Отчет по лабораторной работе № VIII по курсу
Алгоритмы и структуры данных

Студент группы М8О-103Б-22 Клименко Виталий Максимович, № по списку 11

Контакты www, e-mail, icq, skype vitalikklimenko96@gmail.com

Работа выполнена: 28 мая 2023 г.

Преподаватель: доцент Никулин С.П.

Входной контроль знаний с оценкой _____

Отчет сдан « » _____ 202 __ г., итоговая оценка ___

Подпись преподавателя _____

1. **Тема:** Линейные списки

2. **Цель работы:** Составить и отладить программу на языке Си для обработки линейного списка заданной организации с отображением списка на динамические структуры

3. **Задание** (*вариант № 11 (4, 6, 12)*): Вид списка - линейный двунаправленный с барьерным элементом, тип элемента списка - строковый, нестандартное действие - проверка упорядоченности

4. **Оборудование** (*лабораторное*):
ЭВМ _____, процессор _____, имя узла сети _____ с ОП _____ Мб,
НМД _____ Мб. Терминал _____ адрес _____. Принтер _____
Другие устройства _____

Оборудование ПЭВМ студента, если использовалось:
Процессор Intel 4x 3.5GHz с ОП 16 ГБ НМД HDD 200 ГБ . Монитор Встроенный 1920x1080
Другие устройства Touchpad Synaptics

5. **Программное обеспечение** (*лабораторное*):
Операционная система семейства _____, наименование _____ версия _____
интерпретатор команд _____ версия _____
Система программирования _____ версия _____
Редактор текстов _____ версия _____
Утилиты операционной системы _____
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных _____

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства UNIX, наименование Pop!_OS версия 22.04 jammy
интерпретатор команд bash версия 5.1.16
Система программирования _____ версия _____
Редактор текстов _____ версия _____
Утилиты операционной системы _____
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных на домашнем компьютере _____

6. **Идея, метод, алгоритм** решение задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

Составить структуру данных linked_list, в которой будут находиться указатели на предыдущий и следующий элементы и значение - указатель на начало строки
В качестве барьерного элемента я использую пустую строку

7. **Сценарий выполнения работы** (план работы, первоначальный текст программы в черновике [можно на отдельном листе] и тесты либо соображения по тестированию)

Для удобства написания функций над этой структурой данных были написаны функции, которые возвращают указатель на самый левый и на самый правый элементы переданного списка
Удаление элемента происходит так: сначала освобождается значение элемента, указатели на предыдущий и следующий элемент "склеиваются", чтобы не проходить удаленный элемент в других функциях, потом освобождается память под указателем, который был передан изначально
В ходе отладки программы была использована программа valgrind, которая помогла избавиться от утечек памяти

Пункты 1-7 отчета составляются строго до начала лабораторной работы.
Допущен к выполнению работы. **Подпись преподавателя** _____

8. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем)

```
vitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ ls
./  ../  linked-list.h*  main.c*  makefile*
vitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ cat main.c
#define LINKED_LIST_IMPLEMENTATION
#include "linked-list.h"

#include <assert.h>

int main() {
    linked_list *ll = ll_alloc();

    ll_set_value(ll, "1");

    ll_push_right(ll, "7");
    ll_push_right(ll, "2");
    ll_push_right(ll, "3");
    ll_push_right(ll, "6");
    ll_push_right(ll, "5");
    printf("Initial list:\n");
    ll_print(ll);

    ll = ll_remove(ll->next);
    printf("Without second element:\n");
    ll_print(ll);

    ll = ll_remove_left(ll);
    printf("Without leftmost element:\n");
    ll_print(ll);

    if (ll_is_sorted(ll)) {
        printf("Linked list is sorted!\n");
    } else {
        printf("Linked list is not sorted!\n");
    }

    ll = ll_remove_right(ll);
    printf("Without rightmost element:\n");
    ll_print(ll);

    if (ll_is_sorted(ll)) {
        printf("Linked list is sorted!\n");
    } else {
        printf("Linked list is not sorted!\n");
    }

    ll_dealloc(ll);

    return 0;
}vitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ cat makefile
CC = gcc
CFLAGS = -std=c99 -Wall -Wextra -Wformat=0

main:
        $(CC) $(CFLAGS) -o main.out main.c
debug:
        $(CC) $(CFLAGS) -g -o main.out main.c
clean:
        rm -f *.o main.outvitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ cat linked-list.h
#ifndef LINKED_LIST_H
#define LINKED_LIST_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

typedef struct linked_list {
    struct linked_list *prev;
    struct linked_list *next;
    char *value;
} linked_list;


linked_list* ll_alloc();

void ll_set_value(linked_list *ll, const char *value);
void ll_push_right(linked_list *ll, const char *value);

linked_list* ll_leftmost(linked_list *ll);
```

```
linked_list* ll_rightmost(linked_list *ll);

linked_list* ll_remove(linked_list *ll);
linked_list* ll_remove_right(linked_list *ll);
linked_list* ll_remove_left(linked_list *ll);

void ll_print(linked_list *ll);
void ll_print_debug(linked_list *ll);

size_t ll_is_sorted(linked_list *ll);

void ll_dealloc(linked_list *ll);

#endif // LINKED_LIST_H



#ifdef LINKED_LIST_IMPLEMENTATION

linked_list* ll_alloc() {
    linked_list *ll = (linked_list*) calloc(1, sizeof(linked_list));

    ll->prev = NULL;
    ll->next = NULL;
    ll->value = (char*) calloc(1, sizeof(char)); // barrier element = "\0";

    return ll;
}


void ll_set_value(linked_list *ll, const char *value) {
    if (strcmp(ll->value, "") == 0) {
        linked_list *pl, *nl;

        if (ll->prev == NULL) {
            pl = ll_alloc();
            pl->next = ll;
            pl->prev = ll->prev;
            ll->prev = pl;
        }

        if (ll->next == NULL) {
            nl = ll_alloc();
            nl->next = ll->next;
            nl->prev = ll;
            ll->next = nl;
        }
    }

    free(ll->value);

    char* s = (char*) calloc(strlen(value) + 1, sizeof(char));
    strcpy(s, value);

    ll->value = s;
}

void ll_push_right(linked_list *ll, const char *value) {
    ll_set_value(ll_rightmost(ll), value);
}


linked_list* ll_leftmost(linked_list *ll) {
    while (strcmp(ll->value, "") != 0) {
        ll = ll->prev;
    }

    return ll;
}

linked_list* ll_rightmost(linked_list *ll) {
    while (strcmp(ll->value, "") != 0) {
        ll = ll->next;
    }

    return ll;
}


linked_list* ll_remove(linked_list *ll) {
    linked_list *next = ll->next;
    linked_list *prev = ll->prev;

    free(ll->value);
    free(ll);
```

```
    ll = prev;

    ll->next = next;
    ll->next->prev = prev;

    return prev;
}

linked_list* ll_remove_right(linked_list *ll) {
    ll = ll_rightmost(ll);
    return ll_remove(ll->prev)->prev;
}

linked_list* ll_remove_left(linked_list *ll) {
    ll = ll_leftmost(ll);
    return ll_remove(ll->next)->next;
}


void ll_print(linked_list *ll) {
    if (strcmp(ll->value, "") == 0) {
        printf("\n");
        return;
    }

    ll = ll_leftmost(ll);
    ll = ll->next;

    while (strcmp(ll->value, "") != 0) {
        printf("%s, ", ll->value);
        ll = ll->next;
    }

    printf("\b\b \b\n");
}

void ll_print_debug(linked_list *ll) {
    printf("ll: *%d, prev: *%d, next: *%d, value: %s\n", ll, ll->prev, ll->next, ll->value);
}


size_t ll_is_sorted(linked_list *ll) {
    ll = ll_leftmost(ll);

    while (strcmp(ll->next->value, "") != 0 && strcmp(ll->value, ll->next->value) <= 0) {
        ll = ll->next;
    }

    if (strcmp(ll->next->value, "") == 0) {
        return 1;
    } else {
        return 0;
    }
}


void ll_dealloc(linked_list *ll) {
    ll = ll_leftmost(ll);

    // do this outside of while loop, because ll.value = "\0"
    free(ll->value);
    ll = ll->next;

    while (strcmp(ll->value, "") != 0) {
        // printf("freeing value = %s\n", ll->value);
        // ll_print(ll);
        free(ll->value);
        free(ll->prev);

        ll = ll->next;
    }

    free(ll->prev);
    free(ll->value);
    free(ll);
}

#endif // LINKED_LIST_IMPLEMENTATIONvitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ make
gcc -std=c99 -Wall -Wextra -Wformat=0 -o main.out main.c
vitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ ./ma
main.c    main.out  makefile
vitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ ./main.out
Initial list:
1, 7, 2, 3, 6, 5
```

```
Without second element:
1, 2, 3, 6, 5
Without leftmost element:
2, 3, 6, 5
Linked list is not sorted!
Without rightmost element:
2, 3, 6
Linked list is sorted!
vitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ cat main.c
#define LINKED_LIST_IMPLEMENTATION
#include "linked-list.h"

#include <assert.h>

int main() {
    linked_list *ll = ll_alloc();

    ll_set_value(ll, "aaa");

    ll_push_right(ll, "b");
    ll_push_right(ll, "c");
    ll_push_right(ll, "eerh");
    ll_push_right(ll, "d");
    ll_push_right(ll, "faa");
    printf("Initial list:\n");
    ll_print(ll);

    ll = ll_remove(ll->next);
    printf("Without second element:\n");
    ll_print(ll);

    ll = ll_remove_left(ll);
    printf("Without leftmost element:\n");
    ll_print(ll);

    if (ll_is_sorted(ll)) {
        printf("Linked list is sorted!\n");
    } else {
        printf("Linked list is not sorted!\n");
    }

    ll = ll_remove_right(ll);
    printf("Without rightmost element:\n");
    ll_print(ll);

    if (ll_is_sorted(ll)) {
        printf("Linked list is sorted!\n");
    } else {
        printf("Linked list is not sorted!\n");
    }

    ll_dealloc(ll);

    return 0;
}vitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ make
gcc -std=c99 -Wall -Wextra -Wformat=0 -o main.out main.c
vitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ ./main.
main.c    main.out
vitos@vitos-hp16:/mnt/c/important/docs/mai/labs/lviii$ ./main.out
Initial list:
aaa, b, c, eerh, d, faa
Without second element:
aaa, c, eerh, d, faa
Without leftmost element:
c, eerh, d, faa
Linked list is not sorted!
Without rightmost element:
c, eerh, d
Linked list is not sorted!
```

9. **Дневник отладки** должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. **Замечания автора** по существу работы: _____

_____

_____

_____

11. **Выводы:** Я научился реализовывать связные списки на языке Си. В ходе работы над лабораторной работой больше узнал про утечки памяти. _____

_____

_____

Недочёты при выполнении задания могут быть устранены следующим образом: _____

_____

_____

_____

_____

Подпись студента _____