|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ**

по дисциплине: «Основы программирования»

|  |  |  |
| --- | --- | --- |
| Студент |  | Камалов Антон Павлович |
| Группа |  | РК6-15Б |
| Тип задания |  | Рейтинг №3 |

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ \_**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ \_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

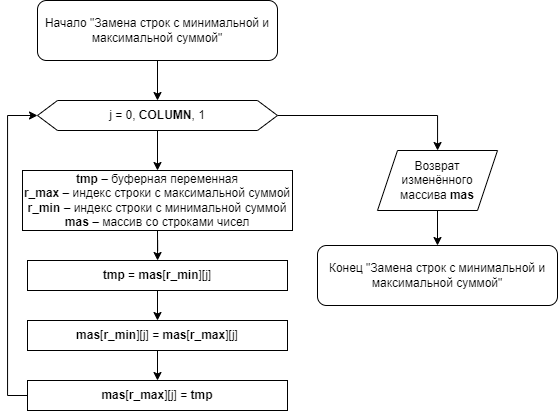
*Москва, 2022 г.*

# *Вариант 8.*

# Задание: Разработайте программу, которая вводит из файла, имя которого определяется в командной строке, слова и помещает их в линейный односвязный список. Затем удаляет из списка все элементы, у которых длина слова меньше 3. Слова из результирующего списка вывести в стандартный поток вывода.

# Блок-схема программы

# 



# Текст программы

# #include <stdio.h>

# #include <stdlib.h>

# #include <string.h>

# #include <ctype.h>

# #define LINE\_LEN 2048

# #define STR\_LEN 3

# typedef struct Node {

# char \* string;

# struct Node \* next;

# }

# WORD;

# // List

# WORD \* in(FILE \* f);

# // Add word to list

# void append(WORD \*\* , char \* );

# // Delete word, which length less than STR\_LEN

# void delete(WORD \*\* );

# // Print words to stdout

# void print(WORD \*\* );

# // Clear list

# void clear(WORD \*\* );

# int main(int argc, char \* argv[]) {

# FILE \* f;

# if (argc < 2) {

# printf("Incorrect number of arguments.\n");

# exit(-1);

# }

# if ((f = fopen(argv[1], "r")) == NULL) {

# printf("Can't open file `%s`\n", argv[1]);

# exit(-3);

# }

# WORD \* words = NULL;

# words = in(f);

# fclose(f);

# if (words) {

# printf("Before delete: \n");

# print(&words);

# delete(&words);

# printf("\nAfter delete:\n");

# 

# print(&words);

# clear(&words);

# } else{

# printf("List is empty!\n");

# }

# }

# WORD \* in(FILE \* f) {

# WORD \* head = NULL;

# int j = 0, c;

# char line[LINE\_LEN], \* elem;

# while ((c = fgetc(f)) != EOF) {

# if (c != ' ' && iscntrl(c) == 0) {

# if (j >= LINE\_LEN) {

# printf("BUFFER OVERFLOW!\n");

# exit(-1);

# }

# line[j++] = c;

# } else {

# line[j] = '\0';

# j = 0;

# if (strlen(line) == 0) continue;

# if ((elem = (char \* ) malloc(strlen(line))) == NULL) {

# printf("NOT ENOUGH MEMORY!\n");

# exit(-1);

# }

# strcpy(elem, line);

# append(&head, elem);

# }

# }

# if (j > 0 && strlen(line) != 0) {

# line[j] = '\0';

# if ((elem = (char \* ) malloc(strlen(line))) == NULL) {

# printf("NOT ENOUGH MEMORY!\n");

# exit(-1);

# }

# strcpy(elem, line);

# append(&head, elem);

# }

# return head;

# }

# void append(WORD \*\* headRef, char \* line) {

# WORD \* newWord, \* p;

# newWord = (WORD \* ) malloc(sizeof(WORD));

# newWord -> string = line;

# newWord -> next = NULL;

# if ( \* headRef == NULL) {

# \* headRef = newWord;

# } else {

# p = \* headRef;

# while (p -> next != NULL) {

# p = p -> next;

# }

# p -> next = newWord;

# }

# }

# void print(WORD \*\* head) {

# WORD \* p;

# for (p = \* head; p != NULL; p = p -> next) {

# printf("%s\n", p -> string);

# }

# }

# void delete(WORD \*\* head) {

# WORD \* p = \* head, \* q;

# if (strlen(p -> string) < STR\_LEN) {

# q = p;

# p = p -> next;

# free(q -> string);

# free(q);

# \* head = p;

# }

# 

# while (p != NULL) {

# q = p -> next;

# if (q != NULL && strlen(q -> string) < STR\_LEN) {

# p -> next = q -> next;

# free(q -> string);

# free(q);

# continue;

# }

# p = p -> next;

# }

# 

# }

# void clear(WORD \*\* head) {

# WORD \* p, \* q;

# while ( \* head) {

# p = \* head;

# q = p -> next;

# free(p);

# \* head = q;

# }

# }

# Результаты отладки

После первой компиляции была найдена синтаксическая ошибка: пропущена «;» после вызова функции ***delete()***.

# Результаты работы программы

# 

# Выводы по работе

Разработана программа, запоминающая в двумерном массиве числа от –5 до 15, получаемые случайным образом, определяющая строки с максимальной и минимальной суммами и меняющая эти строки местами.