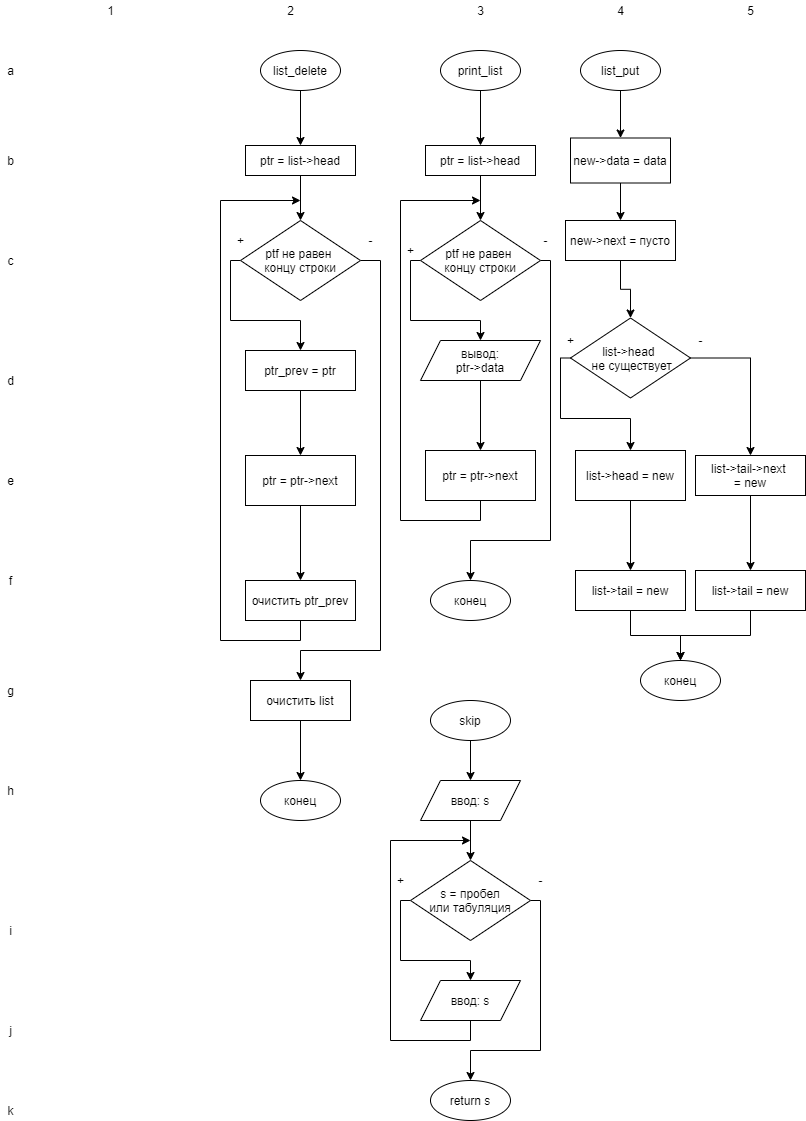
Самородов Роман Б20-513

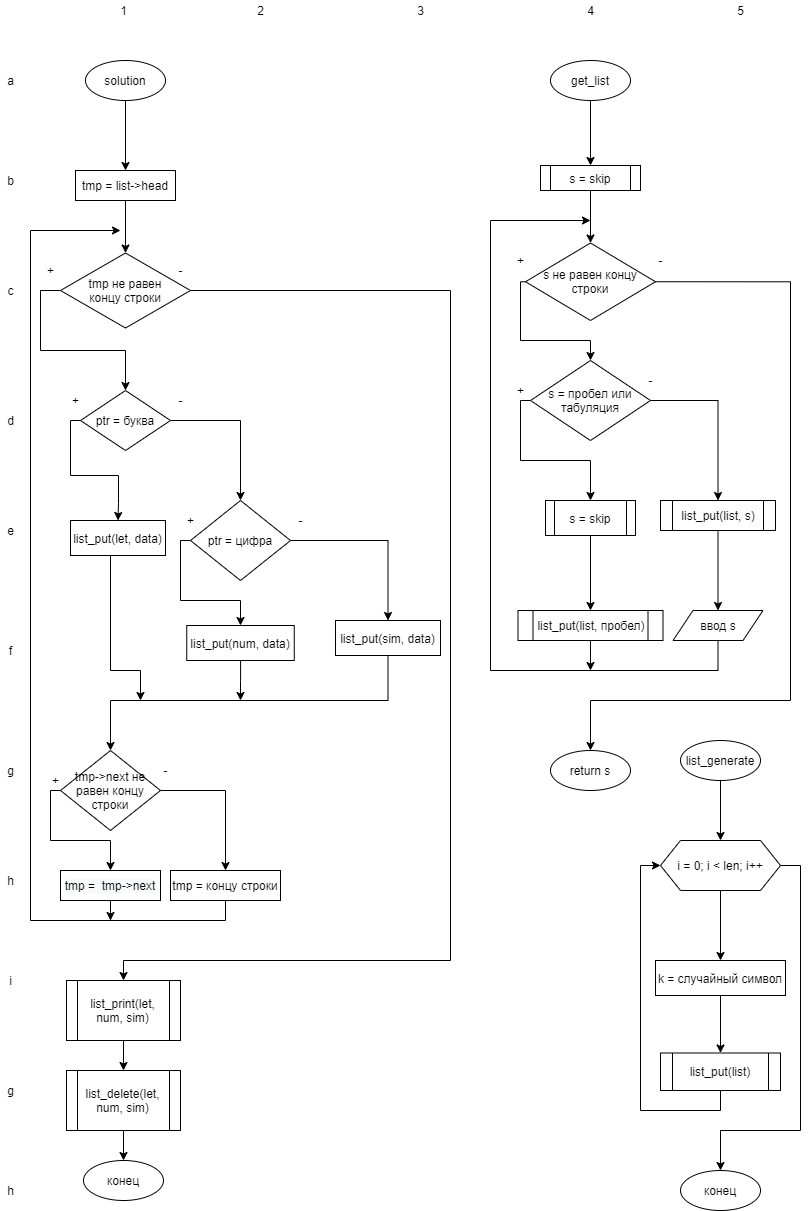
5.27 Для каждой строки сформировать три новые строки, полученные путем добавления в них слов, состоящих из символов каждого слова исходной строки, относящихся к буквам (1-я строка), цифрам (2-я строка) и прочим символам (3-я строка).

**1. Блок-схемы алгоритмов работы программы и основных функций.**

**Изображение выглядит как текст, электроника

Автоматически созданное описание**

****

****

**2. Исходные коды основной программы.**

**Lab5.c**

/\*5.27 Для каждой строки сформировать три новые строки, полученные путем

добавления в них слов, состоящих из символов каждого слова исходной строки,

относящихся к буквам (1-я строка), цифрам (2-я строка) и прочим символам (3-я

строка).

Самородов Роман Б20-513

\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "lab5\_1.h"

#include "lab5\_2.h"

int main()

{

int f;

printf("\n1.Свои данные\n2.Случайная генерация\n");

scanf("%d", &f);

if(f == 1)

{

sol1();

} else {

sol2();

}

return 0;

}

**Lab5\_1.h**

#pragma once

typedef struct item {

char data;

struct item \*next;

} item;

typedef struct list {

item \*head;

item \*tail;

} list;

list \*list\_new() {

return (list \*) calloc(1, sizeof(list));

}

void list\_delete(list \*list)//удаление списка

{

item \*ptr = list->head, \*ptr\_prev;

while (ptr)

{

ptr\_prev = ptr;

ptr = ptr->next;

free(ptr\_prev);

}

free(list);

}

void list\_print(list \*list)//вывод списка

{

item \*ptr = list->head;

printf("\n\"");

while (ptr)

{

printf("%c", ptr->data);

ptr = ptr->next;

}

printf("\"\n");

}

int list\_put(list \*list, char data)//добавление элемента в конец списка

{

item \*new = (item \*) calloc(1, sizeof(item));

if (!new)

{

return -1;

}

new->data = data;

new->next = NULL;

if (!list->head)

{

list->head = new;

list->tail = new;

} else {

list->tail->next = new;

list->tail = new;

}

return 0;

}

void solution(list\* list)//получение строк решения

{

struct list\* let = (struct list\*)calloc(1, sizeof(struct list));

struct list\* num = (struct list\*)calloc(1, sizeof(struct list));

struct list\* sim = (struct list\*)calloc(1, sizeof(struct list));

item \*tmp = list->head;

while(tmp)

{

if (((tmp->data >= 'A')&&(tmp->data <='Z'))||((tmp->data >= 'a')&&(tmp->data <='z')))

{

list\_put(let, tmp->data);//добавляем букву в первую строку

}

else if ((tmp->data >= '0')&&(tmp->data <='9'))

{

list\_put(num, tmp->data);//добавляем цифру в первую строку

}

else if((tmp->data != ' ')&&(tmp->data != '\t'))

{

list\_put(sim, tmp->data);//добавляем символ в первую строку

}

if (tmp->next)//переход на следующий символ

{

tmp = tmp->next;

}

else tmp = NULL; }

list\_print(let);

list\_print(num);

list\_print(sim);

list\_delete(let);

list\_delete(num);

list\_delete(sim);

}

char skip()// пропуск пробелов и табуляций входного потока до первого символа

{

char s = getchar();

while((s == ' ') || (s == '\t'))

{

s = getchar();

}

return s;

}

void list\_generate(list\* list, int len)// генерация строки

{

char k;

for (int i = 0; i < len; i++)

{

k = (char) (rand()%94 + 33);

list\_put(list, k);

}

}

char get\_list(list\* list)//ввод списка без лишних пробелов и табуляций

{

char s;

printf("\nВведите строку:\n");

s = skip();

while((s != '\n') && (s != EOF))

{

if((s == ' ') || (s == '\t'))

{

s = skip();

list\_put(list, ' ');

}

else {

list\_put(list, s);

s = getchar();

}

}

return s;

}

double sol\_time(list\* list)

{

list\_print(list);

printf("\nРешение:");

clock\_t begin = clock();

solution(list);

clock\_t end = clock();

double time\_spent = (double)(end - begin);

printf("\nВремя обработки строки: %f мс\n", time\_spent);

list\_delete(list);

return time\_spent;

}

**Lab5\_2.c**

#pragma once

void sol1()//Ввод данных

{

char s = getchar();

double time\_spent = 0;

while(s != EOF)

{

list\* list = list\_new();

s = get\_list(list);

printf("\nВходная строка без лишник пробелов и табуляций:");

sol\_time(list);

}

}

void sol2()//Генерация данных

{

char s = 0;

double time\_spent = 0;

int n, len;

printf("Введите количество строк\n");

scanf("%d", &n);

printf("Введите длину строк\n");

scanf("%d", &len);

srand(time(NULL));

for(int i = 0; i < n; i++)

{

list\* list = list\_new();

list\_generate(list, len);

printf("\nСгенерированная строка:");

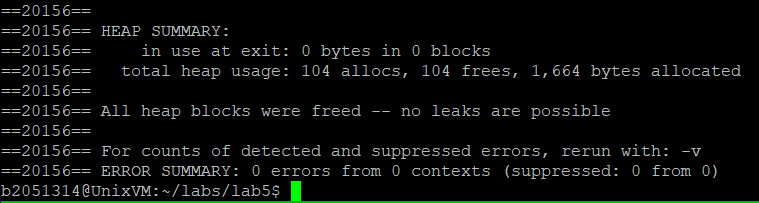
time\_spent += sol\_time(list);

}

printf("\nСреднее время обработки строки: %f мс\n", time\_spent/n);

}

**3.** **Valgrind**



**4.** **Тестовые наборы для основной программы.**

|  |  |
| --- | --- |
| **Входная строка** | **Результат** |
| else if ((tmp->data >= '0')&&(tmp->data <='9')) | Входная строка без лишних пробелов и табуляций:  "else if ((tmp->data >= '0')&&(tmp->data <='9'))"  Решение:  "elseiftmpdatatmpdata"  "09"  "((->>='')&&(-><=''))" |
| 32if (((tmp->data >= 'A')&&(tmp->data <='Z')||((tmp->data >= 'a')&&(tmp->data <='z'))) | Входная строка без лишних пробелов и табуляций:  "if (((tmp->data >= 'A')&&(tmp->data <='Z')||((tmp->data >= 'a')&&(tmp->data <='z')))"  Решение:  "iftmpdataAtmpdataZtmpdataatmpdataz"  "32"  "(((->>='')&&(-><='')||((->>='')&&(-><='')))" |
| One - 1 two - 2 ? three ! 3 ((QWER 2 | Входная строка без лишних пробелов и табуляций:  "One - 1 two - 2 ? three ! 3 ((QWER 2"  Решение:  "OnetwothreeQWER"  "1232"  "--?!((" |
| **Генерированная строка** | **Результат** |
| Mqt7r`WaeR>B1nj | Решение:  "MqtrWaeRBnj"  "71"  "`>" |
| 4kPHK)k"|U+3'16!Sf3A | Решение:  "kPHKkUSfA"  "43163"  ")"|+'!" |
| "KZEZw5KiDZWb)j@Dh|oXo | Решение:  "KZEZwKiDZWbjDhoXo"  "5"  ")@|" |

**5. Результаты таймирования, содержащие таблицы, графики зависимости**

**времени выполнения обработки строки от ее длины и краткие выводы.**

Для точности результатов бралось большое количество строк:

|  |  |  |
| --- | --- | --- |
| **Количество строк** | **Длина строки (символов)** | **Среднее время обработки (мс)** |
| 30 | 5 | 32.46 |
| 30 | 10 | 33.72 |
| 30 | 15 | 37.32 |
| 30 | 20 | 37.5 |
| 30 | 25 | 38.6 |
| 30 | 30 | 39.13 |
| 30 | 35 | 39.37 |
| 30 | 40 | 41.07 |
| 30 | 45 | 41.73 |
| 30 | 50 | 42.31 |

**Вывод:** время обработки строки линейно зависит от ее длины