Санкт-Петербургский государственный политехнический университет

Институт компьютерных наук и технологий

**Кафедра «Компьютерные системы и программные технологии»**

**КУРСОВОЙ ПРОЕКТ**

**Симулятор Johniac**по дисциплине «Прикладное программирование»

Выполнил

студент гр.23531/1 К.А. Казаченко

Руководитель К.А. Гагарский

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2018 г.

Санкт-Петербург

2018

Санкт-Петербургский государственный политехнический университет

**ЗАДАНИЕ**

**НА ВЫПолнение курсовоГО ПРОЕКТА**

студент группы 23531/1 Казаченко Константин Андреевич

***1. Тема проекта (работы):*** Симулятор машины JOHNNIAC

***2. Срок сдачи законченного проекта (работы)*** \_\_. \_\_.18 г.

***3. Исходные данные к проекту (работе)***:

Описание машины, содержимое входной ленты задаются во входных текстовых файлах (разных), результат работы машины выводится в текстовый файл.

Необходимо обеспечить возможность имитации работы машины до ее останова или в течение заданного числа шагов.

Необходимо обеспечить возможность отладки машины (вывод промежуточных конфигураций, возобновление работы машины).

Допускается введение разумных ограничений на число инструкций (миллионы). Размер входных данных должен быть ограничен только доступным симулятору объемом памяти.

Разработка кода и документации должна вестись в репозитории git.

Разработанное ПО должно собираться с помощью команды make компилятором gcc (mingw) с опциями -std=c11 -pedantic -Wall -Wextra без предупреждений компилятора.

***4. Содержание пояснительной записки***: введение, основная часть (раскрывается структура основной части), заключение, список использованных источников, приложения.

***Дата получения задания***: «05» апреля 2018 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.А. Гагарский

*(подпись)*

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_ К.А. Казаченко

*(подпись студента)*

05.04.2018г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc517616657)

[1. Методика решения 5](#_Toc517616658)

[2. Код программы, реализующей симулятор 7](#_Toc517616659)

[2.1. constant.c 7](#_Toc517616660)

[2.2. constant.h 7](#_Toc517616661)

[2.3. double\_john.c 7](#_Toc517616662)

[2.4. double\_john.h 9](#_Toc517616663)

[2.5. help\_func.c 9](#_Toc517616664)

[2.6. help\_func.h 14](#_Toc517616665)

[2.7. main.c 14](#_Toc517616666)

[3. Сборка разработанного ПО с помощью команды make 17](#_Toc517616667)

[Makefile: 17](#_Toc517616668)

[4. Методика испытаний 18](#_Toc517616669)

[4.1 Исходные данные для каждого запуска 18](#_Toc517616670)

[4.2 Полный запуск 19](#_Toc517616671)

[4.3 Выполнить определенные кол-во шагов 21](#_Toc517616672)

[4.4 Выполнить код пошагово 22](#_Toc517616673)

[4.4 Выполнить код пошагово в режиме debug 23](#_Toc517616674)

[ЗАКЛЮЧЕНИЕ 27](#_Toc517616675)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#_Toc517616676)

## ВВЕДЕНИЕ

В данном курсовом проекте мною будет написана программа, являющаяся симулятором Johniac, что соответствует моему индивидуальному заданию по прикладному программированию.

В ней будут реализованы основные опции Johniac, такие как загрузка команд из соответствующих входных файлов, печать команд на экран, пошаговое и мгновенное выполнение программы, содержащейся в файле с командами.

## 1. Методика решения

Имена входных/выходных файлов заданы по умолчанию и равны “input.txt” и “output.txt” соответственно.

input.txt - файл, содержащий список команд программы, которую должен будет выполнить симулятор, в следующем формате:

\_\_\_\_\_\_\_\_\_\_\_\_

Cc' 12

M 555

R

h 1  
 \_\_\_\_\_\_\_

где до пробела - команда, после – числовое значение команды(аргумент).

output.txt - файл вывода, содержащий пошаговую работу выполненной программы.

Набор допустимых команд:

1. “ x” очистить аккумулятор и добавить в него число х
2. “- x” очистить аккумулятор и вычесть из него число х
3. “М x” очистить аккумулятор и добавить в него модуль числа х
4. “-М x” очистить аккумулятор и вычесть из него модуль числа х
5. “h x” добавить в аккумулятор число х
6. “h- x” вычесть из аккумулятора число х
7. “hM x” добавить в аккумулятор модуль числа х
8. “h-М x” вычесть из аккумулятора модуль числа х
9. “R x” очистить регистр на позиции R и добавить число х
10. “А x” очистить аккумулятор и добавить в него число из регистра на позиции R
11. “X x” умножить значение регистра на позиции R на х
12. “-- x” поделить значение регистра на позиции R на х
13. “С ” сместить R на одно значение влево
14. “С’ ” сместить R на одно значение вправо
15. “Сс x” если значение аккумулятора больше 0 значению R присвоить x
16. “Сс’ x” если значение аккумулятора меньше 0 значению R присвоить x
17. “S x” регистру на позиции x присвоить значение аккумулятора
18. “Sp x” значение числителя R в регистре присвоить значению x
19. “Sp’ x” значение делителя R в регистре присвоить значению x
20. “L” умножить значение аккумулятора на 2
21. “R” поделить значение аккумулятора на 2

Данные в машине хранятся в виде doble\_john – значение разбита на числитель и знаменатель.

**typedef struct** double\_john {  
 **int** num;  
 **int** den;  
} double\_john;

Функциональность файлов программы:

**main.c –** главная функция программы **constant.h –** структура, котораясодержит все необходимые переменные для работы программы  
**help\_func.h –** функции для помощи в main.c **help\_func.c –** реализация help\_func.h, а так же реализация всех команд программы  
**double\_john.h –** заголовочный файл для структуры хранения и обработки данных в типе double\_john  
**double\_john.c –** реализация double\_john.h

Работа с пользователем осуществляется посредством меню, представленного на

Рис 1.1:

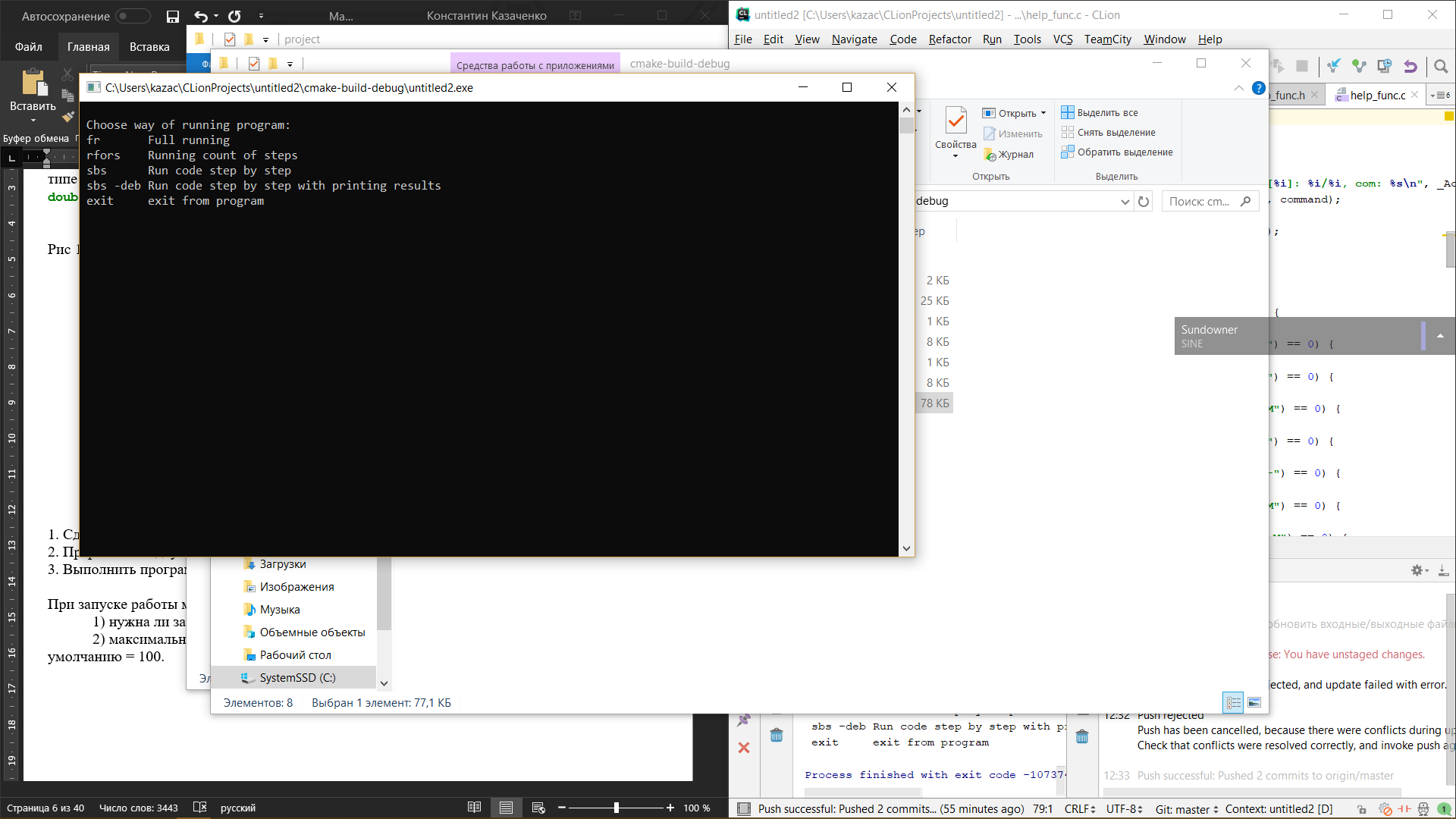


Рис 1.1 Меню работы симулятора

При работе в режиме отладки (sbs -deb) предусмотрены 4 возможных действия:

w - write memory

s - not write memory

f - run to end

q – exit

## 2. Код программы, реализующей симулятор

### 2.1. constant.h

#ifndef **UNTITLED2\_CONSTANT\_H**#include **<stdio.h>**#include **"double\_john.h"**#define **\_WHITH "w"**#define **\_EXIT "q"**#define **\_WHITHOUT "s"**#define **\_FULL "f"**#define **\_R\_DATA** 1023  
#define **\_L\_DATA** 0  
  
**typedef struct** constant {  
  
 **int** \_R;  
  
 double\_john \*\*registerData;  
 double\_john \*\_Ac;  
 FILE \*fp, \*out;  
} constant;  
  
*//extern double\_john \_Ac;*#define **UNTITLED2\_CONSTANT\_H**#endif

### 2.2. double\_john.c

*//  
// Created by kazac on 22.06.2018.  
//*#include **<stdio.h>**#include **<stdlib.h>**#include **"double\_john.h"  
  
void** putInt(double\_john \*this, **int** num) {  
  
 this->num = num;  
 this->den = 1;  
};  
  
**int** NOD(**int** n1, **int** n2) {  
 **int** div;  
 **if** (n1 == n2) *// если числа равны, НОД найден* **return** n1;  
 **int** d = n1 - n2; *// Находим разность чисел* **if** (d < 0) *// если разность отрицательная,* {  
 d = -d; *// меняем знак* div = NOD(n1, d); *// вызываем функцию NOD() для двух наименьших чисел* } **else** *// если разность n1-n2 положительная* {  
 div = NOD(n2, d); *// вызываем функцию NOD() для двух наименьших чисел* }  
 **return** div;  
}  
  
**void** NOD\_john(double\_john \*one) {  
 **int** nod = NOD(abs(one->den), abs(one->num));  
 one->den /= nod;  
 one->num /= nod;  
 **if** (one->den == 0) one->den = 1;  
 **if** (one->den < 0) {  
 one->den \*= -1;  
 one->num \*= -1;  
 }  
}  
  
**void** add(double\_john \*one, double\_john \*two) {  
 **int** d2 = two->den;  
 **int** d1 = one->den;  
 one->num = one->num \* d2 + two->num \* d1;  
 one->den = one->den \* d2;  
  
 NOD\_john(one);  
}  
  
**void** addInt(double\_john \*one, **int** two) {  
 double\_john \*new = (double\_john \*) malloc(**sizeof**(double\_john));  
 new->num = two;  
 new->den = 1;  
 add(one, new);  
}  
  
**void** sub(double\_john \*one, double\_john \*two) {  
 **int** d2 = two->den;  
 **int** d1 = one->den;  
 one->num = one->num \* d2 - two->num \* d1;  
 one->den = one->den \* d2;  
 NOD\_john(one);  
}  
  
**void** subInt(double\_john \*one, **int** two) {  
 double\_john \*new = (double\_john \*) malloc(**sizeof**(double\_john));  
 new->num = two;  
 new->den = 1;  
 sub(one, new);  
}  
  
**void** mult(double\_john \*one, double\_john \*two) {  
 one->num = one->num \* two->num;  
 one->den = one->den \* two->den;  
 NOD\_john(one);  
}  
  
**void** multInt(double\_john \*one, **int** two) {  
 double\_john \*new = (double\_john \*) malloc(**sizeof**(double\_john));  
 new->num = two;  
 new->den = 1;  
 mult(one, new);  
}  
  
**void** dev(double\_john \*one, double\_john \*two) {  
 one->num = one->num \* two->den;  
 one->den = one->den \* two->num;  
 NOD\_john(one);  
}  
  
**void** devInt(double\_john \*one, **int** two) {  
 double\_john \*new = (double\_john \*) malloc(**sizeof**(double\_john));  
 new->num = two;  
 new->den = 1;  
 dev(one, new);  
}

### 2.3. double\_john.h

#ifndef **UNTITLED2\_DOUBLE\_JOHN\_H  
typedef struct** double\_john {  
 **int** num;  
 **int** den;  
} double\_john;  
  
**void** putInt(double\_john\*, **int**);  
  
**void** add(double\_john\*, double\_john\*);  
  
**void** addInt(double\_john\* one, **int** two);  
  
**void** sub(double\_john\* one, double\_john\* two);  
  
**void** subInt(double\_john\* one, **int** two);  
  
**void** mult(double\_john\* one, double\_john\* two);  
  
**void** multInt(double\_john\* one, **int** two);  
  
**void** dev(double\_john\* one, double\_john\* two);  
  
**void** devInt(double\_john\* one, **int** two);  
  
#define **UNTITLED2\_DOUBLE\_JOHN\_H**#endif *//UNTITLED2\_DOUBLE\_JOHN\_H*

### 2.4. help\_func.c

#include **<stdlib.h>**#include **"help\_func.h"**#include **"constant.h"**#include **<string.h>  
  
int** func\_add();  
  
**int** func\_sub();  
  
**int** func\_minM();  
  
**int** func\_M();  
  
**int** func\_h();  
  
**int** func\_hmin();  
  
**int** func\_hm();  
  
**int** func\_hminM();  
  
**int** func\_Rr();  
  
**int** func\_A();  
  
**int** func\_X();  
  
**int** func\_minmin();  
  
**int** func\_C();  
  
**int** func\_C\_();  
  
**int** func\_Cc();  
  
**int** func\_Cc\_();  
  
**int** func\_S();  
  
**int** func\_Sp();  
  
**int** func\_R();  
  
**int** func\_L();  
  
**int** func\_Sp\_();  
  
**char** \*numbcom;  
**char** \*com;  
**char** \*command;  
**int** \_curi;  
**int** \_x;  
constant \*constantThis;  
  
**int** pars(**int** debag, constant \*constant) {  
 com = malloc(**sizeof**(**char**) \* 100);  
 command = malloc(**sizeof**(**char**) \* 10);  
 numbcom = malloc(**sizeof**(**char**) \* 100);  
 memset(command, 0, **sizeof**(command));  
 memset(numbcom, 0, **sizeof**(numbcom));  
 memset(com, 0, **sizeof**(com));  
 constantThis = constant;  
  
 **int** i = 0;  
 **for** (i = 0; i < 10; i++)  
 com[i] = **' '**;  
 fgets(command, 100, constant->fp);*//читаем очередную строку* **if** (debag > 0) {  
 printf(command);  
 **for** (i = 0; i < **\_R\_DATA** + 1; i++) {  
 **if** (debag > 1)  
 printf(**"%i/%i - %i ;"**, constant->registerData[i]->num, constant->registerData[i]->den, i);  
 fprintf(constant->out, **"%i/%i - %i ;"**, constant->registerData[i]->num,  
 constant->registerData[i]->den, i);  
 }  
 }  
 \_curi = 0;  
 printf(**"%s\n"**, command);  
 **for** (i = 0; i < strlen(command); i++) {  
  
 **if** (command[i] != **' '** && command[i] != **'\n'** && command[i] != **'\0'**) {  
 numbcom[i] = command[i];  
 \_curi++;  
 } **else break**;  
 }  
 \_curi++;  
 **for** (i = \_curi; i < strlen(command); i++) {  
 **if** ((**'0'** <= command[i] && command[i] <= **'9'**) || command[i] == **'-'** || command[i] == **' '** || command[i] == **'\n'** ||  
 command[i] == **'\0'**)  
 com[i - \_curi] = command[i];  
 **else** {  
 printf(**"Exception in code: Unsupported number."**);  
 **return** 665;  
 }  
 }  
  
 com[i - \_curi] = **'\0'**;  
 \_x = atoi(com);  
 **int** ans = sw(numbcom);  
 printf(**"Ac: %i/%i, data[%i]: %i/%i, com: %s\n"**, constant->\_Ac->num, constant->\_Ac->den,  
 constant->\_R, constant->registerData[constant->\_R]->num,  
 constant->registerData[constant->\_R]->den, command);  
 fprintf(constant->out, **"Ac: %i/%i, data[%i]: %i/%i, com: %s\n"**, constant->\_Ac->num, constant->\_Ac->den,  
 constant->\_R, constant->registerData[constant->\_R]->num,  
 constant->registerData[constant->\_R]->den, command);  
 fclose(constant->out);  
 fopen\_s((FILE \*\*) constant->out, **"output.txt"**, **"a"**);  
 **return** ans;  
}  
  
**int** sw(**char** \*numbcom) {  
 **if** (strcmp(numbcom, **""**) == 0) {  
 **return** func\_add();  
 } **else if** (strcmp(numbcom, **"-"**) == 0) {  
 **return** func\_sub();  
 } **else if** (strcmp(numbcom, **"M"**) == 0) {  
 **return** func\_M();  
 } **else if** (strcmp(numbcom, **"-M"**) == 0) {  
 **return** func\_minM();  
 } **else if** (strcmp(numbcom, **"h"**) == 0) {  
 **return** func\_h();  
 } **else if** (strcmp(numbcom, **"h-"**) == 0) {  
 **return** func\_hmin();  
 } **else if** (strcmp(numbcom, **"hM"**) == 0) {  
 **return** func\_hm();  
 } **else if** (strcmp(numbcom, **"h-M"**) == 0) {  
 **return** func\_hminM();  
 } **else if** (strcmp(numbcom, **"Rr"**) == 0) {  
 **return** func\_Rr();  
 } **else if** (strcmp(numbcom, **"A"**) == 0) {  
 **return** func\_A();  
 } **else if** (strcmp(numbcom, **"X"**) == 0) {  
 **return** func\_X();  
 } **else if** (strcmp(numbcom, **"--"**) == 0) {  
 **return** func\_minmin();  
 } **else if** (strcmp(numbcom, **"C"**) == 0) {  
 **return** func\_C();  
 } **else if** (strcmp(numbcom, **"C'"**) == 0) {  
 **return** func\_C\_();  
 } **else if** (strcmp(numbcom, **"Cc"**) == 0) {  
 **return** func\_Cc();  
 } **else if** (strcmp(numbcom, **"Cc'"**) == 0) {  
 **return** func\_Cc\_();  
 } **else if** (strcmp(numbcom, **"S"**) == 0) {  
 **return** func\_S();  
 } **else if** (strcmp(numbcom, **"Sp"**) == 0) {  
 **return** func\_Sp();  
 } **else if** (strcmp(numbcom, **"Sp'"**) == 0) {  
 **return** func\_Sp\_();  
 } **else if** (strcmp(numbcom, **"L"**) == 0) {  
 **return** func\_L();  
 } **else if** (strcmp(numbcom, **"R"**) == 0) {  
 **return** func\_R();  
 } **else** {  
 printf(**"Exception in code: Unsupported command."**);  
 **return** 666;  
 }  
}  
  
*////////////////////////////////////////////////////////////////***int** func\_add() {  
 putInt(constantThis->\_Ac, 0);  
 addInt(constantThis->\_Ac, \_x);  
 **return** 0;  
}  
  
**int** func\_sub() {  
 putInt(constantThis->\_Ac, 0);  
 subInt(constantThis->\_Ac, \_x);  
 **return** 0;  
}  
  
**int** func\_M() {  
 putInt(constantThis->\_Ac, 0);  
 addInt(constantThis->\_Ac, abs(\_x));  
 **return** 0;  
}  
  
**int** func\_minM() {  
 putInt(constantThis->\_Ac, 0);  
 subInt(constantThis->\_Ac, abs(\_x));  
 **return** 0;  
}  
  
**int** func\_h() {  
 addInt(constantThis->\_Ac, \_x);  
 **return** 0;  
}  
  
**int** func\_hmin() {  
 subInt(constantThis->\_Ac, \_x);  
 **return** 0;  
}  
  
**int** func\_hm() {  
 addInt(constantThis->\_Ac, abs(\_x));  
 **return** 0;  
}  
  
**int** func\_hminM() {  
 subInt(constantThis->\_Ac, abs(\_x));  
 **return** 0;  
}  
  
**int** func\_Rr() {  
 **if** (constantThis->\_R < **\_L\_DATA** || constantThis->\_R > **\_R\_DATA**) {  
 printf(**"Exception in code: Index of bound."**);  
 **return** 10;  
 }  
 putInt(constantThis->registerData[constantThis->\_R], 0);  
 addInt(constantThis->registerData[constantThis->\_R], \_x);  
 **return** 0;  
}  
  
**int** func\_A() {  
 **if** (constantThis->\_R < **\_L\_DATA** || constantThis->\_R > **\_R\_DATA**) {  
 printf(**"Exception in code: Index of bound."**);  
 **return** 10;  
 }  
 putInt(constantThis->\_Ac, 0);  
 constantThis->\_Ac = constantThis->registerData[constantThis->\_R];  
 **return** 0;  
}  
  
**int** func\_X() {  
 **if** (constantThis->\_R < **\_L\_DATA** || constantThis->\_R > **\_R\_DATA**) {  
 printf(**"Exception in code: Index of bound."**);  
 **return** 10;  
 }  
 putInt(constantThis->\_Ac, 0);  
 multInt(constantThis->registerData[constantThis->\_R], \_x);  
 **return** 0;  
}  
  
**int** func\_minmin() {  
 **if** (constantThis->\_R < **\_L\_DATA** || constantThis->\_R > **\_R\_DATA**) {  
 printf(**"Exception in code: Index of bound."**);  
 **return** 10;  
 }  
 putInt(constantThis->registerData[constantThis->\_R], 0);  
 devInt(constantThis->\_Ac, \_x);  
 constantThis->registerData[constantThis->\_R] = constantThis->\_Ac;  
 **return** 0;  
}  
  
**int** func\_C() {  
 constantThis->\_R--;  
 **if** (constantThis->\_R < **\_L\_DATA** || constantThis->\_R > **\_R\_DATA**) {  
 printf(**"Exception in code: Index of bound."**);  
 **return** 10;  
 }  
 **return** 0;  
}  
  
**int** func\_C\_() {  
 constantThis->\_R++;  
 **if** (constantThis->\_R < **\_L\_DATA** || constantThis->\_R > **\_R\_DATA**) {  
 printf(**"Exception in code: Index of bound."**);  
 **return** 10;  
 }  
 **return** 0;  
}  
  
**int** func\_Cc() {  
 **if** (constantThis->\_Ac->num > 0) constantThis->\_R = \_x;  
 **return** 0;  
}  
  
**int** func\_Cc\_() {  
 **if** (constantThis->\_Ac->num <= 0) constantThis->\_R = \_x;  
 **return** 0;  
}  
  
**int** func\_S() {  
 constantThis->\_R = \_x;  
 **if** (constantThis->\_R < **\_L\_DATA** || constantThis->\_R > **\_R\_DATA**) {  
 printf(**"Exception in code: Index of bound."**);  
 **return** 10;  
 }  
 constantThis->registerData[\_x] = constantThis->\_Ac;  
 **return** 0;  
}  
  
**int** func\_Sp() {  
 **if** (constantThis->\_R < **\_L\_DATA** || constantThis->\_R > **\_R\_DATA** || \_x < **\_L\_DATA** || \_x > **\_R\_DATA**) {  
 printf(**"Exception in code: Index of bound."**);  
 **return** 10;  
 }  
 constantThis->registerData[constantThis->\_R]->num = constantThis->registerData[\_x]->num;  
  
 **return** 0;  
}  
  
**int** func\_Sp\_() {  
 **if** (constantThis->\_R < **\_L\_DATA** || constantThis->\_R > **\_R\_DATA** || \_x < **\_L\_DATA** || \_x > **\_R\_DATA**) {  
 printf(**"Exception in code: Index of bound."**);  
 **return** 10;  
 }  
 constantThis->registerData[constantThis->\_R]->den = constantThis->registerData[\_x]->den;  
  
 **return** 0;  
}  
  
**int** func\_L() {  
 multInt(constantThis->\_Ac, 2);  
 **return** 0;  
}  
  
**int** func\_R() {  
 devInt(constantThis->\_Ac, 2);  
 **return** 0;  
}

### 2.5. help\_func.h

#ifndef **UNTITLED2\_HELP\_FUNC\_H**#define **UNTITLED2\_HELP\_FUNC\_H**#include **<stdio.h>  
  
  
int** pars(**int** debag, constant \*constant);  
  
**int** sw(**char** \*numbcom);  
  
#endif *//UNTITLED2\_HELP\_FUNC\_H*

### 2.6. main.c

#include **<stdio.h>**#include **<stdlib.h>**#include **<string.h>**#include **"constant.h"**#include **"help\_func.h"  
  
int** initialization(constant \*);  
  
**int** running\_count\_of\_steps(constant \*);  
  
**int** full\_running(constant \*);  
  
**int** run\_code\_step\_by\_step(constant \*);  
  
**int** run\_code\_step\_by\_step\_with\_printing\_results(constant \*);  
  
**char** key[2];  
**char** answer[255];  
  
  
**int** main() {  
 **int** runing\_state = 1;  
 constant \*constant = malloc(**sizeof**(constant));  
  
*// {\*\_Ac,  
// \_x = 0,  
// \_R = 0,  
// \_curi = 0,  
// double\_john \*\*registerData,  
// \*numbcom,  
// \*com,  
// \*command,  
// \*out,  
// \*fp};* **while** (1) {  
 initialization(constant);  
 **if** (runing\_state != 3) {  
 printf(**"\n Choose way of running program: \n"  
  
 " fr Full running \n"  
 " rfors Running count of steps \n"  
 " sbs Run code step by step \n"  
 " sbs -deb Run code step by step with printing results \n"  
 " exit exit from program\n"**);  
 }  
 gets(answer); *//fgets - это чтение из файла а не из с консоли))) но чем это заменить я так и не придумал, хотя можно использовать алфавит вмсето клавишь но так не интересно* **if** (strcmp(answer, **"fr"**) == 0) { *//полное выполнение кода* **if** (full\_running(constant) != 0) runing\_state = 2;  
 } **else if** (strcmp(answer, **"rfors"**) == 0) { *//выполнение определенного числа шагов* **if** (running\_count\_of\_steps(constant) != 0) runing\_state = 2;  
 } **else if** (strcmp(answer, **"sbs"**) == 0) { *//выполнить код пошагово* **if** (run\_code\_step\_by\_step(constant) != 0) runing\_state = 2;  
 } **else if** (strcmp(answer, **"sbs -deb"**) == 0) { *//выполнить код пошагово с выводом результата* **if** (run\_code\_step\_by\_step\_with\_printing\_results(constant) != 0) runing\_state = 2;  
 } **else if** (strcmp(answer, **"exit"**) == 0) {  
 **return** 0;  
 } **else if** (strcmp(answer, **"help"**) == 0 || strcmp(answer, **"h"**) == 0) {  
 runing\_state = 2;  
 } **else if** (strcmp(answer, **""**) == 0) {  
 runing\_state = 3;  
 } **else** {  
 printf(**"Not found command\n"**);  
 runing\_state = 2;  
 }  
 **if** (runing\_state == 1) {  
 **for** (**int** i = 0; i < **\_R\_DATA** + 1; i++) {  
 fprintf(constant->out, **"%i/%i - %i ;"**, constant->registerData[i]->num, constant->registerData[i]->den,  
 i);  
 }  
 printf(**"You code completed successful"**);  
 }  
 **if** (runing\_state == 2) runing\_state = 1;  
 fclose(constant->fp);  
 fclose(constant->out);  
 }  
}  
  
**int** initialization(constant \*constant) {  
 constant->registerData = malloc(**sizeof**(double\_john) \* (**\_R\_DATA** + 1));  
 constant->\_Ac = malloc(**sizeof**(double\_john));  
 constant->\_Ac->num = 0;  
 constant->\_Ac->den = 1;  
 constant->\_R = 0;  
  
  
 memset(constant->registerData, 0, **sizeof**(constant->registerData));  
 **for** (**int** i = 0; i < **\_R\_DATA** + 1; i++) {  
 double\_john \*john = malloc(**sizeof**(double\_john));  
 john->num = 0;  
 john->den = 1;  
 constant->registerData[i] = john;  
 }  
 constant->out = fopen(**"output.txt"**, **"a"**);  
 constant->fp = fopen(**"input.txt"**, **"r"**);  
}  
  
  
**int** StrToInt(**const char** \*s) {  
 **int** temp = 0; *// число* **int** i = 0;  
 **int** sign = 0; *// знак числа 0- положительное, 1 — отрицательное* **if** (s[i] == **'-'**) {  
 sign = 1;  
 i++;  
 }  
 **while** (s[i] >= 0x30 && s[i] <= 0x39) {  
 temp = temp + (s[i] & 0x0F);  
 temp = temp \* 10;  
 i++;  
 }  
 temp = temp / 10;  
 **if** (sign == 1)  
 temp = -temp;  
 **return** (temp);  
}  
  
**int** running\_count\_of\_steps(constant \*constant) {  
 printf(**"Input count of steps \n"**);  
 **char** \*string = malloc(**sizeof**(**char**) \* 100);  
 memset(string, 0, **sizeof**(string));  
 fgets(string, 10, **stdin**);  
 **int** countSteps = StrToInt(string);  
 **while** (countSteps > 0) {  
 **int** result = pars(0, constant);  
 **if** (result != 0) **return** result;  
 countSteps--;  
 }  
 **return** 0;  
};  
  
**int** full\_running(constant \*constant) {  
 **while** (!feof(constant->fp)) {  
 **int** result = pars(0, constant);  
 **if** (result != 0) **return** result;  
 }  
 **return** 0;  
};  
  
**int** run\_code\_step\_by\_step(constant \*constant) {  
 **while** (!feof(constant->fp)) {  
  
 printf(**"s - next step \n"**);  
 printf(**"f - run to end \n"**);  
 printf(**"q - exit \n"**);  
  
 fgets(key, 2, **stdin**);  
  
 **if** (strcmp(key, **\_WHITHOUT**) == 0) {  
 **int** result = pars(0, constant);  
 **if** (result != 0) **return** result;  
 }  
 *// break;* **if** (strcmp(key, **\_EXIT**) == 0) {  
 **return** 0;  
 }*//завершаем работу программы* **if** (strcmp(key, **\_FULL**) == 0) {  
 **while** (!feof(constant->fp)) {  
 **int** result = pars(0, constant);  
 **if** (result != 0) **return** result;  
 }  
 }  
 }  
 **return** 0;  
};  
  
**int** run\_code\_step\_by\_step\_with\_printing\_results(constant \*constant) {  
 **while** (!feof(constant->fp)) {  
 printf(**"w - write memory \n"**);  
 printf(**"s - not write memory \n"**);  
 printf(**"f - run to end \n"**);  
 printf(**"q - exit \n"**);  
  
 fgets(key, **sizeof**(key), **stdin**);  
  
 **if** (strcmp(key, **\_WHITH**) == 0) {*// Space- следующая команда* {  
 **int** result = pars(2, constant);  
 **if** (result != 0) **return** result;  
 printf(**" Ac: %i/%i R: %i; "**, constant->\_Ac->num, constant->\_Ac->den, constant->\_R);  
 }  
 }  
 **if** (strcmp(key, **\_WHITHOUT**) == 0) {  
 **int** result = pars(1, constant);  
 **if** (result != 0) **return** result;  
 printf(**" Ac: %i/%i R: %i; "**, constant->\_Ac->num, constant->\_Ac->den, constant->\_R);  
 }  
 **if** (strcmp(key, **\_EXIT**) == 0) {*//Esc - прекратить выполнение программы* **return** 0;  
 }  
 **if** (strcmp(key, **\_FULL**) == 0) {  
 **while** (!feof(constant->fp)) {  
 **int** result = pars(1, constant);  
 **if** (result != 0) **return** result;  
 }  
 }  
  
 }  
 **return** 0;  
};

## 3. Сборка разработанного ПО с помощью команды make

### Makefile:

\_\_\_\_\_\_\_\_\_\_\_\_\_

CC=gcc  
CFLAGS=-std=c11 -pedantic -Wall -Wextra -c -o  
  
untitled2 : main.o help\_func.o constant.o double\_john.o  
 $(CC) -o untitled2 main.o constant.o help\_func.o double\_john.o  
  
main.o : main.c  
 $(CC) $(CFLAGS) main.o main.c  
  
constant.o : constant.c  
 $(CC) $(CFLAGS) constant.o constant.c  
  
help\_func.o : help\_func.c  
 $(CC) $(CFLAGS) help\_func.o help\_func.c  
  
double\_john.o : double\_john.c  
 $(CC) $(CFLAGS) double\_john.o double\_john.c

\_\_\_\_\_\_\_\_\_\_\_\_\_

## 4. Методика испытаний

### 4.1 Исходные данные для каждого запуска

Будет два основных типа проверки:

Первый- код полностью корректный и должен выполниться целиком.

Второй- код имеет ошибку и выполнение должно прерваться.

10

- 10

M 1

M -2

-M 3

-M -7

C'

C'

C

C'

Cc 10

Cc' 12

M 555

R

h 1

40

R

10

- 10

M 1

M -2

-M 3

-M -7

C'

C'

C

C'

Cc 10

Cc' 12

Mq 555 🡨тут не верная команда

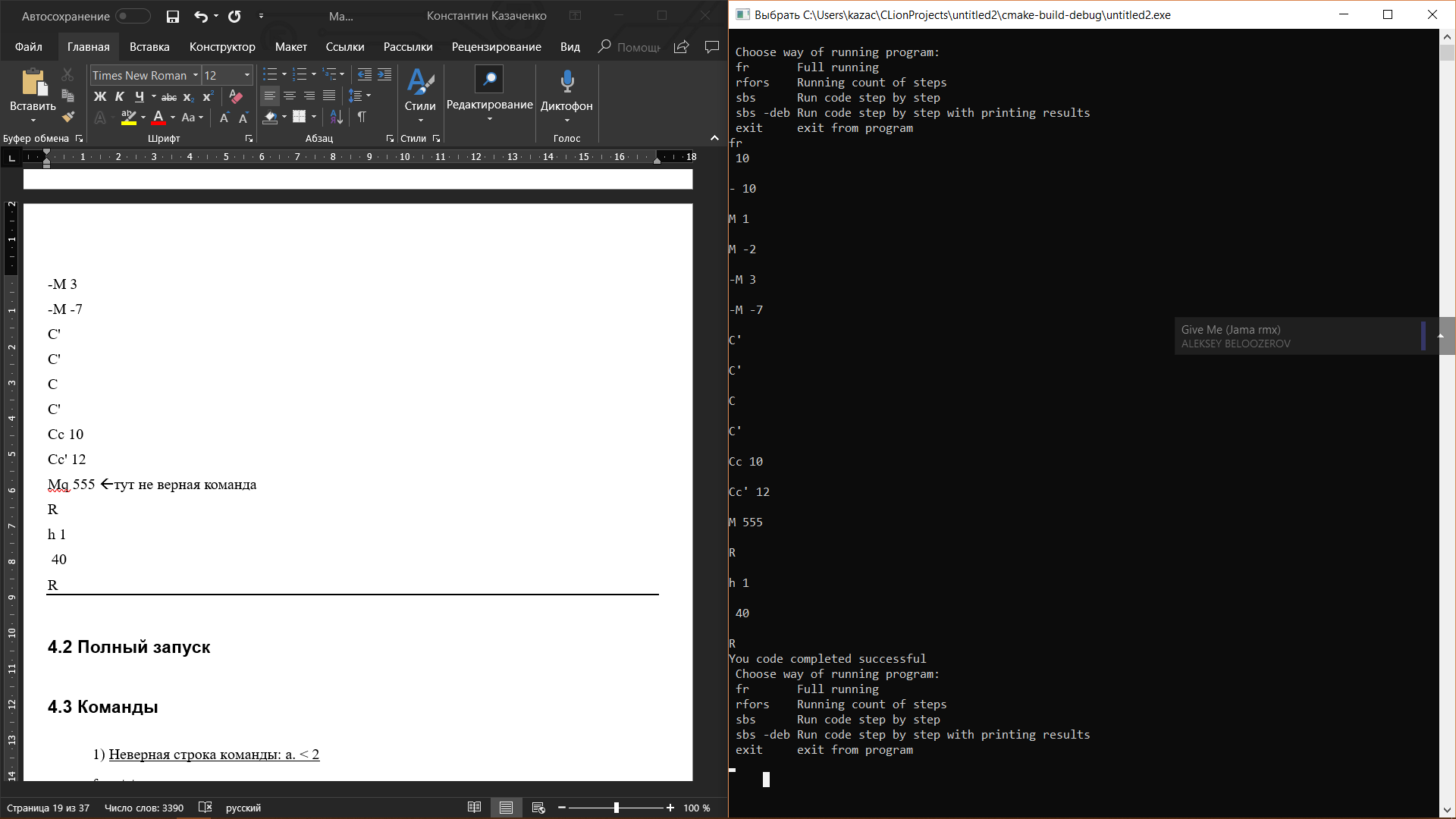
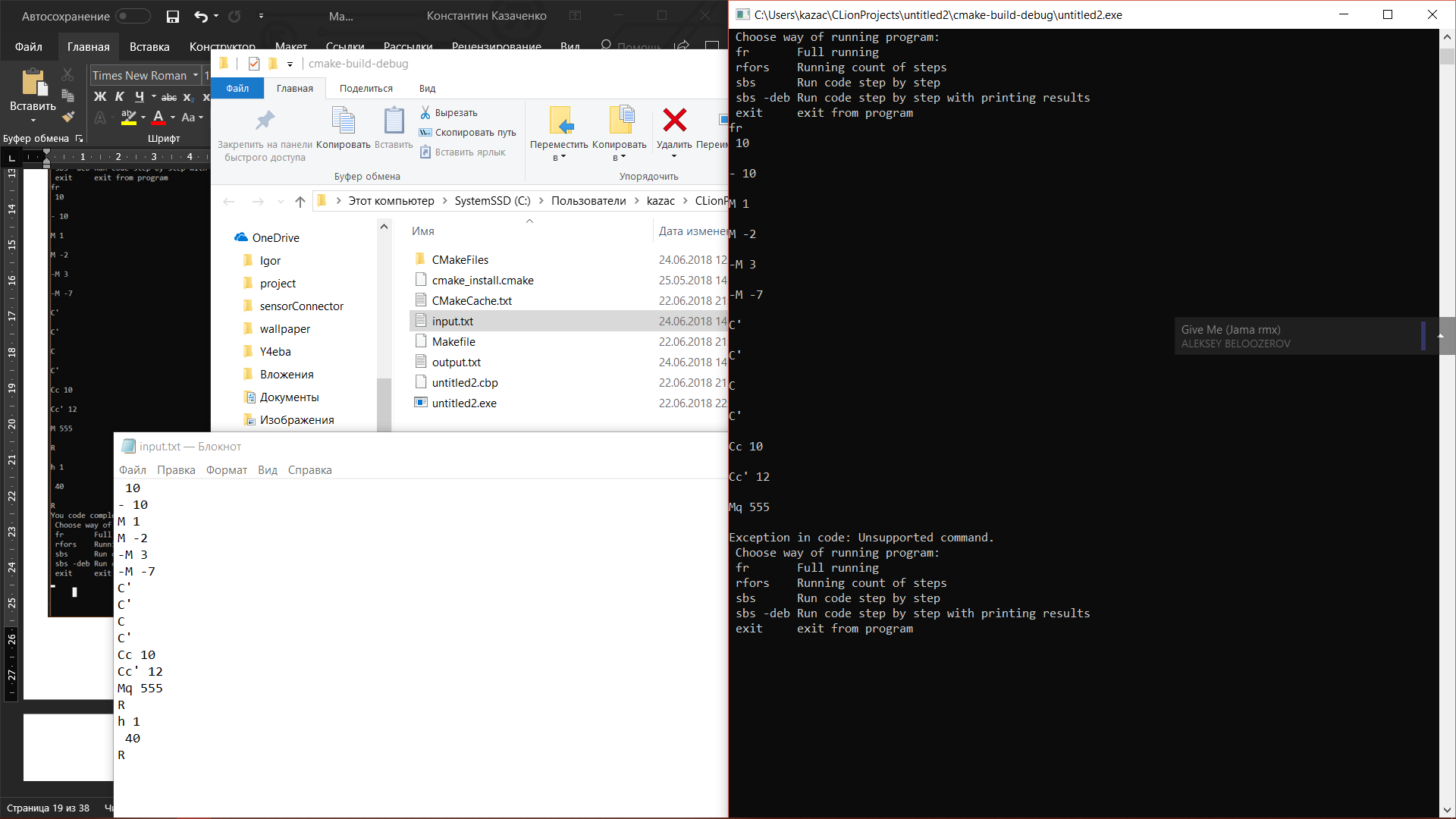
R

h 1

40

R

### 4.2 Полный запуск

Ac: 10/1, data[0]: 0/1, com: 10

Ac: -10/1, data[0]: 0/1, com: - 10

Ac: 1/1, data[0]: 0/1, com: M 1

Ac: 2/1, data[0]: 0/1, com: M -2

Ac: -3/1, data[0]: 0/1, com: -M 3

Ac: -7/1, data[0]: 0/1, com: -M -7

Ac: -7/1, data[1]: 0/1, com: C'

Ac: -7/1, data[2]: 0/1, com: C'

Ac: -7/1, data[1]: 0/1, com: C

Ac: -7/1, data[2]: 0/1, com: C'

Ac: -7/1, data[2]: 0/1, com: Cc 10

Ac: -7/1, data[12]: 0/1, com: Cc' 12

Ac: 555/1, data[12]: 0/1, com: M 555

Ac: 555/2, data[12]: 0/1, com: R

Ac: 557/2, data[12]: 0/1, com: h 1

Ac: 40/1, data[12]: 0/1, com: 40

Ac: 20/1, data[12]: 0/1, com: R

Ac: 10/1, data[12]: 0/1, com: 10

Ac: -10/1, data[12]: 0/1, com: - 10

Ac: 1/1, data[12]: 0/1, com: M 1

Ac: 2/1, data[12]: 0/1, com: M -2

Ac: -3/1, data[12]: 0/1, com: -M 3

Ac: -7/1, data[12]: 0/1, com: -M -7

Ac: -7/1, data[13]: 0/1, com: C'

Ac: -7/1, data[14]: 0/1, com: C'

Ac: -7/1, data[13]: 0/1, com: C

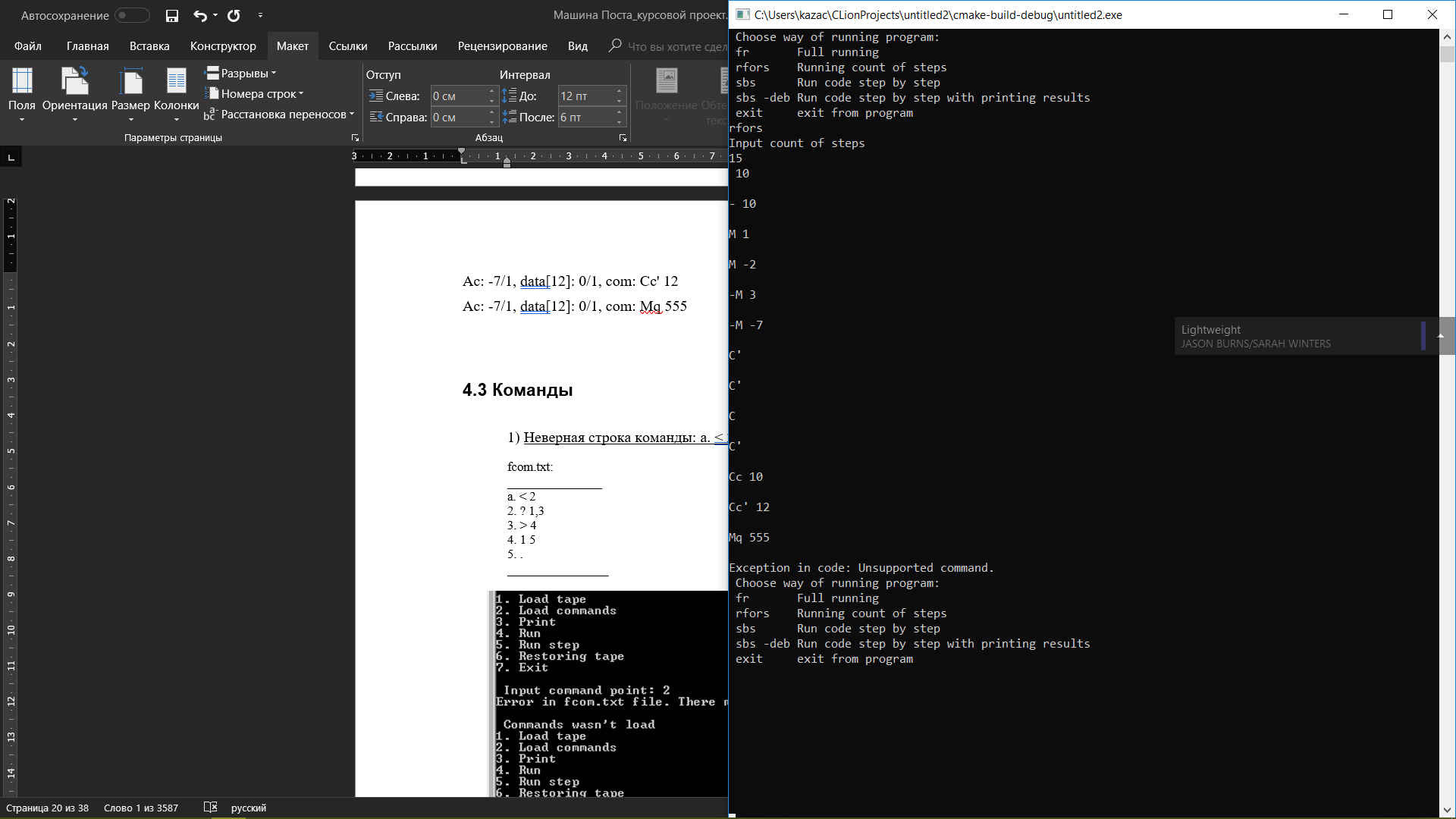
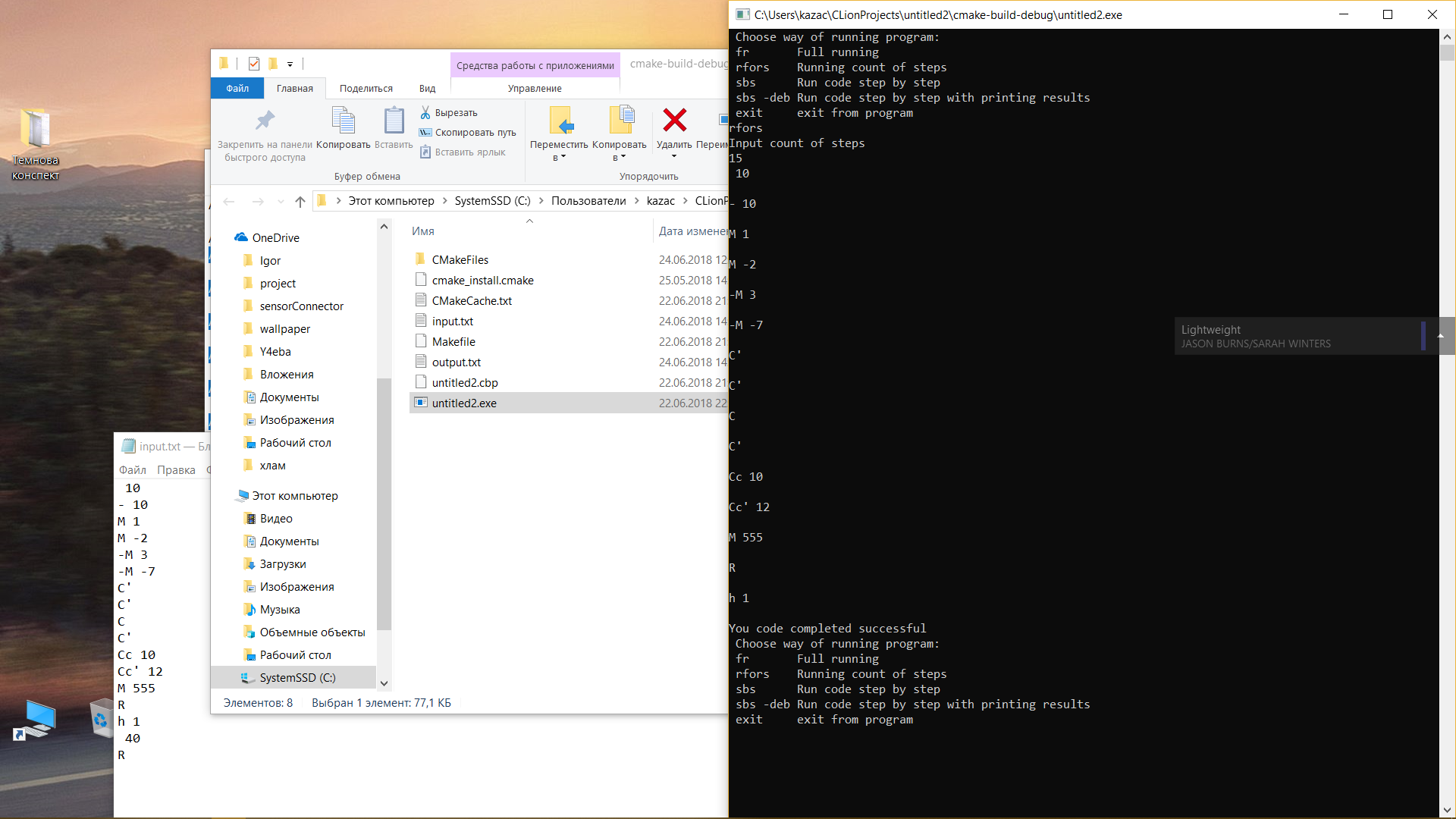
Ac: -7/1, data[14]: 0/1, com: C'

Ac: -7/1, data[14]: 0/1, com: Cc 10

Ac: -7/1, data[12]: 0/1, com: Cc' 12

Ac: -7/1, data[12]: 0/1, com: Mq 555

### 4.3 Выполнить определенные кол-во шагов



Ac: 10/1, data[0]: 0/1, com: 10

Ac: -10/1, data[0]: 0/1, com: - 10

Ac: 1/1, data[0]: 0/1, com: M 1

Ac: 2/1, data[0]: 0/1, com: M -2

Ac: -3/1, data[0]: 0/1, com: -M 3

Ac: -7/1, data[0]: 0/1, com: -M -7

Ac: -7/1, data[1]: 0/1, com: C'

Ac: -7/1, data[2]: 0/1, com: C'

Ac: -7/1, data[1]: 0/1, com: C

Ac: -7/1, data[2]: 0/1, com: C'

Ac: -7/1, data[2]: 0/1, com: Cc 10

Ac: -7/1, data[12]: 0/1, com: Cc' 12

Ac: 555/1, data[12]: 0/1, com: M 555

Ac: 555/2, data[12]: 0/1, com: R

Ac: 557/2, data[12]: 0/1, com: h 1

Ac: 10/1, data[12]: 0/1, com: 10

Ac: -10/1, data[12]: 0/1, com: - 10

Ac: 1/1, data[12]: 0/1, com: M 1

Ac: 2/1, data[12]: 0/1, com: M -2

Ac: -3/1, data[12]: 0/1, com: -M 3

Ac: -7/1, data[12]: 0/1, com: -M -7

Ac: -7/1, data[13]: 0/1, com: C'

Ac: -7/1, data[14]: 0/1, com: C'

Ac: -7/1, data[13]: 0/1, com: C

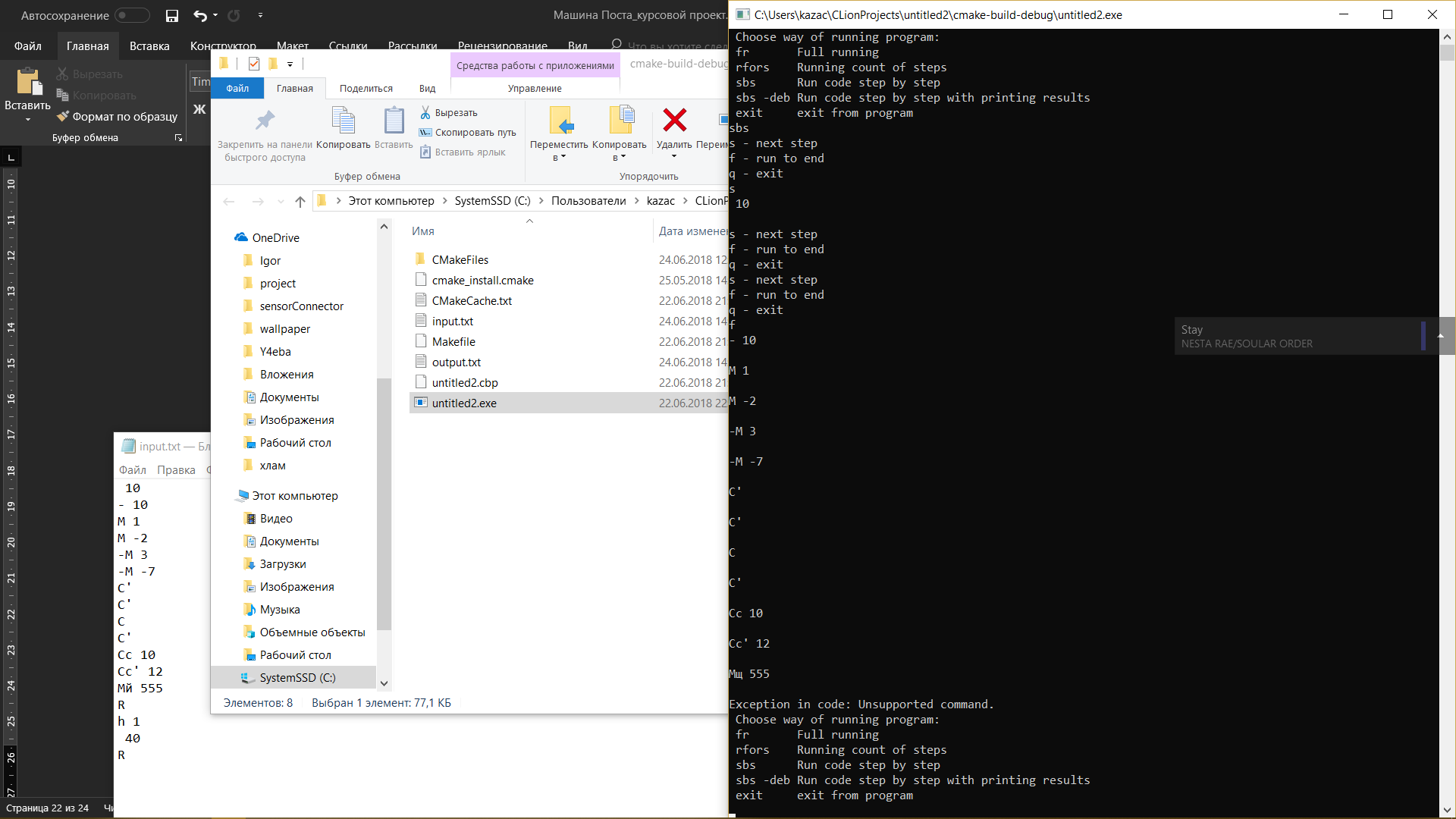
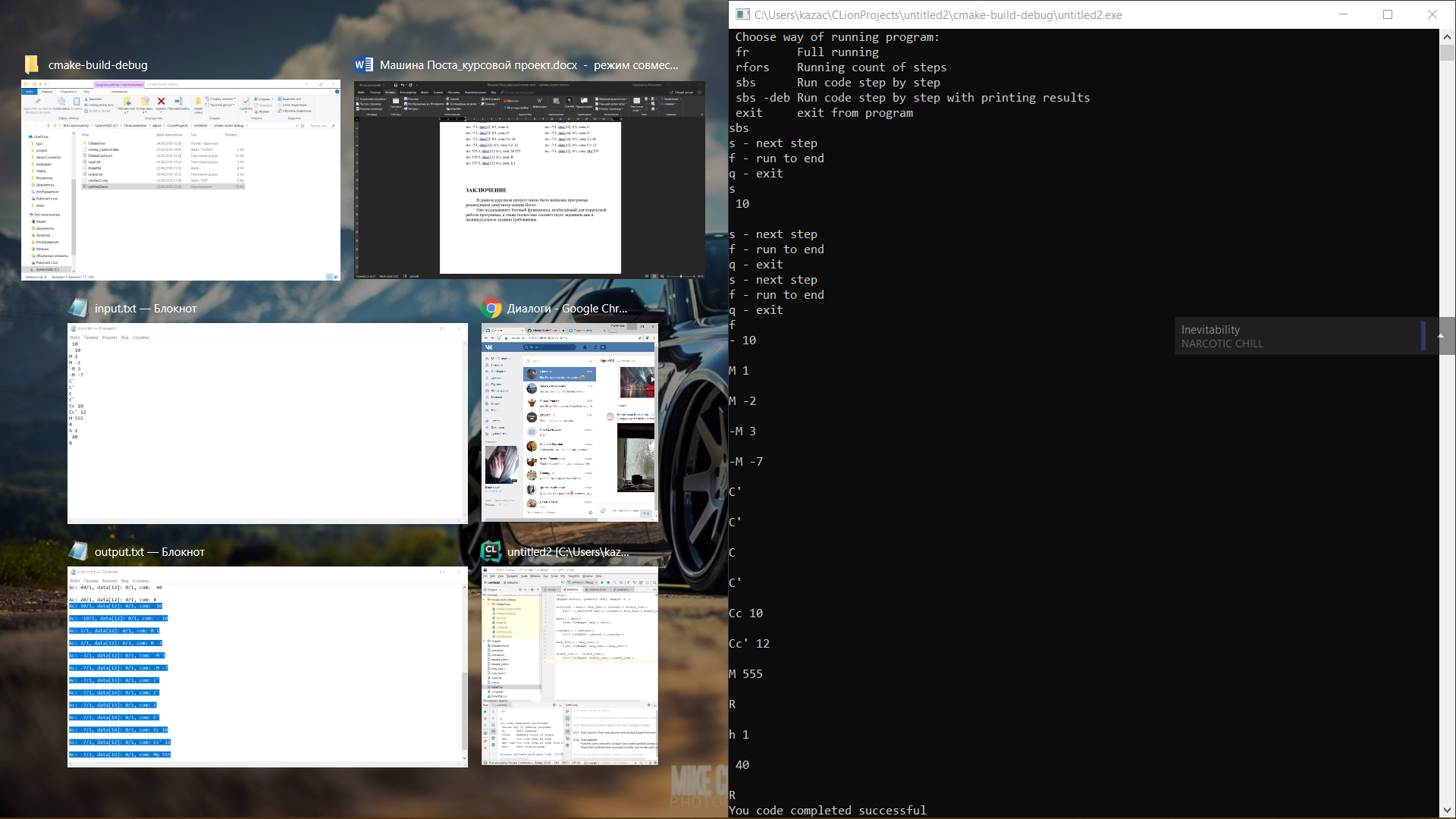
Ac: -7/1, data[14]: 0/1, com: C'

Ac: -7/1, data[14]: 0/1, com: Cc 10

Ac: -7/1, data[12]: 0/1, com: Cc' 12

Ac: -7/1, data[12]: 0/1, com: Mq 555

### 4.4 Выполнить код пошагово



Ac: 10/1, data[0]: 0/1, com: 10

Ac: -10/1, data[0]: 0/1, com: - 10

Ac: 1/1, data[0]: 0/1, com: M 1

Ac: 2/1, data[0]: 0/1, com: M -2

Ac: -3/1, data[0]: 0/1, com: -M 3

Ac: -7/1, data[0]: 0/1, com: -M -7

Ac: -7/1, data[1]: 0/1, com: C'

Ac: -7/1, data[2]: 0/1, com: C'

Ac: -7/1, data[1]: 0/1, com: C

Ac: -7/1, data[2]: 0/1, com: C'

Ac: -7/1, data[2]: 0/1, com: Cc 10

Ac: -7/1, data[12]: 0/1, com: Cc' 12

Ac: 555/1, data[12]: 0/1, com: M 555

Ac: 555/2, data[12]: 0/1, com: R

Ac: 557/2, data[12]: 0/1, com: h 1

Ac: 40/1, data[12]: 0/1, com: 40

Ac: 20/1, data[12]: 0/1, com: R

Ac: 10/1, data[12]: 0/1, com: 10

Ac: -10/1, data[12]: 0/1, com: - 10

Ac: 1/1, data[12]: 0/1, com: M 1

Ac: 2/1, data[12]: 0/1, com: M -2

Ac: -3/1, data[12]: 0/1, com: -M 3

Ac: -7/1, data[12]: 0/1, com: -M -7

Ac: -7/1, data[13]: 0/1, com: C'

Ac: -7/1, data[14]: 0/1, com: C'

Ac: -7/1, data[13]: 0/1, com: C

Ac: -7/1, data[14]: 0/1, com: C'

Ac: -7/1, data[14]: 0/1, com: Cc 10

Ac: -7/1, data[12]: 0/1, com: Cc' 12

Ac: -7/1, data[12]: 0/1, com: Mq 555

### 4.4 Выполнить код пошагово в режиме debug

Choose way of running program:

fr Full running

rfors Running count of steps

sbs Run code step by step

sbs -deb Run code step by step with printing results

exit exit from program

sbs -deb

w - write memory

s - not write memory

f - run to end

q - exit

w

10

0/1 - 0 ;0/1 - 1 ;0/1 - 2 ;0/1 - 3 ;0/1 - 4 ;0/1 - 5 ;0/1 - 6 ;0/1 - 7 ;0/1 - 8 ;0/1 - 9 ;0/1 - 10 ;0/1 - 11 ;0/1 - 12 ;0/1 - 13 ;0/1 - 14 ;0/1 - 15 ;0/1 - 16 ;0/1 - 17 ;0/1 - 18 ;0/1 - 19 ;0/1 - 20 ;0/1 - 21 ;0/1 - 22 ;0/1 - 23 ;0/1 - 24 ;0/1 - 25 ;0/1 - 26 ;0/1 - 27 ;0/1 - 28 ;0/1 - 29 ;0/1 - 30 ;0/1 - 31 ;0/1 - 32 ;0/1 - 33 ;0/1 - 34 ;0/1 - 35 ;0/1 - 36 ;0/1 - 37 ;0/1 - 38 ;0/1 - 39 ;0/1 - 40 ;0/1 - 41 ;0/1 - 42 ;0/1 - 43 ;0/1 - 44 ;0/1 - 45 ;0/1 - 46 ;0/1 - 47 ;0/1 - 48 ;0/1 - 49 ;0/1 - 50 ;0/1 - 51 ;0/1 - 52 ;0/1 - 53 ;0/1 - 54 ;0/1 - 55 ;0/1 - 56 ;0/1 - 57 ;0/1 - 58 ;0/1 - 59 ;0/1 - 60 ;0/1 - 61 ;0/1 - 62 ;0/1 - 63 ;0/1 - 64 ;0/1 - 65 ;0/1 - 66 ;0/1 - 67 ;0/1 - 68 ;0/1 - 69 ;0/1 - 70 ;0/1 - 71 ;0/1 - 72 ;0/1 - 73 ;0/1 - 74 ;0/1 - 75 ;0/1 - 76 ;0/1 - 77 ;0/1 - 78 ;0/1 - 79 ;0/1 - 80 ;0/1 - 81 ;0/1 - 82 ;0/1 - 83 ;0/1 - 84 ;0/1 - 85 ;0/1 - 86 ;0/1 - 87 ;0/1 - 88 ;0/1 - 89 ;0/1 - 90 ;0/1 - 91 ;0/1 - 92 ;0/1 - 93 ;0/1 - 94 ;0/1 - 95 ;0/1 - 96 ;0/1 - 97 ;0/1 - 98 ;0/1 - 99 ;0/1 - 100 ;0/1 - 101 ;0/1 - 102 ;0/1 - 103 ;0/1 - 104 ;0/1 - 105 ;0/1 - 106 ;0/1 - 107 ;0/1 - 108 ;0/1 - 109 ;0/1 - 110 ;0/1 - 111 ;0/1 - 112 ;0/1 - 113 ;0/1 - 114 ;0/1 - 115 ;0/1 - 116 ;0/1 - 117 ;0/1 - 118 ;0/1 - 119 ;0/1 - 120 ;0/1 - 121 ;0/1 - 122 ;0/1 - 123 ;0/1 - 124 ;0/1 - 125 ;0/1 - 126 ;0/1 - 127 ;0/1 - 128 ;0/1 - 129 ;0/1 - 130 ;0/1 - 131 ;0/1 - 132 ;0/1 - 133 ;0/1 - 134 ;0/1 - 135 ;0/1 - 136 ;0/1 - 137 ;0/1 - 138 ;0/1 - 139 ;0/1 - 140 ;0/1 - 141 ;0/1 - 142 ;0/1 - 143 ;0/1 - 144 ;0/1 - 145 ;0/1 - 146 ;0/1 - 147 ;0/1 - 148 ;0/1 - 149 ;0/1 - 150 ;0/1 - 151 ;0/1 - 152 ;0/1 - 153 ;0/1 - 154 ;0/1 - 155 ;0/1 - 156 ;0/1 - 157 ;0/1 - 158 ;0/1 - 159 ;0/1 - 160 ;0/1 - 161 ;0/1 - 162 ;0/1 - 163 ;0/1 - 164 ;0/1 - 165 ;0/1 - 166 ;0/1 - 167 ;0/1 - 168 ;0/1 - 169 ;0/1 - 170 ;0/1 - 171 ;0/1 - 172 ;0/1 - 173 ;0/1 - 174 ;0/1 - 175 ;0/1 - 176 ;0/1 - 177 ;0/1 - 178 ;0/1 - 179 ;0/1 - 180 ;0/1 - 181 ;0/1 - 182 ;0/1 - 183 ;0/1 - 184 ;0/1 - 185 ;0/1 - 186 ;0/1 - 187 ;0/1 - 188 ;0/1 - 189 ;0/1 - 190 ;0/1 - 191 ;0/1 - 192 ;0/1 - 193 ;0/1 - 194 ;0/1 - 195 ;0/1 - 196 ;0/1 - 197 ;0/1 - 198 ;0/1 - 199 ;0/1 - 200 ;0/1 - 201 ;0/1 - 202 ;0/1 - 203 ;0/1 - 204 ;0/1 - 205 ;0/1 - 206 ;0/1 - 207 ;0/1 - 208 ;0/1 - 209 ;0/1 - 210 ;0/1 - 211 ;0/1 - 212 ;0/1 - 213 ;0/1 - 214 ;0/1 - 215 ;0/1 - 216 ;0/1 - 217 ;0/1 - 218 ;0/1 - 219 ;0/1 - 220 ;0/1 - 221 ;0/1 - 222 ;0/1 - 223 ;0/1 - 224 ;0/1 - 225 ;0/1 - 226 ;0/1 - 227 ;0/1 - 228 ;0/1 - 229 ;0/1 - 230 ;0/1 - 231 ;0/1 - 232 ;0/1 - 233 ;0/1 - 234 ;0/1 - 235 ;0/1 - 236 ;0/1 - 237 ;0/1 - 238 ;0/1 - 239 ;0/1 - 240 ;0/1 - 241 ;0/1 - 242 ;0/1 - 243 ;0/1 - 244 ;0/1 - 245 ;0/1 - 246 ;0/1 - 247 ;0/1 - 248 ;0/1 - 249 ;0/1 - 250 ;0/1 - 251 ;0/1 - 252 ;0/1 - 253 ;0/1 - 254 ;0/1 - 255 ;0/1 - 256 ;0/1 - 257 ;0/1 - 258 ;0/1 - 259 ;0/1 - 260 ;0/1 - 261 ;0/1 - 262 ;0/1 - 263 ;0/1 - 264 ;0/1 - 265 ;0/1 - 266 ;0/1 - 267 ;0/1 - 268 ;0/1 - 269 ;0/1 - 270 ;0/1 - 271 ;0/1 - 272 ;0/1 - 273 ;0/1 - 274 ;0/1 - 275 ;0/1 - 276 ;0/1 - 277 ;0/1 - 278 ;0/1 - 279 ;0/1 - 280 ;0/1 - 281 ;0/1 - 282 ;0/1 - 283 ;0/1 - 284 ;0/1 - 285 ;0/1 - 286 ;0/1 - 287 ;0/1 - 288 ;0/1 - 289 ;0/1 - 290 ;0/1 - 291 ;0/1 - 292 ;0/1 - 293 ;0/1 - 294 ;0/1 - 295 ;0/1 - 296 ;0/1 - 297 ;0/1 - 298 ;0/1 - 299 ;0/1 - 300 ;0/1 - 301 ;0/1 - 302 ;0/1 - 303 ;0/1 - 304 ;0/1 - 305 ;0/1 - 306 ;0/1 - 307 ;0/1 - 308 ;0/1 - 309 ;0/1 - 310 ;0/1 - 311 ;0/1 - 312 ;0/1 - 313 ;0/1 - 314 ;0/1 - 315 ;0/1 - 316 ;0/1 - 317 ;0/1 - 318 ;0/1 - 319 ;0/1 - 320 ;0/1 - 321 ;0/1 - 322 ;0/1 - 323 ;0/1 - 324 ;0/1 - 325 ;0/1 - 326 ;0/1 - 327 ;0/1 - 328 ;0/1 - 329 ;0/1 - 330 ;0/1 - 331 ;0/1 - 332 ;0/1 - 333 ;0/1 - 334 ;0/1 - 335 ;0/1 - 336 ;0/1 - 337 ;0/1 - 338 ;0/1 - 339 ;0/1 - 340 ;0/1 - 341 ;0/1 - 342 ;0/1 - 343 ;0/1 - 344 ;0/1 - 345 ;0/1 - 346 ;0/1 - 347 ;0/1 - 348 ;0/1 - 349 ;0/1 - 350 ;0/1 - 351 ;0/1 - 352 ;0/1 - 353 ;0/1 - 354 ;0/1 - 355 ;0/1 - 356 ;0/1 - 357 ;0/1 - 358 ;0/1 - 359 ;0/1 - 360 ;0/1 - 361 ;0/1 - 362 ;0/1 - 363 ;0/1 - 364 ;0/1 - 365 ;0/1 - 366 ;0/1 - 367 ;0/1 - 368 ;0/1 - 369 ;0/1 - 370 ;0/1 - 371 ;0/1 - 372 ;0/1 - 373 ;0/1 - 374 ;0/1 - 375 ;0/1 - 376 ;0/1 - 377 ;0/1 - 378 ;0/1 - 379 ;0/1 - 380 ;0/1 - 381 ;0/1 - 382 ;0/1 - 383 ;0/1 - 384 ;0/1 - 385 ;0/1 - 386 ;0/1 - 387 ;0/1 - 388 ;0/1 - 389 ;0/1 - 390 ;0/1 - 391 ;0/1 - 392 ;0/1 - 393 ;0/1 - 394 ;0/1 - 395 ;0/1 - 396 ;0/1 - 397 ;0/1 - 398 ;0/1 - 399 ;0/1 - 400 ;0/1 - 401 ;0/1 - 402 ;0/1 - 403 ;0/1 - 404 ;0/1 - 405 ;0/1 - 406 ;0/1 - 407 ;0/1 - 408 ;0/1 - 409 ;0/1 - 410 ;0/1 - 411 ;0/1 - 412 ;0/1 - 413 ;0/1 - 414 ;0/1 - 415 ;0/1 - 416 ;0/1 - 417 ;0/1 - 418 ;0/1 - 419 ;0/1 - 420 ;0/1 - 421 ;0/1 - 422 ;0/1 - 423 ;0/1 - 424 ;0/1 - 425 ;0/1 - 426 ;0/1 - 427 ;0/1 - 428 ;0/1 - 429 ;0/1 - 430 ;0/1 - 431 ;0/1 - 432 ;0/1 - 433 ;0/1 - 434 ;0/1 - 435 ;0/1 - 436 ;0/1 - 437 ;0/1 - 438 ;0/1 - 439 ;0/1 - 440 ;0/1 - 441 ;0/1 - 442 ;0/1 - 443 ;0/1 - 444 ;0/1 - 445 ;0/1 - 446 ;0/1 - 447 ;0/1 - 448 ;0/1 - 449 ;0/1 - 450 ;0/1 - 451 ;0/1 - 452 ;0/1 - 453 ;0/1 - 454 ;0/1 - 455 ;0/1 - 456 ;0/1 - 457 ;0/1 - 458 ;0/1 - 459 ;0/1 - 460 ;0/1 - 461 ;0/1 - 462 ;0/1 - 463 ;0/1 - 464 ;0/1 - 465 ;0/1 - 466 ;0/1 - 467 ;0/1 - 468 ;0/1 - 469 ;0/1 - 470 ;0/1 - 471 ;0/1 - 472 ;0/1 - 473 ;0/1 - 474 ;0/1 - 475 ;0/1 - 476 ;0/1 - 477 ;0/1 - 478 ;0/1 - 479 ;0/1 - 480 ;0/1 - 481 ;0/1 - 482 ;0/1 - 483 ;0/1 - 484 ;0/1 - 485 ;0/1 - 486 ;0/1 - 487 ;0/1 - 488 ;0/1 - 489 ;0/1 - 490 ;0/1 - 491 ;0/1 - 492 ;0/1 - 493 ;0/1 - 494 ;0/1 - 495 ;0/1 - 496 ;0/1 - 497 ;0/1 - 498 ;0/1 - 499 ;0/1 - 500 ;0/1 - 501 ;0/1 - 502 ;0/1 - 503 ;0/1 - 504 ;0/1 - 505 ;0/1 - 506 ;0/1 - 507 ;0/1 - 508 ;0/1 - 509 ;0/1 - 510 ;0/1 - 511 ;0/1 - 512 ;0/1 - 513 ;0/1 - 514 ;0/1 - 515 ;0/1 - 516 ;0/1 - 517 ;0/1 - 518 ;0/1 - 519 ;0/1 - 520 ;0/1 - 521 ;0/1 - 522 ;0/1 - 523 ;0/1 - 524 ;0/1 - 525 ;0/1 - 526 ;0/1 - 527 ;0/1 - 528 ;0/1 - 529 ;0/1 - 530 ;0/1 - 531 ;0/1 - 532 ;0/1 - 533 ;0/1 - 534 ;0/1 - 535 ;0/1 - 536 ;0/1 - 537 ;0/1 - 538 ;0/1 - 539 ;0/1 - 540 ;0/1 - 541 ;0/1 - 542 ;0/1 - 543 ;0/1 - 544 ;0/1 - 545 ;0/1 - 546 ;0/1 - 547 ;0/1 - 548 ;0/1 - 549 ;0/1 - 550 ;0/1 - 551 ;0/1 - 552 ;0/1 - 553 ;0/1 - 554 ;0/1 - 555 ;0/1 - 556 ;0/1 - 557 ;0/1 - 558 ;0/1 - 559 ;0/1 - 560 ;0/1 - 561 ;0/1 - 562 ;0/1 - 563 ;0/1 - 564 ;0/1 - 565 ;0/1 - 566 ;0/1 - 567 ;0/1 - 568 ;0/1 - 569 ;0/1 - 570 ;0/1 - 571 ;0/1 - 572 ;0/1 - 573 ;0/1 - 574 ;0/1 - 575 ;0/1 - 576 ;0/1 - 577 ;0/1 - 578 ;0/1 - 579 ;0/1 - 580 ;0/1 - 581 ;0/1 - 582 ;0/1 - 583 ;0/1 - 584 ;0/1 - 585 ;0/1 - 586 ;0/1 - 587 ;0/1 - 588 ;0/1 - 589 ;0/1 - 590 ;0/1 - 591 ;0/1 - 592 ;0/1 - 593 ;0/1 - 594 ;0/1 - 595 ;0/1 - 596 ;0/1 - 597 ;0/1 - 598 ;0/1 - 599 ;0/1 - 600 ;0/1 - 601 ;0/1 - 602 ;0/1 - 603 ;0/1 - 604 ;0/1 - 605 ;0/1 - 606 ;0/1 - 607 ;0/1 - 608 ;0/1 - 609 ;0/1 - 610 ;0/1 - 611 ;0/1 - 612 ;0/1 - 613 ;0/1 - 614 ;0/1 - 615 ;0/1 - 616 ;0/1 - 617 ;0/1 - 618 ;0/1 - 619 ;0/1 - 620 ;0/1 - 621 ;0/1 - 622 ;0/1 - 623 ;0/1 - 624 ;0/1 - 625 ;0/1 - 626 ;0/1 - 627 ;0/1 - 628 ;0/1 - 629 ;0/1 - 630 ;0/1 - 631 ;0/1 - 632 ;0/1 - 633 ;0/1 - 634 ;0/1 - 635 ;0/1 - 636 ;0/1 - 637 ;0/1 - 638 ;0/1 - 639 ;0/1 - 640 ;0/1 - 641 ;0/1 - 642 ;0/1 - 643 ;0/1 - 644 ;0/1 - 645 ;0/1 - 646 ;0/1 - 647 ;0/1 - 648 ;0/1 - 649 ;0/1 - 650 ;0/1 - 651 ;0/1 - 652 ;0/1 - 653 ;0/1 - 654 ;0/1 - 655 ;0/1 - 656 ;0/1 - 657 ;0/1 - 658 ;0/1 - 659 ;0/1 - 660 ;0/1 - 661 ;0/1 - 662 ;0/1 - 663 ;0/1 - 664 ;0/1 - 665 ;0/1 - 666 ;0/1 - 667 ;0/1 - 668 ;0/1 - 669 ;0/1 - 670 ;0/1 - 671 ;0/1 - 672 ;0/1 - 673 ;0/1 - 674 ;0/1 - 675 ;0/1 - 676 ;0/1 - 677 ;0/1 - 678 ;0/1 - 679 ;0/1 - 680 ;0/1 - 681 ;0/1 - 682 ;0/1 - 683 ;0/1 - 684 ;0/1 - 685 ;0/1 - 686 ;0/1 - 687 ;0/1 - 688 ;0/1 - 689 ;0/1 - 690 ;0/1 - 691 ;0/1 - 692 ;0/1 - 693 ;0/1 - 694 ;0/1 - 695 ;0/1 - 696 ;0/1 - 697 ;0/1 - 698 ;0/1 - 699 ;0/1 - 700 ;0/1 - 701 ;0/1 - 702 ;0/1 - 703 ;0/1 - 704 ;0/1 - 705 ;0/1 - 706 ;0/1 - 707 ;0/1 - 708 ;0/1 - 709 ;0/1 - 710 ;0/1 - 711 ;0/1 - 712 ;0/1 - 713 ;0/1 - 714 ;0/1 - 715 ;0/1 - 716 ;0/1 - 717 ;0/1 - 718 ;0/1 - 719 ;0/1 - 720 ;0/1 - 721 ;0/1 - 722 ;0/1 - 723 ;0/1 - 724 ;0/1 - 725 ;0/1 - 726 ;0/1 - 727 ;0/1 - 728 ;0/1 - 729 ;0/1 - 730 ;0/1 - 731 ;0/1 - 732 ;0/1 - 733 ;0/1 - 734 ;0/1 - 735 ;0/1 - 736 ;0/1 - 737 ;0/1 - 738 ;0/1 - 739 ;0/1 - 740 ;0/1 - 741 ;0/1 - 742 ;0/1 - 743 ;0/1 - 744 ;0/1 - 745 ;0/1 - 746 ;0/1 - 747 ;0/1 - 748 ;0/1 - 749 ;0/1 - 750 ;0/1 - 751 ;0/1 - 752 ;0/1 - 753 ;0/1 - 754 ;0/1 - 755 ;0/1 - 756 ;0/1 - 757 ;0/1 - 758 ;0/1 - 759 ;0/1 - 760 ;0/1 - 761 ;0/1 - 762 ;0/1 - 763 ;0/1 - 764 ;0/1 - 765 ;0/1 - 766 ;0/1 - 767 ;0/1 - 768 ;0/1 - 769 ;0/1 - 770 ;0/1 - 771 ;0/1 - 772 ;0/1 - 773 ;0/1 - 774 ;0/1 - 775 ;0/1 - 776 ;0/1 - 777 ;0/1 - 778 ;0/1 - 779 ;0/1 - 780 ;0/1 - 781 ;0/1 - 782 ;0/1 - 783 ;0/1 - 784 ;0/1 - 785 ;0/1 - 786 ;0/1 - 787 ;0/1 - 788 ;0/1 - 789 ;0/1 - 790 ;0/1 - 791 ;0/1 - 792 ;0/1 - 793 ;0/1 - 794 ;0/1 - 795 ;0/1 - 796 ;0/1 - 797 ;0/1 - 798 ;0/1 - 799 ;0/1 - 800 ;0/1 - 801 ;0/1 - 802 ;0/1 - 803 ;0/1 - 804 ;0/1 - 805 ;0/1 - 806 ;0/1 - 807 ;0/1 - 808 ;0/1 - 809 ;0/1 - 810 ;0/1 - 811 ;0/1 - 812 ;0/1 - 813 ;0/1 - 814 ;0/1 - 815 ;0/1 - 816 ;0/1 - 817 ;0/1 - 818 ;0/1 - 819 ;0/1 - 820 ;0/1 - 821 ;0/1 - 822 ;0/1 - 823 ;0/1 - 824 ;0/1 - 825 ;0/1 - 826 ;0/1 - 827 ;0/1 - 828 ;0/1 - 829 ;0/1 - 830 ;0/1 - 831 ;0/1 - 832 ;0/1 - 833 ;0/1 - 834 ;0/1 - 835 ;0/1 - 836 ;0/1 - 837 ;0/1 - 838 ;0/1 - 839 ;0/1 - 840 ;0/1 - 841 ;0/1 - 842 ;0/1 - 843 ;0/1 - 844 ;0/1 - 845 ;0/1 - 846 ;0/1 - 847 ;0/1 - 848 ;0/1 - 849 ;0/1 - 850 ;0/1 - 851 ;0/1 - 852 ;0/1 - 853 ;0/1 - 854 ;0/1 - 855 ;0/1 - 856 ;0/1 - 857 ;0/1 - 858 ;0/1 - 859 ;0/1 - 860 ;0/1 - 861 ;0/1 - 862 ;0/1 - 863 ;0/1 - 864 ;0/1 - 865 ;0/1 - 866 ;0/1 - 867 ;0/1 - 868 ;0/1 - 869 ;0/1 - 870 ;0/1 - 871 ;0/1 - 872 ;0/1 - 873 ;0/1 - 874 ;0/1 - 875 ;0/1 - 876 ;0/1 - 877 ;0/1 - 878 ;0/1 - 879 ;0/1 - 880 ;0/1 - 881 ;0/1 - 882 ;0/1 - 883 ;0/1 - 884 ;0/1 - 885 ;0/1 - 886 ;0/1 - 887 ;0/1 - 888 ;0/1 - 889 ;0/1 - 890 ;0/1 - 891 ;0/1 - 892 ;0/1 - 893 ;0/1 - 894 ;0/1 - 895 ;0/1 - 896 ;0/1 - 897 ;0/1 - 898 ;0/1 - 899 ;0/1 - 900 ;0/1 - 901 ;0/1 - 902 ;0/1 - 903 ;0/1 - 904 ;0/1 - 905 ;0/1 - 906 ;0/1 - 907 ;0/1 - 908 ;0/1 - 909 ;0/1 - 910 ;0/1 - 911 ;0/1 - 912 ;0/1 - 913 ;0/1 - 914 ;0/1 - 915 ;0/1 - 916 ;0/1 - 917 ;0/1 - 918 ;0/1 - 919 ;0/1 - 920 ;0/1 - 921 ;0/1 - 922 ;0/1 - 923 ;0/1 - 924 ;0/1 - 925 ;0/1 - 926 ;0/1 - 927 ;0/1 - 928 ;0/1 - 929 ;0/1 - 930 ;0/1 - 931 ;0/1 - 932 ;0/1 - 933 ;0/1 - 934 ;0/1 - 935 ;0/1 - 936 ;0/1 - 937 ;0/1 - 938 ;0/1 - 939 ;0/1 - 940 ;0/1 - 941 ;0/1 - 942 ;0/1 - 943 ;0/1 - 944 ;0/1 - 945 ;0/1 - 946 ;0/1 - 947 ;0/1 - 948 ;0/1 - 949 ;0/1 - 950 ;0/1 - 951 ;0/1 - 952 ;0/1 - 953 ;0/1 - 954 ;0/1 - 955 ;0/1 - 956 ;0/1 - 957 ;0/1 - 958 ;0/1 - 959 ;0/1 - 960 ;0/1 - 961 ;0/1 - 962 ;0/1 - 963 ;0/1 - 964 ;0/1 - 965 ;0/1 - 966 ;0/1 - 967 ;0/1 - 968 ;0/1 - 969 ;0/1 - 970 ;0/1 - 971 ;0/1 - 972 ;0/1 - 973 ;0/1 - 974 ;0/1 - 975 ;0/1 - 976 ;0/1 - 977 ;0/1 - 978 ;0/1 - 979 ;0/1 - 980 ;0/1 - 981 ;0/1 - 982 ;0/1 - 983 ;0/1 - 984 ;0/1 - 985 ;0/1 - 986 ;0/1 - 987 ;0/1 - 988 ;0/1 - 989 ;0/1 - 990 ;0/1 - 991 ;0/1 - 992 ;0/1 - 993 ;0/1 - 994 ;0/1 - 995 ;0/1 - 996 ;0/1 - 997 ;0/1 - 998 ;0/1 - 999 ;0/1 - 1000 ;0/1 - 1001 ;0/1 - 1002 ;0/1 - 1003 ;0/1 - 1004 ;0/1 - 1005 ;0/1 - 1006 ;0/1 - 1007 ;0/1 - 1008 ;0/1 - 1009 ;0/1 - 1010 ;0/1 - 1011 ;0/1 - 1012 ;0/1 - 1013 ;0/1 - 1014 ;0/1 - 1015 ;0/1 - 1016 ;0/1 - 1017 ;0/1 - 1018 ;0/1 - 1019 ;0/1 - 1020 ;0/1 - 1021 ;0/1 - 1022 ;0/1 - 1023 ;

10

Ac: 10/1 R: 0;

w - write memory

s - not write memory

f - run to end

q – exit  
Это один шаг с выводом всех значений памяти на экран. Далее аналогично…

## ЗАКЛЮЧЕНИЕ

В данном курсовом проекте мною была написана программа, реализующая симулятор Johniac.

Она поддерживает базовый функционал, необходимый для корректной работы программы, а также полностью соответствует заданным мне в индивидуальном задании требованиям.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Arthur W. Burks. Preliminary discussion of the logical design of an electronic computing instrument [Электронный ресурс] 1946г. – Режим доступа: <https://library.ias.edu/files/Prelim_Disc_Logical_Design.pdf> – (Дата обращения: 20.05.2018).
2. M. Уэйт, С. Прата, Д. Мартин - Язык Си руководство для начинающих [Электронный ресурс] 1988г. – Режим доступа: <http://storage.ded32.net.ru/Lib/TX/CPrimerPlus.pdf> – (Дата обращения: 20.05.2018).
3. Структуры – сайт – Режим доступа: <https://learnc.info/c/structures.html> - (Дата обращения: 05.06.2018).
4. Черновик стандарта C11.
5. Различные циклы уроков (tutorials) по make (например, http://habrahabr.ru/post/211751).