**Отчет по лабораторной работе № 25,26** по курсу “Фундаментальная информатика”

Студент группы М80-103Б-21 Катин Иван Вячеславович, № по списку 12

Контакты e-mail: ikatin.2003.sokol@gmail.com, telegram: @Dazz1e

Работа выполнена: «8» апреля2022г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

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

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

**Тема:** Абстрактные типы данных. Модульное программирование на Си. Makefile.

1. **Цель работы:** Изучить абстрактные типы данных, работу Makefile-оф.
2. **Задание:** 1-1. Стек, метод и продецура: сортировка линейным выбором, удаление и поиск макс. элемента
3. **Оборудование** (студента):

Процессор *Intel® Core™ i5-9300H CPU @ 2.40GHz × 8* с ОП 7,6 GiB, НМД *1024* Гб. Монитор *1920x1080*

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

Операционная система семейства: *linux*, наименование: *ubuntu*, версия *20.04.3 LTS*

интерпретатор команд: *bash* версия *4.4.20(1)-release*.

Система программирования -- CLion--**,** редактор текстов *emacs* версия *25.2.2*

Утилиты операционной системы --

Прикладные системы и программы – **LibreOffice**

Местонахождение и имена файлов программ и данных на домашнем компьютере – *home*/dazzle

**6**. **Идея, метод, алгоритм.** Реализовать структуру данных - стек. Будет состоять из указателя последнего элемента, массива элементов и его максимальной размерности. Если места в массиве не хватает, вызываем реалок. Удаление и вставка, получение элемента - делается для последнего элемента в массиве на него указывает ptr -1. Сложность вставки : O(1)(если место в массиве заканчивается, то за O(n). Сортировка выполняется с помощью двух циклов => сложность = O(). Поиск и удаление максимального элемента за O(n).

**7**. **Сценарий выполнения работы**

1. Создание структуры в STACK.H
2. Объявление методов(print, pop, top, max\_top\_delete, clear, sort, push.size, isEmpty) в этом файле.

* print- распечатка
* pop - удаление последнего элемента
* top - получение последнего элемента
* push - вставка
* clear - очистка стека
* max\_top\_delete - поиск и удаление максимального элемента
* size - размер стека
* isEmpty - проверка на наличие элементов

1. Создаем файл stack.c с реализацией методов
2. Создаем main.c, в котором обрабатываем запросы и вызывает нужный метод.

**8. Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми

примерами,подписанныйпреподавателем).

**STACK.h**

| #ifndef UNTITLED4\_STACK\_H  #define UNTITLED4\_STACK\_H  #include <stdbool.h>  typedef struct {  int ptr;  int maxSize;  int\* elements;  }stack;  stack\* create();  bool is\_empty(stack\* s);  void push(stack\* s, int value);  void pop(stack\* s);  int top(stack\* s);  void sort(stack\* s);  void clear(stack\* s);  int max\_top\_delete(stack\* s);  void print(stack\* s);  int size(stack\* s);  #endif //UNTITLED4\_STACK\_H |
| --- |

**stack.c**

| **#include "STACK.h"**  **#include <stdio.h>**  **#include <stdbool.h>**  **#include <stdlib.h>**  **#include <malloc.h>**  **stack\* create (){**  **stack\* s = (stack\*) malloc(sizeof (stack));**  **s->maxSize = 100;**  **s->elements = (int\*) malloc(sizeof(int)\*100);**  **s->ptr = 0;**  **return s;**  **}**  **void push(stack\* s, int value){**  **if(s->maxSize == s->ptr){**  **realloc(s->elements,sizeof(int) \*s->maxSize \* 2);**  **s->maxSize \*= 2;**  **}**  **s->elements[s->ptr++] = value;**  **}**  **void pop(stack\* s){**  **if(s->ptr == 0){**  **perror("STACK IS EMPTY");**  **exit(1);**  **}**  **if(s->ptr == 0) return;**  **s->ptr--;**  **}**  **int top(stack\* s){**  **if(s->ptr == 0){**  **perror("STACK IS EMPTY");**  **exit(1);**  **}**  **return s->elements[s->ptr-1];**  **}**  **int size(stack\* s){**  **return s->ptr;**  **}**  **bool is\_empty(stack\* s){**  **if(s->ptr == 0) return true;**  **return false;**  **}**  **void print(stack\* s){**  **for(int i =0; i < s->ptr; i++){**  **printf("%d ", s->elements[i]);**  **}**  **printf("\n");**  **}**  **void clear(stack \*s)**  **{**  **free(s->elements);**  **s->ptr = 0;**  **}**  **int max\_top\_delete(stack\* s){**  **if(s->ptr == 0){**  **perror("STACK IS EMPTY");**  **exit(1);**  **}**  **if(s->ptr == 0) {**  **perror("STACK IS EMPTY");**  **exit(1);**  **}**  **int maxEl = -1e9;**  **int index = 0;**  **for (int i = 0; i < s->ptr; ++i) {**  **if(s->elements[i] > maxEl){**  **maxEl = s->elements[i];**  **index = i;**  **}**  **}**  **for(; index < s->ptr; index++){**  **if(index == s->ptr - 1){**  **s->ptr--;**  **return maxEl;**  **}**  **s->elements[index] = s->elements[index+1];**  **}**  **}**  **void sort(stack\* s){**  **if(s->ptr == 0){**  **perror("STACK IS EMPTY");**  **exit(1);**  **}**  **for(int i =0; i < s->ptr; i++){**  **int tmp = s->elements[i];**  **int minEl = i;**  **for (int j = i+1; j < s->ptr; ++j) {**  **if(s->elements[minEl] > s->elements[j]){**  **minEl = j;**  **}**  **}**  **s->elements[i] = s->elements[minEl];**  **s->elements[minEl] = tmp;**  **}**  **}** |
| --- |

**main.c**

| **#include "STACK.h"**  **#include <stdio.h>**  **#include <stdbool.h>**  **#include <stdlib.h>**  **#include <string.h>**  **int main() {**  **char zapros[100];**  **stack\* s = create();**  **printf("pop, top, clear, isEmpty, push, max\_top\_delete, size, print, sort\n");**  **while(scanf("%s", zapros) != EOF){**  **if(strcmp(zapros, "pop") == 0){**  **pop(s);**  **}**  **if(strcmp(zapros, "top") == 0){**  **printf("%d\n", top(s));**  **}**  **if(strcmp(zapros, "clear") == 0){**  **clear(s);**  **}**  **if(strcmp(zapros, "isEmpty") == 0){**  **if(is\_empty(s)){**  **printf("true\n");**  **} else {**  **printf("false\n");**  **}**  **}**  **if(strcmp(zapros, "push") == 0){**  **int value;**  **printf("Value = ");**  **scanf("%d", &value);**  **push(s, value);**  **}**  **if(strcmp(zapros, "max\_top\_delete") == 0){**  **printf("%d\n", max\_top\_delete(s));**  **}**  **if(strcmp(zapros, "size") == 0){**  **printf("%d\n", size(s));**  **}**  **if(strcmp(zapros, "print") == 0){**  **print(s);**  **}**  **if(strcmp(zapros, "sort") == 0){**  **sort(s);**  **}**  **printf("pop, top, clear, isEmpty, push, max\_top\_delete, size, print, sort\n");**  **}**  **}** |
| --- |

**9. Дневник отладки**

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

1. **Замечания автора**
2. **Выводы**

Поработал снова с заголовочными файлами. Узнал о makefile, что позволяет сделать компиляцию не только проще,но и быстрее , можно(компилировать файлы только те, которые изменились). Узнал о нескольких реализациях стека,очереди, дека,cписка, получил понимание, как работают эти структуры данных.

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