|  |  |
| --- | --- |
|  | **Отчёт по лабораторной работе** №5 по курсу 1 Прикладная мат. и инф.  студента группы 08-108 Мохлякова Павла., № по списку 16. .  Адреса www, e-mail, jabber, skype. pmokhliakov@gmail.com  Работа выполнена: “ “ 2001г.  Преподаватель: каф.806. Поповкин Александр .  Входной контроль знаний с оценкой  Отчёт сдан “19“ марта 2020 г., итоговая оценка  Подпись преподавателя |

1. **Тема**: Абстрактные типы данных.Рекурсия.Модульное программирование на Си
2. **Цель работы**: Составить отладить модуль определений и модуль реализации по зааданной схеме модуля определениц для абстрактного типа данных.
3. **Задание** (*вариант №* 16 ): Сортировка очереди методом вставки
4. **Оборудование** (*лабораторное*):

ЭВМ PC , процессор i7-3770 , имя узла сети alisa с ОП 16384 МБ

НМД 400 ГБ. Терминал GNOME адрес 192.168.2.255 . Принтер

Другие устройства

*Оборудование ПЭВМ студента, если использовалось:*

Процессор Intel Core i5-3470 , ОП 8192 МБ, НМД 120 ГБ. Монитор Acer IPS 23’

Другие устройства

1. **Программное обеспечение** (*лабораторное*):

Операционная система семейства Linux , наименование Ubuntu версия 18.04.03

Интерпретатор команд bash версия 4.4.19

Система программирования gcc версия

Редактор текстов nano версия

Утилиты операционной системы make

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства Linux , наименование Manjaro версия 5.24.4

Интерпретатор команд bash версия 5.0.16

Система программирования gcc версия

Редактор текстов atom версия

Утилиты операционной системы make

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

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

**[pavel@lenovo lb2526]$** cat main.c      
#include <stdio.h>   
#include "queue.h"   
void sdvig(queue \*q,int d)   
{   
 for(int i=0;i<d;i++)   
 {   
   enqueue(q,dequeue(q));   
 }   
}   
void paste(queue \*q,int k,int i,int f)   
{   
 sdvig(q,k-i-2);   
 int pr;   
 int s=0;   
 if(f>=peek(q))   
 {   
   pr=dequeue(q);   
   enqueue(q,pr);   
   while(!((f>=pr)&&(f<peek(q))))   
   {   
     enqueue(q,dequeue(q));   
     s++;   
   }   
 }   
 else   
 {   
   i++;   
 }   
 enqueue(q,f);   
 sdvig(q,i-s);   
}   
int main()   
{   
 queue a;   
 creat(&a);   
 printf("Введите количество элементов(не больше 100)\n");   
 int k;   
 scanf("%d",&k);   
 printf("Вводите элементы\n");   
 for(int i=0;i<k;i++)   
 {   
   int j;   
   scanf("%d",&j);   
   enqueue(&a,j);   
 }   
 printq(&a);   
 int f=dequeue(&a);   
 enqueue(&a,f);   
 for(int i=0;i<k-1;i++)   
 {   
   if(peek(&a)<f)   
   {   
     paste(&a,k,i,dequeue(&a));   
   }   
   else {   
     f=dequeue(&a);   
     enqueue(&a,f);   
   }   
 }   
 printq(&a);   
 return 0;   
}   
**[pavel@lenovo lb2526]$** cat queue.h      
#ifndef \_QUEUE\_   
#define \_QUEUE\_   
  
typedef struct   
{   
 int node[100];   
 int start;   
 int end;   
 int count;   
} queue;   
  
int count(queue \*q);   
void creat(queue \*q);   
int empty(queue \*q);   
void enqueue(queue \*q,int key);   
int peek(queue \*q);   
int dequeue(queue \*q);   
void printq(queue \*q);   
#endif   
**[pavel@lenovo lb2526]$** cat queuef.c     
#include <stdio.h>   
#include "queue.h"   
  
void creat(queue \*q)   
{   
 q->count=0;   
 q->start=0;   
 q->end=-1;   
}   
  
int count(queue \*q)   
{   
 return q->count;   
}   
  
int empty(queue \*q)   
{   
 if(q->count == 0) return 1;   
 else return 0;   
}   
  
void enqueue(queue \*q,int key)   
{   
 if((q->count+1)>100)   
 {   
   printf("Queue if full\n");   
 }   
 else   
 {   
   q->count++;   
   q->end=(q->end+1)%100;   
   q->node[q->end]=key;   
 }   
}   
  
int peek(queue \*q)   
{   
 return q->node[q->start];   
}   
  
int dequeue(queue \*q)   
{   
 int key = peek(q);   
 q->count--;   
 q->start=(q->start+1)%100;   
 return key;   
}   
  
void printq(queue \*q)   
{   
 for(int i=0;i<q->count;i++)   
 {   
   printf("%d ",peek(q));   
   enqueue(q,dequeue(q));   
 }   
 printf("\n");   
}   
**[pavel@lenovo lb2526]$** cat Makefile    
CC=gcc   
  
CFLAGS=-c -Wall   
  
all: lb25-26   
  
lb25-26: main.o queuef.o   
       $(CC) main.o queuef.o -o hello   
  
main.o: main.c   
       $(CC) $(CFLAGS) main.c   
  
queuef.o: queuef.c   
       $(CC) $(CFLAGS) queuef.c   
  
clean:   
       rm -rf \*.o hello

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

Тесты:

**[pavel@lenovo lb2526]$** ./hello    
Введите количество элементов(не больше 100)   
10   
Вводите элементы   
1   
2   
7   
4   
9   
4   
7   
3   
6   
3   
1 2 7 4 9 4 7 3 6 3    
1 2 3 3 4 4 6 7 7 9    
**[pavel@lenovo lb2526]$** ./hello    
Введите количество элементов(не больше 100)   
5   
Вводите элементы   
1   
2   
3   
2   
5   
1 2 3 2 5    
1 2 2 3 5      
**[pavel@lenovo lb2526]$** ./hello    
Введите количество элементов(не больше 100)   
10   
Вводите элементы   
8   
34   
3   
7   
2   
56   
2   
34536   
7   
3   
8 34 3 7 2 56 2 34536 7 3    
2 2 3 3 7 7 8 34 56 34536

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

Допущен к выполнению работы. Подпись преподавателя

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)
2. **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 |
|  |  |  |  |  |  |  |

1. Замечание автора по существу работы
2. Выводы Научился работать с абстрактными типами данных и модульным программированием на Си

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

Подпись студента