Отчет по лабораторной работе №25-26 по курсу практикум на ЭВМ

Студент группы М8О-107Б-22 Брюханов Захар Дмитриевич, № по списку 5

Контакты e-mail: br_zahar@mail.ru; telegram: @br_zahar

Работа выполнена: «30» апреля 2023 г.

Преподаватель: Аносов Наталья Павловна

Входной контроль знаний с оценкой _____

Отчет сдан « » _____202 __ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Автоматизация программ модульной сборки на языке С с использованием утилиты make. Абстрактные типы данных. Рекурсия. Модульное программирование на языке С.

2. **Цель работы:** Изучить утилиту make, абстрактные типы данных, модульное программирование и рекурсию.

3. **Задание** (*вариант № 5*): АТД: Дек. Процедура: поиск и удаление максимального (для стека, дека, списка) зли минимального (для очереди) элемента. Метод: сортировка линейным выбором

4. **Оборудование** (лабораторное):
ЭВМ Intel Pentium G2140, процессор 3.30 GHz, имя узла сети Cameron с ОП 8096 Мб, НМД 7906 Мб. Терминал ASUS адрес dev/pets/3 Принтер HP Laserjet 6P
Другие устройства _____

Оборудование ПЭВМ студента, если использовалось:
Процессор M1 Pro с 10-ядерным процессором и 14-ядерным графическим процессором с ОП 16 Гб, НМД 512 Гб. Дисплей Liquid Retina XDR
Другие устройства _____

5. **Программное обеспечение (лабораторное):**
Операционная система семейства Unix, наименование Ubuntu версия 18.15.0
интерпретатор команд bash версия 4.4.20
Система программирования GNU версия 5.8.13
Редактор текстов emacs версия 25.2.2
Утилиты операционной системы cat
Прикладные системы и программы
Местонахождение и имена файлов программ и данных stud/208104

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система Mac OS версия 13.2.1
интерпретатор команд bash версия 5.0.17
Система программирования Clion версия 2022.3.3
Редактор текстов emacs версия 25.2.2
Утилиты операционной системы cat
Прикладные системы и программы
Местонахождение и имена файлов программ и данных на домашнем компьютере /Users/br_zahar

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

25: Проанализируем файловую структуру модуля. На ее основе создадим Makefile с зависимостями программных файлов модуля.

26: Отдельно реализуем модуль дека на языке Си заголовочным файлом (deq.h) и реализуем методы модуля (deq.c). Модуль имеет операции добавления в конец, добавления в начало, удаления из начала, удаления из конца, печати очереди, получения элемента начала, конца.

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

Методы модуля:

```
udt *create_udt(udt *d) {
    d = (udt *) malloc(sizeof(udt));
    d->first = d->last = -1;
    d->size = 0;
    return d;
}

bool udt_empty(const udt *d) {
    return d->size == 0;
}

int udt_size(const udt *d) {
    return d->size;
}

bool udt_push_back(udt *d, data t) {
    if (d->size == 10) {
        return false;
    }
    if (!d->size) {
        d->last = d->first = 0;
    } else {
        d->last = (d->last + 1) % 10;
    }
    d->arr[d->last] = t;
    d->size++;
    return true;
}

bool udt_push_front(udt *d, data t) {
    if (d->size == 10) {
        return false;
    }
    if (!d->size) {
        d->first = d->last = 0;
    } else {
        d->first = (10 + (d->first - 1) % 10) % 10;
    }
    d->arr[d->first] = t;
    d->size++;
    return true;
}

bool udt_pop_front(udt *d) {
    if (!d->size) {
        return false;
    }
    if (d->size == 1) {
        d->first = d->last = -1;
        d->size--;
        return true;
    }
    d->first++;
    d->first %= 10;
    d->size--;
    return true;
```

```
}

bool udt_pop_back(udt *d) {
    if (!d->size) {
        return false;
    }
    if (d->size == 1) {
        d->first = d->last = -1;
        d->size--;
        return true;
    }
    d->last--;
    d->last = (10 + d->last % 10) % 10;
    d->size--;
    return true;
}

data udt_top_left(udt *d) {
    if (d->size) {
        return d->arr[d->first];
    }
}

data udt_top_right(udt *d) {
    if (d->size) {
        return d->arr[d->last];
    }
}

void udt_print(udt *d) {
    printf("Key\tValue\n");
    int size = udt_size(d);
    for (int i = 0; i < size; i++) {
        data a = udt_top_left(d);
        udt_pop_front(d);
        printf("%d\t", a.key);
        printf("%s\n", a.value);
        udt_push_back(d, a);
    }
}
```

Тесты:

Протестируем наихудший для сортировки случай: когда входные данные отсортированы в обратном порядке.

Ключ	**Строка**
5	5
4	4
3	3
2	2
1	1

Пункты 1-7 отчета составляются строго до начала лабораторной работы.

Допущен к выполнению работы. **Подпись преподавателя** _____

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

```
br_zahar@mbp-zahar laba26 % cat Makefile
laba26: deq.o main.o
    gcc deq.o main.o
deque.o : deq.h deq.c
    gcc -c deq.c
main.o : deq.h main.c
    gcc -c main.c%
br_zahar@mbp-zahar laba26 % cat deq.c
#include <stdio.h>
#include <stdlib.h>
#include "deq.h"

udt *create_udt(udt *d) {
    d = (udt *) malloc(sizeof(udt));
    d->first = d->last = -1;
    d->size = 0;
    return d;
}

bool udt_empty(const udt *d) {
    return d->size == 0;
}

int udt_size(const udt *d) {
    return d->size;
}

bool udt_push_back(udt *d, data t) {
    if (d->size == 10) {
        return false;
    }
    if (!d->size) {
        d->last = d->first = 0;
    } else {
        d->last = (d->last + 1) % 10;
    }
    d->arr[d->last] = t;
    d->size++;
    return true;
}

bool udt_push_front(udt *d, data t) {
    if (d->size == 10) {
        return false;
    }
    if (!d->size) {
        d->first = d->last = 0;
    } else {
        d->first = (10 + (d->first - 1) % 10) % 10;
    }
    d->arr[d->first] = t;
    d->size++;
    return true;
}

bool udt_pop_front(udt *d) {
```

```
    if (!d->size) {
        return false;
    }
    if (d->size == 1) {
        d->first = d->last = -1;
        d->size--;
        return true;
    }
    d->first++;
    d->first %= 10;
    d->size--;
    return true;
}

bool udt_pop_back(udt *d) {
    if (!d->size) {
        return false;
    }
    if (d->size == 1) {
        d->first = d->last = -1;
        d->size--;
        return true;
    }
    d->last--;
    d->last = (10 + d->last % 10) % 10;
    d->size--;
    return true;
}

data udt_top_left(udt *d) {
    if (d->size) {
        return d->arr[d->first];
    }
}

data udt_top_right(udt *d) {
    if (d->size) {
        return d->arr[d->last];
    }
}

void udt_print(udt *d) {
    printf("Key\tValue\n");
    int size = udt_size(d);
    for (int i = 0; i < size; i++) {
        data a = udt_top_left(d);
        udt_pop_front(d);
        printf("%d\t", a.key);
        printf("%s\n", a.value);
        udt_push_back(d, a);
    }
}%
br_zahar@mbp-zahar laba26 % cat deq.h
#ifndef _UDT_H_
#define _UDT_H_

#include <stdbool.h>

typedef struct {
```

```
    int key;
    char value[40];
} data;

typedef struct {
    int first;
    int last;
    int size;
    data arr[10];
} udt;

udt* create_udt(udt *);

bool udt_empty(const udt *);

bool udt_push_front(udt *, data);

bool udt_push_back(udt *, data);

bool udt_pop_front(udt *);

bool udt_pop_back(udt *);

data udt_top_left(udt *);

data udt_top_right(udt *);

void udt_print(udt *);

int udt_size(const udt *);

#endif%
br_zahar@mbp-zahar laba26 % cat main.c
#include <stdio.h>
#include <string.h>
#include "deq.h"
#include <limits.h>

#define INF INT_MAX

data procedure(udt *d) {
    data t, a;
    int max = -INF - 1;
    int size = d->size;
    for (int i = 0; i < size; i++) {
        a = udt_top_left(d);
        if (a.key > max) {
            max = udt_top_left(d).key;
            t = udt_top_left(d);
        }
        udt_pop_front(d);
        udt_push_back(d, a);
    }
    udt *d1 = create_udt(NULL);
    while (udt_top_left(d).key != t.key && udt_top_right(d).key != t.key)
{
        a = udt_top_left(d);
        udt_push_front(d1, a);
        udt_pop_front(d);
```

```
        if (!udt_empty(d)) {
            a = udt_top_right(d);
            udt_push_back(d1, a);
            udt_pop_back(d);
        }
    }
    if (udt_top_left(d).key == t.key) {
        udt_pop_front(d);
    } else {
        udt_pop_back(d);
    }
    while (!udt_empty(d1)){
        udt_push_front(d, udt_top_left(d1));
        udt_pop_front(d1);
    }
    return t;
}

udt* sort(udt *d) {
    int size = udt_size(d);
    udt *d1 = create_udt(NULL);
    data t;
    for (int i = 0; i < size; i++) {
        t = procedure(d);
        udt_push_front(d1, t);
    }
    return d1;
}

int main() {
    int c = 1, ans;
    udt *d = NULL;
    while (c) {
        printf("1. Create deq\t 2. Empty\t 3. Size\t 4. Push back\t 5.
Push front\t 6. Top left\t 7. Top right\t 8.Pop back\t 9. Pop front\t
10.Print\t 11. Sort\t 12. Exit\n");
        scanf("%d", &ans);
        switch (ans) {
            case 1: {
                d = create_udt(d);
                break;
            }
            case 2: {
                if (d == NULL) {
                    printf("Deq doesn't exist\n");
                } else {
                    udt_empty(d) ? printf("Deq is empty\n") : printf("Deq
isn't empty\n");
                }
                break;
            }
            case 3: {
                if (d == NULL) {
                    printf("Deq doesn't exist\n");
                } else {
                    printf("%d\n", udt_size(d));
                }
                break;
            }
```

```
case 4: {
    if (d == NULL) {
        printf("Deq doesn't exist\n");
    } else {
        data tb;
        printf("Print key\n");
        scanf("%d", &tb.key);
        printf("Print string\n");
        scanf("%s", tb.value);
        if (!udt_push_back(d, tb)) {
            printf("Deq is full\n");
        }
    }
    break;
}
case 5: {
    if (d == NULL) {
        printf("Deq doesn't exist\n");
    } else {
        data tf;
        printf("Print key\n");
        scanf("%d", &tf.key);
        printf("Print string\n");
        scanf("%s", tf.value);
        if (!udt_push_front(d, tf)) {
            printf("Deq is full\n");
        }
    }
    break;
}
case 6: {
    if (d == NULL) {
        printf("Deq doesn't exist\n");
    } else {
        if (udt_empty(d)) {
            printf("Deq is empty\n");
        } else {
            data a = udt_top_left(d);
            printf("Key\n%d\nValue\n%s\n", a.key, a.value);
        }
    }
    break;
}
case 7: {
    if (d == NULL) {
        printf("Deq doesn't exist\n");
    } else {
        if (udt_empty(d)) {
            printf("Deq is empty\n");
        } else {
            data a = udt_top_right(d);
            printf("Key\n%d\nValue\n%s\n", a.key, a.value);
        }
    }
    break;
}
case 8: {
    if (d == NULL) {
        printf("Deq doesn't exist\n");
```

```
                } else {
                    if (!udt_pop_back(d)) {
                        printf("Deq is empty\n");
                    }
                }
                break;
            }
            case 9: {
                if (d == NULL) {
                    printf("Deq doesn't exist\n");
                } else {
                    if (!udt_pop_front(d)) {
                        printf("Deq is empty\n");
                    }
                }
                break;
            }
            case 10: {
                if (d == NULL) {
                    printf("Deq doesn't exist\n");
                } else {
                    udt_print(d);
                }
                break;
            }
            case 11: {
                if (d == NULL) {
                    printf("Deq doesn't exist\n");
                } else {
                    d = sort(d);
                }
                break;
            }
            case 12: {
                c = 0;
                break;
            }
            default: {
                printf("Wrong answer\n");
            }
        }
    }
    return 0;
}%
br_zahar@mbp-zahar laba26 % make
cc    -c -o deq.o deq.c
gcc -c main.c
gcc deq.o main.o
br_zahar@mbp-zahar laba26 % ./a.out
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
1
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
2
Deq is empty
```

1. Create deq 2. Empty 3. Size 4. Push back 5. Push front 6. Top left 7. Top right 8.Pop back 9. Pop front 10.Print 11. Sort 12. Exit
3
0
1. Create deq 2. Empty 3. Size 4. Push back 5. Push front 6. Top left 7. Top right 8.Pop back 9. Pop front 10.Print 11. Sort 12. Exit
6
Deq is empty
1. Create deq 2. Empty 3. Size 4. Push back 5. Push front 6. Top left 7. Top right 8.Pop back 9. Pop front 10.Print 11. Sort 12. Exit
7
Deq is empty
1. Create deq 2. Empty 3. Size 4. Push back 5. Push front 6. Top left 7. Top right 8.Pop back 9. Pop front 10.Print 11. Sort 12. Exit
4
Print key
5
Print string
5
1. Create deq 2. Empty 3. Size 4. Push back 5. Push front 6. Top left 7. Top right 8.Pop back 9. Pop front 10.Print 11. Sort 12. Exit
4
Print key
4
Print string
4
1. Create deq 2. Empty 3. Size 4. Push back 5. Push front 6. Top left 7. Top right 8.Pop back 9. Pop front 10.Print 11. Sort 12. Exit
4
Print key
3
Print string
3
1. Create deq 2. Empty 3. Size 4. Push back 5. Push front 6. Top left 7. Top right 8.Pop back 9. Pop front 10.Print 11. Sort 12. Exit
4
Print key
2
Print string
2
1. Create deq 2. Empty 3. Size 4. Push back 5. Push front 6. Top left 7. Top right 8.Pop back 9. Pop front 10.Print 11. Sort 12. Exit
4
Print key
1
Print string
1
1. Create deq 2. Empty 3. Size 4. Push back 5. Push front 6. Top left 7. Top right 8.Pop back 9. Pop front 10.Print 11. Sort 12. Exit
10

```
Key Value
5   5
4   4
3   3
2   2
1   1
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
11
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
10
Key Value
1   1
2   2
3   3
4   4
5   5
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
2
Deq isn't empty
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
3
5
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
6
Key
1
Value
1
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
7
Key
5
Value
5
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
8
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
9
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
10
Key Value
```

```
2   2
3   3
4   4
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
5
Print key
1
Print string
1
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
4
Print key
5
Print string
5
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
10
Key  Value
1    1
2    2
3    3
4    4
5    5
1. Create deq     2. Empty     3. Size  4. Push back 5. Push front 6. Top
left 7. Top right     8.Pop back   9. Pop front     10.Print 11. Sort 12.
Exit
12
br_zahar@mbp-zahar laba26 % make
gcc deq.o main.o
br_zahar@mbp-zahar laba26 % touch deq.h
br_zahar@mbp-zahar laba26 % ls -l
total 136
-rw-r--r--@ 1 br_zahar  staff    111 29 апр 21:29 Makefile
-rwxr-xr-x  1 br_zahar  staff  34226 30 апр 13:23 a.out
-rw-r--r--@ 1 br_zahar  staff   1804 29 апр 21:32 deq.c
-rw-r--r--@ 1 br_zahar  staff    492 30 апр 13:24 deq.h
-rw-r--r--  1 br_zahar  staff   3248 30 апр 13:19 deq.o
-rw-r--r--@ 1 br_zahar  staff   5259 29 апр 21:03 main.c
-rw-r--r--  1 br_zahar  staff   5472 30 апр 13:19 main.o
br_zahar@mbp-zahar laba26 % make
gcc -c main.c
gcc deq.o main.o
br_zahar@mbp-zahar laba26 % touch main.c
br_zahar@mbp-zahar laba26 % ls -l
total 136
-rw-r--r--@ 1 br_zahar  staff    111 29 апр 21:29 Makefile
-rwxr-xr-x  1 br_zahar  staff  34226 30 апр 13:24 a.out
-rw-r--r--@ 1 br_zahar  staff   1804 29 апр 21:32 deq.c
-rw-r--r--@ 1 br_zahar  staff    492 30 апр 13:24 deq.h
-rw-r--r--  1 br_zahar  staff   3248 30 апр 13:19 deq.o
-rw-r--r--@ 1 br_zahar  staff   5259 30 апр 13:24 main.c
-rw-r--r--  1 br_zahar  staff   5472 30 апр 13:24 main.o
br_zahar@mbp-zahar laba26 % make
```

```
gcc -c main.c
gcc deq.o main.o
```

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.	Дат	Врем	Событие	Действие по	Примечание

10. **Замечания автора** по существу работы:_____

11. **Выводы:** Я изучил принцип работы утилиты make, а также абстрактный тип данных, рекурсию и модульное программирование._____

Недочёты при выполнении задания могут быть устранены следующим образом:

_____

_____

_____

Подпись студента _____