Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра вычислительных методов и программирования

Отчет по лабораторной работе №3
Динамическая структура ОЧЕРЕДЬ
Вариант 11

Выполнил:
студент 1 курса
группы № 348602
Трошкин Дмитрий Сергеевич

Проверил:
Матюшкин Светослав Иванович

Минск 2023

1 ДИНАМИЧЕСКАЯ СТРУКТУРА ОЧЕРЕДЬ

Цель работы: изучить алгоритмы работы со списками, организованными в виде очереди.

1.1 Условие

Написать программу, содержащую основные функции обработки двунаправленного списка, информационная часть которого представляет собой целые числа.

1.2 Исходный код

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

typedef struct List {
        double data;
        struct List *next, *prev;
} list;

list *create_list(double data)
{
    list *t = malloc(sizeof(list));
    t->data = data;
    t->prev = t->next = NULL;
    return t;
}

void init(list **b, list **e, double val)
{
    *b = *e = create_list(val);
}


// derection (0 - start, anything - end);
void add(int direction, list **b, list **e, double val)
{
    list *t = malloc(sizeof(list));
    t->data = val;
    if(direction) { // start
        t->prev = NULL;
        t->next = *b;
        (*b)->prev = t;
        *b = t;
    } else { // end
        t->next = NULL;
        t->prev = *e;
```

```
        (*e)->next = t;
        (*e) = t;
    }
}

int del_all(list **b)
{
    int i; list *t;
    for(i = 0; *b; i++)
    {
        t = *b;
        *b = t->next;
        free(t);
    }
    return i;
}


void swap_data(list *p1, list *p2) {
    double tmp;
    if(!p1 || !p2) return;
    tmp = p1->data;
    p1->data = p2->data;
    p2->data = tmp;
}

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

int print_list(int direction, list *b, list *e) {
        int i;
    if(direction) for(i = 0; b; b = b->next, i++)
            printf("%i: %lf\n", i, b->data);
    else for(i = 0; e; e = e->prev, i++)
            printf("%i: %lf\n", i, e->data);

         return i;
}

double avg_calc(list *s)
{
        int i; double sum = 0;
```

```
        for(i = 0; s; i++, s = s->next)
                sum += s->data;
        return sum / i;
}

void task(list *b, list *e) {
        if(!b) return;
        e->data = avg_calc(b);
}

int main()
{
        list *begin = NULL, *end = NULL;

    init(&begin, &end, rand() % 1000);

        for(int i = 0; i < 9; i++)
                add(0, &begin, &end, rand() % 100);

        puts("Исходный стек:\n");
        print_list(0, begin, end);
        sort_data(begin);
        puts("Отсортированный стек:\n");
        print_list(0, begin, end);

        task(begin, end);
        puts("Задание:\n");
        print_list(0, begin, end);

        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: 383.000000
Отсортированный стек:

0: 15.000000
1: 21.000000
```

```
2: 35.000000
3: 49.000000
4: 77.000000
5: 86.000000
6: 86.000000
7: 92.000000
8: 93.000000
9: 383.000000
Задание:

0: 93.700000
1: 21.000000
2: 35.000000
3: 49.000000
4: 77.000000
5: 86.000000
6: 86.000000
7: 92.000000
8: 93.000000
9: 383.000000
```