассемблерной вставки на примере, представленном выше

Результат выведен верно, следовательно, ассемблерная вставка, подсчитывающая числа и их повторения работает правильно