Национальный исследовательский ядерный университет «МИФИ»
Институт интеллектуальных кибернетических систем
Кафедра №12 «Компьютерные системы и технологии»

ОТЧЕТ

О выполнении лабораторной работы №4

«Работа с массивами данных»

Студент: Кругликова М. В.
Группа: Б22-504
Преподаватель: Комаров Т. И.

Москва – 2023

1. **Формулировка индивидуального задания**
   Вариант №26
   Каждое слово строки, которое является записью некоторого числа в десятичной системе счисления, заменить на представление данного числа в шестнадцатеричной системе счисления с добавлением префикса «0x»

2. **Описание использованных типов данных**
   При выполнении данной лабораторной работы использовались такие типы данных, как:
   1. Встроенный тип данных char *, предназначенный для работы с указателями на символы и массивы из них
   2. Тип данных clock_t из библиотеки time.h, предназначенный для представления времени
   3. Встроенный тип данных double, предназначенный для работы с вещественными числами
   4. Встроенный тип данных int, предназначенный для работы с целыми числами
   5. Встроенный тип данных char, предназначенный для работы с символами
   6. Беззнаковый целый тип size_t, предназначенный для представления размера любого объекта в памяти
3. **Описание использованного алгоритма**

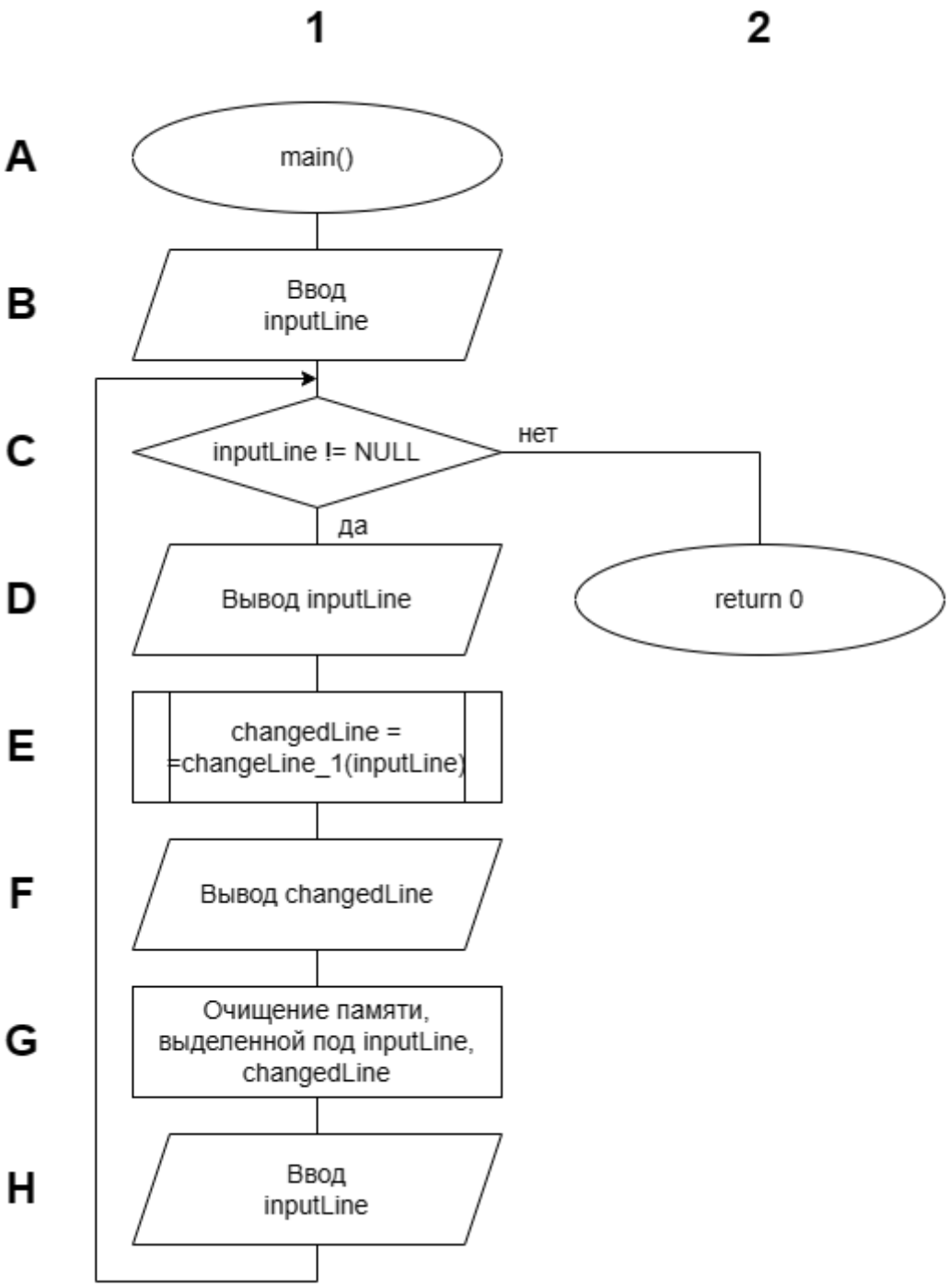

Рисунок 1. Блок-схема алгоритма работы функции main (файл main4_1.c)

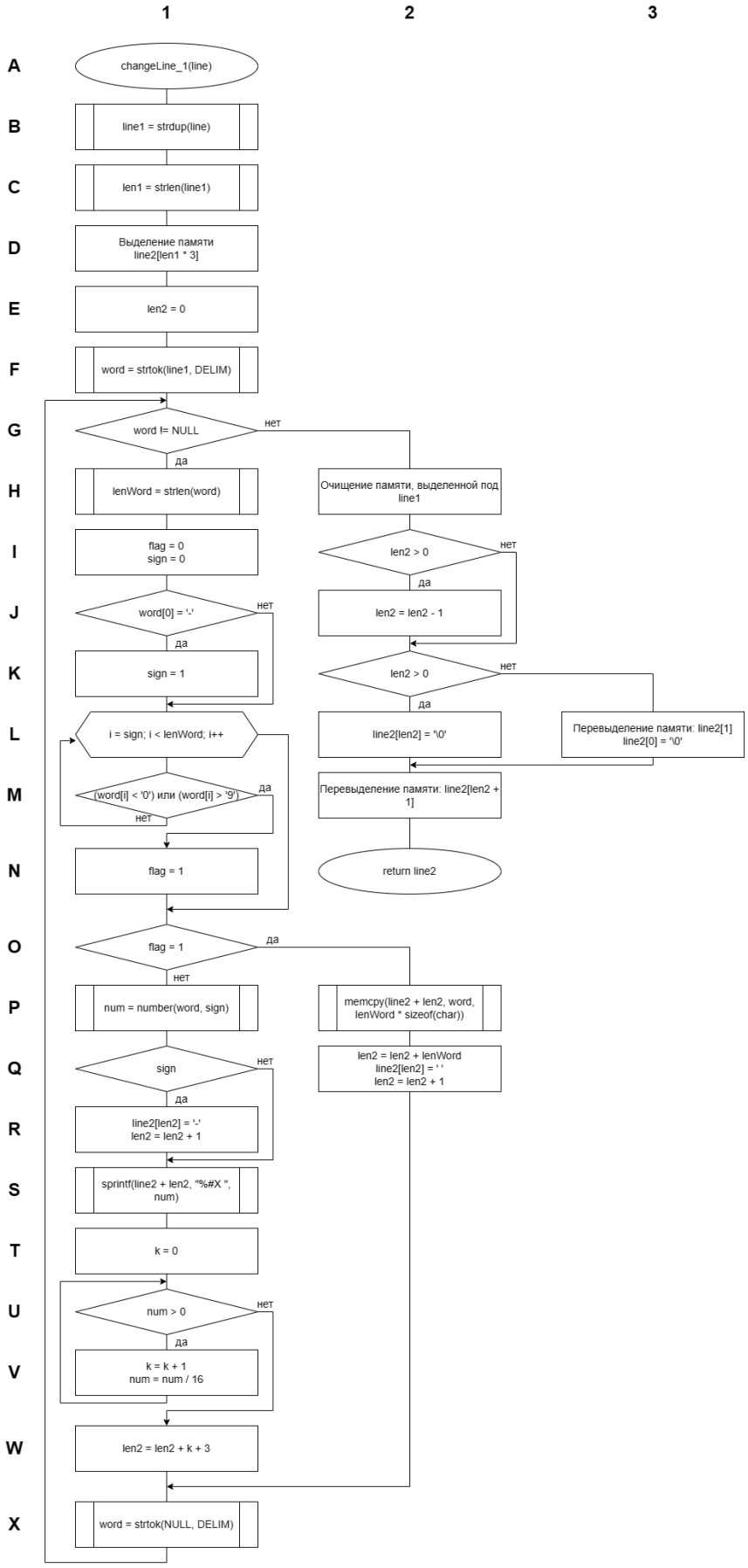

Рисунок 2. Блок-схема алгоритма работы функции changeLine_1 (файл changeLine_1.c)

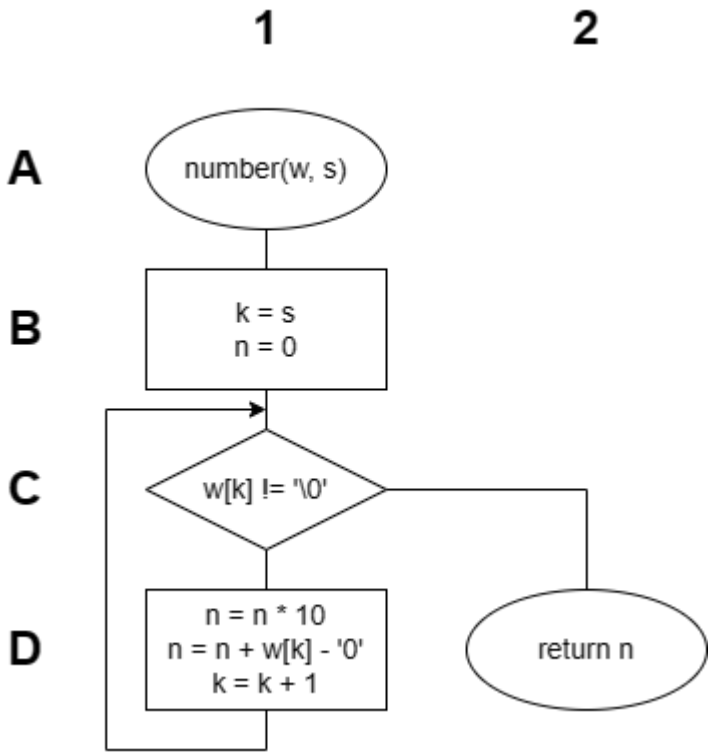

Рисунок 3. Блок-схема алгоритма работы функции number (файлы changeLine_1.c, changeLine_2.c)

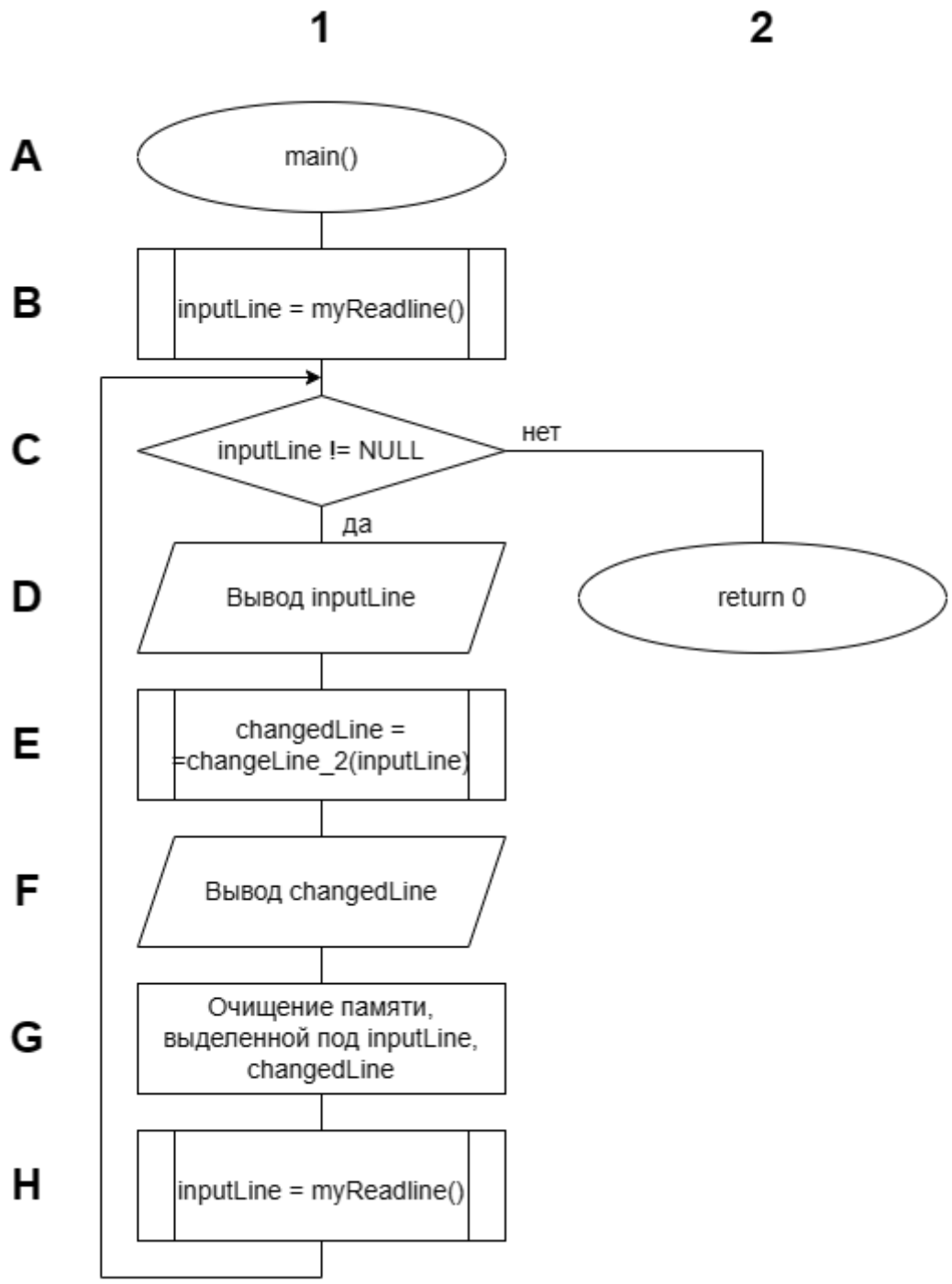

Рисунок 4. Блок-схема алгоритма работы функции main (файл main4_2.c)

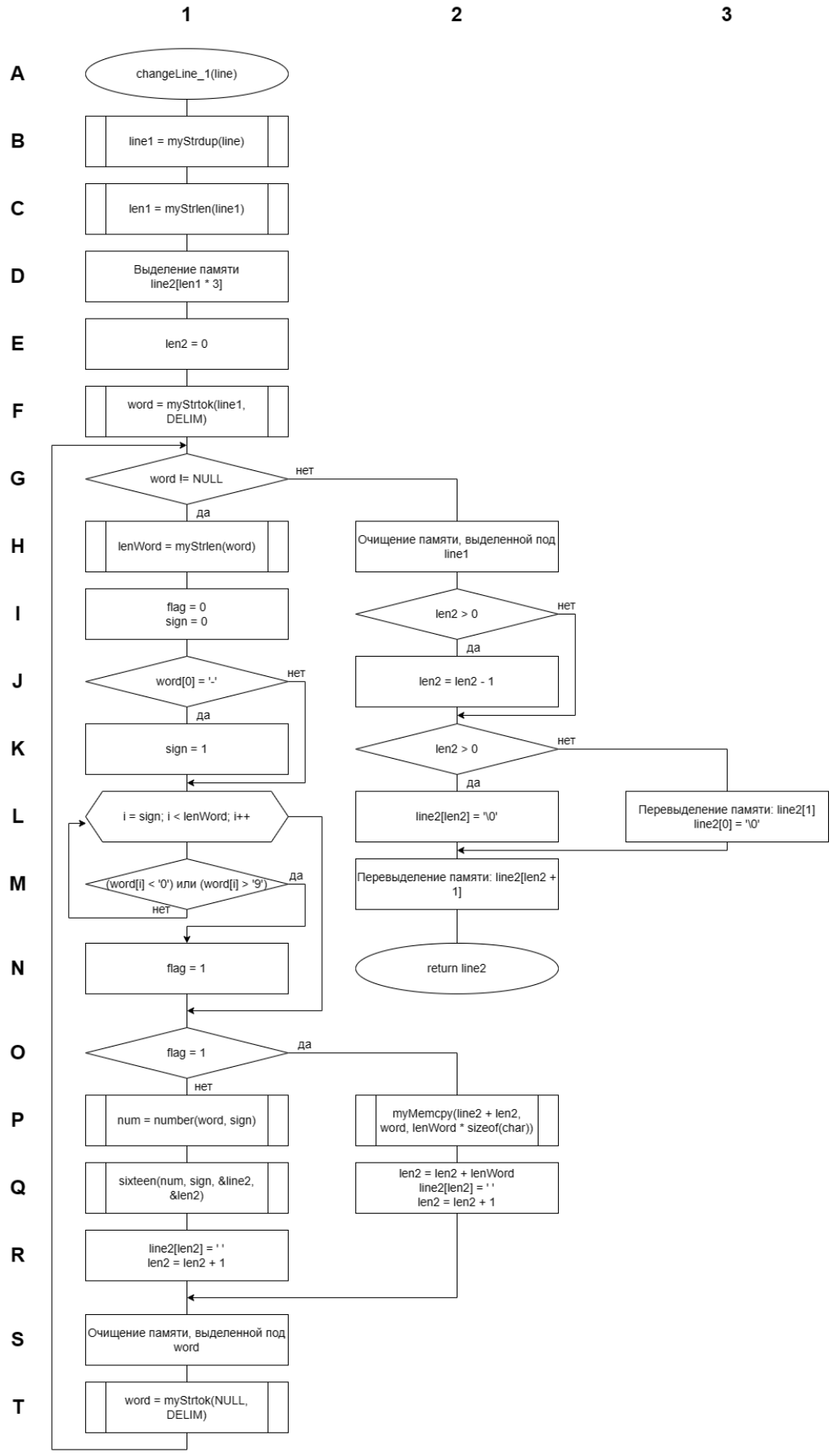

Рисунок 5. Блок-схема алгоритма работы функции changeLine_2 (файл changeLine_2.c)

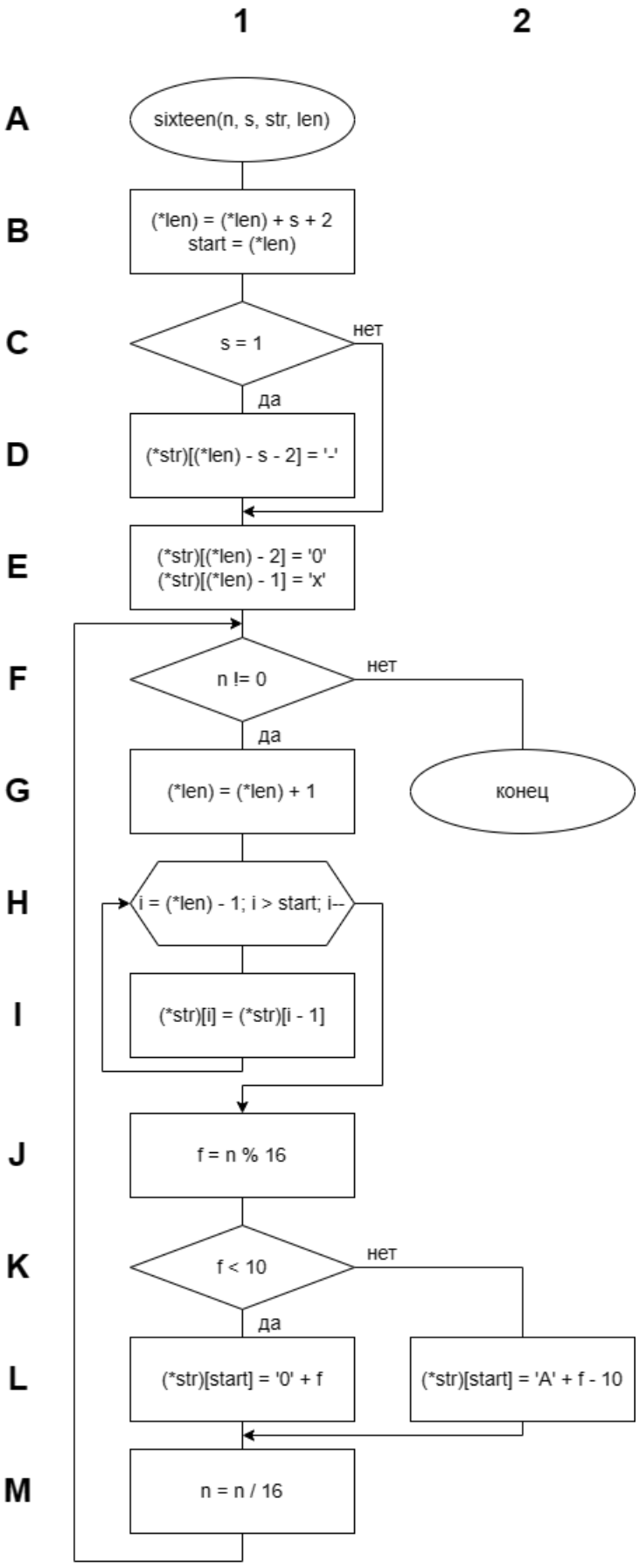

Рисунок 6. Блок-схема алгоритма работы функции sixteen (файл changeLine_2.c)

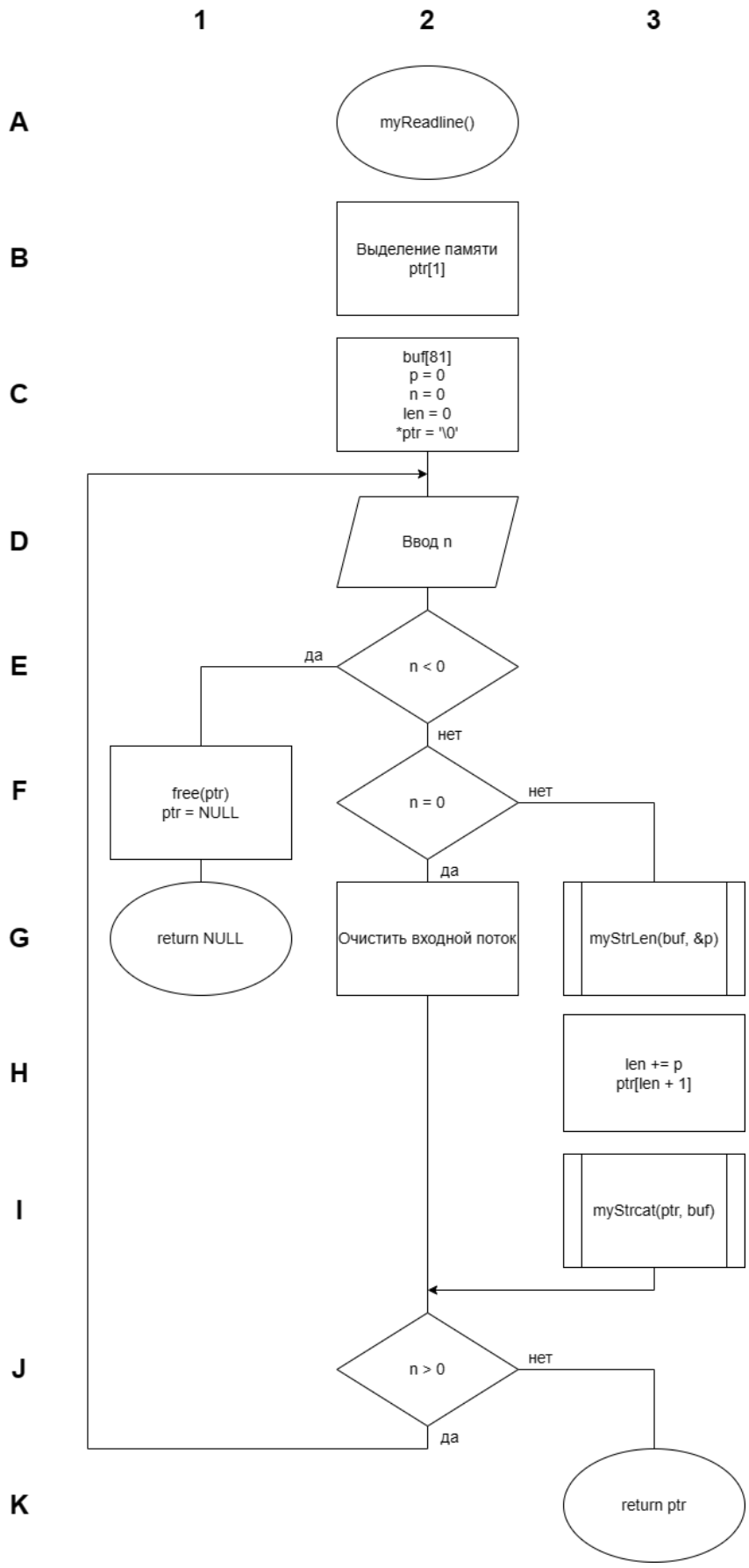

Рисунок 7. Блок-схема алгоритма работы функции myReadline (файл myReadline.c)

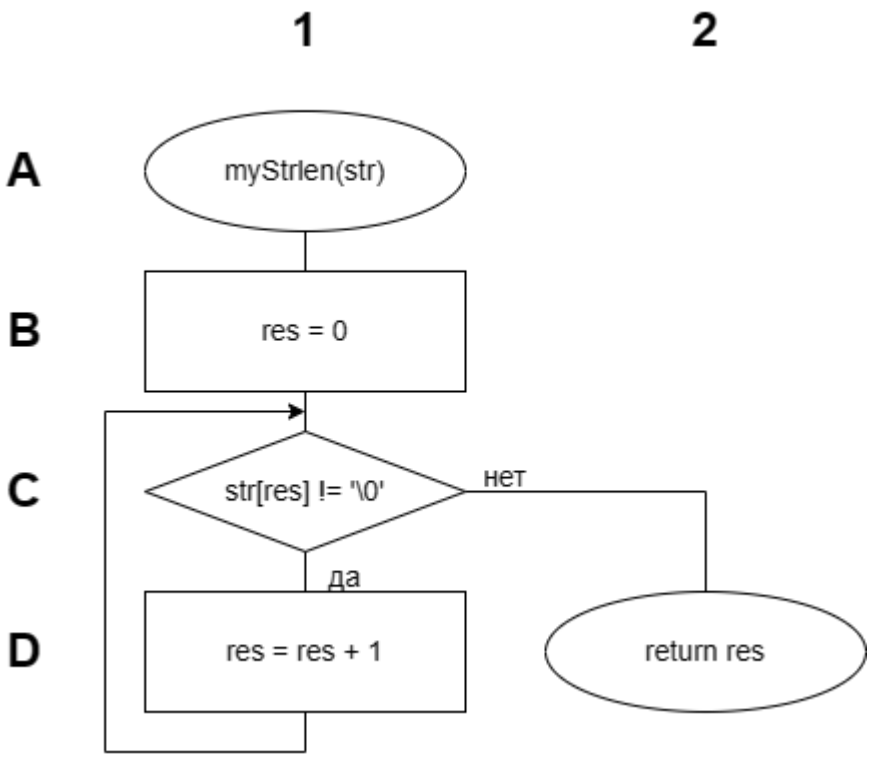

Рисунок 8. Блок-схема алгоритма работы функции myStrlen (файл myString.c)

Рисунок 9. Блок-схема алгоритма работы функции myStrdup (файл myString.c)

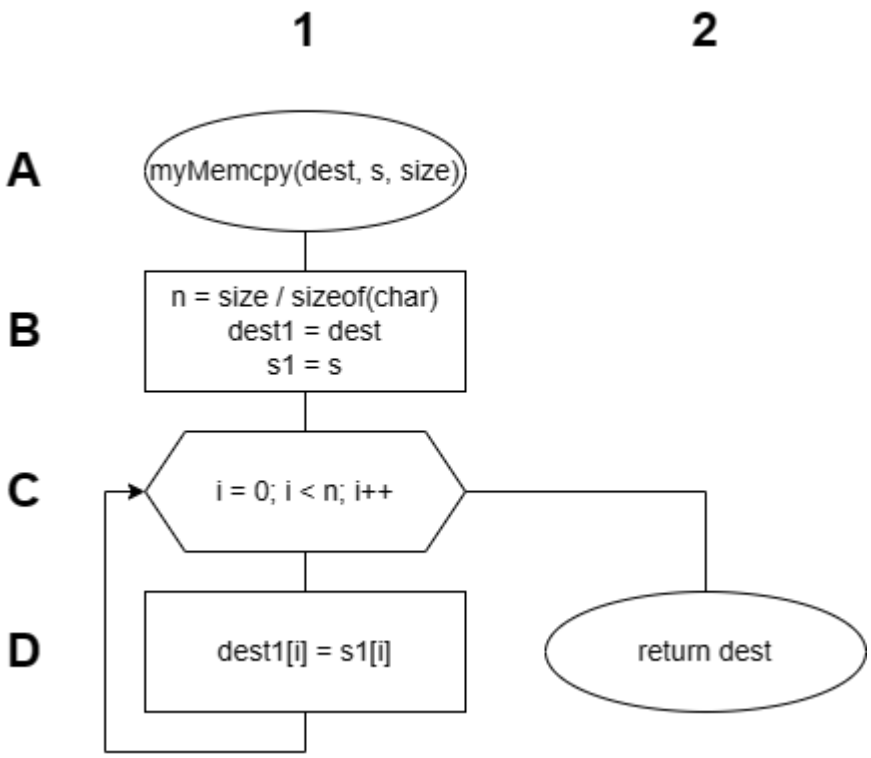

Рисунок 10. Блок-схема алгоритма работы функции myMemcpy (файл myString.c)

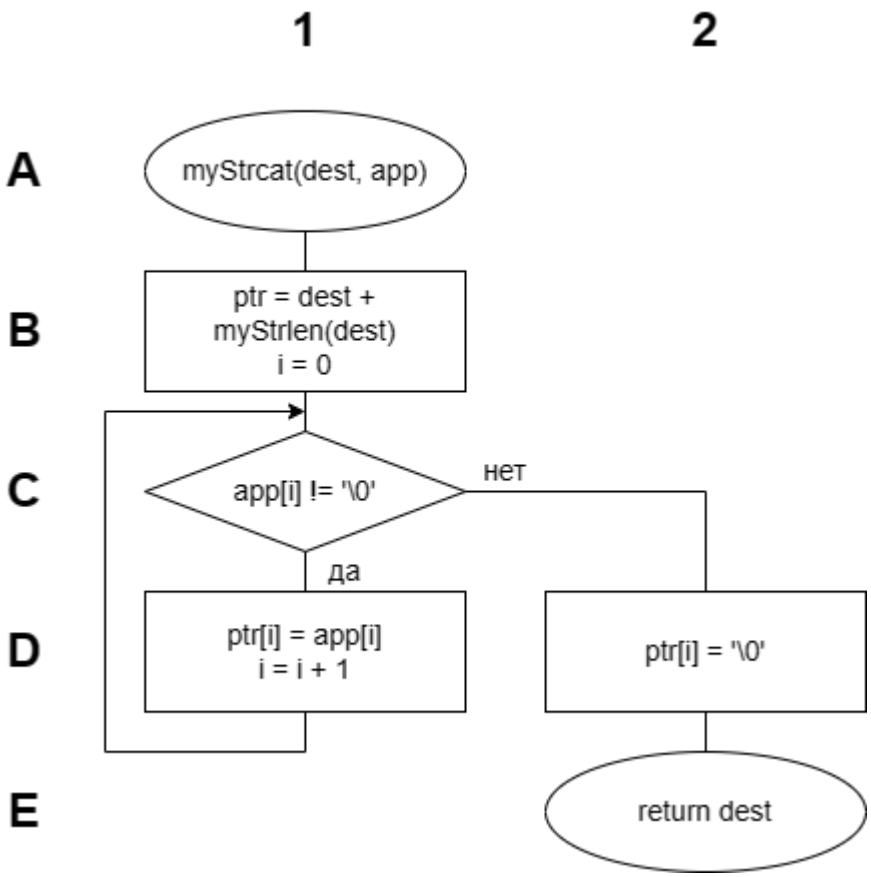

Рисунок 11. Блок-схема алгоритма работы функции myStrcat (файл myString.c)

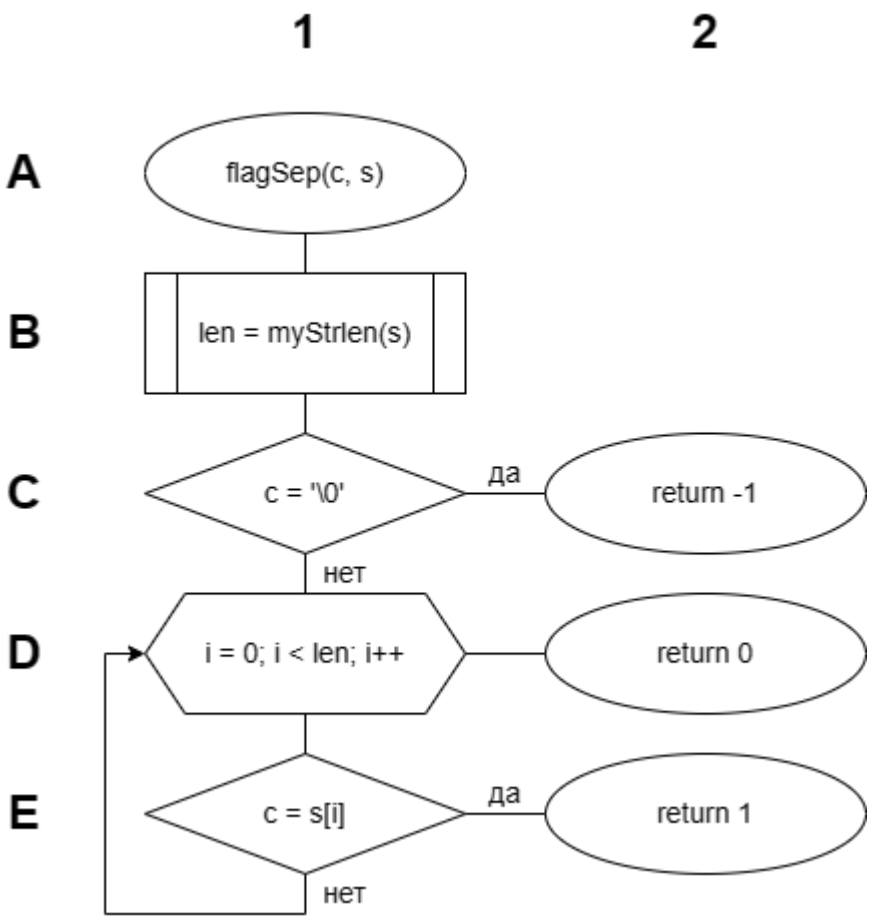

Рисунок 12. Блок-схема алгоритма работы функции flagSep (файл myString.c)

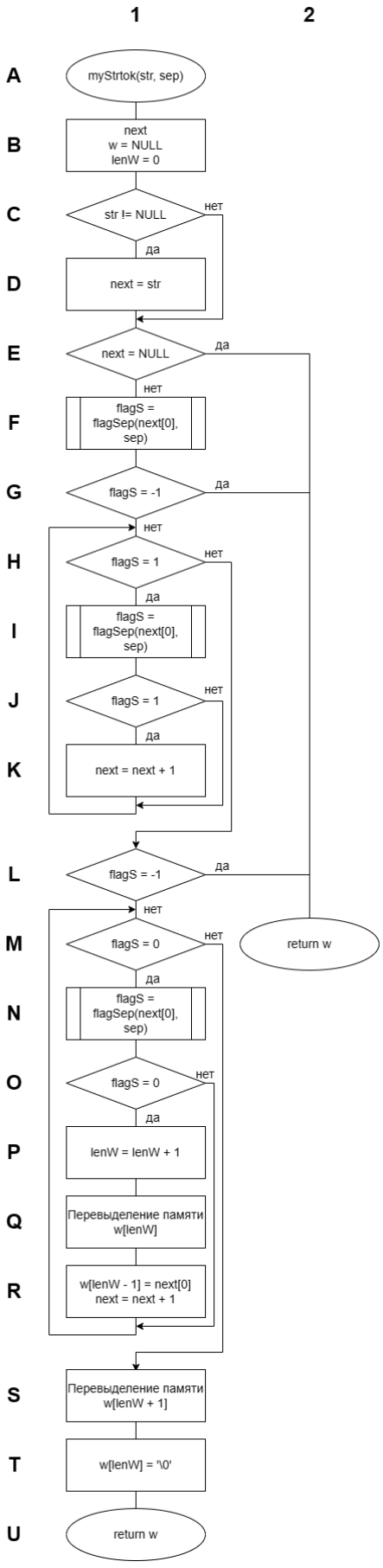

Рисунок 13. Блок-схема алгоритма работы функции myStrtok (файл myString.c)

4. Исходные коды разработанных программ

Листинг 1. Исходные коды программы main (файл main4_1.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <readline/readline.h>
#include <time.h>
#include "changeLine_1.h"

int main() {
    printf("Введите строку: ");
    char *inputLine = readline("");
    while (inputLine != NULL) {
        printf("Введенная строка: \"%s\"\n",
inputLine);
        clock_t start = clock();
        char *changedLine =
changeLine_1(inputLine);
        clock_t end = clock();
        printf("Новая строка: \"%s\"\n",
changedLine);
        double time = (double)(end -
start)/CLOCKS_PER_SEC;
        printf("Время выполнения программы: %.10lf
\n", time);
        free(inputLine);
        free(changedLine);
        printf("Введите строку: ");
        inputLine = readline("");
    }
    return 0;
}
```

Листинг 2. Исходные коды программы changeLine_1 (файл changeLine_1.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "changeLine_1.h"

#define DELIM " \t"
```

```
char *changeLine_1(const char *line) {
    char *line1 = strdup(line);
    int len1 = strlen(line1);
    char *line2 = (char *) calloc(len1 * 3,
sizeof(char));
    int len2 = 0;
    char *word = strtok(line1, DELIM);
    while (word != NULL) {
        int lenWord = strlen(word);
        int flag = 0;
        int sign = 0;
        if (word[0] == '-') {
            sign = 1;
        }
        for (int i = sign; i < lenWord; i++) {
            if ((word[i] < '0') || (word[i] >
'9')) {
                flag = 1;
                break;
            }
        }
        if (flag == 1) {
            memcpy(line2 + len2, word, lenWord *
sizeof(char));
            len2 += lenWord;
            line2[len2] = ' ';
            len2++;
        } else {
            int num = number(word, sign);
            if (sign) {
                line2[len2] = '-';
                len2++;
            }
            sprintf(line2 + len2, "%#X ", num);
            int k = 0;
            while (num > 0) {
                k++;
                num /= 16;
            }
            len2 += k + 3;
        }
```

```
        word = strtok(NULL, DELIM);
    }
    free(line1);
    if (len2 > 0) {
        --len2;
    }
    if (len2 > 0) {
        line2[len2] = '\0';
    } else {
        line2 = realloc(line2, sizeof(char));
        line2[0] = '\0';
    }
    line2 = realloc(line2, (len2 + 1) *
sizeof(char));
    return line2;
}

int number(const char *w, int s) {
    int k = s;
    int n = 0;
    while(w[k] != '\0') {
        n *= 10;
        n += w[k] - '0';
        k++;
    }
    return n;
}
```

Листинг 3. Исходные коды программы main (файл main4_2.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "myReadline.h"
#include "changeLine_2.h"

int main() {
    printf("Введите строку: ");
    char *inputLine = myReadline("");
    while (inputLine != NULL) {
        printf("Введенная строка: \"%s\"\n",
inputLine);
        clock_t start = clock();
```

```
        char *changedLine =
changeLine_2(inputLine);
        clock_t end = clock();
        printf("Новая строка: \"%s\"\n",
changedLine);
        double time = (double)(end -
start)/CLOCKS_PER_SEC;
        printf("Время выполнения программы: %.10lf
\n", time);
        free(inputLine);
        free(changedLine);
        printf("Введите строку: ");
        inputLine = myReadline("");
    }
    return 0;
}
```

Листинг 4. Исходные коды программы changeLine_2 (файл changeLine_2.c)

```
#include <stdio.h>
#include <stdlib.h>
#include "myString.h"
#include "changeLine_2.h"

#define DELIM " \t"

char *changeLine_2(const char *line) {
    char *line1 = myStrdup(line);
    int len1 = myStrlen(line1);
    char *line2 = (char *) calloc(len1 * 3,
sizeof(char));
    int len2 = 0;
    char *word = myStrtok(line1, DELIM);
    while (word != NULL) {
        int lenWord = myStrlen(word);
        int flag = 0;
        int sign = 0;
        if (word[0] == '-') {
            sign = 1;
        }
        for (int i = sign; i < lenWord; i++) {
```

```
            if ((word[i] < '0') || (word[i] >
'9')) {
                flag = 1;
                break;
            }
        }
        if (flag == 1) {
            myMemcpy(line2 + len2, word, lenWord *
sizeof(char));
            len2 += lenWord;
            line2[len2] = ' ';
            len2++;
        } else {
            int num = number(word, sign);
            sixteen(num, sign, &line2, &len2);
            line2[len2] = ' ';
            len2++;
        }
        free(word);
        word = myStrtok(NULL, DELIM);
    }
    free(line1);
    if (len2 > 0) {
        --len2;
    }
    if (len2 > 0) {
        line2[len2] = '\0';
    } else {
        line2 = realloc(line2, sizeof(char));
        line2[0] = '\0';
    }
    line2 = realloc(line2, (len2 + 1) *
sizeof(char));
    return line2;
}

int number(const char *w, int s) {
    int k = s;
    int n = 0;
    while(w[k] != '\0') {
        n *= 10;
        n += w[k] - '0';
```

```
            k++;
        }
        return n;
}

void sixteen(int n, int s, char **str, int *len) {
        (*len) += (s + 2);
        int start = (*len);
        if (s == 1) {
            (*str)[(*len) - s - 2] = '-';
        }
        (*str)[(*len) - 2] = '0';
        (*str)[(*len) - 1] = 'x';
        while (n != 0) {
            (*len)++;
            for (int i = (*len) - 1; i > start; i--) {
                (*str)[i] = (*str)[i - 1];
            }
            int f = n % 16;
            if (f < 10) {
                (*str)[start] = '0' + f;
            } else {
                (*str)[start] = 'A' + f - 10;
            }
            n /= 16;
        }
}
```

Листинг 5. Исходные коды программы myReadline (файл myReadline.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "myString.h"

char *myReadline() {
        char *ptr = (char *) malloc(1);
        char buf[81];
        int n, len = 0;
        *ptr = '\0';
        do {
            n = scanf("%80[^\n]", buf);
            if (n < 0) {
```

```
            free(ptr);
            ptr = NULL;
            continue;
        }
        if (n == 0) {
            scanf("%*c");
        } else {
            len += myStrlen(buf);
            ptr = (char *) realloc(ptr, len + 1);
            myStrcat(ptr, buf);
        }
    } while (n > 0);
    return ptr;
}
```

Листинг 6. Исходные коды программы myString (файл myString.c)

```
#include <stdio.h>
#include <stdlib.h>
#include "myString.h"

size_t myStrlen(const char *str) {
    int res = 0;
    while (str[res] != '\0') {
        res ++;
    }
    return res;
}

char *myStrdup(const char *str) {
    int len = myStrlen(str);
    char *res = (char *) malloc((len + 1)
*sizeof(char));
    myMemcpy(res, str, len *sizeof(char));
    res[len] = '\0';
    return res;
}

void *myMemcpy(void *dest, const void *s, size_t
size) {
    int n = size / sizeof(char);
    char *dest1 = (char *) dest;
    char *s1 = (char *) s;
```

```
    for (int i = 0; i < n; i++) {
        dest1[i] = s1[i];
    }
    return dest;
}

char *myStrcat(char *dest, const char *app) {
    char *ptr = dest + myStrlen(dest);
    int i = 0;
    while (app[i] != '\0') {
        ptr[i] = app[i];
        i++;
    }
    ptr[i] = '\0';
    return dest;
}

int flagSep(const char c, const char *s) {
    int len = myStrlen(s);
    if (c == '\0'){
        return -1;
    }
    for (int i = 0; i < len; i++) {
        if (c == s[i]) {
            return 1;
        }
    }
    return 0;
}

char *myStrtok(char *str, const char *sep) {
    static char *next;
    char *w = NULL;
    int lenW = 0;
    if (str != NULL) {
        next = str;
    }
    if (next == NULL) {
        return w;
    }
    int flagS = flagSep(next[0], sep);
    if (flagS == -1) {
```

```
            return w;
        }
        while (flagS == 1) {
            flagS = flagSep(next[0], sep);
            if (flagS == 1) {
                next++;
            }
        }
        if (flagS == -1) {
            return w;
        }
        while (flagS == 0) {
            flagS = flagSep(next[0], sep);
            if (flagS == 0) {
                lenW++;
                w = realloc(w, lenW * sizeof(char));
                w[lenW - 1] = next[0];
                next++;
            }
        }
        w = realloc(w, (lenW + 1) * sizeof(char));
        w[lenW] = '\0';
        return w;
    }
```

5. Описание тестовых примеров

Таблица 1. Тестовые примеры работы программ lab4_1 и lab4_2

Введенная строка	Новая строка (ожидание)	Новая строка (реальность)	Время выполнения lab4_1	Время выполнения lab4_2
Пустая (enter)	пустая	пустая	0.002484	0.005743
“ “	пустая	пустая	0.000646	0.001292
“ 12345 qwerty -98765 “	“0X3039 qwerty -0X181CD”	“0X3039 qwerty -0X181CD”	0.007348	0.007368

6. Скриншоты

```
[kruglikova.mv@unix:~/inf/lab4/lab4_1]$ valgrind ./lab4_1
==4481== Memcheck, a memory error detector
==4481== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==4481== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==4481== Command: ./lab4_1
==4481==
Введенная строка: ""
Новая строка: ""
Время выполнения программы: 0.0024840000
Введите строку:
Введенная строка: "                         "
Новая строка: ""
Время выполнения программы: 0.0006460000
Введите строку:    12345     qwerty     -98765
Введенная строка: "   12345     qwerty     -98765   "
Новая строка: "0X3039 qwerty -0X181CD"
Время выполнения программы: 0.0073480000
Введите строку:
==4481==
==4481== HEAP SUMMARY:
==4481==     in use at exit: 204,072 bytes in 219 blocks
==4481==   total heap usage: 2,820 allocs, 2,601 frees, 380,580 bytes allocated
==4481==
==4481== LEAK SUMMARY:
==4481==    definitely lost: 0 bytes in 0 blocks
==4481==    indirectly lost: 0 bytes in 0 blocks
==4481==      possibly lost: 0 bytes in 0 blocks
==4481==    still reachable: 204,072 bytes in 219 blocks
==4481==         suppressed: 0 bytes in 0 blocks
==4481== Rerun with --leak-check=full to see details of leaked memory
==4481==
==4481== For lists of detected and suppressed errors, rerun with: -s
==4481== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[kruglikova.mv@unix:~/inf/lab4/lab4_1]$
```

Рисунок 14. Запуск программы lab4_1

```
Введите строку: [kruglikova.mv@unix:~/inf/lab4/lab4_2]$ valgrind ./lab4_2
==4384== Memcheck, a memory error detector
==4384== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==4384== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==4384== Command: ./lab4_2
==4384==
Введите строку:
Введенная строка: ""
Новая строка: ""
Время выполнения программы: 0.0057430000
Введите строку:
Введенная строка: "                       "
Новая строка: ""
Время выполнения программы: 0.0012920000
Введите строку:    12345     qwerty     -98765
Введенная строка: "   12345     qwerty     -98765   "
Новая строка: "0x3039 qwerty -0x181CD"
Время выполнения программы: 0.0073680000
Введите строку: ==4384==
==4384== HEAP SUMMARY:
==4384==     in use at exit: 0 bytes in 0 blocks
==4384==   total heap usage: 39 allocs, 39 frees, 2,441 bytes allocated
==4384==
==4384== All heap blocks were freed -- no leaks are possible
==4384==
==4384== For lists of detected and suppressed errors, rerun with: -s
==4384== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Рисунок 15. Запуск программы lab4_2

7. Выводы

В работе был изучен способ представления строк в виде вектора на физическом уровне. Также были изучены функции библиотеки string.h для работы со строками и написаны их реализации.

Из запусков программ для разных тестовых наборов видно, что программа lab4_1 (в которой использованы библиотечные функции) работает быстрее, чем lab4_2 (в которой библиотечные функции заменены реализациями без исполльзования библиотек string.h и readline/readline.h). Это объясняется тем, что библиотечные функции лучше оптимизированы и отлажены, также они пользуются различными низкоуровневыми преимуществами, поэтому работают быстрее аналогов.