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

Студент группы М80-103Б-21 Первухин Алексей Сергеевич, № по списку 18

Контакты pervukhin.alexey@mail.ru, telegram @alioxa

Работа выполнена: «29» апреля 2022г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

Отчет сдан « » _____20__ г., итоговая оценка _____

Подпись преподавателя

_____

1. **Тема:** Абстрактные типы данных. Рекурсия. Модульное программирование на языке Си. Автоматизация сборки программ модульной структуры.

2. **Цель работы:** Научиться работать с абстрактным типом данных и собирать программу с помощью утилиты make.

3. **Задание:** Поиск и удаление максимального элемента дека с помощью сортировки линейным выбором.

4. **Оборудование** (студента):
Процессор *Intel Core i5-8265U @ 8x 3.9GH* с ОП *7851* Мб, НМД *1024* Гб. Монитор *1920x1080*

5. **Программное обеспечение (**студента**):**
Операционная система семейства: *linux*, наименование: *ubuntu*, версия *18.10 cosmic* интерпретатор команд: *bash* версия *4.4.19*.
Система программирования --GNU версия **--,** редактор текстов *emacs* версия *25.2.2*

6. **Идея, метод, алгоритм решения задачи**
Изучить принцип работы с абстрактными типами данных, создать дек, применить к нему сортировку линейным выбором, удалить максимальный элемент, оказавшийся в конце дека.

7. **Сценарий выполнения работы**
Написал программу и протестировал ее работу.

8. **Распечатка протокола**
deque.h

```
#ifndef LAB26_DEQUE_H

typedef struct {
    struct DeqElem *prev;
    struct DeqElem *next;
    int value;
    int index;
} DeqElem;

typedef struct {
```

```
    DeqElem *head;
    DeqElem *tail;
    int size;
} DeqStruct;

int RInput(char *str, int op);
int GiveIndex(DeqStruct* deq);
int PushFrontDeq(DeqStruct* deq, int value);
int PopBackDeq(DeqStruct* deq);
int PushBackDeq(DeqStruct* deq, int value);
int PopFrontDeq(DeqStruct* deq);
int PrintDeq(DeqStruct* deq);
DeqStruct* CreateDeq();
void DestroyDequeue(DeqStruct * deq);
void Swap(DeqElem* a, DeqElem* b);
DeqElem* GetElement(DeqStruct* deq, int i);
int SelectionSort(DeqStruct* deq);

#define LAB26_DEQUE_H

#endif //LAB26_DEQUE_H
```

dequeue.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "deque.h"

int RInput(char *str, int op) {
    char *str1 = str;
    bool f = 1;
    while (*str) {
        if (*str < '0' || *str >'9')
            f = 0;
        str++;
    }
    if (f)
        return atoi(str1);
    else {
        if (op == 0) {
            printf("Wrong input, try again\n");
            printf("Enter option: ");
        }
        else {
            printf("Wrong input, try again\n");
            printf("Enter value: ");
        }
        return -1;
    }
}

int GiveIndex(DeqStruct* deq) {
    DeqElem* curElement = deq->head;
    int i = 0;
    while (curElement != NULL) {
        curElement->index = i;
        i++;
        curElement = curElement->next;
    }
    return 0;
}

int PushFrontDeq(DeqStruct* deq, int value) {
    DeqElem* new = malloc(sizeof(DeqElem));
    new->next = deq->head;
    new->prev = NULL;
    new->value = value;
    if (deq->tail == NULL) {
```

```
        deq->head = deq->tail = new;
    } else {
        deq->head->prev = new;
        deq->head = new;
    }
    deq->size++;
    GiveIndex(deq);
    return 0;
}

int PushBackDeq(DeqStruct* deq, int value) {
    DeqElem* new = malloc(sizeof(DeqElem));
    new->next = NULL;
    new->prev = deq->tail;
    new->value = value;
    if (deq->head == NULL) {
        deq->head = deq->tail = new;
    } else {
        deq->tail->next = new;
        deq->tail = new;
    }
    deq->size++;
    GiveIndex(deq);
    return 0;
}

int PopFrontDeq(DeqStruct* deq) {
    DeqElem *temp = deq->head;
    if (deq->head == deq->tail)
        deq->head = deq->tail = NULL;
    else
        deq->head = temp->next;
    deq->size--;
    free(temp);
    GiveIndex(deq);
    return 0;
}

int PopBackDeq(DeqStruct* deq) {
    DeqElem *temp = deq->tail;
    if (deq->head == deq->tail)
        deq->head = deq->tail = NULL;
    else {
        deq->tail = temp->prev;
        deq->tail->next = NULL;
    }
    deq->size--;
    free(temp);
    GiveIndex(deq);
    return 0;
}

int PrintDeq(DeqStruct* deq) {
    DeqElem* cur = deq->head;
    if (cur != NULL) {
        while (cur != NULL) {
            if (cur->next != NULL)
                printf("%d -> ", cur->value);
            else
                printf("%d\n", cur->value);
            cur = cur->next;
        }
    }
    else
        printf("The dequeue is empty\n");
    return 0;
}

DeqStruct* CreateDeq() {
```

```
    DeqStruct *deq = malloc(sizeof(DeqStruct));
    if (deq != NULL) {
        deq->head = deq->tail = NULL;
    }
    deq->size = 0;
    return deq;
}

void DestroyDequeue(DeqStruct* deq) {
    while (deq->head != deq->tail) {
        struct ItemDeque * i = deq->head;
        deq->head = deq->head->next;
        deq->head->prev = 0;
        free(i);
    }
    deq->head = deq->tail = 0;
}

void Swap(DeqElem* a, DeqElem* b) {
    DeqElem *temp = malloc(sizeof(DeqElem));
    temp->value = a->value;
    a->value = b->value;
    b->value = temp->value;
}

DeqElem* GetElement(DeqStruct* deq, int i) {
    DeqElem* cur = deq->head;
    while (cur != NULL) {
        if (cur->index == i) {
            return cur;
        }
        cur = cur->next;
    }
}

int SelectionSort(DeqStruct* deq) {
    for (int step = 0; step < deq->size - 1; step++) {
        int min_idx = step;
        for (int i = step + 1; i < deq->size; i++) {
            if (GetElement(deq, i)->value < GetElement(deq, min_idx)->value)
                min_idx = i;
        }
        Swap(GetElement(deq, min_idx), GetElement(deq, step));
    }
    return 0;
}
```

main.c

```
#include <stdio.h>
#include <stdbool.h>
#include "deque.h"

int main() {
    DeqStruct* deq = CreateDeq();
    int n;
    int op;
    char op1[] = "";
    char n1[] = "";
    printf("-----------------------------------------------\n");
    printf("0 - add new element to the front\n");
    printf("1 - add new element to the back\n");
    printf("2 - delete the first element\n");
    printf("3 - delete the last element\n");
    printf("4 - print the dequeue\n");
    printf("5 - sort the dequeue and delete the maximum\n");
    printf("6 - destroy the dequeue\n");
    printf("7 - end the program\n");
```

```
    printf("-------------------------------------------------\n");
    while (true) {
        printf("Enter option: ");
        do {
            scanf("%s", op1);
            op = RInput(op1, 0);
        } while (op == -1);
        if (op == 0) {
            printf("Enter value: ");
            do {
                scanf("%s", n1);
                n = RInput(n1, 1);
            } while (n == -1);
            PushFrontDeq(deq, n);
        }
        else if (op == 1) {
            printf("Enter value: ");
            do {
                scanf("%s", n1);
                n = RInput(n1, 1);
            } while (n == -1);
            PushBackDeq(deq, n);
        }
        else if (op == 2){
            if (deq->head != NULL) {
                printf("Deleting first element\n");
                PopFrontDeq(deq);
            }
            else
                printf("The dequeue is empty");
        }
        else if (op == 3){
            if (deq->head != NULL) {
                printf("Deleting last element\n");
                PopBackDeq(deq);
            }
            else
                printf("The dequeue is empty\n");
        }
        else if (op == 4) {
            PrintDeq(deq);
        }
        else if (op == 5) {
            if (deq->head != NULL) {
                SelectionSort(deq);
                PopBackDeq(deq);
            }
            else
                printf("Nothing to sort\n");
        }
        else if (op == 6) {
            DestroyDequeue(deq);
        }
        else if (op == 7) {
            break;
        }
    }
    return 0;
}
```

makefile

```
CC = gcc
LD = gcc
CCFLAGS = -std=c99
LDFLAGS =
OBJ = main.o dequeue.o
```

```
HDR = deque.h
all: $(OBJ)
    @$(LD) $(LDFLAGS) -o main $(OBJ)
main.o: main.c $(HDR)
    @$(CC) $(CCFLAGS) -c main.c
dequeue.o: dequeue.c $(HDR)
    @$(CC) $(CCFLAGS) -c dequeue.c
clean:
    @rm -f *.o main
```

9. Дневник отладки

10. Замечания автора по существу работы

11. Выводы

В результате работы у меня получилось выполнить поставленную задачу. Работа была не слишком сложной и интересной, мне понравилось работать с динамическими структурами данных.

Подпись студента

_____