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 , @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**

2. **Цель работы: Научиться польщоваться make и понять структры данных**

3. **Задание** (*вариант №* **8): 1;3 — Стэк, соединение двух стэков, сортировка Хоасе на стэке**

4. **Оборудование** (студента):
   Процессор *Intel Pentium N4200 1.1 ГГц* с ОП *8* Гб, SSD *128* Гб. Монитор *1920x1080*

5. **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) и тесты либо соображения по тестированию].

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

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. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. **Замечания автора** по существу работы

11. **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

_____