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. Поиск одинаковых чисел в массиве

По дисциплине «Аппаратное обеспечение вычислительных систем»

Вариант №

Выполнил студент группы
№M3112
Тимофеев Вячеслав [redacted]

[redacted]

Проверила
Шевчик [redacted]

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 ≤ N ≤ 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*.

Код:

main.c × INPUT.txt OUTPUT.txt ×

```c
#include <stdio.h>
#include <stdlib.h>
#define ll long long
#define F FILE
#define MAX(i, j) (((i) > (j)) ? (i) : (j))
#define DEmodulation(a, mod) (a - mod)

void sortContainer(ll *Container, ll n) {
    asm(
            "mov %[Container_ptr], %%rsi;"  // rsi = Container_ptr
            "mov %[N], %%rcx;"          //  rcx=N

            "out:;"
            "mov $1, %%rax;"            // j=1

            "next:;"
            "cmp %%rcx, %%rax;"         // rcx=rax ? end : do
            "jge end;"

            "mov (%%rsi, %%rax, 8), %%rbx;"  // Загружаем текущий элемент в rbx
            "mov %%rax, %%rdx;"          // rax=rdx
            "dec %%rdx;"                 // rdx--

            "in:;"
            "cmp $0, %%rdx;"             // rdx=0 ? go swap : do
            "jl insert;"
            "mov (%%rsi, %%rdx, 8), %%rdi;"

            "cmp %%rbx, %%rdi;"          // rbx>=rdi ? insert : do
            "jge insert;"

            "mov %%rdi, 8(%%rsi, %%rdx, 8);"  // Сдвигаем предыдущий элемент вправо
            "dec %%rdx;" // j--
            "jmp in;"   // next i

            "insert:;"
            "mov %%rbx, 8(%%rsi, %%rdx, 8);"  // Вставляем текущий элемент на нужное место
            "inc %%rax;"                      // rax++
            "jmp next;"                       // for(i++)

            "end:;"
            :                                                 // Выходные операнды
            : [Container_ptr] "r" (Container), [N] "r" (n) // Входные операнды
    : "rax", "rbx", "rcx", "rdx", "rsi", "rdi"              // Используемые регистры
    );
}
```

```
// rsi - i текущего элемента (Исходный индексный регистр)
// rax - для выполнения арифметических и логических операций (Накопительный регистр)
// rcx - счетчик цикла в операциях с повторяющимися командами и циклами (Регистр счетчика)
// rdi - указатель на данные, которые будут обрабатываться, особенно в строковых операциях (Целевой индексный регистр)
// rbx - для хранения данных и адресов (Базовый регистр)
// rdx - для хранения данных и в качестве вспомогательного регистра для некоторых операций (Регистр данных)
// ecx - текущий элемент массива (Расширенный регистр счетчика)
// edx - предыдущий элемент массива (Расширенный реестр данных)
// eax - для хранения арифм и лог операций (Накопительный регистр)

void NotAloneCount(ll n, ll border, ll *Container, ll *result) {
    asm (
        // Инициализация
            "mov %[Container_ptr], %%rsi;"       // ptr на массив Container в rsi
            "mov %[result_ptr], %%rdi;"          // ptr на массив result в rdi
            "xor %%rcx, %%rcx;"                 // Обнуление rcx (индекс массива)
            "mov %[border], %%rax;"              // Загрузка border в rax

            "begin:;"
         // Проверка окончания цикла
            "cmp %[size], %%rcx;"
            "jge done;"             // rcx <= size ? do : end

         // Обработка текущего элемента массива
            "mov (%%rsi, %%rcx, 8), %%rdx;"  // Загрузка текущего элемента массива в rdx
            "add %%rax, %%rdx;"              // Прибавляем border к текущему элементу (case с отрицательными числами)
            "mov (%%rdi, %%rdx, 8), %%rbx;"  // rbx=result[i]
            "inc %%rbx;"                     // rbx++
            "mov %%rbx, (%%rdi, %%rdx, 8);"  // result[i]=rbx


            "inc %%rcx;"  // i++
            "jmp begin;"  // for (i++)

            "done:;"
            :
            : [Container_ptr] "r" (Container), [result_ptr] "r" (result), [size] "r" (n), [border] "r" (border)
    : "rax", "rbx", "rcx", "rdx", "rdi", "rsi"
    );
}
```

```
int main() {
    F *inputF = fopen( Filename: "input.txt",  Mode: "r");
    ll n;
    fscanf( stream: inputF,  format: "%lld", &n);
    ll *Container = (ll *)calloc( NumOfElements: n,  SizeOfElements: sizeof(ll));
    if (Container == NULL) {
        return 1;
    }

    for (ll i = 0; i < n; i++) {
        fscanf( stream: inputF,  format: "%lld", &Container[i]);
    }
    fclose( File: inputF);
    sortContainer(Container, n);

    ll border = MAX((Container[0]), (Container[n-1]));
    ll modulationSize = border*2+1;
    ll *result = (ll *)calloc( NumOfElements: (modulationSize),  SizeOfElements: sizeof(ll));
    if (!result) {
        return 1;
    }

    NotAloneCount(n, border, Container, result);
    F *outputF = fopen( Filename: "output.txt",  Mode: "w");

    bool flag = false;
    for (ll i = 0; i < modulationSize; i++) {
        if (result[i] > 1) {
            fprintf( stream: outputF,  format: "%lld - %lld\n", DEmodulation(i,border), result[i]);
            if (!flag) flag = true;
        }
    }
    if (!flag) fprintf( stream: outputF,  format: "None");
    free( Memory: Container);
    free( Memory: result);

    fclose( File: outputF);
    return 0;
}
```

Пример 1:

main.c | INPUT.txt | OUTPUT.txt

```
15
-3
10
-4
-3
5
8
-3
10
88
3
10
-3
100
66
5
```

main.c | INPUT.txt | OUTPUT.txt

```
-3 - 4
5 - 2
10 - 3
```

Пример 2:

main.c | INPUT.txt | OUTPUT.txt

```
5
-1
0
1
2
3
```

main.c | INPUT.txt | OUTPUT.txt

```
None
```