**Отчет по лабораторной работе №25-26** по курсу Алгоритмы и структуры данных

Студент группы М8О-107Б-2022 Калуцкий Максим Витальевич, № по списку 13

Контакты e-mail max.kalutskij17@gmail.com

Работа выполнена: «18» .05.2023г.

Преподаватель: Аносова Наталья Павловна каф. 806

Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет сдан « » \_\_\_\_\_\_\_\_\_202 \_\_ г., итоговая оценка \_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. **Цель работы:**  Составить и отладить модуль определений и модуль реализации по заданной схеме модуля определений для абстрактного типа данных.
2. **Задание:** Реализовать дек с возможностью конкатенации двух деков с использованием быстрой сортировки Хоара. Составить Makefile, который будет собирать модульный проект.
3. **Оборудование** (лабораторное):

*Оборудование ПЭВМ студента, если использовалось:*

Процессор Intel Core i3-4005U CPU с ОП 8096 Мб, НМД \_ 131072\_ Мб. Монитор lenovo

Другие устройства \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства \_\_\_Unix\_\_\_\_, наименование \_\_ Ubuntu \_\_\_\_\_\_\_\_\_\_\_\_ версия 4.4.2

интерпретатор команд \_\_\_\_bash\_\_\_\_\_\_\_\_ версия \_\_5.0.17\_\_\_\_.

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

Редактор текстов \_\_\_\_\_nano\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_25.2.2 \_\_\_\_\_\_\_\_\_

Утилиты операционной системы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Прикладные системы и программы **\_\_\_**

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

Местонахождение и имена файлов программ и данных на домашнем компьютере \_\_\_\_\_home/jekich228\_\_\_\_\_\_\_\_\_

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

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

Для реализации дека я решил создать 2 класса: первый класс – класс узла дека (“node”), а второй класс – класс самой структуры (“deq”), экземпляр которого будет связывать экземпляры класса узла.

**Класс “deq”.**

В этом классе в поле “private” находится целочисленное поле “Size”, которое содержит в себе длину дека, то есть количество узлов внутри него, а также два поля с указателями на узлы, первый указатель – “head”, указывающий на первый элемент дека, второй – “tail”, указывающий на последний элемент дека. В поле “public” находятся:

* Конструктор: срабатывает при создании экземпляра класса, инициализируется поле “Size”, заполняя значением нуль, а также задаются указатели “head” и “tail” значениями nullptr;
* Деструктор: срабатывает автоматически при выходе объекта из поля видимости компилятора, то есть после завершения основного блока программы, он очищает динамически выделенную память под узлы дека, чтобы не было утечки памяти;
* Конструктор копирования: срабатывает в том случае, когда создается объект, который основан на другом объекте какого-то класса, он необходим, чтобы компилятор не очищал одну и ту же область памяти. Так как по умолчанию конструктор копирования создает новый адрес на ту же самую область памяти и, следовательно, деструктор для этой области будет вызван дважды, что приведет к непредсказуемым последствиям;
* Метод “size”: необходим для доступа к длине дека, так как само поле с длиной является приватным внутри класса;
* Метод “push\_back”: метод, который добавляет в конец структуры новый элемент;
* Метод “back”: метод, который возвращает значение последнего элемента дека;
* Метод “pop\_back”: позволяет удалять последний элемент структуры;
* Метод “pop\_front”: позволяет удалять первый элемент структуры;
* Метод “front”: возвращает значение первого элемента структуры;
* Метод “pop\_front”: удаляет первый элемент структуры;
* Метод “sort”: метод, который сортирует текущий экземпляр класса и возвращает новый экземпляр, но уже отсортированный;
* Перегруженный оператор “[ ]”: оператор был перегружен таким образом, чтобы была возможность через него получать значение узла по его индексу;
* Перегруженный оператор “+”: данный оператор был перегружен для конкатенации двух и более деков.

**Класс “node”.**

В данном классе есть 2 “private” поля: значение узла, а также указатель на следующий узел.

В “public” же находятся:

* Конструктор: принимает на вход целочисленное число и создает экземпляр узла, помещая в поле значения переданное число, а также устанавливает указатель на следующий элемент nullptr;
* Метод “print”: выводит в консоль значение узла и указатель на следующий узел;
* Метод “get\_next”: позволяет получить доступ к адресу следующего узла относительно текущего;
* Метод “get\_value”: позволяет получить доступ к значению в узле;
* Метод “change\_next”: получает на вход указатель на узел, который после будет записан в поле “next” относительно того элемента, от лица которого был вызван метод.

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

**Файл “main.cpp”:**

#include <bits/stdc++.h>

using namespace std;

#include "node.h"

#include "deq.h"

int main(){

    deq lol1,lol2;

    lol2.push\_back(1111);

    lol2.push\_back(22);

    deq t = (lol1+lol2);

    t = t.sort();

    for(int i=0;i<t.size();++i){cout<<t[i]<<" ";}

    cout<<endl;

    // cout<<t.size()<<endl;

}

**Файл “node.cpp”:**

#include "node.h"

Node::Node(int value)

{

    this->value = value;

    next = nullptr;

}

void Node::print()

{

    cout<<value<<endl;

    cout<<next<<endl;

}

Node\* Node::get\_next()

{

    return next;

}

int Node::get\_value()

{

    return value;

}

void Node::change\_next(Node\* i)

{

    next = i;

}

**Файл “deq.cpp”:**

#include "deq.h"

    deq::deq()

{

    Size = 0;

    head = nullptr;

    tail = nullptr;

    // cout<<"constructor"<<" "<<this<<" "<< endl;

}

deq::~deq() {

    Node\* phead = head;

    while(phead!=tail){

        Node\* plhead = phead;

        phead = phead->get\_next();

        delete plhead;

        plhead = nullptr;

    }

    delete tail;

    phead = nullptr;

    head = nullptr;

    tail = nullptr;

    // cout<<"destructor"<<" "<<this<<" "<< endl;

}

    deq::deq(const deq &other)

    {

        this->Size = other.Size;

        this->head = nullptr;

        this->tail = nullptr;

        for(int i=0;i<other.Size;i++){

                if(this->head==nullptr){this->head = this->tail = new Node(other[i]);}

                else{

                    this->tail->change\_next(new Node(other[i]));

                    this->tail = this->tail->get\_next();

                }

        }

        // cout<<"copy"<<" "<<this<<" "<<"from"<<&other<<endl;

    }

int deq::size(){return Size;}

void deq::push\_back(int i)

{

    if(head == nullptr){

        head = new Node(i);

        tail = head;

        Size++;

    }

    else{

        tail->change\_next(new Node(i));

        tail = tail->get\_next();

        Size++;

    }

}

int deq::back()

{

    return tail->get\_value();

}

void deq::pop\_back()

{

    if(Size == 0) return;

    else if(Size == 1){

        delete head;

        head = nullptr;

        tail = nullptr;

        Size--;

    }

    else if(Size == 2){

        delete tail;

        tail = head;

        Size--;

    }

    else{

        Node \* phead = head;

        while((phead->get\_next()) != tail){

            phead = phead->get\_next();

        }

        delete tail;

        tail = phead;

        phead = nullptr;

        Size--;

    }

}

void deq::push\_front(int i)

{

    if(head == nullptr){

        head = new Node(i);

        tail = head;

        Size++;

    }

    else{

        Node\* phead = head;

        head = new Node(i);

        head->change\_next(phead);

        phead = nullptr;

        Size++;

    }

}

int deq::front()

{

    return head->get\_value();

}

void deq::pop\_front()

{

    if(Size == 0) return;

    else if(Size == 1){

        delete head;

        head = nullptr;

        tail = nullptr;

        Size--;

    }

    else if(Size == 2){

        delete head;

        head = tail;

        Size--;

    }

    else{

        Node \* phead = head;

        head = head->get\_next();

        delete phead;

        phead = nullptr;

        Size--;

    }

}

int deq::operator[](int i) const

{

    if(i>Size or i < 0) return -10000000;

    else{

        Node\* phead = head;

        i--;

        while(i!=-1){

            phead = phead->get\_next();

            i--;

        }

        return phead->get\_value();

    }

}

deq operator\*(const deq &lol1, const deq &lol2)

{

    int Size\_lol1 = lol1.Size;

    int Size\_lol2 = lol2.Size;

    deq keke;

    for(int i = 0;i<min(Size\_lol1,Size\_lol2);++i){

        keke.push\_back(lol1[i] + lol2[i]);

    }

    return keke;

}

    deq operator+(const deq &lol1,const  deq &lol2)

    {

        deq keke;

        for(int i = 0; i<lol1.Size; ++i){

            keke.push\_back(lol1[i]);

            }

        for(int i = 0; i<lol2.Size; ++i){

            keke.push\_back(lol2[i]);

            }

        return keke;

    }

    deq deq::sort()

    {

        if(Size <= 1){

            deq keke;

            for(int i=0;i<Size;++i){keke.push\_back((\*this)[i]);

            }

            return keke;

        }

        int s = Size;

        int k = (\*this)[s/2];

        deq l1, l2, l3;

        for(int i=0;i<s;++i){

            if((\*this)[i]>k) l3.push\_back((\*this)[i]);

            else if((\*this)[i]<k) l1.push\_back((\*this)[i]);

            else l2.push\_back((\*this)[i]);

        }

        return l1.sort() + l2 + l3.sort();

    }

**Файл “node.hpp”:**

#pragma once

#include <bits/stdc++.h>

using namespace std;

class Node{

public:

    Node(int value);

    void print();

    Node\* get\_next();

    int get\_value();

    void change\_next(Node\* i);

private:

    int value;

    Node\* next;

};

**Файл “deq.hpp”:**

#pragma once

#include <bits/stdc++.h>

#include "node.h"

using namespace std;

//description of functions for creating a deq

class deq {

private:

    int Size;

    Node\* head;

    Node\* tail;

public:

    deq();

    ~deq();

    deq(const deq &other);

    int size();

    // for back

    void push\_back(int i);

    int back();

    void pop\_back();

    // for front

    void push\_front(int i);

    int front();

    void pop\_front();

    int operator [] (int i) const;

    friend deq operator \* (const deq &lol1,const deq &lol2);//for fun

    friend deq operator + ( const deq &lol1,  const deq &lol2);

    // friend deq operator + (  deq &lol1,   deq &lol2);

    deq sort();

};

deq operator \* (const deq &lol1,const deq &lol2);

deq operator + (const deq &lol1, const deq &lol2);

// deq operator + ( deq &lol1,  deq &lol2);56565

**Файл “Makefile”:**

TARGET = My\_lab\_25

CC = g++

$(TARGET) : main.o deq.o node.o

    $(CC) main.o deq.o node.o -o $(TARGET)

main.o : main.cpp

    $(CC) -c main.cpp -o main.o

deq.o : deq.cpp

    $(CC) -c deq.cpp -o deq.o

node.o : node.cpp

    $(CC) -c node.cpp -o node.o

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).
2. int main(){
3. deq lol1,lol2;
4. lol2.push\_back(1111);
5. lol2.push\_back(22);
6. deq t = (lol1+lol2);
7. t = t.sort();
8. for(int i=0;i<t.size();++i){cout<<t[i]<<" ";}
9. cout<<endl;
10. // cout<<t.size()<<endl;
11. }

PS C:\Users\heleu\Desktop\laba\_2\_sem\_kek\laba26> make

g++ -c main.cpp -o main.o

g++ -c node.cpp -o node.o

g++ main.o deq.o node.o -o My\_lab\_25

PS C:\Users\heleu\Desktop\laba\_2\_sem\_kek\laba26> ./a.exe

2 22 1111

PS C:\Users\heleu\Desktop\laba\_2\_sem\_kek\laba26>

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. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы очень крутая лаба, теперь буду во всех работах использовать модульное разбиение программы

**Выводы**: Проделав данную работу, я научился составлять модули определений и модули реализаций по заданной схеме для абстрактных типов данных, а также составлять Makefile для сборки модульного проекта.

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

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