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

Студент группы М80-101Б-21 Ершова Станислава Григорьевича, № по списку 8

Контакты e-mail, telegram: [stas.ershov57@gmail.com](mailto:stas.ershov57@gmail.com) , @stas\_orel

Работа выполнена: «2» декобря 2020г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

Отчет сдан «2» декобря 2021 г., итоговая оценка \_\_\_\_\_\_\_\_

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

* 1. **Тема:** **Makefile, Структуры данных стэк, дэк, очередь, связанный списко**

1. **Цель работы: Научиться польщоваться make и понять структры данных**
2. **Задание** (*вариант №* **8**)**:** **1;3 — Стэк, соединение двух стэков, сортировка Хоасе на стэке**
3. **Оборудование** (студента):

Процессор *Intel Pentium N4200 1.1 ГГц* с ОП *8* Гб, SSD *128* Гб. Монитор *1920x1080*

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

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

интерпретатор команд: *bash* версия *5.0.17*

Система программирования -- версия --**,** редактор текстов *nano* версия *4.8*

Утилиты операционной системы --

Прикладные системы и программы --

Местонахождение и имена файлов программ и данных на домашнем компьютере --

**6. Идея, метод, алгоритм:**

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

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

**Makefile:**

all: prog

prog: stack\_list.h stack\_list.o main.o

gcc main.o stack\_list.o -o prog

main.o: main.c

gcc -c main.c

stack\_list.o: stack\_list.c

gcc -c stack\_list.c

clean:

rm \*.o

**Stack\_list.c:**

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include "stack\_list.h"

Stack \*stack\_create()

{

Stack \*s = malloc(sizeof(Stack));

s->head = NULL;

s->size = 0;

return s;

}

StackNode \*insert\_stack\_node(StackNode \*parent, Item value)

{

StackNode \*new\_node = malloc(sizeof(StackNode));

new\_node->data = value;

new\_node->next = parent;

return new\_node;

}

void stack\_push(Stack \*s, Item value)

{

s->head = insert\_stack\_node(s->head, value);

s->size++;

}

Item stack\_top(Stack \*s) {

if (s->head != NULL)

return s->head->data;

else

printf("Стек пустой, невозможно выолнить top\n");

}

void stack\_pop(Stack \*s)

{

if (s->head != NULL) {

StackNode \*tmp = s->head->next;

free(s->head);

s->head = tmp;

s->size--;

}

else

printf("Стек пустой, невозможно выолнить top\n");

}

bool stack\_is\_empty(Stack \*s)

{

return (s->head == NULL);

}

void stack\_join(Stack \*a, Stack \*b)

{

struct stacknode \*buttom\_b = b->head;

while (buttom\_b->next != NULL)

buttom\_b = buttom\_b->next;

buttom\_b->next = a->head;

a->head = b->head;

a->size += b->size;

}

void print\_stack(Stack \*s) {

StackNode \*s\_node = s->head;

while (s\_node->next != NULL) {

printf("%d ", s\_node->data);

s\_node = s\_node->next;

}

if (s->size > 0)

printf("%d", s\_node->data);

printf("\n");

}

**hoare\_stack.c:**

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include "stack\_list.h"

typedef struct

{

StackNode \*l;

StackNode \*r;

} Pivot\_lr;

Pivot\_lr partition(Stack \*st, StackNode \*low, StackNode \*high)

{

StackNode \*pivot = low;

StackNode \*i = NULL;

StackNode \*j = high;

StackNode \*right\_pivot;

while (j != NULL) {

if (j->next == low) {

right\_pivot = j;

}

if (j->data <= pivot->data) {

if (i == NULL)

i = high;

else

i = i->next;

int temp\_data = i->data;

i->data = j->data;

j->data = temp\_data;

j = j->next;

} else {

j = j->next;

}

}

Pivot\_lr ret;

ret.l = pivot->next;

ret.r = right\_pivot;

return ret;

}

int stack\_len(StackNode \*l, StackNode \*r) {

int cnt = 1;

while (r != l) {

cnt++;

r = r->next;

}

return cnt;

}

void quickSort(Stack \*st, StackNode \*low, StackNode \*high)

{

if (low != high)

{

Pivot\_lr pivot\_lr = partition(st, low, high);

if (pivot\_lr.l != NULL) {

int size = stack\_len(pivot\_lr.l, low);

quickSort(st, pivot\_lr.l, low);

}

if (pivot\_lr.r != NULL) {

int size = stack\_len(pivot\_lr.r, high);

quickSort(st, pivot\_lr.r, high);

}

}

}

int main() {

Stack \*st = stack\_create();

stack\_push(st, 7);

stack\_push(st, 0);

stack\_push(st, -4);

stack\_push(st, 3);

stack\_push(st, 1);

stack\_push(st, -2);

stack\_push(st, -5);

StackNode \*buttom\_st = st->head;

while (buttom\_st->next != NULL) {

buttom\_st = buttom\_st->next;

}

quickSort(st, buttom\_st, st->head);

print\_stack(st);

}

**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. **Замечания автора** по существу работы
2. **Выводы: Makefile полезен при сборке больших проектов, т. к. уже собранные файлы не будут компилироваться повторно, также он хорошо подходит для сборки проектов, содержащих несколько различных ЯП.**

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