**Отчет по лабораторной работе № 25/26** по курсу “Языки и методы программирования”

Студент группы М80-103Б-21 Зверева Елизавета Леонидовна, № по списку 11

e-mail: [elizavetka.zvereva.2003@mail.ru](mailto:elizavetka.zvereva.2003@mail.ru) , telegram: @banshee

Работа выполнена: «» сентября 2021г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

Отчет сдан « » \_\_\_\_\_\_\_\_\_20\_\_ г., итоговая оценка \_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Тема:** **Автоматизация сборки программ модульной структуры на языке Си с использованием утилиты make. Абстрактные типы данных. Рекурсия. Модульное программирование на языке Си.**

1. **Цель работы:** Изучить работу с утилитой make. Создать и отладить модуль с реализацией заданного абстрактного типа данных.
2. **Задание (*вариант № 4/6* ):** Создать тип данных линейный список. Сортировка вставка.
3. **Оборудование** (студента):

Процессор *AMD A9-9420 RADEON R5, 5 COMPUTE CORES 2C+3G 3.00 GHz* с ОП *8* Гб, НМД *512* Гб. Монитор *1920x1080*

1. **Программное обеспечение (**студента**):**

Операционная система семейства: *linux*, наименование: *ubuntu*, версия *20.04 focal*

интерпретатор команд: *bash* версия *5.0.17*

Редактор текстов *emacs* версия *3.24.14*

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

Необходимо написать отдельный файл с Си кодом, в котором будут реализованы основные методы для работы с линейным списком. Для сборки проекта напишем Makefile

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

**написать функции:**

**1) создание линейного списка**

**2)нахождение размера**

**3)вставка**

**4)нахождение элемента в списке**

**5)печать списка**

**6)проверка на пустоту**

**liza@liza-HLYL-WXX9:~/Рабочий стол/laba26$ make**

**gcc -c main.c**

**gcc -c list.c**

**cc main.o list.o -o main**

**liza@liza-HLYL-WXX9:~/Рабочий стол/laba26$ ./main**

**Working with list.Enter Code of command:**

**Code a: create list.**

**code h: add element**

**code f:find element.**

**code d: delete element**

**code e: check list it is empty**

**code p: print list**

**code s: sort**

**code #:exit**

**a1**

**h4**

**h8**

**h4**

**h9**

**h3**

**h5**

**p**

**1 4 8 4 9 3 5**

**e**

**List is not empty**

**f5**

**Found 1 element**

**d5**

**p**

**1 4 8 4 9 3**

**s**

**p**

**1 3 4 4 8 9**

#

liza@liza-HLYL-WXX9:~/Рабочий стол/laba26$ make clean

rm -rf \*.o main

**8. Распечатка протокола**

**list.h**

**#ifndef list\_h**

**#define list\_h**

**#include <stdio.h>**

**#include "stdlib.h"**

**typedef struct list**

**{**

**int k;**

**struct list \*next;**

**} list;**

**typedef struct list Node;**

**struct list \*create\_list(struct list \*l, int n);**

**int empty(struct list \*l);**

**void push(struct list \*l, int k);**

**int find(struct list \*l, int n, int r);**

**struct list \*delete( struct list \*l, int n);**

**void output(struct list \*l);**

**int length(struct list \*l, int r);**

**struct list \*sort(struct list \*l);**

**#endif**

**list.c**

**#include "list.h"**

**#include <stdio.h>**

**#include <stdlib.h>**

**struct list \*create\_list(struct list \*l, int n) {**

**if (l == NULL)**

**{**

**struct list \*l = malloc(sizeof(struct list));**

**l->k = n;**

**l->next = NULL;**

**return l;**

**}**

**l->next = create\_list(l->next, n);**

**}**

**void push(struct list \*l, int k) {**

**struct list \*current = l;**

**while (current->next != NULL) {**

**current = current->next;**

**}**

**current->next = malloc(sizeof(struct list));**

**current->next->k = k;**

**current->next->next = NULL;**

**}**

**int empty(struct list \*l) {**

**if (l == NULL)**

**return 0;**

**else**

**return 1;**

**}**

**struct list \*delete(struct list \*l, int n)**

**{**

**if (l->k == n)**

**return l->next;**

**if (l == NULL)**

**return l;**

**if ((l->next)->k == n)**

**{**

**l->next = (l->next)->next;**

**return l;**

**}**

**delete(l->next, n);**

**return l;**

**}**

**void output(struct list \*l)**

**{**

**if (l == NULL)**

**return;**

**printf("%d ", l->k);**

**output(l->next);**

**}**

**int find(struct list \*l, int n, int r)**

**{**

**if (l == NULL)**

**return r;**

**if (l->k == n)**

**r++;**

**find(l->next, n, r);**

**}**

**int length(struct list \*l, int r)**

**{**

**if (l == NULL)**

**return r;**

**r++;**

**length(l->next, r);**

**}**

**struct list\* sort(struct list\* head){**

**if(head == NULL || head->next ==NULL)**

**return head;**

**Node\* SortList = (Node\*)malloc(sizeof(Node));**

**SortList->next = head;**

**head = head->next;**

**SortList->next->next = NULL;**

**// Остальные узлы вставлены**

**Node\* cur = head;**

**while(cur)**

**{**

**Node\* next = cur->next;**

**// Начиная с заголовка отсортированного списка, находим подходящую позицию для вставляемого узла**

**Node\* sortprev = SortList;**

**Node\* sorttail = SortList->next;**

**while(sorttail)**

**{**

**if(cur->k > sorttail->k)**

**{**

**sortprev = sorttail;**

**sorttail = sorttail->next;**

**}**

**else**

**{**

**break;**

**}**

**}**

**// Вставляем в нужное место**

**sortprev->next = cur;**

**cur->next = sorttail;**

**cur = next;**

**}**

**Node\* list = SortList->next;**

**free(SortList);**

**return list;**

**}**

**main.c**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include "list.h"**

**int main() {**

**struct list \*L = NULL;**

**printf("Working with list.Enter Code of command:\n Code a: create list.\n code h: add element\n code f:find element.\n code d: delete element\n code e: check list it is empty\n code p: print list\n code s: sort\n code #:exit\n");**

**char op = 'a';**

**int e = 1;**

**while (op != '#')**

**{**

**scanf("%c", &op);**

**if (op == 'a')**

**{**

**scanf("%d", &e);**

**L = create\_list(L, e);**

**}**

**if (op == 'h')**

**{**

**scanf("%d", &e);**

**push(L, e);**

**}**

**if (op == 'f')**

**{**

**scanf("%d", &e);**

**int r = find(L, e, 0);**

**if (r == 0)**

**printf("Not found\n");**

**if (r == 1)**

**printf("Found 1 element\n");**

**if (r > 1)**

**printf("Found %d elements\n", r);**

**}**

**if (op == 'd')**

**{**

**scanf("%d", &e);**

**int r = find(L, e, 0);**

**for (int i = 0; i < r; i++)**

**L = delete(L, e);**

**}**

**if (op == 'e')**

**{**

**if (empty(L) == 0)**

**printf("List is empty\n");**

**else**

**printf("List is not empty\n");**

**}**

**if (op == 'p')**

**{**

**output(L);**

**printf("\n");**

**}**

**if (op == 's')**

**{**

**L = sort(L);**

**}**

**}**

**return 0;**

**}**

**makefile**

**main: main.o list.o**

**main.o: main.c**

**gcc -c main.c**

**list.o: list.h list.c**

**gcc -c list.c**

**clean:**

**-rm -rf \*.o main**

**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 других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
| 1 | - |  |  | - | - | - |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора.** Нет.

**Выводы.** Результатом лабораторной работы стала программа и файл для её сборки. В процессе выполнения задания были изучены основы работы с утилитой make. Определённо, утилита make значительно упрощает жизнь при отладки программ и их сборки.

  Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_