Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра вычислительных методов и программирования

Отчет по лабораторной работе №2
Динамическая структура СТЕК
Вариант 11

Выполнил:
студент 1 курса
группы № 348602
Трошкин Дмитрий Сергеевич

Проверил:
Матюшкин Светослав Иванович

Минск 2023

1 ДИНАМИЧЕСКАЯ СТРУКТУРА СТЕК

Цель работы: изучить алгоритмы работы с динамическими структурами данных в виде стека.

1.1 Условие

Составить алгоритм в виде блок-схемы, написать и отладить поставленную задачу с использованием рекурсивной и обычной функций. Сравнить полученные результаты.

1.2 Исходный код

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

struct Stack {
        double data;
        struct Stack *next;
};

void push(struct Stack **s, double val)
{
        struct Stack *t = malloc(sizeof(struct Stack));
        t->next = *s;
        t->data = val;
        *s = t;
}

double pop(struct Stack **s, double *val)
{
        struct Stack *t;
        if(*s == NULL) return -1;
        *val = (*s)->data;
        t = *s;
        *s = t->next;
        free(t);
        return 0;
}

void sort_p(struct Stack **s)
{
        double tmp;
        struct Stack *t = NULL, *t1, *r;
        push(s, 0);
        if((*s)->next->next == NULL) return;
        do {
                for(t1 = *s; t1->next->next != t; t1 = t1->next)
```

```
                        if(t1->next->data > t1->next->next->data) {
                                r = t1->next->next;
                                t1->next->next = r->next;
                                r->next = t1->next;
                                t1->next = r;
                        }
                t = t1->next;
        } while((*s)->next->next != t);
        pop(s, &tmp);
}

void sort_data(struct Stack *s)
{
        struct Stack *t = NULL, *t1; double r;
        do {
                for(t1 = s; t1->next != t; t1 = t1->next)
                        if(t1->data > t1->next->data) {
                                r = t1->data;
                                t1->data = t1->next->data;
                                t1->next->data = r;
                        }
                t = t1;
        } while(s->next != t);
}

int print_stack(struct Stack *s) {
        int i;
        for(i = 0; s; s = s->next, i++)
                printf("%i: %lf\n", i, s->data);
        puts("\n");
        return i;
}

void del_all(struct Stack **s)
{
        struct Stack *t;
        while(*s != NULL)
        {
                t = *s;
                *s = t->next;
                free(t);
        }
}

double avg_calc(struct Stack *s)
{
        int i; double sum = 0;
        for(i = 0; s; i++, s = s->next)
                sum += s->data;
        return sum / i;
}
```

```
void task(struct Stack *s) {
        if(!s) return;

        s->data = avg_calc(s);
}

int main()
{
        struct Stack *begin = NULL;

        for(int i = 1; i <= 10; i++)
                push(&begin, rand() % 100);

        puts("Исходный стек:\n");
        print_stack(begin);
        sort_data(begin);
        puts("Отсортированный стек:\n");
        print_stack(begin);
        del_all(&begin);

        for(int i = 1; i <= 25; i++)
                push(&begin, rand() % 100);


        puts("Исходный стек:\n");
        print_stack(begin);
        sort_p(&begin);
        puts("Отсортированный стек:\n");
        print_stack(begin);

        task(begin);
        puts("Задание:\n");
        print_stack(begin);

        del_all(&begin);
}
```

1.3 Пример

```
 $ ./task
Исходный стек:

0: 21.000000
1: 49.000000
2: 92.000000
3: 86.000000
4: 35.000000
5: 93.000000
6: 15.000000
7: 77.000000
```

```
8: 86.000000
9: 83.000000


Отсортированный стек:

0: 15.000000
1: 21.000000
2: 35.000000
3: 49.000000
4: 77.000000
5: 83.000000
6: 86.000000
7: 86.000000
8: 92.000000
9: 93.000000


Исходный стек:

0: 69.000000
1: 58.000000
2: 22.000000
3: 2.000000
4: 29.000000
5: 35.000000
6: 67.000000
7: 23.000000
8: 62.000000
9: 30.000000
10: 82.000000
11: 29.000000
12: 67.000000
13: 68.000000
14: 11.000000
15: 36.000000
16: 72.000000
17: 26.000000
18: 40.000000
19: 26.000000
20: 63.000000
21: 59.000000
22: 90.000000
23: 27.000000
24: 62.000000


Отсортированный стек:

0: 2.000000
1: 11.000000
```

2: 22.000000
3: 23.000000
4: 26.000000
5: 26.000000
6: 27.000000
7: 29.000000
8: 29.000000
9: 30.000000
10: 35.000000
11: 36.000000
12: 40.000000
13: 58.000000
14: 59.000000
15: 62.000000
16: 62.000000
17: 63.000000
18: 67.000000
19: 67.000000
20: 68.000000
21: 69.000000
22: 72.000000
23: 82.000000
24: 90.000000

Задание:

0: 46.200000
1: 11.000000
2: 22.000000
3: 23.000000
4: 26.000000
5: 26.000000
6: 27.000000
7: 29.000000
8: 29.000000
9: 30.000000
10: 35.000000
11: 36.000000
12: 40.000000
13: 58.000000
14: 59.000000
15: 62.000000
16: 62.000000
17: 63.000000
18: 67.000000
19: 67.000000
20: 68.000000
21: 69.000000
22: 72.000000
23: 82.000000

```
24: 90.000000
```