Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №12 «Компьютерные системы и технологии»

ОТЧЁТ

О выполнении лабораторной работы № 4

"Работа со строками"

Студент: Гаврилов Д. А.

Группа: Б23-516

Преподаватель: Бабалова И. Ф.

Москва – 2023

1. Формулировка задания

Вариант №23

Из входного потока вводится произвольное количество строк произвольной длины. Каждая строка в общем случае содержит одно или более слов, разделенных пробелами и/или знаками табуляции. Завершение ввода определяется концом файла. Для каждой входной строки формируется новая выходная строка, в которую помещается результат. В полученной строке слова разделяются только одним пробелом, пробелов в её начале и в конце быть не должно. Введённая и сформированная строки выводятся на экран в двойных кавычках. В ходе выполнения лабораторной работы должны быть разработаны:

1. Программа, использующая функцию readline() из состава библиотеки GNU readline для ввода строк и функции стандартной библиотеки для их обработки (<string.h>).

2. Программа, идентичная п. 1, за исключением того, что все библиотечные функции заменены на собственную реализацию данных функций, представленную в отдельных файлах (например: mystring.h, mystring.c).

Отчётность по выполнению лабораторной работы должна включать:

1. Блок-схему алгоритма работы основной программы.

2. Блок-схемы алгоритмов работы функций по обработке строк.

3. Исходные коды всех программ.

4. Тестовые наборы для программ п. 1 и п. 2.

5. Сравнительный анализ времени, потраченного на решение задачи программами п. 1 и п. 2 (на конкретных примерах).

Примечания:

1. Каждая строка представлена на физическом уровне вектором.

2. Использование массивов переменной длины (VLA — variable length arrays) не допускается.

3. Ввод строк в п. 2 должен быть организован с помощью функции scanf() со спецификациями для ввода строк. Использование функций семейства gets(), getchar(), а также спецификаций %c и %m в scanf() не допускается.

4. Целочисленные и строковые константы, используемые в формулировках индивидуальных заданий, должны быть заданы в исходном коде с помощью директив препроцессора #define.

5. Программа должна корректным образом завершаться при обнаружении EOF — конца файла (в UNIX-подобных ОС инициируется нажатием клавиш Ctrl + D, в Windows — Ctrl + Z).

6. Логически законченные части алгоритма решения задачи должны быть оформлены в виде отдельных функций с параметрами.

7. Исходные коды программ должны быть логичным образом разбиты на несколько файлов (необходимо использовать как *.c-файлы, так и *.h-файлы).

8. Использование глобальных переменных не допускается.

9. Программы должна корректным образом работать с памятью. Для проверки необходимо использовать соответствующие программные средства, например, valgrind (при тестировании и отладке программ п. 1 и п. 2 необходимо запускать их командой вида valgrind ./lab4, а при анализе производительности — ./lab4).

Индивидуальное задание

Разбить строку на предложения по символам «.», «!», «?» и «...». Каждое предложение должно начинаться с новой строки с заглавной буквы.

2. Описание использованных типов данных

Целочисленный тип данных – int (спецификатор формата %d) для хранения целых чисел и длин динамических массивов/строк. Символьный тип данных - char – для работы с отдельными символами. Указатель на char (char *) – для хранения и создания строк.

3. Описание использованных алгоритмов

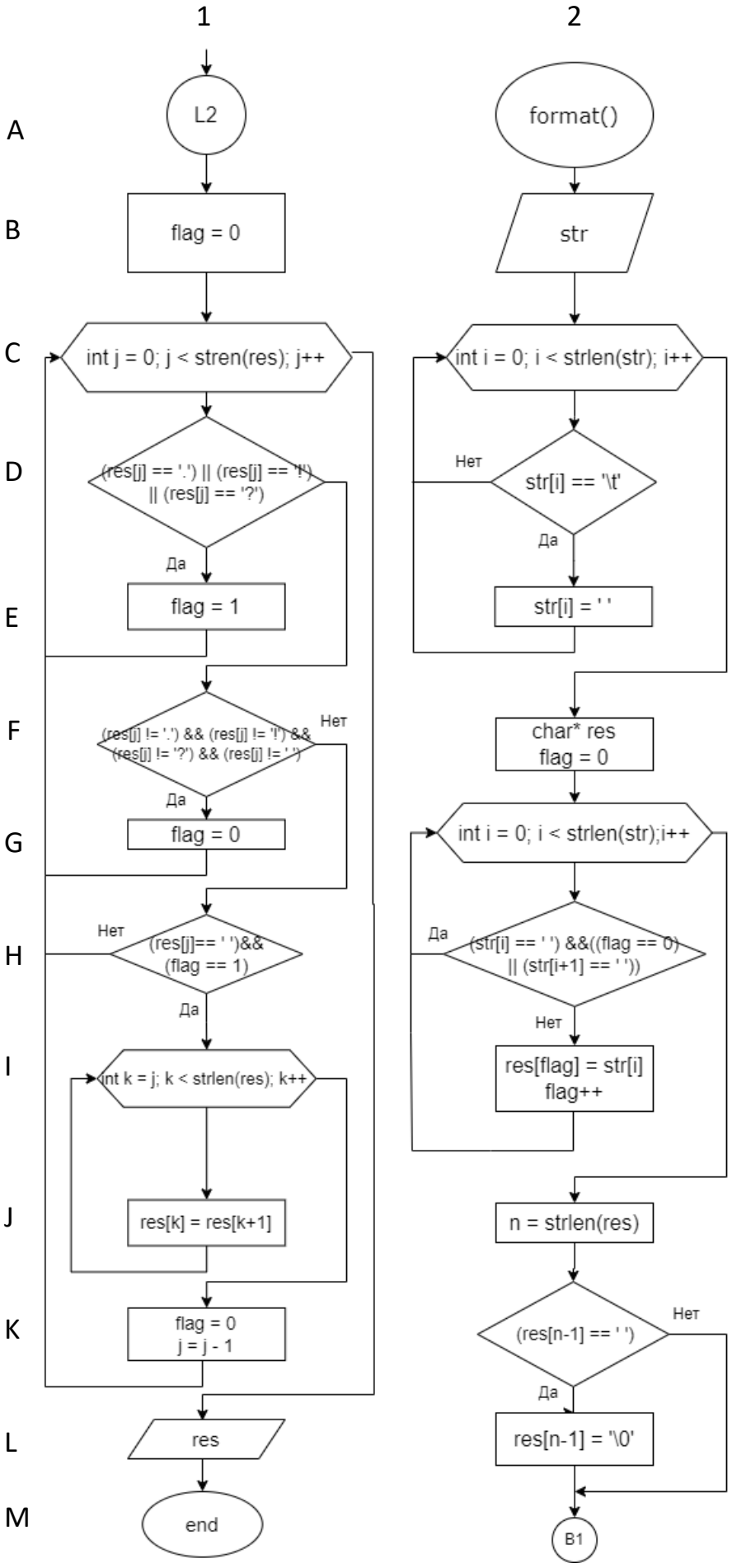


Рис. 1: Блок-схема алгоритма форматирование пробелов в строке format(str).

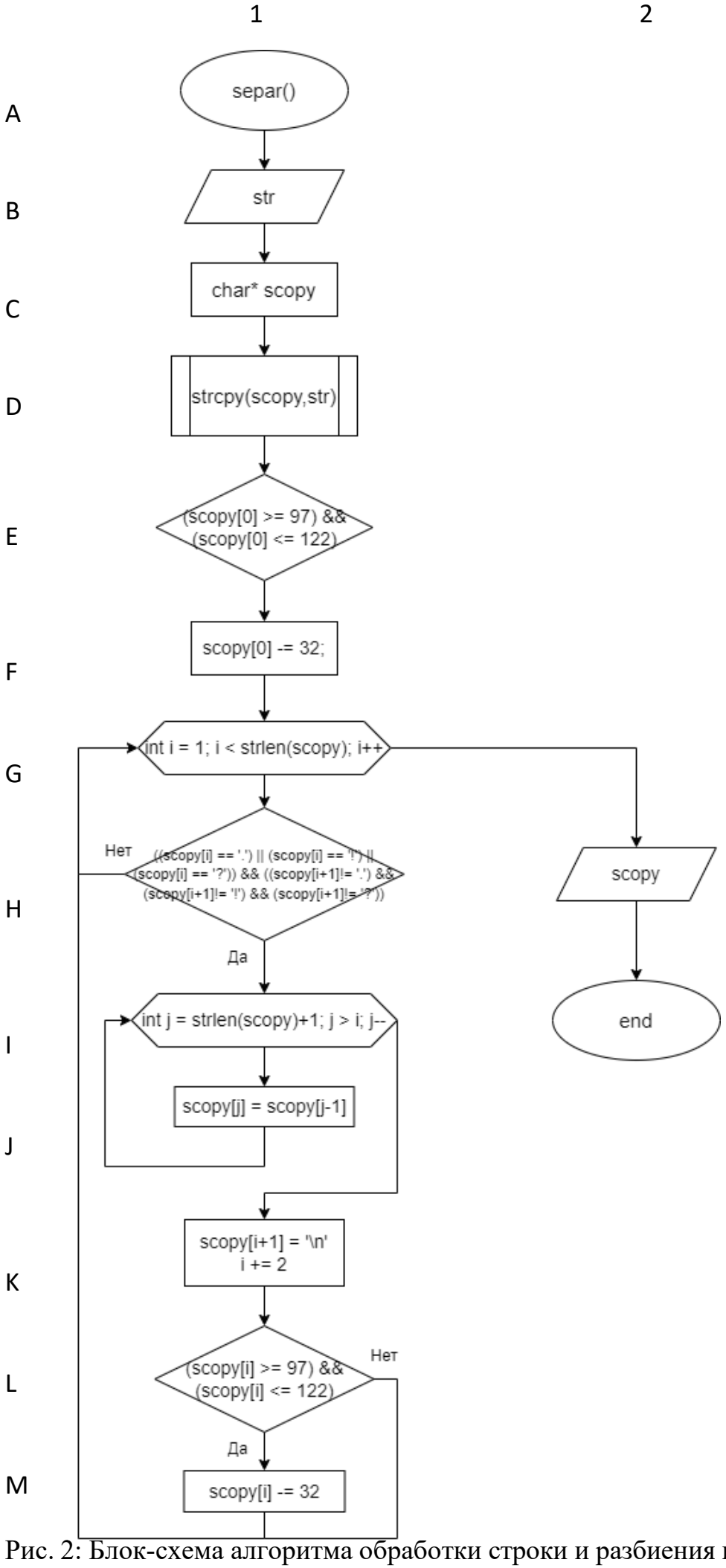


Рис. 2: Блок-схема алгоритма обработки строки и разбиения на предложения separ().

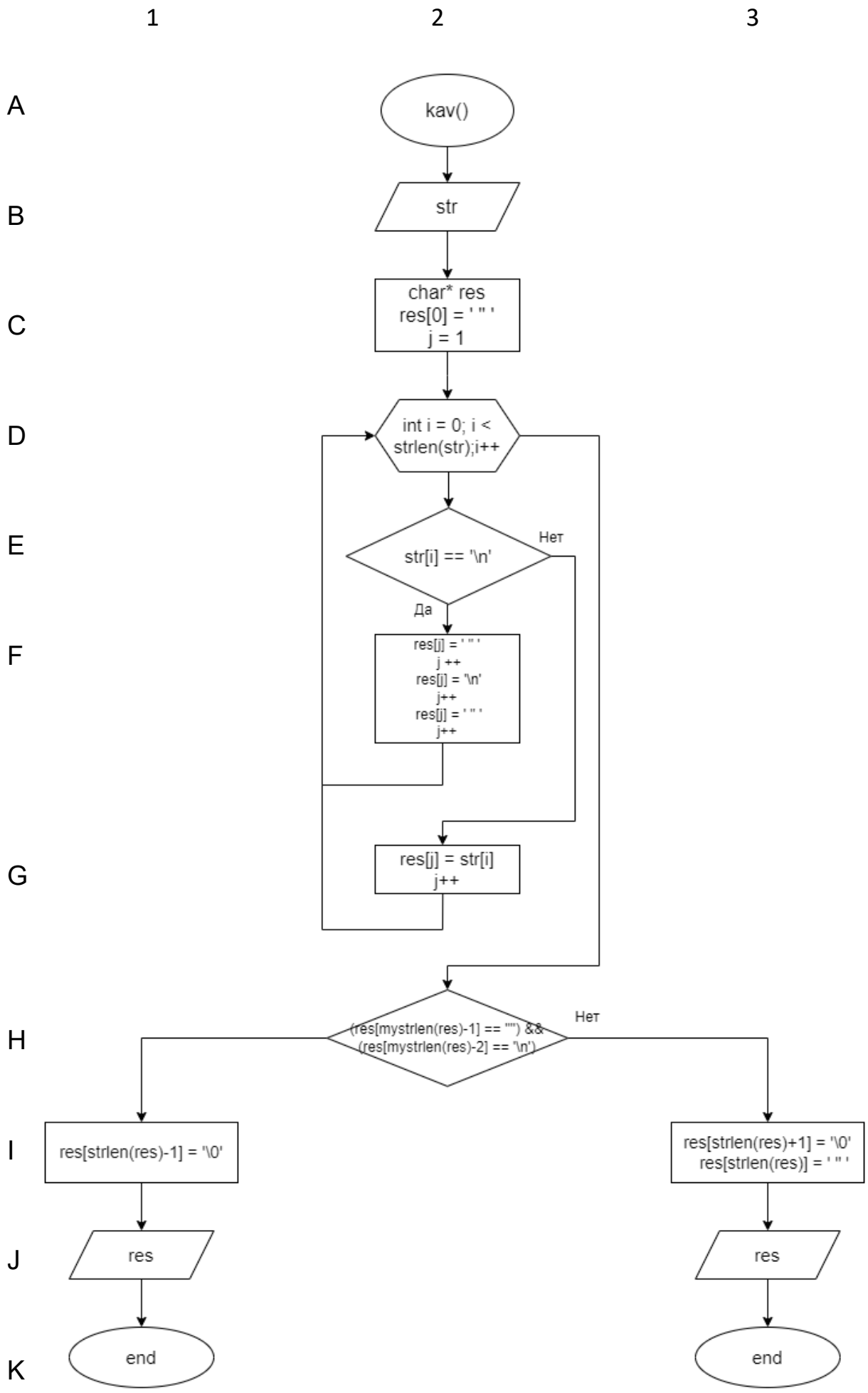


Рис. 3: Блок-схема алгоритма функции оформления предложений в кавычки kav(str).

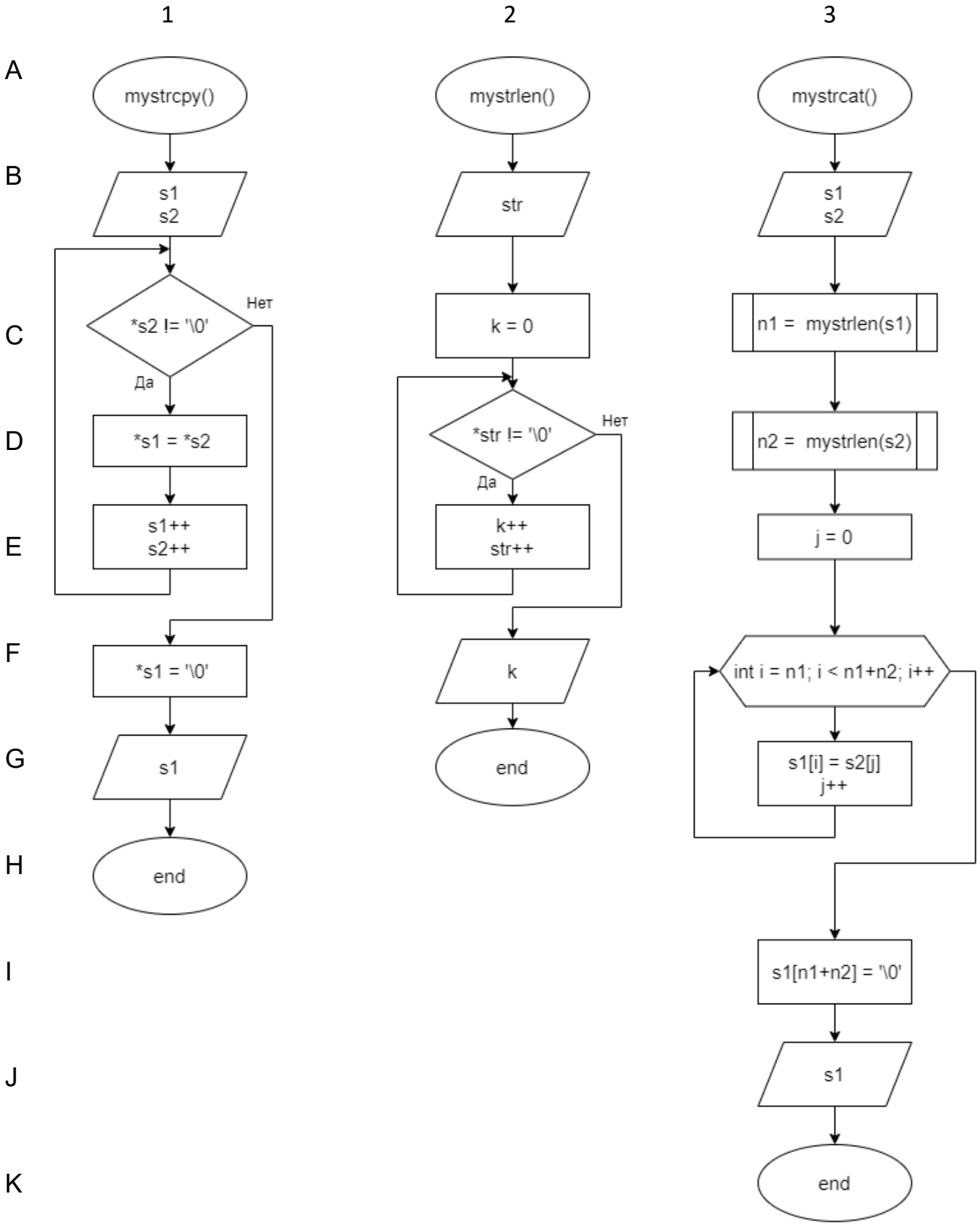


Рис. 4: Блок-схема алгоритмов функций работы со строками.

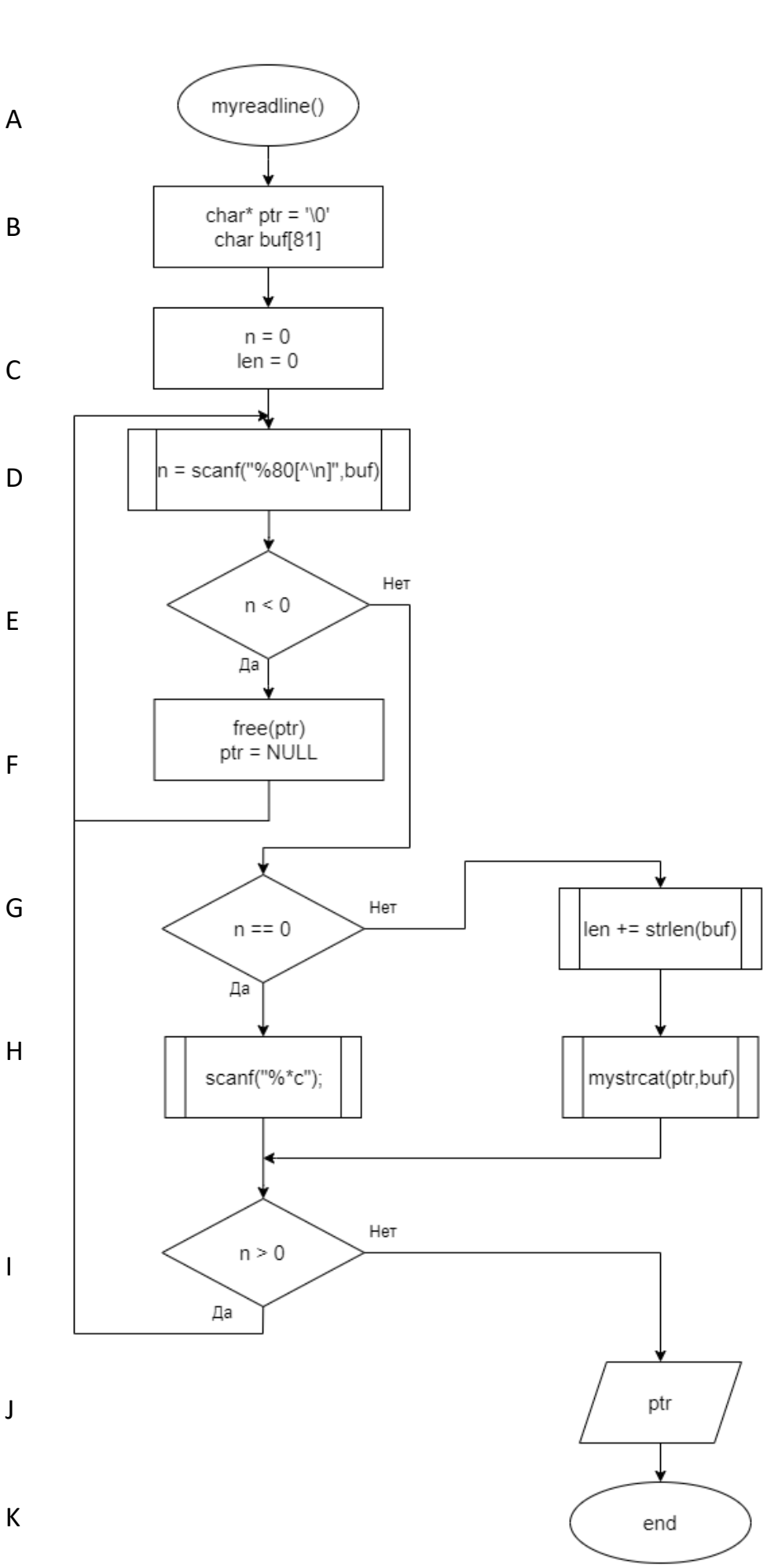


Рис. 5: Блок-схема алгоритма функции ввода строк myreadline().

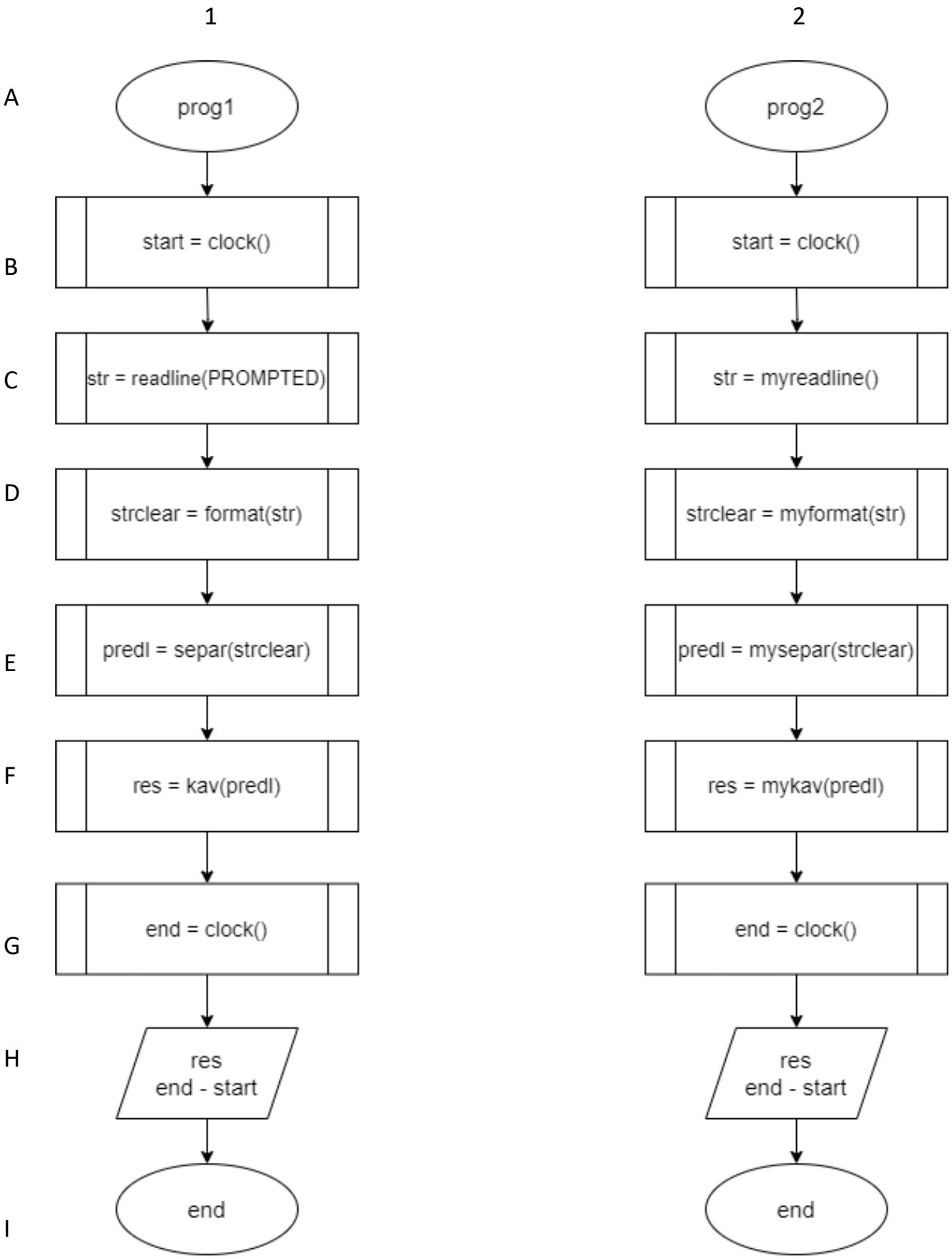


Рис. 6: Блок-схема алгоритма программ запуска prog1() и prog2().

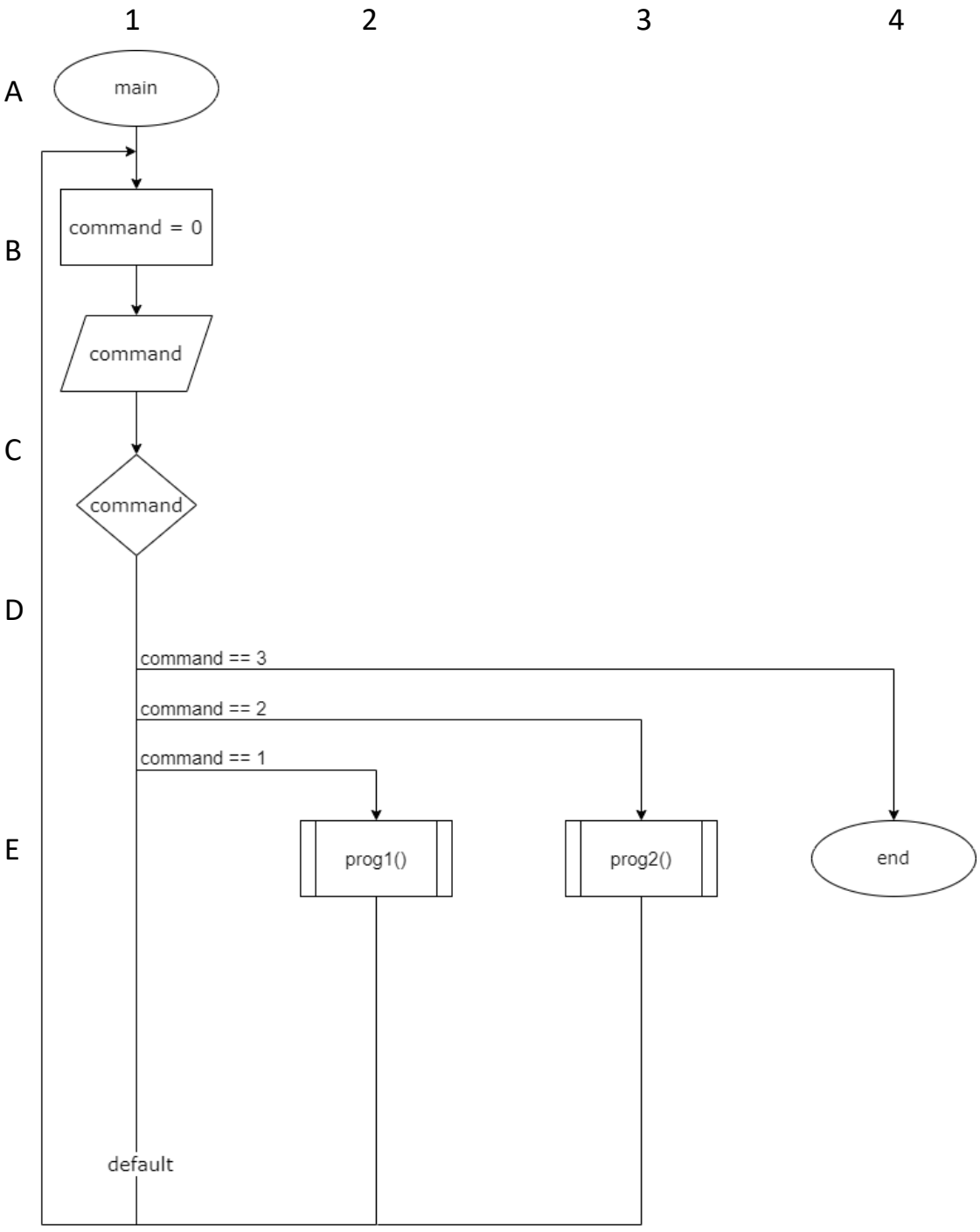


Рис. 7: Блок-схема алгоритма функции main().

4. Исходные коды разработанных программ

Структура проекта состоит из 3 файлов: заголовочного FILELAB.h и двух с исходным кодом – lab4.c, functions.c (исходники своих функций для работы со строками).

Листинг 1: Исходный код программы main (файл: lab4.c)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <readline/readline.h>
#include <time.h>
#include "FILELAB.h"

int main() {
    int command;
    printf("Список команд:\n1) Выполнение задания со встроенными функциями\n2) Выполнение
    do {
    printf("Enter the command: ");
    command = 0;
    scanf("%d",&command);
    switch (command){
        case 1: prog1();break;
        case 2: prog2();break;
        case 3: return 0;
        default: scanf("%*[^\n]");scanf("%*c");printf("Wrong command! ");break;
    }
    }while(1);
}
```

Листинг 2: Исходный код программы functions (файл: functions.c)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <readline/readline.h>
#include <time.h>
#include "FILELAB.h"

char* format(char* str){
        for (int i = 0; i < strlen(str); i++){
            if (str[i] == '\t'){str[i] = ' ';}
        }
        char* res = calloc(strlen(str) + 1, sizeof(char));
        int flag = 0;
        for (int i = 0; i < strlen(str); i++){ // создание копии введённой строки с отформатированными пробелами
                if (str[i] == ' ') {
                        if ((flag == 0) ||(str[i+1] == ' ')){continue;} // флаг = 0 только когда пробелы в начале
                        }
                res[flag] = str[i];
                flag++;
        }
        int n = strlen(res);
        if (res[n-1] == ' '){res[n-1] = '\0';} // удаление пробела в конце
        res = realloc(res, strlen(res)+1); // все двойные пробелы, пробелы в начале и в конце удалены
        flag = 0;
        for (int j = 0; j < strlen(res); j++){ // цикл удаления пробелов после знаков конца предложения (для индивидуального задания)
                if ((res[j] == '.') || (res[j] == '!') || (res[j] == '?')) {flag = 1; continue;} // пред символ знак препинания
                if ((res[j] != '.') && (res[j] != '!') && (res[j] != '?') && (res[j] != ' ')) {flag = 0; continue;}
                if ((res[j] == ' ') && (flag == 1)){ // если нашли пробел после окончания предложения ->
                        for(int k = j; k < strlen(res); k++ ){
                                res[k] = res[k+1]; // сдвиг влево части строки после пробела
                        }
                        flag = 0;
                        j = j - 1; // коррекция прохода по строке
                        res = realloc(res, strlen(res)+1);
                }
        }
        return res;
}
```

```c

char* myformat(char* str){
        for (int i = 0; i < mystrlen(str); i++){
            if (str[i] == '\t'){str[i] = ' ';}
        }
        char* res = calloc(mystrlen(str) + 1, sizeof(char));
        int flag = 0;
        for (int i = 0; i < mystrlen(str); i++){ // создание копии введённой строки с отформатированными пробелами
                if (str[i] == ' ') {
                        if ((flag == 0) ||(str[i+1] == ' ')){continue;} // флаг = 0 только когда пробелы в начале
                        }
                res[flag] = str[i];
                flag++;
        }
        int n = mystrlen(res);
        if (res[n-1] == ' '){res[n-1] = '\0';} // удаление пробела в конце
        res = realloc(res, mystrlen(res)+1); // все двойные пробелы, пробелы в начале и в конце удалены
        flag = 0;
        for (int j = 0; j < mystrlen(res); j++){ // цикл удаления пробелов после знаков конца предложения (для индивидуального задания)
                if ((res[j] == '.') || (res[j] == '!') || (res[j] == '?')) {flag = 1; continue;} // пред символ знак препинания
                if ((res[j] != '.') && (res[j] != '!') && (res[j] != '?') && (res[j] != ' ')) {flag = 0; continue;}
                if ((res[j] == ' ') && (flag == 1)){ // если нашли пробел после окончания предложения ->
                        for(int k = j; k < mystrlen(res); k++ ){
                                res[k] = res[k+1]; // сдвиг влево части строки после пробела
                        }
                        flag = 0;
                        j = j - 1; // коррекция прохода по строке
                        res = realloc(res, mystrlen(res)+1);
                }
        }
        return res;
}
```

```
71
72 char* kav(char* str){ // на вход подается готовая строка, используем эту функцию для подготовки ответа
73         char* res = calloc(strlen(str)*2,sizeof(char)); // расстановка кавычек для предложений, разделённых началом или \n или \0
74         res[0] = '"';
75         int j = 1;
76         for (int i = 0; i < strlen(str); i++){ // проход по готовой строке и формирование новой, строки - результата
77                 if (str[i] == '\n'){ // если нашли переход(конец предложения) ->
78                         res[j] = '"';
79                         j ++;
80                         res[j] = '\n';
81                         j ++;
82                         res[j] = '"';
83                         j ++;
84                         continue;
85                 }
86                 res[j] = str[i]; // копирование элементов
87                 j++;
88         }
89         res = realloc(res, strlen(res)+1);
90         if ((res[strlen(res)-1] == '"') && (res[strlen(res)-2] == '\n')){ //коррекция кавычек в зависимости от конца строки
91                 res[strlen(res)-1] = '\0';
92                 return res;
93         }
94         res = realloc(res, strlen(res)+2); // добавление кавычки в конце
95         res[strlen(res)+1] = '\0';
96         res[strlen(res)] = '"';
97         return res;
98 }
```

```
99
100 char* mykav(char* str){ // на вход подается готовая строка, используем эту функцию для подготовки ответа
101         char* res = calloc(mystrlen(str)*2,sizeof(char)); // расстановка кавычек для предложений, разделённых началом или \n или \0
102         res[0] = '"';
103         int j = 1;
104         for (int i = 0; i < mystrlen(str); i++){ // проход по готовой строке и формирование новой, строки - результата
105                 if (str[i] == '\n'){ // если нашли переход(конец предложения) ->
106                         res[j] = '"';
107                         j ++;
108                         res[j] = '\n';
109                         j ++;
110                         res[j] = '"';
111                         j ++;
112                         continue;
113                 }
114                 res[j] = str[i]; // копирование элементов
115                 j++;
116         }
117         res = realloc(res, mystrlen(res)+1);
118         if ((res[mystrlen(res)-1] == '"') && (res[mystrlen(res)-2] == '\n')){ //коррекция кавычек в зависимости от конца строки
119                 res[mystrlen(res)-1] = '\0';
120                 return res;
121         }
122         res = realloc(res, mystrlen(res)+2); // добавление кавычки в конце
123         res[mystrlen(res)+1] = '\0';
124         res[mystrlen(res)] = '"';
125         return res;
126 }
```

```
126 }
127
128 char* mysepar(char* str){ // функция разделения отформатированной строки на предложения + заглавная буква
129         char* scopy = calloc(mystrlen(str)+1, sizeof(char));
130         mystrcpy(scopy,str);
131         if ((scopy[0] >= 97) && (scopy[0] <= 122)){scopy[0] -= 32;} // заглавная буква в начале строки
132         for(int i = 1; i < mystrlen(scopy); i++) // проход по строке и поиск нужных разделителей
133             {
134             // если найдены крайние "!?.", то делим строку после них
135                 if (((scopy[i] == '.') || (scopy[i] == '!') || (scopy[i] == '?')) && ((scopy[i+1]!= '.') && (scopy[i+1]!= '!') && (scopy[i+1]!= '?')))
136                         {
137                                 scopy = realloc(scopy, (mystrlen(scopy)+2) * sizeof(char));
138                                 for (int j = mystrlen(scopy)+1; j > i; j--)
139                                 {
140                                         scopy[j] = scopy[j-1]; // сдвиг элементов вправо начиная с (i + 1) ого (элементы после разделителя)
141                                 }
142                                 scopy[i+1] = '\n'; // разделение предложения
143                                         i += 2; // коррекция прохода, теперь scopy[i] указывает на 1ую букву следующего предложения или на \0
144                                         if ((scopy[i] >= 97) && (scopy[i] <= 122)){scopy[i] -= 32;} // заглавная буква
145                         }
146                 }
147         scopy = realloc(scopy, (mystrlen(scopy)+1) * sizeof(char));
148         return scopy;
149 }
```

```c
char* separ(char* str){ // функция разделения отформатированной строки на предложения + заглавная буква
	char* scopy = calloc(strlen(str)+1, sizeof(char));
	strcpy(scopy,str);
	if ((scopy[0] >= 97) && (scopy[0] <= 122)){scopy[0] -= 32;} // заглавная буква в начале строки
	for(int i = 1; i < strlen(scopy); i++) // проход по строке и поиск нужных разделителей
		{
		// если найдены крайние "!?.", то делим строку после них
			if (((scopy[i] == '.') || (scopy[i] == '!') || (scopy[i] == '?')) && ((scopy[i+1]!= '.') && (scopy[i+1]!= '!') && (scopy[i+1]!= '?')))
				{
					scopy = realloc(scopy, (strlen(scopy)+2) * sizeof(char));
					for (int j = strlen(scopy)+1; j > i; j--)
					{
						scopy[j] = scopy[j-1]; // сдвиг элементов вправо начиная с (i + 1) ого (элементы после разделителя)
					}
					scopy[i+1] = '\n'; // разделение предложения
						i += 2; // коррекция прохода, теперь scopy[i] указывает на 1ую букву следующего предложения или на \0
						if ((scopy[i] >= 97) && (scopy[i] <= 122)){scopy[i] -= 32;} // заглавная буква
				}
		}
	scopy = realloc(scopy, (strlen(scopy)+1) * sizeof(char));
	return scopy;
}

char* mystrcpy(char* s1, const char* s2){ // реализация strcpy
	while (*s2 != '\0'){
		*s1 = *s2;
		s1++;
		s2++;
	}
	*s1 = '\0';
	return s1;
}

size_t mystrlen(const char* str){ //реализация mystrlen
	size_t k = 0;
	while(*str != '\0'){
		k++;
		str++;
	}
	return k;
}

char* mystrcat(char* s1, char* s2){ // реализация strcat
    int n1 =  mystrlen(s1);
    int n2 =  mystrlen(s2);
    int j = 0;
    for (int i = n1; i < n1+n2; i++){
        s1[i] = s2[j];
        j++;
    }
    s1[n1+n2] = '\0';
    return s1;
}

char* myreadline(){
	printf(PROMPTED);
	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 += strlen(buf);
			ptr = (char*) realloc(ptr,len + 1);
			mystrcat(ptr,buf);
		}
	} while(n > 0);
	return ptr;
}
```

```
227
228 void prog1(){
229         clock_t start = clock();
230         char* str = readline(PROMPTED);
231         printf("Введенная строка: \"%s\"\n", str);
232         char* strclear = format(str); //форматирование пробелов
233         char* predl = separ(strclear);
234         char* res = kav(predl);
235         clock_t end = clock();
236         printf("Полученные предложения (встроенные функции):\n");
237         printf("%s\nВремя выполнения задания программой 1: %lf c\n\n", res, (double)(end-start)/CLOCKS_PER_SEC);
238         free(str);free(strclear);free(predl);free(res);
239 }
240
241 void prog2(){
242         scanf("%*c"); // удаление символа \n из буфера который выходит после ввода command
243         clock_t start = clock();
244         char* str = myreadline();
245         printf("Введенная строка: \"%s\"\n", str);
246         char* strclear = myformat(str); //форматирование пробелов
247         char* predl = mysepar(strclear);
248         char* res = mykav(predl);
249         clock_t end = clock();
250         printf("Полученные предложения (свои функции):\n");
251         printf("%s\nВремя выполнения задания программой 2: %lf c\n\n", res,(double)(end-start)/CLOCKS_PER_SEC);
252         free(str);free(strclear);free(predl);free(res);
253 }
254
```

5. Текстовые примеры работы программы

Тесты алгоритма вычисления длины строки (mystrlen)

Входные данные	Вывод
“”	0
“ “	2
“dcnjsdjv ”	9
“scjnnj nj k”	11

Тесты алгоритма конкатенации строк (mystrcat)

Входные данные	Вывод
“dvdsdv”, “b”	“dvdsdvb”
“qq”, “”	“qq”
“ yyy”, “pop ”	“ yyypop ”
“”, “”	“”

Тесты алгоритма обработки строки (удаление лишней табуляции и пробелов) (format(str))

Входные данные	Вывод
“ vdhduhh \t xnjk . jj xsc ”	“vdhduhh xnjk .jj xsc”
“”	“”
“\t dvdv. \t”	“dvdv.”
“\t wasd. okjvdv\t .\t”	“wasd.okjvdv.”

Тесты алгоритма выполнения индивидуального задания

Входные данные	Вывод
“ \t bdbh.scnjsj d. \t\t jvfn.\t”	“Bdbh.” “Scnjsj d.” “Jvfn.”
“”	“”
“ cscsmmk xcscm. scr? cnsx! ”	“Cscsmmk xcscm.” “Scr?” “Cnsx!”
“ dsdvds szc dv \t d s. df !”	“Dsdvds szc dv d s.” “Df !”
“v”	“V”
“hfujkukutc… \t dvds… sfb scs ?.”	“Hfujkukutc…” “Dvds…” “Sfb scs ?.”
“fd dgh ….. ry h . y. y. y.!”	“Fd dgh …..” “Ry h .” “Y.” “Y.” “Y.!”

6. Скриншоты работы программы

Команда для сборки программы:

cc –o reslab4 lab4.c functions.c -lreadline

Запуск с valgrind:

valgrind ./reslab4

```
[gavrilov.da@unix lab4]$ cc -o reslab4 lab4.c functions.c -lreadline
[gavrilov.da@unix lab4]$ valgrind ./reslab4
==22447== Memcheck, a memory error detector
==22447== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==22447== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==22447== Command: ./reslab4
==22447==
Список команд:
1) Выполнение задания со встроенными функциями
2) Выполнение задания со своими функциями
3) Завершить выполнение
Enter the command: 2
Enter the str >>         bdbh.scnjsj   d.               jvfn.
Введенная строка: "      bdbh.scnjsj   d.               jvfn."
Полученные предложения (свои функции):
"Bdbh."
"Scnjsj d."
"Jvfn."

Время выполнения задания программой 2: 0.025304 с

Enter the command: 2
Enter the str >>
Введенная строка: ""
Полученные предложения (свои функции):
""
Время выполнения задания программой 2: 0.001320 с

Enter the command: 2
Enter the str >>   cscsmmk     xcscm. scr?  cnsx!
Введенная строка: "  cscsmmk     xcscm. scr?  cnsx! "
Полученные предложения (свои функции):
"Cscsmmk xcscm."
"Scr?"
"Cnsx!"

Время выполнения задания программой 2: 0.000533 с

Enter the command: 3
==22447==
==22447== HEAP SUMMARY:
==22447==     in use at exit: 0 bytes in 0 blocks
==22447==   total heap usage: 29 allocs, 29 frees, 2,700 bytes allocated
==22447==
==22447== All heap blocks were freed -- no leaks are possible
==22447==
==22447== For lists of detected and suppressed errors, rerun with: -s
==22447== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
[gavrilov.da@unix lab4]$ cc -o reslab4 lab4.c functions.c -lreadline
[gavrilov.da@unix lab4]$ valgrind ./reslab4
==28226== Memcheck, a memory error detector
==28226== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==28226== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==28226== Command: ./reslab4
==28226==
Список команд:
1) Выполнение задания со встроенными функциями
2) Выполнение задания со своими функциями
3) Завершить выполнение
Enter the command: 2
Enter the str >> dsdvds szc dv    \t d s.  df !
Введенная строка: "dsdvds szc dv    \t d s.  df !"
Полученные предложения (свои функции):
"Dsdvds szc dv \t d s."
"Df !"

Время выполнения задания программой 2: 0.024788 c

Enter the command: 2
Enter the str >> v
Введенная строка: "v"
Полученные предложения (свои функции):
"V"
Время выполнения задания программой 2: 0.000292 c

Enter the command: 3
==28226==
==28226== HEAP SUMMARY:
==28226==     in use at exit: 0 bytes in 0 blocks
==28226==   total heap usage: 22 allocs, 22 frees, 2,384 bytes allocated
==28226==
==28226== All heap blocks were freed -- no leaks are possible
==28226==
==28226== For lists of detected and suppressed errors, rerun with: -s
==28226== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[gavrilov.da@unix lab4]$
```

```
[gavrilov.da@unix lab4]$ cc -o reslab4 lab4.c functions.c -lreadline
[gavrilov.da@unix lab4]$ valgrind ./reslab4
==8587== Memcheck, a memory error detector
==8587== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==8587== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==8587== Command: ./reslab4
==8587==
Список команд:
1) Выполнение задания со встроенными функциями
2) Выполнение задания со своими функциями
3) Завершить выполнение
Enter the command: 2
Enter the str >> hfujkukutc...          dvds... sfb scs ?.
Введенная строка: "hfujkukutc...        dvds... sfb scs ?."
Полученные предложения (свои функции):
"Hfujkukutc..."
"Dvds..."
"Sfb scs ?."

Время выполнения задания программой 2: 0.024049 c

Enter the command: 2
Enter the str >> fd  dgh ….. ry h . y. y. y.!
Введенная строка: "fd  dgh ….. ry h . y. y. y.!"
Полученные предложения (свои функции):
"Fd dgh ….."
"Ry h ."
"Y."
"Y."
"Y.!"

Время выполнения задания программой 2: 0.000623 c

Enter the command: 3
==8587==
==8587== HEAP SUMMARY:
==8587==     in use at exit: 0 bytes in 0 blocks
==8587==   total heap usage: 32 allocs, 32 frees, 2,999 bytes allocated
==8587==
==8587== All heap blocks were freed -- no leaks are possible
==8587==
==8587== For lists of detected and suppressed errors, rerun with: -s
==8587== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[gavrilov.da@unix lab4]$
```

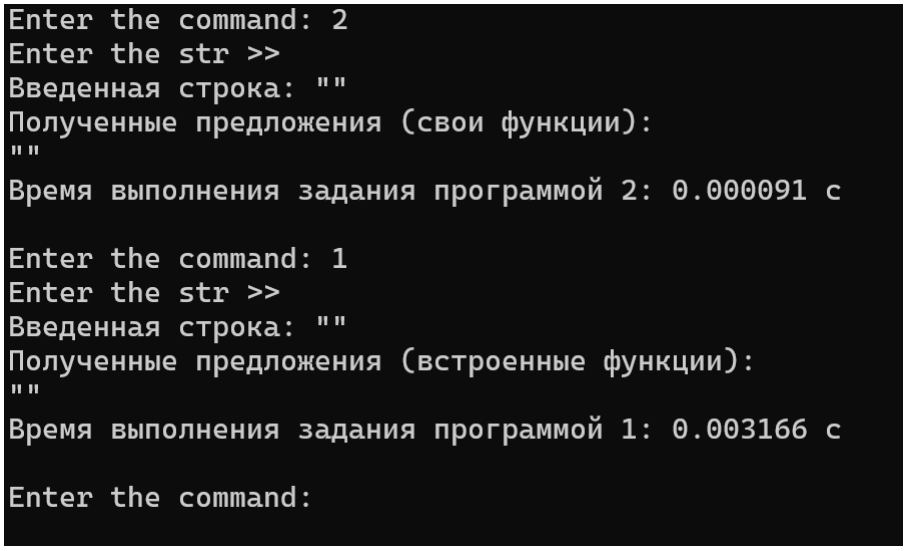
С помощью встроенного в язык таймера (clock()) было выяснено время обработки одной строки, а также, что стандартные функции работают немного быстрее чем аналогичная своя реализация. Кроме функции readline(). Самостоятельно написанная функция myreadline() для своих задач работает быстрее.

```
[gavrilov.da@unix lab4]$ cc -o reslab4 lab4.c functions.c -lreadline
[gavrilov.da@unix lab4]$ valgrind ./reslab4
==15536== Memcheck, a memory error detector
==15536== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==15536== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==15536== Command: ./reslab4
==15536==
Список команд:
1) Выполнение задания со встроенными функциями
2) Выполнение задания со своими функциями
3) Завершить выполнение
Enter the command: 2
Enter the str >> hfujkukutc...          dvds... sfb scs ?.
Введенная строка: "hfujkukutc...        dvds... sfb scs ?."
Полученные предложения (свои функции):
"Hfujkukutc..."
"Dvds..."
"Sfb scs ?."

Время выполнения задания программой 2: 0.024343 с

Enter the command: 1
Enter the str >> hfujkukutc...          dvds... sfb scs ?.
Введенная строка: "hfujkukutc...        dvds... sfb scs ?."
Полученные предложения (встроенные функции):
"Hfujkukutc..."
"Dvds..."
"Sfb scs ?."

Время выполнения задания программой 1: 0.404782 с

Enter the command: 2
Enter the str >>   cscsmmk     xcscm. scr?  cnsx!
Введенная строка: "  cscsmmk     xcscm. scr?  cnsx! "
Полученные предложения (свои функции):
"Cscsmmk xcscm."
"Scr?"
"Cnsx!"

Время выполнения задания программой 2: 0.000825 с

Enter the command: 1
Enter the str >>   cscsmmk     xcscm. scr?  cnsx!
Введенная строка: "  cscsmmk     xcscm. scr?  cnsx!"
Полученные предложения (встроенные функции):
"Cscsmmk xcscm."
"Scr?"
"Cnsx!"

Время выполнения задания программой 1: 0.009875 с
```

Сравнительная таблица временных результатов работы 1 и 2 программы при разных значениях входных элементов:

Входные данные	Вывод	Время prog1 (встроенные), с	Время prog2 (свои), с
“ \t bdbh.scnjsj d. \t\t jvfn.\t”	“Bdbh.” “Scnjsj d.” “Jvfn.”	0.003614	0.000092
“hfujkukutc… \t dvds… sfb scs ?.”	“Hfujkukutc…” “Dvds…” “Sfb scs ?.”	0.003351	0.000092
“ cscsmmk xcscm. scr? cnsx! ”	“Cscsmmk xcscm.” “Scr?” “Cnsx!”	0.003595	0.000096
“”	“”	0.003166	0.000091
“ dsdvds szc dv \t d s. df !”	“Dsdvds szc dv d s.” “Df !”	0.003378	0.000090
“fd dgh ….. ry h . y. y. y.!”	“Fd dgh …..” “Ry h .” “Y.” “Y.” “Y.!”	0.003418	0.000169
“v”	“V”	0.003190	0.000119

Скриншоты к таблице:



```
Enter the command: 2
Enter the str >>   cscsmmk     xcscm. scr?  cnsx!
Введенная строка: "  cscsmmk     xcscm. scr?  cnsx! "
Полученные предложения (свои функции):
"Cscsmmk xcscm."
"Scr?"
"Cnsx!"

Время выполнения задания программой 2: 0.000096 с

Enter the command: 1
Enter the str >>   cscsmmk     xcscm. scr?  cnsx!
Введенная строка: "  cscsmmk     xcscm. scr?  cnsx! "
Полученные предложения (встроенные функции):
"Cscsmmk xcscm."
"Scr?"
"Cnsx!"

Время выполнения задания программой 1: 0.003595 с
```

```
Enter the command: 2
Enter the str >> fd  dgh …... ry h . y. y. y.!
Введенная строка: "fd  dgh …... ry h . y. y. y.!"
Полученные предложения (свои функции):
"Fd dgh …..."
"Ry h ."
"Y."
"Y."
"Y.!"

Время выполнения задания программой 2: 0.000169 с

Enter the command: 1
Enter the str >> fd  dgh …... ry h . y. y. y.!
Введенная строка: "fd  dgh …... ry h . y. y. y.!"
Полученные предложения (встроенные функции):
"Fd dgh …..."
"Ry h ."
"Y."
"Y."
"Y.!"

Время выполнения задания программой 1: 0.003418 с
```

```
Enter the command: 2
Enter the str >> v
Введенная строка: "v"
Полученные предложения (свои функции):
"V"
Время выполнения задания программой 2: 0.000119 с

Enter the command: 1
Enter the str >> v
Введенная строка: "v"
Полученные предложения (встроенные функции):
"V"
Время выполнения задания программой 1: 0.003190 с
```

```
Enter the command: 2
Enter the str >>        bdbh.scnjsj d.          jvfn.
Введенная строка: "     bdbh.scnjsj d.          jvfn. "
Полученные предложения (свои функции):
"Bdbh."
"Scnjsj d."
"Jvfn."

Время выполнения задания программой 2: 0.000092 c

Enter the command: 1
Enter the str >>        bdbh.scnjsj d.          jvfn.
Введенная строка: "     bdbh.scnjsj d.          jvfn."
Полученные предложения (встроенные функции):
"Bdbh."
"Scnjsj d."
"Jvfn."

Время выполнения задания программой 1: 0.003614 c
```

```
Enter the command: 2
Enter the str >>  dsdvds szc dv          d s.  df !
Введенная строка: " dsdvds szc dv        d s.  df !"
Полученные предложения (свои функции):
"Dsdvds szc dv d s."
"Df !"

Время выполнения задания программой 2: 0.000090 c

Enter the command: 1
Enter the str >>  dsdvds szc dv          d s.  df !
Введенная строка: " dsdvds szc dv        d s.  df !"
Полученные предложения (встроенные функции):
"Dsdvds szc dv d s."
"Df !"

Время выполнения задания программой 1: 0.003378 c
```

```
Enter the command: 2
Enter the str >> hfujkukutc…    dvds... sfb scs ?.
Введенная строка: "hfujkukutc…          dvds... sfb scs ?."
Полученные предложения (свои функции):
"Hfujkukutc… dvds..."
"Sfb scs ?."

Время выполнения задания программой 2: 0.000092 c

Enter the command: 1
Enter the str >> hfujkukutc…    dvds... sfb scs ?.
Введенная строка: "hfujkukutc…          dvds... sfb scs ?."
Полученные предложения (встроенные функции):
"Hfujkukutc… dvds..."
"Sfb scs ?."

Время выполнения задания программой 1: 0.003351 c
```

Итого мы можем сделать вывод, что за счёт своего readline() prog2() выигрывает у prog1() в несколько раз закономерно по времени: среднее время 0.00009 с против 0.0035 с.

7. Трудности при выполнении работы

1. В ходе выполнения данной работы я столкнулся с проблемой возникновения учетек с памятью при работе с динамическими выделенными строками при их передаче в качестве аргументов в функцию. Проблема была решена с помощью грамотного построения и проектирования интерфейсов функций по работе со строками (массивами char), а также с помощью встроенных функций языка Си по работе с памятью и тестирование программы утилитой valgrind.

2. Также при форматировании строк я иногда не следил за реальным количеством ячеек выделенной памяти:

   Например, функция realloc при добавлении символа в строку (строке уже выделена память) должна выглядеть так: res = realloc(res, strlen(res)+**2**); потому что realloc(res, strlen(res)+**1**); нам никакой пользы не принесёт, так как 1 место в динамическом массиве зарезервировано ‘\0’ и мы просто не выделим память под символ. Проблема была решена исправлением и отлову ошибок при помощи valgrind с ключом компиляции -g и ключом –leak-check=full

8. Выводы

В ходе выполнения данной работы на примере программы, выполняющей базовые алгоритмы над строками, были рассмотрены базовые принципы работы со строками в языке Си:

1. Основные алгоритмы по работе над строками (длина, конкатенация и т. д.).
2. Выделение и освобождение нужного количества памяти под строки.
3. Организация системы подпрограмм, находящихся в разных файлах.
4. Создание пользовательского интерфейса в виде меню с выбором действий (выбор своих или стандартных функций в работе со строками).
5. Создание и работа с заголовочными файлами (.h) в языке Си.

6. Измерение времени работы программ на Си.