

Федеральное государственное автономное  
образовательное учреждение высшего  
образования  
«Национальный исследовательский университет  
ИТМО»

Факультет Информационных технологий и программирования

Аппаратное обеспечение вычислительных систем

Дополнительное задание №2

Выполнил студент:  
Гаджиев Саид Ильясович  
Группа: М3115

Санкт-Петербург  
2023 г.

Разработать программу на языке C. Программа должна считать из файла целые числа типа `int`, количество чисел от 100 до 1000. Найти первые 10 наибольших значений. Записать эти значения в файл. Поиск первых 10 наибольших значений реализовать в виде ассемблерной вставки.

### Решение:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>
#define MAX_SIZE 1000
#define MAX_SORTED_SIZE 10

int main() {
    int* arr = (int*)malloc(MAX_SIZE * sizeof(int));
    int n = 0;
    FILE* input_file = fopen("input.txt", "r");
    if (input_file == NULL) {
        printf("ERROR input.txt");
        return 1;
    }
    while (fscanf(input_file, "%d", &arr[n]) == 1 && n < MAX_SIZE) {
        n++;
    }
    fclose(input_file);
    __asm {
        xor ecx, ecx; Обнуляем регистр ecx
        mov ecx, n; Загружаем значение переменной n в ecx (ecx = n)
        dec ecx; Уменьшаем значение ecx на 1 (n - 1)
        xor eax, eax; Обнуляем регистр eax (индекс i)
        mov eax, 0; Присваиваем eax значение 0 (i = 0)
        xor edx, edx; Обнуляем регистр edx (индекс j)
        mov edx, 0; Присваиваем edx значение 0 (j = 0)
        xor esi, esi; Обнуляем регистр esi (указатель на массив arr)
        mov esi, arr; Загружаем адрес массива arr в esi
        jmp First_if; Переходим к метке First_if

    First_if :
        cmp eax, ecx; Сравниваем значение eax(i) с ecx(n - 1)
        jl First_for; Если i < n - 1, переходим к метке First_for
        jmp End; Иначе переходим к метке End

    First_for :
        push ecx; Сохраняем текущее значение ecx(n - 1) в стеке
        sub ecx, eax; Вычитаем значение eax(i) из ecx(n - 1)
        mov ebx, ecx; Сохраняем полученное значение в ebx
        pop ecx; Восстанавливаем из стека предыдущее значение ecx(n - 1)
        cmp edx, ebx; Сравниваем значение edx(j) с ebx
        jl Second_if; Если j < n - i - 1, переходим к метке Second_if
        inc eax; Увеличиваем значение eax(i) на 1
        mov edx, 0; Обнуляем edx(j)
        jmp First_if; Переходим к метке First_if

    Second_if :
        push eax; Сохраняем текущее значение eax(i) в стеке
        mov eax, edx; Загружаем значение edx(j) в eax(i)
        inc edx; Увеличиваем значение edx(j) на 1
        mov ebx, [4 * eax + esi]; Загружаем элемент массива arr[i] в ebx
        pop eax; Восстанавливаем из стека предыдущее значение eax(i)
        cmp ebx, [4 * edx + esi]; Сравниваем элементы arr[i] и arr[j]
        jl First_for; Если arr[i] < arr[j], переходим к метке First_for
        jmp Swap; Иначе переходим к метке Swap
```

```

Swap :
    push ebx; Сохраняем значение ebx(arr[j]) в стеке
    dec edx; Уменьшаем значение edx(j) на 1
    mov ebx, [4 * edx + esi]; Загружаем значение arr[j - 1] в ebx
    inc edx; Увеличиваем значение edx(j) на 1 (теперь edx указывает на
arr[j])
    xchg ebx, [4 * edx + esi]; Обмениваем значения arr[j - 1] и arr[j]
местами, используя команду xchg
    dec edx; Уменьшаем значение edx(j) на 1 (теперь edx указывает на arr[j -
1])
    mov[4 * edx + esi], ebx; Сохраняем значение arr[j](теперь в ebx) в arr[j
- 1]
    inc edx; Увеличиваем значение edx(j) на 1
    pop ebx; Восстанавливаем из стека значение ebx(arr[j])
    jmp First_for; Переходим к метке First_for для продолжения сортировки.

End :
}

FILE* output_file = fopen("output.txt", "w");
if (output_file == NULL) {
    printf("ERROR output.txt");
    return 1;
}

int* arr_ten_last_sorted = (int*)malloc(MAX_SORTED_SIZE * sizeof(int));
__asm {
    mov eax, MAX_SORTED_SIZE; Загружаем значение константы MAX_SORTED_SIZE(т.е.
10) в регистр eax
    mov ecx, n; Загружаем значение переменной n(количество элементов в массиве)
в регистр ecx
    mov esi, arr; Загружаем массив arr в регистр esi
    mov edi, arr_ten_last_sorted; Загружаем массив arr_ten_last_sorted в регистр
edi
    mov ebx, n; Загружаем значение переменной n в регистр ebx
    sub ebx, eax; Вычитаем значение константы MAX_SORTED_SIZE из значения
ebx(ebx = n - MAX_SORTED_SIZE)
    mov eax, 0; Обнуляем регистр eax(индекс для массива arr_ten_last_sorted)
    jmp FOR; Переход к метке FOR

FOR :
    mov edx, [4 * ebx + esi]; Загружаем значение из массива arr по адресу
arr[ebx] в регистр edx
    mov[4 * eax + edi], edx; Сохраняем значение из регистра edx в массив
arr_ten_last_sorted по адресу arr_ten_last_sorted[eax]
    inc eax; Увеличиваем значение eax(индекс для массива
arr_ten_last_sorted) на 1
    inc ebx; Увеличиваем значение ebx(индекс для массива arr) на 1
    push ebx; Сохраняем текущее значение ebx на стеке
    mov ebx, 10; Присваиваем ebx значение 10
    cmp eax, ebx; Сравниваем значение eax(индекс для массива
arr_ten_last_sorted) с значением ebx(10)
    pop ebx; Восстанавливаем предыдущее значение ebx с помощью значения из
стека
    jl FOR; Если значение eax меньше значения ebx, переходим к метке FOR
    jmp Close; Иначе безусловно переходим к метке Close

Close :
}

for (int i = 0; i < MAX_SORTED_SIZE; i++) {
    fprintf(output_file, "%d ", arr_ten_last_sorted[i]);
}

```

```
    fclose(output_file);
    free(arr);
    free(arr_ten_last_sorted);

    return 0;
}

//Обычный вид
//void bubbleSort(int arr[], int n)
//{
//    int i, j;
//    for (i = 0; i < n - 1; i++)
//        for (j = 0; j < n - i - 1; j++)
//            if (arr[j] > arr[j + 1])
//                swap(&arr[j], &arr[j + 1]);
//}
```

File input.txt:

5 4 22 3 1 55 57 246 134 59 864 909 35 757 93 24 910

File output.txt:

55 57 59 93 134 246 757 864 909 910