Отчет по лабораторной работе № 25-26 по курсу “Языки и методы программирования”

Студент группы М80-103Б-21 Тысячный Владислав Валерьевич, № 21 по списку

Контакты e-mail: tysycny2003@gmail.com,
telegram: @Bradvurt

Работа выполнена: «7» апреля 2021г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

Отчет сдан « » _____20__ г., итоговая оценка _____

Подпись преподавателя _____

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

2. **Цель работы:** Изучить работу с утилитой make. Создать и отладить модуль с реализацией заданного абстрактного типа данных.

3. **Задание №2;4:** Тип данных очередь, сортировка пузырьком.

4. **Оборудование** :

   Процессор *IIntel® Core™ i7-7700HQ CPU @ 2.80GHz × 8* с ОП *7,7* Гб, НМД *1024* Гб. Монитор *1920x1080*

5. **Программное обеспечение:**

   Операционная система семейства: *linux*, наименование: *ubuntu*, версия *20.04.3 LTS*
   интерпретатор команд: *bash* версия *4.4.19*.
   Система программирования -- версия --, редактор текстов *emacs* версия *25.2.2*
   Утилиты операционной системы --
   Прикладные системы и программы --
   Местонахождение и имена файлов программ и данных на домашнем компьютере --

6. Идея, метод, алгоритм

Для выполнения требуемых операций над очередью используются отдельно написанные функции и процедуры:

- init — инициализация очереди
- delite — удаление очереди
- print — вывод очереди
- is_empty — проверка на пустоту
- insert— добавление элемента
- len — вывод длины очереди
- removal — удаление крайнего элемента
- copy — копирование очереди
- replace — перенос наибольшего элемента вправо
- sort — сортировка пузырьком

7. Сценарий выполнения работы

What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue
5: Exit
1
Adding a new element, enter its value:
4
What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue

5: Exit
1
Adding a new element, enter its value:
3
What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue
5: Exit
1
Adding a new element, enter its value:
2
What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue
5: Exit
1
Adding a new element, enter its value:
5
What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue
5: Exit
2

4 3 2 5

What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue
5: Exit
3
What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue
5: Exit
2

3 2 5

What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue
5: Exit
4
What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue
5: Exit
2

2 3 5

What you want to do?
1: Add new element
2: Print queue
3: Delete element
4: Sort the queue
5: Exit

5
End of program.

8. Распечатка протокола

main.c

```
#include <stdio.h>
#include "stdlib.h"
#include "stdbool.h"
#include "string.h"
#include "queue.h"

bool is_int(const char*str) {
    while(*str)  {
        if((*str< '0' || *str > '9') && *str != '-' && *str != '.')
            return false;
        *str++;
    }
    return true;
}


int main() {
    bool program_works = true;
    queue *q;
    q = init();
    while(program_works) {
        char menu[] = "0";
        while (!strcmp("0", menu)) {
            printf("What you want to do?\n1: Add new element\n2: Print queue\n3: Delete element\n4: Sort the queue\n5: Exit\
n");
            scanf("%s", menu);
            if (!strcmp("1", menu)) {
                int input;
                char c[] = "";
                bool value_is_incorrect = true;
                while (value_is_incorrect) {
                    printf("Adding a new element, enter its value:\n");
                    scanf("%s", c);
                    if (is_int(c)) {
                        input = atoi(c);
                        value_is_incorrect = false;
                    } else {
                        printf("\nError, is not number!\n\n");
                    }
                }
                insert(q, input);
            } else if (!strcmp("2", menu)) {
                printf("\n");
                print(q);
                printf("\n");
            } else if (!strcmp("3", menu)) {
                removal(q);
            } else if (!strcmp("4", menu)) {
                q = sort(q);
            } else if (!strcmp("5", menu)) {
                printf("End of program.\n");
                program_works = false;
            } else {
                printf("\nError, incorrect input!\n\n");
            }
        }
    }
    delite(q);
    return 0;
}
```

queue.c

```
#include "queue.h"

queue *init() {
    queue *q = malloc(sizeof(queue));
    q->front = NULL;
    q->rear = NULL;
    return q;
}

bool is_empty(queue *q) {
    return (q->front == NULL) ? true : false;;
}

int len(queue *q) {
    int k = 0;
    for (item *i = q->front; i != NULL; i = i->next) {
        k += 1;
    }
    return k;
}

void insert(queue *q, int t) {
    item *i = (item *)malloc(sizeof(item));
    i->data = t;
    i->next = NULL;
    if (is_empty(q)) {
        q->front = i;
        q->rear = i;
    } else {
        q->rear->next = i;
        q->rear = i;
    }
}

void removal(queue *q) {
    if (is_empty(q)) {
        printf("\nThe queue is empty!\n\n");
        return;
    }
    item *i = q->front;
    q->front = q->front->next;
    free(i);
    return;
}

void print(queue *q) {
    item *i = q->front;
    if(is_empty(q)) {
        printf("The queue is empty!\n");
    }
    else {
        while(i != NULL) {
            printf("%d ", i->data);
            i = i->next;
        }
        printf("\n");
    }
}

void delite(queue * q) {
    while (!is_empty(q)) {
        removal(q);
    }
    q->front = q->rear = NULL;
}
```

```
queue *copy(queue *q) {
    queue *q1;
    q1 = init();
    for (item *i = q->front; i != NULL; i = i->next) {
        insert(q1, i->data);
    }
    return q1;
}

void replace(queue *q) {
    queue *q1;
    int j;
    q1 = copy(q);
    delite(q);
    bool flag = false;
    item *i = q1->front;
    for (int _ = 0; _ < len(q1) - 1; _++) {
        if (!flag && i->data > i->next->data) {
            insert(q, i->next->data);
            j = i->data;
            flag = 1;
        }
        else if (flag) {
            if (j > i->next->data) {
                insert(q, i->next->data);
            }
            else {
                insert(q, j);
                flag = 0;
            }
        }
        else {
            insert(q, i->data);
        }
        i = i->next;
    }
    if (flag) {
        insert(q, j);
    }
    else {
        insert(q, i->data);
    }
}

queue *sort(queue *q) {
    for (int i = 0; i < len(q) - 1; i++) {
        replace(q);
    }
    return q;
}
```

queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

#include "stdlib.h"
#include "stdbool.h"
#include "stdio.h"

typedef struct item {
    int data;
    struct item *next;
} item;

typedef struct queue {
    struct item *front;
```

```
    struct item *rear;
} queue;

queue *init(void);

bool is_empty(queue *q);

int len(queue *q);

void insert(queue *q, int t);

void removal(queue *q);

void print(queue *q);

void delite(queue *q);

void replace(queue *q);

queue *sort(queue *q);

#endif
```

9. Дневник отладки

№	Лаб или дом	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечания автора

11. Выводы

Результатом лабораторной работы стала программа и файл для её сборки. В процессе выполнения задания были изучены основы работы с утилитой make. Реализация абстрактного типа данных очередь оказалось не самой лёгкой задачей, но в итоге получилось добиться выполнения всего функционала. Алгоритм сортировки был одним из самых простых, так что с ним проблем особых не возникло.

Подпись студента _____