|  |  |
| --- | --- |
|  | **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 *Робототехника и комплексная автоматизация*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

по дисциплине: «Основы программирования»

|  |  |  |
| --- | --- | --- |
| Студент |  | Камалов Антон Павлович |
| Группа |  | РК6-15Б |
| Тип задания |  | Лабораторная работа №12 |
| Тема лабораторной работы |  | Списки |

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

*подпись, дата фамилия, и.о.*

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

*подпись, дата фамилия, и.о.*

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

*Москва, 2022 г.*

## Цель работы

## Изучение организации структур в языке Си. Изучение функций управления динамической памятью. Применение этих функций для организации динамических связных списков. Изучение функций работы с файлом.

## Ответы на контрольные вопросы

1. Линейный двусвязный список – структура данных, хранящая некую информацию и указатели на предыдущий и следующий узлы.
2. Буферизация - способ организации ввода-вывода в программе, позволяющий минимизировать число обращений к устройству.
3. Структура FILE — это структура, в которой хранится информация о том, как открыт файл (например, для чтения), и где позиция следующего чтения или записи.
4. Файл открывается с помощью функции fopen(). Первым её аргументом адрес строки, содержащей путевое имя файла, а вторым – адрес строки, определяющей тип работы с открываемым файлом.
5. Чтение из файла осуществляются с помощью функций fgets(), fscanf(), fread().
6. Запись в файл осуществляют функции fputs(), fputc(), fprintf(), fwrite().
7. Во-первых, освобождается место в памяти, которое может быть использовано для открытия нового файла. Во-вторых, буфер, если он есть, записывается за приемлемое время.

## Блок-схема программы

## Текст программы

## #include <stdio.h>

## #include <stdlib.h>

## #include <string.h>

## #include <ctype.h>

## #define LINE\_LEN 2048

## typedef struct Node { // узел в списке

## char \* string;

## struct Node \* next;

## int rep;

## }

## WORD;

## WORD \* in (); // ввод слов

## void push(WORD \*\* , char \* ); // добавление слова в стек

## void printList(WORD \* , FILE \* ); // вывод слов из стека

## void delete(WORD \* ); // удаление узлов

## int repeat(WORD \* , char \* ); // есть ли слово в списке?

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

## WORD \* words = NULL;

## FILE \* f;

## f = fopen(argv[1], "w+"); // создаём или открываем файл с функцией записи

## if (count < 2) {

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

## exit(-1);

## }

## words = in ();

## printList(words, f);

## fclose(f); // закрываем файл

## if(words) delete(words); // удаляем только из непустого списка

## }

## WORD \* in () {

## WORD \* head = NULL;

## int j = 0, c;

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

## while ((c = getchar()) != EOF) { // ввод символов

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

## line[j++] = c;

## } else {

## line[j] = '\0';

## j = 0;

## if (strlen(line) == 0) continue; // пустую строку не обрабатываем

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

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

## exit(-1);

## }

## strcpy(elem, line);

## if (repeat(head, elem)) push(&head, elem);

## }

## }

## if (j > 0 && strlen(line) != 0) { // проверка на содержимое буфера

## line[j] = '\0';

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

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

## exit(-1);

## }

## strcpy(elem, line);

## if (repeat(head, elem)) push(&head, elem); // добавляем слово в список, если его там не было (иначе +1 к повтору)

## }

## return head;

## }

## int repeat(WORD \* headRef, char \* line) {

## WORD \* p;

## for (p = headRef; p != NULL; p = p -> next) {

## if (strcmp(p -> string, line) == 0) {

## p -> rep += 1;

## return 0;

## }

## }

## return 1;

## }

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

## WORD \* newWord, \* p;

## if ((newWord = (WORD \* ) malloc(sizeof(WORD))) == NULL) {

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

## exit(-1);

## }

## newWord -> rep = 0, 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 printList(WORD \* head, FILE \* f) {

## WORD \* p;

## fprintf(f, "Words - repeats:\n\n");

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

## fprintf(f, "%s - %d\n", p -> string, p -> rep);

## }

## }

## void delete(WORD \* head) {

## WORD \* p;

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

## p = head;

## head = head -> next;

## free(p);

## }

## }

## Результаты отладки

После первой компиляции была найдена синтаксическая ошибка: пропущена «;» после инициализации переменной ***words***.

## Результаты работы программы

## Выводы по работе

Я изучил организацию структур в языке Си, функции управления динамической памятью, применение этих функций для организации динамических связных списков и функции работы с файлом.