

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Задание 2 - Алгоритмы поиска значений

тема

Преподаватель

Студент КИ18-17/16 031831229

номер группы, зачетной книжки

подпись, дата

подпись, дата

Р.Ю. Царев

инициалы, фамилия

В.А. Прекель

инициалы, фамилия

Красноярск 2020

1 Цель работы с постановкой задачи

1.1 Цель работы

Алгоритмы поиска значений.

1.2 Задача работы

Реализовать в программе два алгоритма (по выбору студента) из указанных ниже:

- а) алгоритм поиска минимального (или максимального) значения,
- б) алгоритм поиска моды,
- в) алгоритм поиска медианы,
- г) алгоритм поиска среднего значения,
- д) алгоритм поиска значения, равного заданному.

Сравнить эффективность реализованных алгоритмов по времени их выполнения.

Требования к выполнению лабораторной работы:

1. Строгое соответствие программы и результатов ее работы с полученным заданием.
2. Самостоятельное тестирование и отладка программы.
3. Устойчивость работы программы при любых воздействиях, задаваемых пользователем через интерфейс программы.
4. Предоставление демонстрационного примера и исходного текста программы для защиты.
5. Предоставление отчета по практическому заданию, содержащего описание реализованных алгоритмов, программы, результатов работы программы (отчет необходимо загрузить на сайт курса).

2 Описание реализованного алгоритма

Реализован алгоритм поиска максимального значения, алгоритм поиска среднего значения. Алгоритмическая часть реализована на языке ассемблера (GNU Assembler для Linux x64), часть взаимодействия с пользователем и юнит-тесты на Си.

3 Описание программы (листинги кода)

Листинг 1 – Lab_02/Lab_02_Lib/Lib.s

```
.text

# int MaxInArray(const int* array, int count);
.globl MaxInArray
MaxInArray:
    # пролог
    pushq %rbp
    movq %rsp, %rbp
    # запись в стек адреса первого элемента массива -24(%rbp)
    # он будет смещаться чтобы получить нужный элемент
    movq %rdi, -24(%rbp) # -24(%rbp) <- array
    # запись в стек кол-ва элементов -28(%rbp)
    movl %esi, -28(%rbp) # -28(%rbp) <- count
    # максимум (в начале первый элемент массива) -4(%rbp)
    movq -24(%rbp), %rax # %rax <- array
    movl (%rax), %eax # %eax <- (%rax)[0]
    movl %eax, -4(%rbp) # -4(%rbp) <- %eax
    # номер текущего элемента -8(%rbp)
    movl $0, -8(%rbp) # -8(%rbp) <- 0
Loop1_Start:
    # сравнение номера текущего элемента с кол-вом элементов
    # если номер текущего равен или больше, то конец цикла
    movl -8(%rbp), %eax # %eax <- -8(%rbp)
    cmpl -28(%rbp), %eax # compare -28(%rbp) <= %eax
    jge Loop1_End # if true go to Loop1_End
    # запись текущего элемента в %eax
    movq -24(%rbp), %rax # %rax <- -24(%rbp)
    movl (%rax), %eax # %eax <- %rax[0]
    # сравнение текущего и максимального
    # если максимальный больше, цикл идёт с начала
    cmpl %eax, -4(%rbp) # compare %eax <= -4(%rbp)
    jge Loop1_Continue # if true to to Loop1_Continue
    # иначе максимальному присваивается текущий
    movl %eax, -4(%rbp) # -4(%rbp) <- %eax
Loop1_Continue:
    # номер текущего увеличивается на 1
    addl $1, -8(%rbp) # -8(%rbp) += 1
    # адрес текущего увеличивается на 4 (байта)
    movq -24(%rbp), %rax # %rax <- -24(%rbp)
    addq $4, %rax # %rax += 4
    movq %rax, -24(%rbp) # -24(%rbp) <- %rax
    # цикл идёт с начала
    jmp Loop1_Start
Loop1_End:
    # возврат значения
    # return -4(%rbp)
    movl -4(%rbp), %eax # %eax <- -4(%rbp)
    # эпилог
    popq %rbp
    ret

# double AverageInArray(const int* array, int count);
.globl AverageInArray
AverageInArray:
```

```

# пролог
pushq   %rbp
movq    %rsp, %rbp
# запись в стек адреса первого элемента массива -24(%rbp)
# он будет смещаться чтобы получить нужный элемент
movq    %rdi, -24(%rbp)           # -24(%rbp) <- array
# запись в стек кол-ва элементов -28(%rbp)
movl    %esi, -28(%rbp)           # -28(%rbp) <- count
# запись в стек 0 с плавающей точкой для суммы -12(%rbp)
pxor    %xmm0, %xmm0              # %xmm0 <- 0
movsd   %xmm0, -12(%rbp)          # -12(%rbp) <- %xmm0
# номер текущего элемента -4(%rbp)
movl    $0, -4(%rbp)              # -4(%rbp) <- 0
Loop2_Start:
# сравнение номера текущего элемента с кол-вом элементов
# если номер текущего равен или больше, то конец цикла
movl    -4(%rbp), %eax             # %eax <- -4(%rbp)
cmpl    -28(%rbp), %eax            # compare -28(%rbp) <= %eax
jge     Loop2_End                 # if true go to Loop1_End
# запись текущего элемента в %eax
movq    -24(%rbp), %rax            # %rax <- -24(%rbp)
movl    (%rax), %eax               # %eax <- %rax[0]
# записать текущий элемент в регистр с плавающей точкой
cvtsi2sd %eax, %xmm0              # %xmm0 <- %eax
# увеличить сумму на текущий элемент
movsd   -12(%rbp), %xmm1          # %xmm1 <- -12(%rbp)
addsd   %xmm1, %xmm0              # %xmm0 += %xmm1
movsd   %xmm0, -12(%rbp)          # -12(%rbp) <- %xmm0
# номер текущего увеличивается на 1
addl    $1, -4(%rbp)              # -4(%rbp) += 1
# адрес текущего увеличивается на 4 (байта)
movq    -24(%rbp), %rax            # %rax <- -24(%rbp)
addq    $4, %rax                  # %rax += 4
movq    %rax, -24(%rbp)           # -24(%rbp) <- %rax
# цикл идёт с начала
jmp     Loop2_Start
Loop2_End:
# записываем кол-во элементов в регистр с плавающей точкой
cvtsi2sd -28(%rbp), %xmm1         # %xmm1 <- -28(%rbp)
# делим сумму на кол-во элементов
# результат будет в регистре %xmm0 который возвращается
movsd   -12(%rbp), %xmm0          # %xmm0 <- -12(%rbp)
divsd   %xmm1, %xmm0              # %xmm0 /= %xmm1
# эпилог
popq    %rbp
ret

```

Остальной код доступен в репозитории по адресу
https://github.com/prekel/AlgorithmsAndDataStructures/tree/master/Lab_02.

4 Результаты работы программы

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/AlgorithmsAndDataStructures/cmake-build-debug/Lab_02/Lab_02_Console$ ./Lab_02_Console
Enter count: 5
Enter element 0: 1
Enter element 1: 2
Enter element 2: 3
Enter element 3: 4
Enter element 4: 5
Max: 5
Average: 3.000000
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/AlgorithmsAndDataStructures/cmake-build-debug/Lab_02/Lab_02_Console$ ./Lab_02_Console
Enter count: 3
Enter element 0: 1
Enter element 1: -10
Enter element 2: 4
Max: 4
Average: -1.666667
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/AlgorithmsAndDataStructures/cmake-build-debug/Lab_02/Lab_02_Console$ ./Lab_02_Console
Enter count: 1
Enter element 0: 4
Max: 4
Average: 4.000000
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/AlgorithmsAndDataStructures/cmake-build-debug/Lab_02/Lab_02_Console$
```

Рисунок 1 – Пример работы программы

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/AlgorithmsAndDataStructures/cmake-build-debug/Lab_02/Lab_02_LibTests$ ./Lab_02_LibTests
CUnit - A unit testing framework for C - Version 3.2.3-cunity
http://cunit.sourceforge.net/
----- Начат Test_Max_1
----- Закончен Test_Max_1, прошло 0.000000 секунд
----- Начат Test_Max_Many
----- Закончен Test_Max_Many, прошло 0.125000 секунд
----- Начат Test_Average_1
----- Закончен Test_Average_1, прошло 0.000000 секунд
----- Начат Test_Average_Many
----- Закончен Test_Average_Many, прошло 0.171875 секунд

Run Summary
Suites: 1
Asserts: 20003
Tests: 4
Elapsed Time: 0.297(s)
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/AlgorithmsAndDataStructures/cmake-build-debug/Lab_02/Lab_02_LibTests$
```

Рисунок 2 – Запуск тестов