**Отчет по лабораторной работе №25-26** по курсу\_1\_\_\_\_\_

|  |  |
| --- | --- |
|  | Студент группы М80-105Б-21 Жилов Андрей, № по списку 3  Контакты www, e-mail: [klzxrcn3692@outlook.com](mailto:klzxrcn3692@outlook.com)\_\_\_\_\_\_\_\_\_\_\_  Работа выполнена: «26» мая 2022г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Преподаватель: В. К. Титов каф. 806\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Отчет сдан «26» мая 2022 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 и составить программный модуль, сортирующий экземпляр указанного абстрактного типа данных заданным методом, используя только, операции, импортированные из мудуля UUDT.
3. **Задание:** (*Вариант №3*):

АДТ: Очередь Метод: Сортировка слиянием Процедура: слияние двух очередей, упорядоченных по возрастанию, с сохранением порядка

1. **Оборудование(лабораторное)**:

ЭВМ -, процессор -, имя узла сети - с ОП - ГБ,

НМД - ГБ, терминал- адрес -, принтер –

Другие устройства –

*Оборудование ПЭВМ студента, если использовалось:*

Процессор AMD Ryzen 3500U, с ОП 8 ГБ, НМД 256 ГБ. Монитор 1920x1080 пикс.

Другие устройства –

1. **Программное обеспечение:**

Операционная система семейства -, наименование - версия –

интерпретатор команд – версия

Система программирования - версия –

Редактор текстов - версия –

Утилиты операционной системы –

Прикладные системы и программы –

Местонахождение и имена файлов программ и данных -

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства GNU/Linux, наименование Kali версия 5.10.0

интерпретатор команд Bash версия 5.1.8(1)-release

Система программирования \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_версия\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

редактор текстов emacs версия 27.1

Утилиты операционной системы cmp, diff, tail, od, vi, vim, ed, head, du, xargs, indent, comm, grep, tee, sum, mc, awk, iconv, file,df,split, wc, join, tr,cut,tar,sed,patch,touch, paste, mktemp, dd,sort, uniq, nroff, gzip, bzip2, md5, find, cpp

Прикладные системы и программы –

Местонахождение и имена файлов программ и данных -

1. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок,

**Описание метода:** задача разбивается на подзадачи меньшего размера, которые решаются по отдельности, после чего их решения комбинируются для получения решения исходной задачи. Если в рассматриваемом массиве один элемент, то он уже отсортирован, алгоритм завершает работу. Иначе массив разбивается на две части, которые сортируются рекурсивно. После сортировки двух частей массива к ним применяется процедура слияния, которая по двум отсортированным частям получает исходный отсортированный массив.

**Описание процедуры:** Эта процедура заключается в том, что мы сравниваем элементы массивов (начиная с начала) и меньший из них записываем в финальный. И затем, в массиве у которого оказался меньший элемент, переходим к следующему элементу и сравниваем теперь его. В конце, если один из массивов закончился, мы просто дописываем в финальный другой массив. После мы наш финальный массив записываем заместо двух исходных и получаем отсортированный участок.

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

**lr25.cpp:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "queuem.h"

queue mergeSort(queue &q)

{

if(Empty(q)||Size(q)==1)

{

return q;

}

queue s ,f;

int size = Size(q),i=1;

Init(s); Init(f);

while(!Empty(q))

{

if(i<=size/2)

{

Push(s,Pop(q));

++i;

}

else

Push(f,Pop(q));

}

queue q1 = mergeSort(s);

queue q2 = mergeSort(f);

return merge(q1,q2);

}

int main()

{

srand(time(0));

queue q;

Init(q);

int in;

while(1)

{

printf("Menu:\n");

printf("1)Random queue\n");

printf("2)Print queue\n");

printf("3)Push value\n");

printf("4)Pop value\n");

printf("5)Sort queue\n");

printf("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;

}

}

}

}

**queued.h:**

#include "stdio.h"

#ifndef QUEUEd\_H

#define QUEUEd\_H

#define N 100

#define TypeValue int

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 \*tpr = new element;

q.end->next=tpr;

tpr->value=value;

q.end = tpr;

q.end->next=0;

}

q.size++;

}

TypeValue Pop(queue &q)

{

if(Empty(q))

printf("Queue is empty\n");

else{

TypeValue value = q.start->value;

element \*s=q.start;

q.start=q.start->next;

delete s;

q.size--;

return value;

}

return 0;

}

TypeValue Top(queue &q)

{

if(Empty(q))

printf("Queue is empty\n");

else

{

return q.start->value;

}

return 0;

}

int Size(queue &q)

{

return q.size;

}

void Display(queue q)

{

if(Empty(q))

printf("Queue is empty\n");

else

{

element \*i=q.start;

printf("[");

while(i)

{

printf("%d ",i->value);

i=i->next;

}

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

**queuem.h:**

#include <stdio.h>

#ifndef QUEUEM\_H

#define QUEUEM\_H

#define N 100

#define TypeValue int

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");

else

{

q.value[++q.end%=N]=v;

q.size++;

}

}

TypeValue Pop(queue &q)

{

if(Empty(q))

printf("queue is empty\n");

else

{

q.size--;

int i = q.start++;

q.start%=N;

return q.value[i];

}

return 0;

}

TypeValue Top(queue &q)

{

if(Empty(q))

printf("queue is empty\n");

else

return q.value[q.start];

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

*Пункты 1-7 отчета составляются сторого до начала лабораторной работы.*

*Допущен к выполнению работы.*  **Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

lain@navi-1:/mnt/c/Users/lain/Desktop/23$ cat head\_lr25.txt

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

| |

| Лабораторная работа №25 |

| Абстрактные типы данных. Рекурсия. |

| Модульное программирование на языке Си. |

| Студента группы М8О-105Б-21 |

| Жилова Андрея |

| |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

lain@navi-1:/mnt/c/Users/lain/Desktop/23$ cat lr25-26.cpp

//Очередь слияние двух очередей упорядоченных по вощрастанию, с сохранением порядка сортировка слиянием

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "queuem.h"

queue mergeSort(queue &q)

{

if(Empty(q)||Size(q)==1)

{

return q;

}

queue s ,f;

int size = Size(q),i=1;

Init(s); Init(f);

while(!Empty(q))

{

if(i<=size/2)

{

Push(s,Pop(q));

++i;

}

else

Push(f,Pop(q));

}

queue q1 = mergeSort(s);

queue q2 = mergeSort(f);

return merge(q1,q2);

}

int main()

{

srand(time(0));

queue q;

Init(q);

int in;

while(1)

{

printf("Menu:\n");

printf("1)Random queue\n");

printf("2)Print queue\n");

printf("3)Push value\n");

printf("4)Pop value\n");

printf("5)Sort queue\n");

printf("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;

}

}

}

}lain@navi-1:/mnt/c/Users/lain/Desktop/23$ cat makefile

lr25-26.o: lr25-26.cpp

g++ -c -o lr25-26.o lr25-26.cpp

lr25-26: lr25-26.o queuem.o

g++ -o lr25-26 lr25-26.o queuem.h

queuem.o: queuem.hlain@navi-1:/mnt/c/Users/lain/Desktop/23$ make lr25-26

g++ -c -o lr25-26.o lr25-26.cpp

g++ -o lr25-26 lr25-26.o queuem.h

lain@navi-1:/mnt/c/Users/lain/Desktop/23$ ./lr25-26

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

1

Please enter a size of queue:16

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

2

[94 35 17 83 85 94 54 88 73 23 0 99 46 4 98 90 ]

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

3

Please enter a value:4

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

4

94

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

2

[35 17 83 85 94 54 88 73 23 0 99 46 4 98 90 4 ]

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

5

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

2

[0 4 4 17 23 35 46 54 73 83 85 88 90 94 98 99 ]

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

0

lain@navi-1:/mnt/c/Users/lain/Desktop/23$ cat lr25-26.cpp

//Очередь слияние двух очередей упорядоченных по вощрастанию, с сохранением порядка сортировка слиянием

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "queued.h"

queue mergeSort(queue &q)

{

if(Empty(q)||Size(q)==1)

{

return q;

}

queue s ,f;

int size = Size(q),i=1;

Init(s); Init(f);

while(!Empty(q))

{

if(i<=size/2)

{

Push(s,Pop(q));

++i;

}

else

Push(f,Pop(q));

}

queue q1 = mergeSort(s);

queue q2 = mergeSort(f);

return merge(q1,q2);

}

int main()

{

srand(time(0));

queue q;

Init(q);

int in;

while(1)

{

printf("Menu:\n");

printf("1)Random queue\n");

printf("2)Print queue\n");

printf("3)Push value\n");

printf("4)Pop value\n");

printf("5)Sort queue\n");

printf("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;

}

}

}

}lain@navi-1:/mnt/c/Users/lain/Desktop/23$ cat makefile

lr25-26.o: lr25-26.cpp

g++ -c -o lr25-26.o lr25-26.cpp

lr25-26: lr25-26.o queued.o

g++ -o lr25-26 lr25-26.o queued.h

queued.o: queued.hlain@navi-1:/mnt/c/Users/lain/Desktop/23$ make lr25-26

g++ -c -o lr25-26.o lr25-26.cpp

g++ -o lr25-26 lr25-26.o queued.h

lain@navi-1:/mnt/c/Users/lain/Desktop/23$ ./lr25-26

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

1

Please enter a size of queue:18

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

2

[84 40 65 36 87 74 75 66 82 1 98 64 51 49 8 16 15 3 ]

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

5

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

2

[1 3 8 15 16 36 40 49 51 64 65 66 74 75 82 84 87 98 ]

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

3

Please enter a value:4

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

2

[1 3 8 15 16 36 40 49 51 64 65 66 74 75 82 84 87 98 4 ]

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

5

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

2

[1 3 4 8 15 16 36 40 49 51 64 65 66 74 75 82 84 87 98 ]

Menu:

1)Random queue

2)Print queue

3)Push value

4)Pop value

5)Sort queue

0)Exit

0

1. **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. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  |  |  |  |  |  |  |

1. **Замечания автора по существу работы:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Выводы**:

В результате выполнения данной лабораторной работы я научился преобразовывать арифметические выражения с помощью деревьев на языке Си

Недочеты при выполнении задания могут быть устранены следующим образом:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_