Отчет по лабораторной работе №25-26

по курсу: языки и методы программирования

студент группы : М8О-105Б-21 Козлов Егор Сергеевич, № по списку: 9

Адреса www, e-mail, jabber, skype: iamaghoulzxc@gmail.com

Работа выполнена: "18 мая 2022 г."

Преподаватель: каф. 806 В.К.Титов

Входной контроль знаний с оценкой: _____

Отчет сдан "___"_____20___г., итоговая оценка _____

Подпись преподавателя _____

1. Тема: Абстрактные типы данных. Рекурсия. Модульное программирование на языке Си.

2. Цель работы: Автоматизация сборки программ модульной структуры на языке Си. Процедура: слияние.

3. Задание (*вариант 9*): АТД: очередь; Метод: сортировка слиянием; Процедура: слияние.

4. Оборудование(*лабораторное*):
ЭВМ _____, процессор _____, имя узла сети ____ с ОП ___ ГБ
НМД _____ ГБ. Терминал _____ адрес _____. Принтер _____
Другие устройства _____
Оборудование ПЭВМ студента, если использовалось:
Процессор Ryzen 7 5800 @ 8x 3.2 GHz , ОП 16384 МБ, НМД _____ГБ. Монитор Встроенный
Другие устройства _____

5. Программное обеспечение(*лабораторное*):
Операционная система семейства UNIX, наименование _____ версия _____
Интерпретатор команд: _____ версия _____
Система программирования: _____ версия _____
Редактор текстов: _____ версия _____
Утилиты операционной системы: _____
Прикладные системы и программы: _____
Местонахождение и имена файлов и программ данных: _____

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства UNIX, наименование Ubuntu версия 22.04
Интерпретатор команд: bash версия _____
Система программирования: C версия _____
Редактор текстов: Emacs версия _____
Утилиты операционной системы: _____
Прикладные системы и программы: _____
Местонахождение и имена файлов и программ данных: /usr/bin , а также /bin

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

Метод:

- объект сортировки разбивается на подобъекты, которые сортируются отдельно. После отсортированные подобъекты снова собираются в один объект.
- если в рассматриваемом объекте один элемент, то он уже отсортирован. Алгоритм завершён. В ином случае объект разбивается на две части, которые сортируются рекурсивно.
- по окончании сортировки двух частей объекта к ним применяется процедура слияния, которая по отсортированным частям получает исходный отсортированный объект.

Процедура:

- выбираем первый элемент первого массива и первый элемент второго массива;
- меньший из них удаляем из соответствующего массива и присоединяем к имеющимся элементам результирующего массива.

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

25 – 26.*cpp* :

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "queueDynamic.h"
//include "queueMassive.h"

queue mergeSort(queue &q) {
    if (empty(q) || (size(q) == 1)) {
        return q;
    }
    queue tmp1, tmp2;
    init(tmp1);
    init(tmp2);
    int dim = size(q), split = 1;
    while (!empty(q)) {
        if (split <= dim / 2) {
            push(tmp1, pop(q));
            split++;
        } else push(tmp2, pop(q));
    }
    queue q1 = mergeSort(tmp1);
    queue q2 = mergeSort(tmp2);
    return merge(q1, q2);
}

int main() {
    srand(time(0));
    queue q;
    init(q);
    int in;
    while (1) {
        printf("Menu:\n"
               "1. Generate random queue\n"
               "2. Print queue\n"
               "3. Push value\n"
               "4. Pop value\n"
               "5. Sort queue\n"
               "0. Exit\n"
               "=> ");
        scanf("%d", &in);
        switch (in) {
            case 0: {
                return 0;
            }
            case 1: {
                printf("Please enter a size of queue:");
```

```
                scanf("%d", &in);
                while (in) {
                    push(q, rand() % 100);
                    in--;
                }
                break;
            }
            case 2: {
                display(q);
                break;
            }
            case 3: {
                printf("Please enter a value:");
                scanf("%d", &in);
                push(q, in);
                break;
            }
            case 4: {
                printf("%d\n", pop(q));
                break;
            }
            case 5: {
                q = mergeSort(q);
                break;
            }
        }
    }
}
```

queueDynamic.h :

```
#include <stdio.h>
#ifndef LAB_QUEUEDYNAMIC_H
#define LAB_QUEUEDYNAMIC_H
#define N 100
typedef int typeValue;

struct element {
    element *next;
    typeValue value;
};

struct queue {
    element *start;
    element *end;
    int size;
};

void init(queue &q) {
    q.start = 0;
    q.end = 0;
    q.size = 0;
}

int empty(queue &q) {
    return q.start == 0;
}

void push(queue &q, typeValue value) {
    if (q.size == N) {
        printf("Queue is overflow!\n");
        return;
    } else if (empty(q)) {
        q.end = new element;
        q.end->value = value;
        q.start = q.end;
    } else {
        element *tmp = new element;
        q.end->next = tmp;
        tmp->value = value;
```

```
        q.end = tmp;
        q.end->next = 0;
    }
    q.size++;
}

typeValue pop(queue &q) {
    if (!empty(q)) {
        typeValue value = q.start->value;
        q.start = q.start->next;
        q.size--;
        return value;
    } else printf("Queue is empty!\n");
    return 0;
}

typeValue top(queue &q) {
    if (!empty(q)) return q.start->value;
    else printf("Queue is empty!\n");
    return 0;
}

int size(queue &q) {
    return q.size;
}

void display(queue &q) {
    if (!empty(q)) {
        element *tmp = q.start;
        printf("[");
        while (tmp) {
            printf("%d ", tmp->value);
            tmp = tmp->next;
        }
        printf("]\n");
    } else printf("Queue is empty!\n");
}

queue merge(queue &q1, queue &q2) {
    queue q;
    init(q);
    while (!(empty(q1) || empty(q2))) {
        if (top(q1) < top(q2)) push(q, pop(q1));
        else push(q, pop(q2));
    }
    while (!empty(q1)) push(q, pop(q1));
    while (!empty(q2)) push(q, pop(q2));
    return q;
}

#endif //LAB_QUEUEDYNAMIC_H
```

queueMassive.h :

```
#include <stdio.h>

#ifndef LAB_QUEUEMASSIVE_H
#define LAB_QUEUEMASSIVE_H
#define N 100
typedef int typeValue;

struct queue {
    int start, end, size;
    typeValue value[N];
```

```
};

void init(queue &q) {
    q.start = q.size = 0;
    q.end = -1;
}

int empty(queue &q) {
    return q.size == 0;
}

void push(queue &q, typeValue v) {
    if (q.size == N) printf("Queue is overflow!\n");
    else {
        q.value[++q.end %= N] = v;
        q.size++;
    }
}

typeValue pop(queue &q) {
    if (!empty(q)) {
        q.size--;
        int i = q.start++;
        q.start %= N;
        return q.value[i];
    } else printf("Queue is empty!\n");
    return 0;
}

typeValue top(queue &q) {
    if (!empty(q)) return q.value[q.start];
    else printf("Queue is empty!\n");
    return 0;
}

int size(queue &q) {
    return q.size;
}

void display(queue q) {
    printf("[");
    for (int i = q.start; i < q.start + q.size; i++)
        printf("%d ", q.value[i % N]);
    printf("]\n");
}

queue merge(queue &q1, queue &q2) {
    queue q;
    init(q);
    while (!(empty(q1) || empty(q2))) {
        if (top(q1) < top(q2)) push(q, pop(q1));
        else push(q, pop(q2));
    }
    while (!empty(q1)) push(q, pop(q1));
    while (!empty(q2)) push(q, pop(q2));
    return q;
}

#endif //LAB_QUEUEMASSIVE_H
```

makefile:

```
CC = g++
CFLAGS = -c -Wall
LDFLAGS =
```

```
SOURCES = 25-26.cpp queueDynamic.h
OBJECTS = $(SOURCES:.cpp=.o)
EXECUTABLE = queue.exe

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
  $(CC) $(LDFLAGS) $(OBJECTS) -o $@

.cpp.o:
  $(CC) $(CFLAGS) $< -o $@
```

Пункты 1-7 отчѐта составляются ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя _____

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

```
isitmuse@isitmuse:~/lab/secondSem/25-26$ cat head
--------------------------------
||ЛАБОРАТОРАНЯ  РАБОТА  №25-26||
||||АБСТРАКТНЫЕ  ТИПЫ ДАННЫХ||||
|||||||РЕКУРСИЯ. МОДУЛЬНОЕ||||||
|||||ПРОГРАММИРОВАНИЕ НА СИ|||||
|||ВЫПОЛНИЛ  СТУДЕНТ  ГРУППЫ||||
|||М8О-105Б-21  КОЗЛОВ  ЕГОР||||
\\\\\\\\\\\\\\\\||///////////////
isitmuse@isitmuse:~/lab/secondSem/25-26$ cat 25-26.cpp
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
//#include "queueDynamic.h"
#include "queueMassive.h"

queue mergeSort(queue &q) {
    if (empty(q) || (size(q) == 1)) {
        return q;
    }
    queue tmp1, tmp2;
    init(tmp1);
    init(tmp2);
    int dim = size(q), split = 1;
    while (!empty(q)) {
        if (split <= dim / 2) {
            push(tmp1, pop(q));
            split++;
        } else push(tmp2, pop(q));
    }
    queue q1 = mergeSort(tmp1);
    queue q2 = mergeSort(tmp2);
    return merge(q1, q2);
}

int main() {
    srand(time(0));
    queue q;
    init(q);
    int in;
    while (1) {
        printf("Menu:\n"
               "1. Generate random queue\n"
               "2. Print queue\n"
               "3. Push value\n"
               "4. Pop value\n"
               "5. Sort queue\n"
               "0. Exit\n"
               "=> ");
        scanf("%d", &in);
        switch (in) {
            case 0: {
                return 0;
            }
            case 1: {
                printf("Please enter a size of queue:");
                scanf("%d", &in);
                while (in) {
                    push(q, rand() % 100);
                    in--;
```

```
                }
                break;
            }
            case 2: {
                display(q);
                break;
            }
            case 3: {
                printf("Please enter a value:");
                scanf("%d", &in);
                push(q, in);
                break;
            }
            case 4: {
                printf("%d\n", pop(q));
                break;
            }
            case 5: {
                q = mergeSort(q);
                break;
            }
        }
    }
}isitmuse@isitmuse:~/lab/secondSem/25-26$ cat queueMassive.h
#include <stdio.h>

#ifndef LAB_QUEUEMASSIVE_H
#define LAB_QUEUEMASSIVE_H
#define N 100
typedef int typeValue;

struct queue {
    int start, end, size;
    typeValue value[N];
};

void init(queue &q) {
    q.start = q.size = 0;
    q.end = -1;
}

int empty(queue &q) {
    return q.size == 0;
}

void push(queue &q, typeValue v) {
    if (q.size == N) printf("Queue is overflow!\n");
    else {
        q.value[++q.end %= N] = v;
        q.size++;
    }
}

typeValue pop(queue &q) {
    if (!empty(q)) {
        q.size--;
        int i = q.start++;
        q.start %= N;
        return q.value[i];
    } else printf("Queue is empty!\n");
    return 0;
}
```

```
typeValue top(queue &q) {
    if (!empty(q)) return q.value[q.start];
    else printf("Queue is empty!\n");
    return 0;
}

int size(queue &q) {
    return q.size;
}

void display(queue q) {
    printf("[");
    for (int i = q.start; i < q.start + q.size; i++)
        printf("%d ", q.value[i % N]);
    printf("]\n");
}

queue merge(queue &q1, queue &q2) {
    queue q;
    init(q);
    while (!(empty(q1) || empty(q2))) {
        if (top(q1) < top(q2)) push(q, pop(q1));
        else push(q, pop(q2));
    }
    while (!empty(q1)) push(q, pop(q1));
    while (!empty(q2)) push(q, pop(q2));
    return q;
}

#endif //LAB_QUEUEMASSIVE_H
isitmuse@isitmuse:~/lab/secondSem/25-26$ cat makefile
CC = g++
CFLAGS = -c -Wall
LDFLAGS =
SOURCES = 25-26.cpp queueMassive.h
OBJECTS = $(SOURCES:.cpp=.o)
EXECUTABLE = queue.exe

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
        $(CC) $(LDFLAGS) $(OBJECTS) -o $@

.cpp.o:
        $(CC) $(CFLAGS) $< -o $@isitmuse@isitmuse:~/lab/secondSem/25-26$ make
g++ -c -Wall 25-26.cpp -o 25-26.o
g++  25-26.o queueMassive.h -o queue.exe
isitmuse@isitmuse:~/lab/secondSem/25-26$ ./queue.exe
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 1
Please enter a size of queue:10
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
```

```
0. Exit
=> 2
[72 74 26 29 7 95 54 72 75 1 ]
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 3
Please enter a value:123
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 2
[72 74 26 29 7 95 54 72 75 1 123 ]
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 4
72
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 2
[74 26 29 7 95 54 72 75 1 123 ]
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 5
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 2
[1 7 26 29 54 72 74 75 95 123 ]
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
```

```
=> 0
isitmuse@isitmuse:~/lab/secondSem/25-26$ rm 25-26.o
isitmuse@isitmuse:~/lab/secondSem/25-26$ rm queue.exe
isitmuse@isitmuse:~/lab/secondSem/25-26$ cat 25-26
cat: 25-26: No such file or directory
isitmuse@isitmuse:~/lab/secondSem/25-26$ cat 25-26.cpp
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "queueDynamic.h"
//#include "queueMassive.h"

queue mergeSort(queue &q) {
    if (empty(q) || (size(q) == 1)) {
        return q;
    }
    queue tmp1, tmp2;
    init(tmp1);
    init(tmp2);
    int dim = size(q), split = 1;
    while (!empty(q)) {
        if (split <= dim / 2) {
            push(tmp1, pop(q));
            split++;
        } else push(tmp2, pop(q));
    }
    queue q1 = mergeSort(tmp1);
    queue q2 = mergeSort(tmp2);
    return merge(q1, q2);
}

int main() {
    srand(time(0));
    queue q;
    init(q);
    int in;
    while (1) {
        printf("Menu:\n"
               "1. Generate random queue\n"
               "2. Print queue\n"
               "3. Push value\n"
               "4. Pop value\n"
               "5. Sort queue\n"
               "0. Exit\n"
               "=> ");
        scanf("%d", &in);
        switch (in) {
            case 0: {
                return 0;
            }
            case 1: {
                printf("Please enter a size of queue:");
                scanf("%d", &in);
                while (in) {
                    push(q, rand() % 100);
                    in--;
                }
                break;
            }
            case 2: {
                display(q);
                break;
            }
```

```
            case 3: {
                printf("Please enter a value:");
                scanf("%d", &in);
                push(q, in);
                break;
            }
            case 4: {
                printf("%d\n", pop(q));
                break;
            }
            case 5: {
                q = mergeSort(q);
                break;
            }
        }
    }
}isitmuse@isitmuse:~/lab/secondSem/25-26$ cat queueDynamic.h
#include <stdio.h>
#ifndef LAB_QUEUEDYNAMIC_H
#define LAB_QUEUEDYNAMIC_H
#define N 100
typedef int typeValue;

struct element {
    element *next;
    typeValue value;
};

struct queue {
    element *start;
    element *end;
    int size;
};

void init(queue &q) {
    q.start = 0;
    q.end = 0;
    q.size = 0;
}

int empty(queue &q) {
    return q.start == 0;
}

void push(queue &q, typeValue value) {
    if (q.size == N) {
        printf("Queue is overflow!\n");
        return;
    } else if (empty(q)) {
        q.end = new element;
        q.end->value = value;
        q.start = q.end;
    } else {
        element *tmp = new element;
        q.end->next = tmp;
        tmp->value = value;
        q.end = tmp;
        q.end->next = 0;
    }
    q.size++;
}

typeValue pop(queue &q) {
```

```
    if (!empty(q)) {
        typeValue value = q.start->value;
        q.start = q.start->next;
        q.size--;
        return value;
    } else printf("Queue is empty!\n");
    return 0;
}

typeValue top(queue &q) {
    if (!empty(q)) return q.start->value;
    else printf("Queue is empty!\n");
    return 0;
}

int size(queue &q) {
    return q.size;
}

void display(queue &q) {
    if (!empty(q)) {
        element *tmp = q.start;
        printf("[");
        while (tmp) {
            printf("%d ", tmp->value);
            tmp = tmp->next;
        }
        printf("]\n");
    } else printf("Queue is empty!\n");
}

queue merge(queue &q1, queue &q2) {
    queue q;
    init(q);
    while (!(empty(q1) || empty(q2))) {
        if (top(q1) < top(q2)) push(q, pop(q1));
        else push(q, pop(q2));
    }
    while (!empty(q1)) push(q, pop(q1));
    while (!empty(q2)) push(q, pop(q2));
    return q;
}

#endif //LAB_QUEUEDYNAMIC_H
isitmuse@isitmuse:~/lab/secondSem/25-26$ cat makefile
CC = g++
CFLAGS = -c -Wall
LDFLAGS =
SOURCES = 25-26.cpp queueDynamic.h
OBJECTS = $(SOURCES:.cpp=.o)
EXECUTABLE = queue.exe

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
        $(CC) $(LDFLAGS) $(OBJECTS) -o $@

.cpp.o:
        $(CC) $(CFLAGS) $< -o $@isitmuse@isitmuse:~/lab/secondSem/25-26$ make
g++ -c -Wall 25-26.cpp -o 25-26.o
g++  25-26.o queueDynamic.h -o queue.exe
isitmuse@isitmuse:~/lab/secondSem/25-26$ ./queue
-bash: ./queue: No such file or directory
```

```
isitmuse@isitmuse:~/lab/secondSem/25-26$ ./queue.exe
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 1
Please enter a size of queue:10
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 2
[82 87 88 93 43 19 1 49 8 55 ]
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 3
Please enter a value:123
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 4
82
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 2
[87 88 93 43 19 1 49 8 55 123 ]
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 5
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 2
[1 8 19 43 49 55 87 88 93 123 ]
```

```
Menu:
1. Generate random queue
2. Print queue
3. Push value
4. Pop value
5. Sort queue
0. Exit
=> 0
isitmuse@isitmuse:~/lab/secondSem/25-26$
```

9. Дневник отладки должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) 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.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечание автора по существу работы _____
_____

11. Выводы Я научился автоматизировать сборку модульных программ на языке Си с помощью утилиты make

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом _____
_____

Подпись студента _____