Московский Авиационный Институт

(Национальный исследовательский университет)

**Институт №8 “Информационных технологий и прикладной математики”**

**Курсовой проект**

**по курсу**

**«Практикум программирования»**

**II семестр**

**Задание 8**

Студент: Калуцкий М.В.

Группа: М8О-107Б-22

Руководитель: Аносова Н.П.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. **Тема:** Линейные списки.

1. **Цель работы:**  Составить и отладить программу на языке Си++ для обработки линейного списка заданной организации с отображением списка на динамические структуры. Предусмотреть выполнение одного нестандартного и четырех стандартных действий:
2. Печать списка. 2. Вставка нового элемента в список. 3. Удаление элемента из списка. 4. Подсчет длины списка.
3. **Задание:** Реализовать нестандартное действие – попарный обмен значениями элементов списка.
4. **Оборудование** (лабораторное):

*Оборудование ПЭВМ студента, если использовалось:*

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

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

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)

Реализовывать список я решил через динамическую структуру, которая содержит в себе указатель на следующий элемент и поле значения (мне выпал булевский тип). Для этого я создал свой класс “List”, который содержал в себе указатель на начало списка “head” и на его конец “ tail”, последний элемент списка в поле “next” имеет указатель на первый элемент, так как по условию у меня кольцевой список.

**Печать списка.**

Для вывода всего списка в терминал нужно пробежаться от самого первого элемента списка к последнему. Это можно реализовать через цикл while (phead -> next != head), то есть нужно перебрать все элементы начиная с первого до того момента, пока мы не встретим узел, у которого в поле “next” не будет стоять указатель на “head”, другими словами мы дойдём до последнего элемента списка, ведь именно у него в данном поле стоит такой указатель, так как список кольцевой.

**Добавление элемента в список.**

Для реализации добавления элемента в список был создан метод класса “add”, который на вход принимает булевское значение, динамически выделяет память, создавая новый экземпляр структуры “node”. Далее если только что созданный объект является первым в списке, то указатели “head” и “tail” начинают указывать на него, если же в списке уже существуют узлы, то к последнему узлу через указатель “tail” мы привязываем только что созданный узел, указав в последнем узле в поле “next” адрес созданного узла. Причем у последнего узла в поле “next” должен находиться указатель на первый элемент, на который указывает “head”. По аналогии заполнение происходит далее.

**Удаление элемента из списка.**

Для реализации удаления элемента из списка я перегрузил оператор “[ ]”, на вход которого поступает целочисленное число – значение индекса, после чего оно проверяется на вхождение в допустимый диапазон. Если значение не удовлетворяет диапазону подходящих значений, то программа завершается, выводя на экран сообщение “Out of range”, если же значение было введено корректное, то в цикле вычитывается подходящий элемент. Далее высвобождается из-под него память, а в поле “next” предыдущего значения помещается элемент, который был в поле “next” элемента, который попал под удаление.

**Подсчёт длины списка.**

Для подсчёта длины в класс “List” было введено целочисленное поле “length”, которое при создании объекта класса, то есть самого списка равнялось нулю, а после добавления элемента в список через функцию “add” увеличивалась на 1. Далее метод класса “len” выводит приватное поле “length” в терминал.

**Попарный обмен значений.**

Для реализации попарного обмена был создан метод класса “List” под названием “exchange”. Этот метод на вход принимает 2 целочисленных значения – индексы элементов, значения которых нужно поменять местами. Далее полученные числа проверяются на вхождение в допустимый диапазон значений, после чего при помощи 2-х циклов перебираются элементы списка, берутся подходящие элементы и меняются местами их значения.

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

**Основной файл “main.cpp”:**

#include "List.hpp"

int main(){

    List a;

    a.add(1);

    a.add(0);

    a.print();

    // a.exchange(0,1);

    a[1];

    cout<<endl;

    a.print();

}

**Заголовочный файл “node.hpp”:**

#pragma once

#include <bits/stdc++.h>

using namespace std;

struct node{

    node\* next = nullptr;

    bool value = 0;

};

**Заголовочный файл “List.hpp”:**

#include "node.hpp"

class List{

    int length;

    node \* head = nullptr;

    node \* tail = nullptr;

public:

    List();

    ~List();

    int len();

    void add(bool value);

    void print();

    void operator [] (int i);

    void exchange(int a, int b);

};

**Файл с реализацией “List.cpp”:**

#include "List.hpp"

 List::List()

{

    this -> length = 0;

}

void List::add(bool a)

{

    node \* new\_node = new node;

    new\_node->value = a;

    length++;

    if(head == nullptr && tail == nullptr){

        head = new\_node;

        tail  = head;

        head -> next = head;

        return;

    }

    node \* ptail = tail;

    tail = new\_node;

    ptail->next = tail;

    tail->next = head;

}

void List::print()

{

    node \* phead = head;

    if(length == 1){

        cout<<head->value;

        return;

    }

    while(phead->next!=head){

        cout<<phead->value<<endl;

        phead = phead->next;

    }

    cout<<phead->value<<endl;

}

void List::exchange(int a, int b)

{

    if(a>=length || a<0 || b>=length || b<0 ||a == b){std::cerr << std::runtime\_error("Out of range").what() << std::endl;

    exit(EXIT\_FAILURE);}

    int k = 0;

    int c;

    if(a>b){

        c = b;

        b = a;

        a = c;

    }

    node \* phead = head;

    while(k!=a){

        k++;

        phead = phead->next;

    }

    k = 0;

    node \* pphead = head;

    while(k!=b){

        k++;

        pphead = pphead->next;

    }

    bool t = phead->value;

    phead->value = pphead->value;

    pphead->value = t;

}

void List::operator[](int i)

{

    if(i>=length || i<0){std::cerr << std::runtime\_error("Out of range").what() << std::endl;

    exit(EXIT\_FAILURE);}

    int k = 0;

    node \* phead = head;

    node \* pphead = tail;

    while(k!=i){

        k++;

        pphead = phead;

        phead = phead->next;

    }

    pphead -> next = phead ->next;

    head = tail->next;

    length--;

    delete phead;

}

int List::len()

{

    return length;

}

List::~List()

{

    node \* phead;

    if(length == 1){

        delete head;

        return;

    }

    while(head!=tail){

        phead = head->next;

        delete head;

        head = phead;

    }

    delete head;

}

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

Пример удаления элемента:

int main(){

    List a;

    a.add(1);

    a.add(0);

    a.add(1);

    a.add(0);

    a.print();

    // a.exchange(0,1);

    a[1];

    cout<<endl;

    a.print();

}

PS C:\Users\heleu\Desktop\laba\_2\_sem\_kek\KP8> g++ main.cpp List.cpp

PS C:\Users\heleu\Desktop\laba\_2\_sem\_kek\KP8> ./a.exe

1

0

1

0

1

1

0

Пример замены значений элементов:

int main(){

    List a;

    a.add(1);

    a.add(0);

    a.add(1);

    a.add(0);

    a.print();

    a.exchange(0,1);

    // a[1];

    cout<<endl;

    a.print();

}

PS C:\Users\heleu\Desktop\laba\_2\_sem\_kek\KP8> g++ main.cpp List.cpp

PS C:\Users\heleu\Desktop\laba\_2\_sem\_kek\KP8> ./a.exe

1

0

1

0

0

1

1

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

1. **Замечания автора** по существу работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Выводы**: Проделав данную работу, я научился составлять и отлаживать программы на языке Си++ обработки линейного списка заданной организации с отображением списка на динамические структуры.

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

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

1. **Идея, метод, алгоритм** решения задачи(в формах:словесной,псевдокода,графической[блок-схема,диаграмма,рисунок,таблица] или формальные спецификации с пред- и постусловиями)
2. **Сценарий выполнения работы** [план работы,первоначальный текст программы в черновике(можно на отдельном листе)итесты либо соображения по тестированию].
3. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).
4. **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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Выводы**: Проделав данную работу, я научился составлять программы выполнения заданных действий над файлами на интерпретируемом командном языке Bash

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

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