Отчет по лабораторной работе № 25+26 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: 8 мая 2023 г.

Преподаватель: доцент Никулин С.П.

Входной контроль знаний с оценкой _____

Отчет сдан « » _____ 202 __ 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

2. **Цель работы:** Научиться пользоваться утилитой make. Научиться составлять линейные структуры данных и реализовывать для них алгоритмы

3. **Задание** (*вариант № 1 - 1*): Разработать makefile для компиляции программы с реализацией стека и функций для него.

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)

   Создать makefile, который отдельно компилирует stack.c и main.c в объектные файлы, также отдельную команду для очистки от объектных файлов и от исполняемого файла

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

   Разобратья в том, как собирать makefile, почитать про реализацию стека, про сортировку линейный выбором, составить на основе этих знаний программу

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)

```
vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ cat makefile
CC = gcc
CFLAGS = -std=c99 -Wall -Wextra

main: main.o stack.o
        $(CC) $(CFLAGS) -o main.out main.o stack.o
main_o:
        $(CC) $(CFLAGS) -c main.c
stack_o:
        $(CC) $(CFLAGS) -c stack.c
clean:
        rm -f *.o main.out
vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ cat main.c
#include <stdio.h>
#include "stack.h"

int main() {
    stack* s = stack_empty();

    int64_t arr[9] = {5, 13, 7, 4, 8, 1, -152, 7, 12};

    for (int i = 0; i < 5; ++i) {
        stack_push(s, i);
    }

    for (int i = 0; i < 9; ++i) {
        stack_push(s, arr[i]);
    }
    printf("Stack before popping:\n");
    stack_print(s);

    for (int i = 0; i < 3; ++i) {
        stack_pop(s);
    }
    printf("Stack after popping:\n");
    stack_print(s);

    stack_delete_max(s);
    printf("Stack after deleting max element from it:\n");
    stack_print(s);

    stack_sort(s);
    printf("Stack after sorting:\n");
    stack_print(s);

    stack_delete_max(s);
    printf("Stack after deleting max element from it:\n");
    stack_print(s);

    for (int i = 0; i < 3; ++i) {
        stack_pop(s);
    }
    printf("Stack after popping 3 elements from it:\n");
    stack_print(s);

    for (int i = 10; i < 15; ++i) {
        stack_push(s, -i + 40);
    }
    printf("Stack after pushing to it:\n");
    stack_print(s);

    stack_delete_max(s);
    printf("Stack after deleting max element from it:\n");
    stack_print(s);

    stack_sort(s);
    printf("Stack after sorting:\n");
    stack_print(s);

    return 0;
}

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ cat stack.h
#include <inttypes.h>
#include <stdbool.h>
```

```
typedef struct stack {
    uint64_t _size;
    uint64_t _capacity;
    int64_t *_arr;
} stack;


stack* stack_empty();
stack* stack_from_array(uint64_t size, int64_t* elements);

uint64_t stack_size(stack* s);
uint64_t stack_capacity(stack* s);
bool stack_is_empty(stack* s);
void stack_print(stack* s);

int64_t stack_pop(stack* s);
void stack_delete_at(stack* s, uint64_t index);

void stack_push(stack* s, int64_t e);
void stack_push_array(stack* s, uint64_t size, int64_t* elements);

void stack_free (stack *s);

void stack_sort(stack* s);
void stack_delete_max(stack* s);
vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ cat stack.c
#ifndef STACK_H
#define STACK_H

#include "stack.h"

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


stack* stack_empty() {
    stack* s;
    s = (stack*) calloc(1, sizeof(stack));
    s->_arr = calloc(1, sizeof(int64_t));
    s->_size = 0;
    s->_capacity = 1;
    return s;
}

stack* stack_from_array(uint64_t size, int64_t* elements) {
    stack* s = stack_empty();
    stack_push_array(s, size, elements);
    return s;
}


uint64_t stack_size(stack* s) {
    return s->_size;
}

uint64_t stack_capacity(stack* s) {
    return s->_capacity;
}

bool stack_is_empty(stack* s) {
    return s->_size == 0;
}

void stack_print(stack* s) {
    uint64_t length = s->_size;
    stack* temp = stack_empty();

    for (uint64_t i = 0; i < length; ++i) {
        int64_t top = stack_pop(s);
        stack_push(temp, top);
        printf("%I64d\n", top);
    }

    for (uint64_t i = 0; i < length; ++i) {
        stack_push(s, stack_pop(temp));
    }
}


int64_t stack_pop(stack* s) {
    if (s->_size == 0) {
        printf("Stack size is zero! Bailing...\n");
        exit(1);
```

```
    }
    s->_size--;
    return s->_arr[s->_size];
}

void stack_delete_at(stack* s, uint64_t index) {
    assert(index < s->_size);

    stack* temp = stack_empty();
    for (uint64_t i = 0; i < index; ++i) {
        stack_push(temp, stack_pop(s));
    }

    stack_pop(s);

    for (uint64_t i = index; i > 0; --i) {
        stack_push(s, stack_pop(temp));
    }
}


void _stack_resize(stack* s, uint64_t new_capacity) {
    s->_capacity = new_capacity;
    s->_arr = (int64_t*) realloc(s->_arr, new_capacity * sizeof(int64_t));
}

void stack_push(stack* s, int64_t e) {
    if (s->_size + 1 > s->_capacity) {
        _stack_resize(s, 2 * s->_capacity);
    }
    s->_arr[s->_size] = e;
    s->_size++;
    s->_arr[s->_size] = 0;
}

void stack_push_array(stack* s, uint64_t size, int64_t* elements) {
    for (uint64_t i = 0; i < size; ++i) {
        stack_push(s, elements[i]);
    }
}


void stack_free(stack* s) {
    s->_size = 0;
    s->_capacity = 0;
    free(s->_arr);
}



void stack_sort(stack* s) {
    stack* sorted = stack_empty();

    stack* temp = stack_empty();
    stack* temp1 = stack_empty();

    while (!stack_is_empty(s)) {
        uint64_t length = s->_size;

        uint64_t min_index = 0;
        int64_t min_element = stack_pop(s);
        stack_push(temp, min_element);

        for (uint64_t j = 1; j < length; ++j) {
            int64_t min_element1 = stack_pop(s);
            stack_push(temp1, min_element1);
            if (min_element1 < min_element) {
                min_index = j;
                min_element = min_element1;
            }
        }

        if (min_index != 0) {
            for (uint64_t j = min_index + 1; j < length; ++j)
                stack_push(s, stack_pop(temp1));

            stack_push(sorted, stack_pop(temp1));

            for (uint64_t j = 1; j < min_index; ++j)
                stack_push(s, stack_pop(temp1));

            stack_push(s, stack_pop(temp));
        } else {
            stack_push(sorted, stack_pop(temp));
            for (uint64_t j = 1; j < length; ++j) {
```

```
                stack_push(s, stack_pop(temp1));
            }
        }
    }
    *s = *sorted;
}

void stack_delete_max(stack* s) {
    stack* temp = stack_empty();

    uint64_t max_index = 0;
    int64_t max_element = INT64_MIN;

    uint64_t length = s->_size;

    for (uint64_t i = 0; i < length; ++i) {
        int64_t top = stack_pop(s);
        if (top > max_element) {
            max_element = top;
            max_index = i;
        }
        stack_push(temp, top);
    }

    for (uint64_t i = 0; i < length; ++i) {
        stack_push(s, stack_pop(temp));
    }

    stack_delete_at(s, max_index);
}


#endif
vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ make stack_o
gcc -std=c99 -Wall -Wextra -c stack.c

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ make
gcc -std=c99 -Wall -Wextra   -c -o main.o main.c
gcc -std=c99 -Wall -Wextra -o main.out main.o stack.o

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ ls
./  ../  l25-2012.djvu  l26-2012.djvu  main.c  main.o  main.out*  makefile  protocol.txt  report.pdf  stack.c  stack.h  stack

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ make clean
rm -f *.o main.out

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ ls
./  ../  l25-2012.djvu  l26-2012.djvu  main.c  makefile  protocol.txt  report.pdf  stack.c  stack.h

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ make
gcc -std=c99 -Wall -Wextra   -c -o main.o main.c
gcc -std=c99 -Wall -Wextra   -c -o stack.o stack.c
gcc -std=c99 -Wall -Wextra -o main.out main.o stack.o

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ ls
./  ../  l25-2012.djvu  l26-2012.djvu  main.c  main.o  main.out*  makefile  protocol.txt  report.pdf  stack.c  stack.h  stack

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l25-26 (master)
$ ./main.out
Stack before popping:
12
7
-152
1
8
4
7
13
5
4
3
2
1
0
Stack after popping:
1
8
4
```

```
7
13
5
4
3
2
1
0
Stack after deleting max element from it:
1
8
4
7
5
4
3
2
1
0
Stack after sorting:
8
7
5
4
4
3
2
1
1
0
Stack after deleting max element from it:
7
5
4
4
3
2
1
1
0
Stack after popping 3 elements from it:
4
3
2
1
1
0
Stack after pushing to it:
26
27
28
29
30
4
3
2
1
1
0
Stack after deleting max element from it:
26
27
28
29
4
3
2
1
1
0
Stack after sorting:
29
28
27
26
4
3
2
1
1
0
```

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 продвинуто пользоваться утилитой make. Узнал больше о реализации линейных структур

Недочёты при выполнении задания могут быть устранены следующим образом: _____

Подпись студента _____