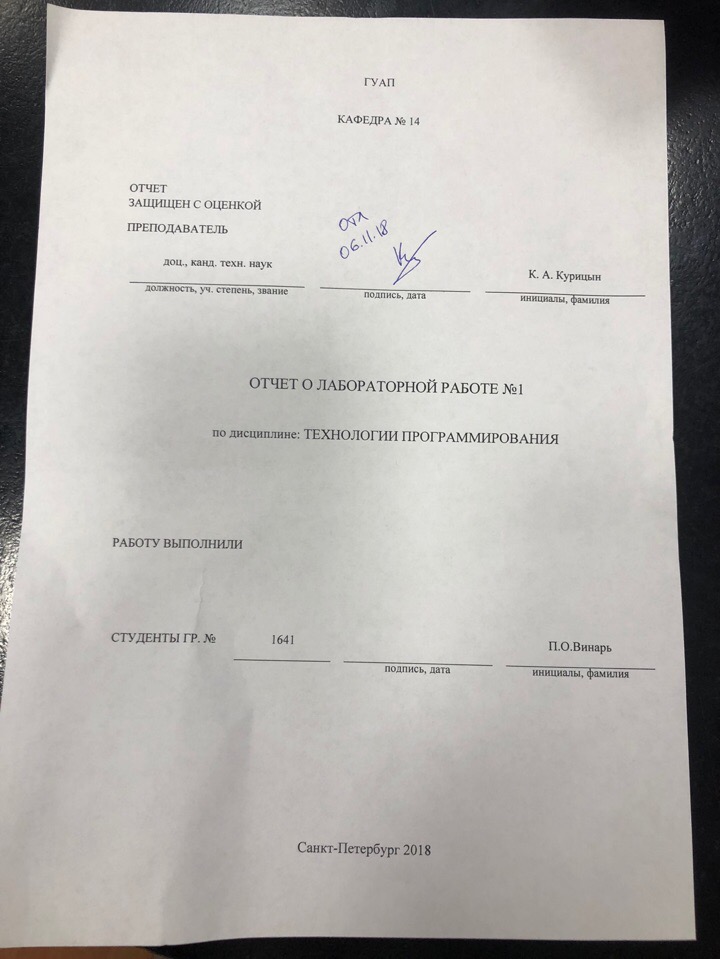
****

**1.Постановка задачи**

Определить класс с именем АЕRОFLОТ, содержащий следующие поля:

● название пункта назначения рейса;

● номер рейса;

● тип самолета.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа АЕRОFLОТ. Написать программу, выполняющую следующие действия:

● ввод с клавиатуры данных в массив, состоящий из семи объектов типа АЕRОFLОТ; записи должны быть упорядочены по возрастанию номера рейса;

● вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры;

● если таких рейсов нет, выдать на дисплей соответствующее сообщение.

**2. Формализация:**

* Добавление элементов осуществляется при помощи динамического массива new.
* Методы и дружественные функции требуется перегрузить.
* Входные данные, которые пользователь вводит с клавиатуры-это количество.
* Для сортировки массива используется шаблонная функция. Она требуется для того, чтобы при разных типах параметров, не приходилось заново писать новую функцию. Сортировка происходит по возрастанию номера рейса
* Все методы, то есть поиск, ввод, вывод, добавление, осуществляется в классе List.
* Указатели \* head, \*tail указывают на начало и конец соответственно.
* Функция to\_low, требуется для того, чтобы в программе не имел значения регистр.
* Исключениями служит:
* А). Если в массиве нет элементов, то программа выведет на экран советующее сообщение.
* Б). Если в массиве нет одинаковых рейсов, то программа выведет только один номер, который соответствует данному.
* Все введенные данные до записываются в конец файла.

**3. Таблица описания классов.**

List

Aeroflot

|  |  |  |
| --- | --- | --- |
| Имя класса | List | Aeroflot |
| Назначение. | Является списком, каждый элемент списка хранит в себе объект типа Aeroflot.  В данном классе находятся все шаблонные методы, которые использует класс Aeroflot. | Класс, который хранит в себе данные. Также используется сортировка в данном классе и изменение регистра букв. |

**Ссылка на GitHub:**

https://github.com/polinavinar/test/tree/master/laba1

**4. Листинг программы.**

4.1. Листинг файла List.h

#pragma once

template<typename T>

struct node{//элемент списка

T \_data; // значенеи элемента

node \*\_next; //ссылка на следущий элемент

node \*\_prev; //ссылка на предыдущий элемент

};

#include <iostream>

#include <cstring>

template<typename T>

class list {

private:

node<T> \*\_head; //начало списка

node<T> \*\_tail; //начало списка

int \_size; //размер

public:

list();//конструктор

void add(const T &\_data);//заносим элемент в конец списка

T remove(int index);//удаляем i-ый элемент

int size(); //размер

T operator[](int index)const;//получение i-го элемента

T & operator[](int index);//получение i-го элемента

template<typename U>

friend std::ostream& operator <<(std::ostream& out, const list<U>& l); //вывод списка на экран

template<typename U>

friend std::istream& operator >>(std::istream& in, list<U>& l); //вввод элементов списка

void PrintRangeBy(const char \*type);

~list(); //деструктор

};

template<typename T>

list<T>::list<T>():\_head(nullptr), \_tail(nullptr), \_size(0){

}

template<typename T>

void list<T>::PrintRangeBy(const char \*type){

bool flag = true;

for (int i = 0; i < \_size; i++) {

if (std::strcmp((\*this)[i].get\_type(), type) == 0) {

std::cout << (\*this)[i];

flag = false;

}

}

if (flag) std::cout << "Таких записей нет \n";

}

template<typename T>

void list<T>::add(const T &\_data){

node<T> \*buf = new node<T>;

buf->\_data = \_data;

buf->\_next = nullptr;

buf->\_prev = \_tail;

if (\_tail) {

\_tail->\_next = buf;

}

if (\_head == nullptr) {

\_head = buf;

}

\_tail = buf;

++\_size;

}

template<typename T>

T list<T>::remove(int index){

if(index < 0 || index >= \_size){//если неверный индекс

return T(0);

}

node<T> \*temp = \_head;

for(int i = 0; i < \_size && i < index; ++i) { // получаем нужный элемент

temp = temp->\_next;

}

if (temp == nullptr) {// если не нашли

return T(0);

}

if (temp->\_prev) {

temp->\_prev->\_next = temp->\_next;

}

if (temp->\_next) {

temp->\_next->\_prev = temp->\_prev;

}

T \_data = temp->\_data;

if (temp->\_prev == nullptr) {

\_head = temp->\_next;

}

if (temp->\_next == nullptr) {

\_tail = temp->\_prev;

}

delete temp;

--\_size;

return \_data;

}

template<typename T>

int list<T>::size(){

return \_size;

}

template<typename T>

T & list<T>::operator[](int index){

node<T> \*buf = \_head;

for(int i = 0; i < \_size && i < index; ++i) { //ищем нужный элемент

buf = buf->\_next;

}

return buf->\_data;

}

template<typename T>

T list<T>::operator[](int index)const{

node<T> \*buf = \_head;

for(int i = 0; i < \_size && i < index; ++i) { //ищем нужный элемент

buf = buf->\_next;

}

return buf->\_data;

}

template<typename T>

std::ostream& operator <<(std::ostream& out, const list<T>& l){

for(int i = 0; i < l.\_size; ++i){//проходим по каждому элементу

out << l[i];

}

return out;

}

template<typename T>

std::istream& operator >>(std::istream& in, list<T>& l){

T buf;

in >> buf;

l.add(buf);

return in;

}

template<typename T>

list<T>::~list(){ //деструктор

for(int i = 0; i < \_size; --\_size){//проходим по каждому элементу

node<T> \*buf = \_head;

\_head = \_head->\_next;//переходим к следующему

delete buf;//удаляем предыдущий

}

}

4.2. Листинг файла var1.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

#include <iomanip>

#include <stdlib.h>

#include "List.h"

using namespace std;

class aeroflot {

private:

char name[30];

char number[10];

char type[30];

public:

aeroflot & operator=(const aeroflot& a) {

strcpy(name, a.name);

strcpy(number, a.number);

strcpy(type, a.type);

return \*this;

}

bool operator==(const aeroflot& a) const {

return strcmp(a.name, name) == 0 && strcmp(a.number, number) == 0 && strcmp(a.type, type) == 0;

}

void set\_name(char \*\_name) {

strcpy(name, \_name);

}

void set\_number(char \*\_number) {

strcpy(number, \_number);

}

void set\_type(char \*\_type) {

strcpy(type, \_type);

}

char\* get\_name() {

return name;

}

char\* get\_number() {

return number;

}

char\* get\_type() {

return type;

}

static int cmp(const aeroflot& a, const aeroflot& b)

{

return strcmp(a.number, b.number);

}

friend istream& operator >> (istream& in, aeroflot& a) {

in >> a.name >> a.number >> a.type;

return in;

}

friend ostream& operator << (ostream& out, const aeroflot& a) {

out << a.name << "\t" << a.number << "\t" << a.type << "\n\n";

return out;

}

};

void to\_low(char mas[]) {

int x = 0;

char b, sumbol;

while (mas[x]) {

sumbol = mas[x]; //выдеделяём символ

b = (char)tolower(sumbol); //функция перевода регистров, из заглавной буквы в строчную

mas[x] = b; //выделенному символу присваем строчный вид

x++;

}

}

void bsort(list<aeroflot>& A, int size) {

int j, n = size;

do {

int nn = 0;

for (j = 1; j < n; ++j)

if (aeroflot::cmp(A[j - 1], A[j]) > 0) {

aeroflot temp = A[j - 1];

A[j - 1] = A[j];

A[j] = temp;

nn = j;

}

n = nn;

} while (n);

}

int main() {

setlocale(LC\_ALL, "Russian");

string path = "File.txt";

char type[30];

char a[30];

list<aeroflot> A;

int i = 0;

int kol = 0;

int n;

char b;

cout << "|1|\tВвести данные с клавиатуры\n|2|\tСчитать данные из файла\n\n";

int swtch = 1;

cin >> swtch;

switch (swtch)

{

case 1:

{

cout << "Количество записей? (максимум 7): ";

do {

cin >> n;

if (n>7) {

cout << "Вы ввели больше 7, введите число, меньше 7: ";

}

} while (n>7);

cout << "\n\n";

//Ввод записей

for (i = 0; i<n; i++) {

cout << "Введите название пункта назначения рейса\n";

cout << "Введите номер рейса\n";

cout << "Введите тип самолёта\n";

cout << "\n\n";

aeroflot temp;

cin >> temp;

A.add(temp);

}

//сортировка по номеру рейса

bsort(A, n);

//запись в файл

cout << "Записать данные в файл? 1 -> ДА, 2 -> НЕТ ";

int chois = 0;

cin >> chois;

if (chois == 1)

{

ofstream fout;

fout.open(path, ofstream::out);

if (!fout.is\_open())

{

cout << "Ошибка открытия файла!" << endl;

}

else

{

cout << "Файл открыт!" << endl;

fout << n << '\n';

fout << A;

}

fout.close();

system("File.txt");

}

break;

}

case 2:

{

//считывание данных из файла

ifstream fin;

fin.open(path);

if (!fin.is\_open())

{

cout << "Ошибка открытия файла!" << endl;

}

else

{

cout << "Файл открыт!" << endl;

i = 0;

fin >> n;

for (i = 0; i < n; ++i)

{

aeroflot temp;

fin >> temp;

A.add(temp);

}

}

fin.close();

system("File.txt");

break;

}

default:

break;

}

//Вывод записей

cout << "Вывод записей на экран \n" << A;

//Вывод записей определённого типа самолёта

cout << "Введите тип самолёта: ";

cin >> type;

to\_low(type);

cout << "Вывод номер рейсов и пунктов назначение, обслуживаемых типом " << type << "\n";

A.PrintRangeBy(type);

system("pause");

}

**5. Результат работы программы**

Тестовый пример 1.

|  |  |  |
| --- | --- | --- |
| Название | Что требуется сделать | Результат(Скриншот) |
| Выбор ввода данных | Пользователю требуется выбрать действие с данными:  1-Ввести их с клавиатуры  2-Считать данные из файла.  При выборе 1. Появится действие ввести количество рейсов. |  |
| Ввод количества записей и данных. | Требуется ввести количество записей, а также ввести название, номер и тип самолета. Данное действие выполнить столько раз, сколько ввели количество. |  |
| Запись данных в файл | Пользователю предлагается записать данные в файл или работать непосредственно с веденными данными, которые не сохраняются после закрытия файла.  При нажатии 2, будем работать с данными которые ввели на экран. |  |
| Работа с данными | Далее требуется ввести тип самолета, чтобы просмотреть общие данные |  |

Тестовый пример 2.

|  |  |  |
| --- | --- | --- |
| Название | Что требуется сделать | Результат(Скриншот) |
| Выбор ввода данных | Пользователю требуется выбрать действие с данными:  1-Ввести их с клавиатуры  2-Считать данные из файла.  При выборе 1. Появится действие, ввести количество рейсов. |  |
| Ввод количества записей и данных. | Требуется ввести количество записей, а также ввести название, номер и тип самолета. Данное действие выполнить столько раз, сколько ввели количество. |  |
| Запись данных в файл | Пользователю предлагается записать данные в файл или работать непосредственно с веденными данными, которые не сохраняются после закрытия файла.  При нажатии 2, будем работать с данными которые ввели на экран. |  |
| Работа с данными | Далее требуется ввести тип самолета, чтобы просмотреть есть ли такой тип самолета |  |

Тестовый пример 3.

|  |  |  |
| --- | --- | --- |
| Название | Что требуется сделать | Результат(Скриншот) |
| Выбор ввода данных | Пользователю требуется выбрать действие с данными:  1-Ввести их с клавиатуры  2-Считать данные из файла.  При выборе 2. Появится действие, «Файл открыт» |  |
| Работа с данными | Далее требуется ввести тип самолета, чтобы просмотреть есть ли такой тип самолета |  |

Тестовый пример 4.

|  |  |  |
| --- | --- | --- |
| Название | Что требуется сделать | Результат(Скриншот) |
| Выбор ввода данных | Пользователю требуется выбрать действие с данными:  1-Ввести их с клавиатуры  2-Считать данные из файла.  При выборе 1. Появится действие ввести количество рейсов. |  |
| Ввод количества записей и данных. | Требуется ввести количество записей, а также ввести название, номер и тип самолета. Данное действие выполнить столько раз, сколько ввели количество. |  |
| Запись данных в файл | Пользователю предлагается записать данные в файл или работать непосредственно с веденными данными, которые не сохраняются после закрытия файла.  При нажатии 1, будем записывать данные в файл. |  |
| Работа с данными | Далее требуется ввести тип самолета, чтобы просмотреть общие данные |  |

**6. Выводы.**

В результаты выполнения лабораторной работы, были закреплены навыки работы с обобщенными функциями, методами. Научилась разрабатывать класс, который хранит в себе данные о другом классе. Также было освоены файловые потоки при чтении и записи в него.