МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

|  |  |  |
| --- | --- | --- |
| ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ: |  |  |

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е. В. Павлов |
| (должность, уч. степень, звание) |  | (подпись, дата) |  | (инициалы, фамилия) |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

«ОЦЕНКА КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

ПО КУРСУ: «МЕТРОЛОГИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

|  |  |  |  |
| --- | --- | --- | --- |
| РАБОТУ ВЫПОЛНИЛ (-А) СТУДЕНТ (-КА): | 4931 | / | Е.Ю.Ильченко |
|  | (номер группы) |  | (инициалы, фамилия) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | / |  | / | 06.04.2021 |
|  |  | (подпись студента) |  | (дата отчета) |

1. Цель работы

Целью данной работы является изучение методологии оценки качества программного обеспечения.

1. Задание на лабораторную работу

Выполнить оценку качества разработанного приложения в соответствии со стандартом ГОСТ Р ИСО / МЭК 25010-2015.

Указать обоснование (пояснение) выполненных оценок качества и привести инфографику для каждой характеристики качества в виде лепестковой диаграммы (если данная характеристика включает в себя три или более подхарактеристики).

Если количество включаемых подхарактеристик меньше трёх, то графическое представление основной характеристики приводить не нужно.

|  |
| --- |
| Вариант задания:  6. Приложение для поиска туров (по стране) |

1. Оценка качества при использовании

Таблица 1 — Характеристики и подхарактеристики качества при использовании

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | | Оценка | Обоснование (пояснение) |
| 3.1 | Эффективность | 60% | Приложение по своим функциональным возможностям отвечает предполагаемым целям использования в заданном контексте работы.  С точки зрения анализа полноты и точности достижения определённых целей имеется ряд ограничений:   * Отсутствует возможность получения полной информации о туре; * Сам поиск туров реализован крайне примитивно и, при увеличении количества позиций (туров), затрудняет использование приложения в заданном контексте работы; * Несмотря на то, что главное меню имеет разграничение для клиента («Поиск тура») и мастера обновления ПО, нет никакой защиты данных ПО и клиент также может их изменить; * Достаточно примитивно реализована и сортировка туров, что также может затруднить использование приложения; |
| 3.2 | Производительность | 43% | В приложении не реализован контроль израсходованных ресурсов.  Все задачи управления ресурсами решаются средствами операционной системы, в рамках которой стабильная работа приложения обеспечивается экономичным форматом хранения данных и низким объёмом потребляемой памяти.  В условиях отсутствия гибкого поиска и при увеличении количества туров, время на выполнение задачи поиска необходимого тура может значительно возрасти, это обуславливает низкую производительность с точки зрения анализа человеческих ресурсов. |
| 3.3 | Удовлетворённость | 48,25% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 3.3.1 | Полноценность | 65% | Реализация функционала приложения подразумевает, что пользователю для покупки или уточнения информации по определённому туру необходимо обращаться в турагентство, что в определённой степени снижает пользу от использования самого приложения. Хотя в рамках сформулированной концепции продукта, оно соответствует заявленным задачам и осуществляет поиск туров по стране. |
| 3.3.2 | Доверие | 85% | Приложение имеет достаточно простой функционал, не работает с персональными данными пользователя, не содержит явных неисправностей и исправно справляется с заявленными требованиями, что позволяет пользователю в достаточно высокой степени быть уверенным в том, что приложение будет выполнять свои функции так, как это предполагалось.  В тоже время в виду спецификации приложения, пользователь не может быть уверен в достоверности и корректности информации о туре (проверка информации осуществляется, но она не затрагивает все случаи, которые могут произойти во время работы с приложением). |
| 3.3.3 | Удовольствие | 19% | Приложение не осуществляет поддержку персональных требований пользователя и функционирует в строго заданном контексте работы в соответствии со спецификацией (сформулированными требованиями). |
| 3.3.4 | Комфорт | 24% | В приложении отсутствует персонализация настроек интерфейса, соответственно при изменении условий использования (например, уменьшение разрешения экрана, изменение освещения, et cetera) эксплуатация приложения может сопровождаться физическим дискомфортом.  Также имеет место проблема поиска нужного тура при увеличении количества предложений, возможности поиска и сортировки достаточно ограничены, не предусмотрены различные фильтры, что достаточно усложняет сам процесс поиска и приносит неудобства при работе с большим количеством информации. |
| 3.4 | Свобода от риска | 6,3% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 3.4.1 | Смягчение отрицательных последствий экономического риска | 4% | Приложение не осуществляет смягчение последствий экономических рисков для пользователей, главным образом, это касается недобросовестных туроператоров и различных схем обмана. Приложение не предоставляет никакой информации о туроператоре, его рейтинга в туристической сфере и отзывов на его работу. В приложение отсутствует какая-либо информация о мерах предосторожности, советы при выборе туров и во время самих путешествий. |
| 3.4.2 | Смягчение отрицательных последствий риска для здоровья и безопасности | 10% | Приложение не осуществляет смягчение последствий рисков, связанных со здоровьем и безопасностью пользователей, например, в случае недостоверной информации указанной в предложении. |
| 3.4.3 | Смягчение отрицательных последствий экологического риска | 5% | Контекст работы приложения не предусматривает возможные экологические риски, а также их снижение. |
| 3.5 | Покрытие контекста | 40% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 3.5.1 | Полнота контекста | 60% | Приложение не поддерживает настройку или персонализацию интерфейса, что в определённых условиях может затруднить его использование (например, при низком разрешении экрана).  Актуальность данных приложения зависит от процессов, которые выходят за рамки его функционирования, следовательно, использование приложения в контексте актуальности предоставляемых им данных может быть сопряжено с определёнными рисками для пользователя (данное положение продиктовано сформулированной концепцией продукта и не должно значительно снижать оценку). |
| 3.5.2 | Гибкость | 20% | Приложение не предполагает использование за рамками его спецификации и не предусматривает какую-либо адаптацию под другие задачи или группы пользователей; в тоже время формат хранения данных допускают возможность изменения для адаптации к другим категориям пользователей (например, размещение предложений на английском языке). |

1. Оценка качества продукта

Таблица 2 — Модель качества продукта

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | | Оценка | Обоснование (пояснение) |
| 4.1 | Функциональная пригодность | 74,3% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 4.1.1 | Функциональная полнота | 63% | Приложение в целом отвечает всем определённым задачам и целям пользователя в указанных условиях использования, однако отсутствие полной информацией о туроператоре и предложении, дополнительных возможностей поиска снижает оценку данной характеристики. |
| 4.1.2 | Функциональная корректность | 95% | В целом все операции ввода-вывода соответствуют ожидаемому результату. |
| 4.1.3 | Функциональная целесообразность | 65% | Приложение обладает простой функциональностью и удобной системой навигации, которая допускает пропуск ненужных шагов для выполнения определённых задач, уточнение выбора пользователя, возврат к главному меню при отменен или выполнении какого-либо действия.  При увеличении количества предложений поиск может быть немного неудобным, но, тем не менее, приложение справляется с поставленными задачами |
| 4.2 | Уровень производительности | 71% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 4.2.1 | Временные характеристики | 95% | Функциональные возможности и архитектура приложения не допускают уменьшение производительности при любых условиях использования. |
| 4.2.2 | Использование ресурсов | 55% | Приложение неэффективно использует человеческие ресурсы (время) в условиях увеличения количества предложений. |
| 4.2.3 | Потенциальные возможности | 63% | Устойчивость работы приложения не зависит от достижения предельных значений параметров (количество предложений), однако подобные условия работы негативно повлияют на функциональную пригодность приложения вследствие отсутствия критериев (фильтров) поиска. |
| 4.3 | Совместимость | 87,5% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 4.3.1 | Сосуществование | 95% | Приложение не влияет на работу других приложений или систем. |
| 4.3.2 | Функциональная совместимость | 80% | Для хранения информации приложение использует текстовые файлы, что обеспечивает возможность стороннего доступа к данным и совместное их использование. Иные способы для взаимодействия и функционирования с другими приложениями не предусмотрены.  Возможности совместимости с другими операционными системами не рассматриваются. |
| 4.4 | Удобство использования | 60,2% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 4.4.1 | Определение пригодности | 87% | С точки зрения пользователя приложение в достаточной мере согласуется с функциональной целесообразностью: наименования пунктов меню соответствую своему назначению, предоставляют пользователю возможность отмены действия (посредством отказа от него при уточнении выбора ), предоставляются необходимы для клиента сведения при выборе определённых пунктов меню. |
| 4.4.2 | Изучаемость | 96% | Наличие простого функционала и удобная навигация исключают необходимость в обучении. |
| 4.4.3 | Управляемость | 86% | Приложение включает в себя необходимые атрибуты удобной навигации (например, пропуск шагов).  Приложение обладает определённой устойчивостью к ошибкам пользователя. (реализованы простые проверки на корректность введённых данных).  Реализованные проверки:   * Проверка корректности введённого пункта меню; * Проверка корректности введённого названия курорта, типа и названия апартаментов, информации о питании; * Проверка корректности введённого кол-ва человек; * Проверка корректности введённой стоимости; * Проверка корректности введённых дат поездки; * Проверка корректности введённой операции;   Типы сортировки данных:   * Сортировка записей в файле по стоимости в порядке возрастания; * Сортировка записей в файле по стоимости в порядке убывания; |
| 4.4.4 | Защищённость от ошибки пользователя | 67% | Реализованы основные проверки данных для ввода на соответствие формату хранения информации, однако перечень проверок нельзя назвать исчерпывающим. Также нет контроля целостности данных в случае редактирования информации с помощью сторонних приложений (свободный доступ к текстовому файлу, который служит базой данных для приложения). Нет контроля над содержательностью вводимой информации. Введённое название курорта, информация о питании, тип и название апартаментов, могут быть несуществующими или не содержать в себе никакого смысла и полезной информации. Нет контроля дублирующейся информации (можно добавить два и более одинаковых предложения с разными индивидуальными номерами). |
| 4.4.5 | Эстетика пользовательского интерфейса | 20% | Учитывая назначение приложения, наличие консольного интерфейса выступает фактором, который в значительной степени снижает оценку данной характеристики. |
| 4.4.6 | Доступность | 5% | Приложение не осуществляет поддержку категорий пользователей с ограниченными возможностями, также отсутствует возможность настройки и персонализации интерфейса. |
| 4.5 | Надежность | 95% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 4.5.1 | Завершенность | 95% | Критерии, которые могли бы негативно сказаться на надёжности приложения в условиях нормальной работы, не выявлены. |
| 4.5.2 | Готовность | 95% | Приблизительная (субъективная) оценка способности приложения находиться в рабочем состоянии, составляет ~ 99,9%.  Возможные отказы приложения могут быть связаны исключительно с внешними факторами. |
| 4.5.3 | Отказоустойчивость | 95% | По результатам тестирования функциональных требований, в приложении не было выявлено критических дефектов, что в сочетании с анализом архитектуры приложения позволяет произвести высокую оценку данного атрибута качества. |
| 4.5.4 | Восстанавливаемость | 95% | Архитектура приложения исключает влияние сбоев на целостность и корректность данных.  Восстановление приложения обеспечивается перезапуском программы и не зависит от причин, которые вызвали отказ. |
| 4.6 | Защищенность | 5% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 4.6.1 | Конфиденциальность | 5% | В приложении не реализовано разграничение прав доступа. |
| 4.6.2 | Целостность | 5% | Нет защиты или контроля целостности и корректности данных, при этом возможно изменение хранимой информации с помощью сторонних приложений, так как все данные приложения хранятся во внешнем файле. |
| 4.6.3 | Неподдельность | 5% | Приложение не поддерживает функциональность, которая бы позволила обеспечить контроль данных и производимых операций над ними. |
| 4.7 | Сопровождаемость, модифицируемость | 82,25% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 4.7.1 | Модульность | 73% | Присутствует разделение пользовательского интерфейса и данных приложения на отдельные составляющие. В данном случае пользовательский интерфейс рассматривается как совокупность уровня представления (отображение данных пользователю) и контроллера (управляющей логики приложения). Хранение данных приложения осуществляется посредством внешних файлов, таким образом, изменение данных может осуществляться независимо от пользовательского интерфейса.  Реализации модульной структуры в приложении не предусмотрена. |
| 4.7.2 | Возможность многократного использования | 76% | Реализация приложения представляет собой простой и самодокументируемый код, некоторые фрагменты которого возможно использовать для создания приложений с подобными задачами.  Однако некоторые участки кода представляют собой решения задач необходимых именно при реализации данного приложения. |
| 4.7.3 | Анализируемость | 85% | В приложении не реализованы возможности генерации отчётов об ошибках, однако анализ и диагностика данного приложения не сопряжены с какими-либо трудностями вследствие его простоты. |
| 4.7.4 | Модифицируемость | 78% | Приложение представляет собой один программный модуль с достаточно простой функциональностью и низкими требованиями к сопровождению. Однако с учетом замечания, сделанного в 4.7.2, оценка данного атрибута может быть снижена. |
| 4.7.5 | Тестируемость | 93% | Функциональные возможности приложения обуславливают простоту его тестирования, как в отношении требований, так и в отношении кода. |
| 4.8 | Переносимость, мобильность | 80% | Оценка данной характеристики выполнена с учётом её подхарактеристик. |
| 4.8.1 | Адаптируемость | 85% | Приложение не требует существенных изменений исходного кода для портирования в иную среду функционирования. |
| 4.8.2 | Устанавливаемость | 60% | Приложение не требует установки, однако функциональность приложения обеспечивается наличием соответствующих библиотек. |
| 4.8.3 | Взаимозаменяемость | 95% | Обновление версии приложения не влияет на используемые форматы хранения данных. |

Выводы по работе

В результате выполнения данной работы произведена оценка разработанного приложения с точки зрения моделей качества при использовании и качества продукта в соответствии со стандартом ГОСТ Р ИСО / МЭК 25010-2015.

Для всех оценок характеристик качества приложения приведено обоснование или формальное уточнение.

Ниже представлены оценки характеристик моделей качества, в том числе в графическом виде для полноты информации (что может быть важно при отсутствии консолидированных метрик качества).

*Оценка качества при использовании:*

1. Эффективность — 60%
2. Производительность — 43%
3. Удовлетворённость — 48,25%
4. Свобода от риска — 6,3%

В приложении не реализованы механизмы для снижения различного рода рисков, поэтому для данной характеристики инфографика не приводится.

1. Покрытие контекста — 40%

*Оценка качества продукта:*

1. Функциональная пригодность — 74,3%
2. Уровень производительности — 71%
3. Совместимость — 87,5%
4. Удобство использования —60,2%
5. Надежность — 95%

Приложение обладает достаточно простой архитектурой, поэтому для данной характеристики инфографика не приводится.

1. Защищенность — 5%

Приложение не поддерживает защиту информации и данных, поэтому для данной характеристики инфографика не приводится.

1. Сопровождаемость, модифицируемость — 82,25%
2. Переносимость, мобильность — 80%

Таким образом, по результатам проделанной работы можно сделать выводы о сильных и слабых сторонах продукта. Отдельно следует отметить относительно низкие оценки удовлетворённости, функциональной пригодности и удобства использования, что обусловлено главным образом следующими факторами:

* Отсутствием полной информации о предложениях и туроператоре;
* Отсутствие какой-либо информации и советов, смягчающих последствия риска для здоровья и безопасности;
* Отсутствие информации о мерах предосторожности, советов при выборе туров и во время самих путешествий;
* Отсутствием критериев и фильтров для поиска туров;
* Реализация консольного интерфейса для задач, где он менее всего для этого подходит.

Указанные факторы представляют собой основные недостатки приложения и в значительной степени оказали отрицательное влияние на оценку некоторых его характеристик качества.

Стоит отметить, что выполненные оценки в значительной степени несут субъективный характер, так как рассматриваемый стандарт не предполагает использование точных численных методов для оценки качества. Соответственно, полученные оценки характеристик качества могут отличаться от результатов подобной работы, выполненной иными лицами, по отношению к данному приложению.

Таким образом, можно заключить, что выполненная работа соответствует поставленной задаче и отвечает всем сформулированным в методических указаниях требованиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Павлов Е. В. Методические рекомендации к выполнению лабораторных работ: Метрология программного обеспечения / Евгений Васильевич Павлов. —   
   СПб ГУАП, 2021
2. ГОСТ Р ИСО/МЭК 25010-2015 Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов, 2015.
3. ГОСТ Р ИСО/МЭК 25051-2017 Информационные технологии (ИТ). Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Требования к качеству готового к использованию программного продукта (RUSP) и инструкции по тестированию, 2017.
4. Липаев В. В. Выбор и оценивание характеристик качества программных средств. Методы и стандарты / В. В. Липаев. — М.: СИНТЕГ, 2001. — 228 с.: ил.
5. Липаев В. В. Проектирование и производство сложных заказных программных продуктов / В. В. Липаев. — М.: СИНТЕГ, 2011. — 408 с.

ПРИЛОЖЕНИЕ А

Исходный код программы

/\*Class.h\*/

#pragma once

#include <string>

#include <fstream>

#include <iostream>

using namespace std;

class Dlist;

class NOTE

{

private:

string code; //у какдого предложения есть индивидуальный код

unsigned short int date[6];//даты поездки

string resort;//курорт

string apart;//тип и название апартаментов

unsigned short int h; //кол-во человек

string food;//информация о питании

unsigned short int price;//стоимость поездки

NOTE\* next = NULL;

NOTE\* prev = NULL;

public:

NOTE(string, unsigned short int[6], string, string, unsigned short int, string, unsigned short int);

friend Dlist;

};

class Dlist

{

private:

NOTE\* head = NULL;

NOTE\* tail = NULL;

public:

void add(string,unsigned short int[6], string, string, unsigned short int, string, unsigned short int);

bool clear();

int remove(string);

void show();

bool free();

void load();

bool editing(string);

void save();

void sorting(int);

void sort(NOTE\*&, NOTE\*&);

void findelem(int, string);

string forcode();

void del();

~Dlist();

};

/\*Class.cpp\*/

#include "Class.h"

#include "Check.h"

#include <iomanip>

#include <string.h>

#include <algorithm>

#include <iostream>

#include <ctype.h>

using namespace std;

Check check;

//конструктор для NOTE

NOTE::NOTE(string code, unsigned short int date[6], string resort, string apart, unsigned short int h, string food, unsigned short int price)

{

this->code = code;

for (int i = 0; i < 6; i++)

{

this->date[i] = date[i];

}

this->resort = resort;

this->apart = apart;

this->h = h;

this->food = food;

this->price = price;

}

//Функция для добавления элементов в список

void Dlist::add(string code,unsigned short int date[6], string resort, string apart, unsigned short int h, string food, unsigned short int price)

{

if (head == NULL) //Проверяем, является ли список пустым

{

NOTE\* cell = new NOTE(code,date, resort, apart, h, food, price); //если является, то предложение становится головой и хвостом списка

head = cell;

tail = cell;

}

else {

//Если в списке что-то есть

NOTE\* temp = head;

while (temp->next != NULL)// переходим в его конец и добавляем туда предложение

{

temp = temp->next;

}

NOTE\* cell = new NOTE(code,date, resort, apart, h, food, price);

tail = cell;

temp->next = cell;

cell->prev = temp;

}

}

//функция для удаления списка

bool Dlist::clear()

{

NOTE\* temp = head;

if (temp != NULL)

{

NOTE\* next;

while (temp != NULL)

{

next = temp->next;

delete temp;

temp = next;

if (temp == NULL)

{

head = NULL;

tail = NULL;

return true;

}

}

}

else

{

return false;

}

}

//функция для удаления элемента списка

int Dlist::remove(string c)

{

NOTE\* temp = head;

while (temp != NULL)

{

string code = temp->code;

if (code == c)

{

if (temp->next == NULL && temp->prev == NULL) //если предложения один в списке

{

head = NULL;

tail = NULL;

delete(temp);

return true;//предложение удалёно, возвращаем 1

}

if (temp->next == NULL) //если предложение расположено в начале списка

{

temp->prev->next = NULL;

delete(temp);

return 1;//предложение удалёно, возвращаем 1

}

if (temp->prev == NULL) //если предложение расположено в конце списка

{

head = temp->next;

temp->next->prev = NULL;

delete(temp);

return 1;//предложение удалёно, возвращаем 1

}

else //если предложение расположено в середине списка

{

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

delete(temp);

return 1; //предложение удалёно, возвращаем 1

}

}

if (code != c && temp->next == NULL)

{

return 2; //если предложение не найдено, возвращаем 2

}

temp = temp->next;

}

}

//функция для вывода списка предложений на экран

void Dlist::show()

{

cout << "\nСписок туров по стране:\n";

NOTE\* temp = head;

int g = 0;

while (temp != NULL) //выводим предложения, пока не дойдём до конца списка

{

cout << string(70, '\_') ;

cout << "\nИндивидуальный код: | "<<temp->code;

cout << "\nКурорт: | " << temp->resort;

cout << "\nДата(заселения/отъезда): | ";

for (int i = 0; i < 6; i++)

{

int b = temp->date[i];

if (i != 2 && i != 5)

{

if (i == 3) {

cout << " - ";

}

if (b < 10)

{

cout << "0" << b << ".";

}

else

{

cout << b << ".";

}

}

else

{

cout << b;

}

}

cout << "\nТип апартаментов и название: | " << temp->apart << "\nКол-во человек: | " << temp->h << "\nПитание: | " << temp->food << "\nЦена: | " << temp->price << " руб.\n";

temp = temp->next;

}

}

//проверка списка на пустоту

bool Dlist::free()

{

if (head != NULL)

{

return true;

}

else

{

return false;

}

}

//Деструктор класса

Dlist::~Dlist()

{

if (head != NULL)

{

NOTE\* temp\_root = tail;

NOTE\* temp = temp\_root->prev;

head = NULL;

tail = NULL;

while (temp != NULL) {

delete(temp\_root);

temp\_root = temp;

temp = temp->prev;

}

}

}

//функция для загрузки списка предложений из файла

void Dlist::load()

{

clear(); //предварительная очистка списка

ifstream fin("infolist.txt");

int crash = 0;

if (!fin)

{

cout << "Файл не открывается!\n";

}

else

{

string resort,apart,food,line,people,sum, code;

int h = 0, price = 0;

unsigned short int date[6];

while (getline(fin, line))

{

int pos = line.find("|");

code = line.substr(0, pos);//присваиваем code часть строки до "|"

line = line.substr(pos + 1);

pos = line.find("|");

resort = line.substr(0, pos); //присваиваем resort часть строки до след "|"

line = line.substr(pos + 1);

pos = line.find("|");

apart = line.substr(0, pos); //присваиваем apart часть строки до след "|"

line = line.substr(pos + 1);

pos = line.find("|");

people = line.substr(0, pos); //присваиваем people часть строки до след "|"

h = stoi(people); //преобразуем в int

line = line.substr(pos + 1);

pos = line.find("|");

food = line.substr(0, pos); //присваиваем food часть строки до след "|"

line = line.substr( pos+1);

pos = line.find("|");

sum = line.substr(0, pos); //присваиваем sum часть строки до след "|"

price = stoi(sum); //преобразуем в int

line = line.substr(pos + 1);

pos = line.find(".");//присваиваем часть строки до "."

sum = line.substr(0, pos);

date[0] = stoi(sum);

line = line.substr(pos + 1);

pos = line.find(".");//присваиваем часть строки до "."

sum = line.substr(0, pos);

date[1] = stoi(sum);

line = line.substr(pos + 1);

pos = line.find("-");//присваиваем часть строки до "-"

sum = line.substr(0, pos);

date[2] = stoi(sum);

line = line.substr(pos + 1);

pos = line.find(".");//присваиваем часть строки до "."

sum = line.substr(0, pos);

date[3] = stoi(sum);

line = line.substr(pos + 1);

pos = line.find(".");//присваиваем часть строки до "."

sum = line.substr(0, pos);

date[4] = stoi(sum);

line = line.substr(pos + 1);

pos = line.find("|");//присваиваем часть строки до "|"

sum = line.substr(0, pos);

date[5] = stoi(sum);

add(code,date,resort,apart,h,food,price); //заносим их в список

}

fin.close();

}

}

//функция для редактирования предложений

bool Dlist::editing(string с)

{

NOTE\* temp = head;

while (temp != NULL) //поиск предложения в списке по индивидуальному коду

{

string code = temp->code;

if (code == с)

{

cout << "\nЗаполните информационные поля для него с учётом изменений\n";

string k = forcode();

temp->code = k;

unsigned short int date[6];

check.fillandcheckdate(date);

for (int i = 0; i < 6; i++)

{

temp->date[i] = date[i];

}

cout << "Введите название курорта: ";

string resort = check.checkname();

temp->resort = resort;

cout << "Введите тип и название апартаментов: ";

string apart = check.checkname();

temp->apart = apart;

cout << "Введите кол-во человек: ";

unsigned short h = check.chec();

temp->h = h;

cout << "Введите информацию о питании: ";

string food = check.checkname();

temp->food = food;

cout << "Введите стоимость: ";

unsigned short int price = check.checkprice();

temp->price = price;

return true;

}

if (code != с && temp->next == NULL)

{

return false;

}

temp = temp->next;

}

}

//Функция для сохранения списка предложений в файл

void Dlist::save()

{

NOTE\* temp = head;

if (temp != NULL)

{

ofstream fout("infolist.txt");

if (!fout.is\_open())

{

cout << "Файл не может быть открыт!\n";

}

else

{

while (temp != NULL)

{

fout << temp->code << '|' << temp->resort << '|'<<temp->apart<<'|'<<temp->h<<'|'<<temp->food<<'|'<<temp->price<<'|';

for (int i = 0; i < 6; i++)

{

int b = temp->date[i];

if (i != 2 && i != 5)

{

if (i == 3) {

fout << " - ";

}

if (b < 10)

{

fout << "0" << b << ".";

}

else

{

fout << b << ".";

}

}

else

{

fout << b;

}

}

fout << "\n";

temp = temp->next;

}

}

fout.close();

}

else

{

cout << "Список предложений пуст!\n";

}

}

void Dlist::sorting(int x)

{

NOTE\* tmph = head;

NOTE\* tmpnext = head->next;

if (x == 1) //Сортировка по возрастанию цены

{

while (tmph != NULL) {

tmpnext = tmph->next;

while (tmpnext != NULL) {

if (tmph->price > tmpnext->price) {

sort(tmph, tmpnext);

}

tmpnext = tmpnext->next;

}

tmph = tmph->next;

}

cout << "\nСписок предложений успешно отсортирован по возрастанию цены!\n\n";

}

else if (x == 2) //Сортировка по убыванию цены

{

while (tmph != NULL) {

tmpnext = tmph->next;

while (tmpnext != NULL) {

if (tmph->price < tmpnext->price) {

sort(tmph, tmpnext);

}

tmpnext = tmpnext->next;

}

tmph = tmph->next;

}

cout << "\nСписок предложений успешно отсортирован по убыванию цены!\n\n";

}

}

//функция для сортировки списка предложений

void Dlist::sort(NOTE\*& first, NOTE\*& second)

{

NOTE\* snext = second->next;

NOTE\* sprev = second->prev;

NOTE\* fprev = first->prev;

NOTE\* fnext = first->next;

if (first->next == second) {

second->next = first;

second->prev = first->prev;

}

else {

second->prev = first->prev;

second->next = first->next;

fnext->prev = second;

}

if (second->prev == NULL)

head = second;

else {

fprev->next = second;

}

if (first->next == second) {

first->next = snext;

first->prev = second;

}

else {

first->next = snext;

first->prev = sprev;

sprev->next = first;

}

if (first->next == NULL)

tail = first;

else

snext->prev = first;

NOTE\* temp = second;

second = first;

first = temp;

}

void Dlist::findelem(int x, string k)

{

NOTE\* temp = head;

if (x == 1) //поиск предложения по коду

{

while (temp != NULL) //ищем код в списке, пока не дойдём до конца

{

string code = temp->code;

if (code == k)

{

cout << string(70, '\_');

cout << "\nИндивидуальный код: | " << temp->code;

cout << "\nКурорт: | " << temp->resort;

cout << "\nДата(заселения/отъезда): | ";

for (int i = 0; i < 6; i++)

{

int b = temp->date[i];

if (i != 2 && i != 5)

{

if (i == 3) {

cout << " - ";

}

if (b < 10)

{

cout << "0" << b << ".";

}

else

{

cout << b << ".";

}

}

else

{

cout << b;

}

}

cout << "\nТип апартаментов и название: | " << temp->apart << "\nКол-во человек: | " << temp->h << "\nПитание: | " << temp->food << "\nЦена: | " << temp->price << " руб.\n";

break;

}

if (code != k && temp->next == NULL)

{

cout << "Такого предложения нет!\n";

}

temp = temp->next;

}

}

if (x == 2) //поиск туров по названию

{

int j = 0;

transform(k.begin(), k.end(), k.begin(), tolower);

while (temp != NULL)

{

string resort = temp->resort;

transform(resort.begin(), resort.end(), resort.begin(), tolower);

if (k == resort)

{

cout << string(70, '\_');

cout << "\nИндивидуальный код: | " << temp->code;

cout << "\nКурорт: | " << temp->resort;

cout << "\nДата(заселения/отъезда): | ";

for (int i = 0; i < 6; i++)

{

int b = temp->date[i];

if (i != 2 && i != 5)

{

if (i == 3) {

cout << " - ";

}

if (b < 10)

{

cout << "0" << b << ".";

}

else

{

cout << b << ".";

}

}

else

{

cout << b;

}

}

cout << "\nТип апартаментов и название: | " << temp->apart << "\nКол-во человек: | " << temp->h << "\nПитание: | " << temp->food << "\nЦена: | " << temp->price << " руб.\n";

j++;

}

if (j == 0 && temp->next == NULL)

{

cout << "Предложений по данному курорту не обнаружено!\n\n";

}

temp = temp->next;

}

}

}

//функция для проверки уникальности кода

string Dlist::forcode()

{

string num;

int t = 1;

NOTE\* temp = head;

while (t != 0) {

t = 0;

cout << "Введите индивидуальный код предложения:\n";

cin >> num;

if (temp != NULL)

{

while (temp != NULL)

{

string code = temp->code;

if (code == num)

{

cout << "Такой код уже сеществует, повторите попытку!\n";

t++;

break;

}

temp = temp->next;

}

}

}

return num;

}

void Dlist::del()

{

ofstream output;

output.open("infolist.txt", ios\_base::trunc);

output.close();

}

/\*Check.h\*/

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Check

{

public:

int checkc();

string checkname();

int chec();

int checkprice();

void fillandcheckdate(unsigned short int\* m);

int checkf();

};

/\*Check.cpp\*/

#include "Check.h"

//функция для проверки и ввода пункта меню

int Check::checkc()

{

cout << "\n\n Номер операции: ";

int var;

cin >> var;

while (cin.fail() || var < 0 || var > 5 || cin.get() != '\n')

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите еще раз:\n ";

cin >> var;

}

return var;

}

//для проверки названия курорта, типа апартаментов, информации о питании

string Check::checkname()

{

string name;

int t = 1;

while (t != 0) {

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

getline(cin, name);

t = 0;

for (int i = 0; i < name.length(); i++) {

if (isdigit((unsigned char)name.at(i))) {

cout << "Некорректный ввод, повторите попытку!\n";

t++;

break;

}

if (name.at(i) < 'А' || name.at(i) > 'я' ) { //проверка на ввод только русских букв

if (name.at(i) == ' ') {

break;

}

cout << "Некорректный ввод, повторите попытку!\n";

t++;

break;

}

}

}

return name;

}

//функция для проверки кол-ва человек

int Check::chec()

{

int var;

cin >> var;

while (cin.fail() || var < 1 || var >30 || cin.get() != '\n')

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите еще раз:\n ";

cin >> var;

}

return var;

}

//Функция для проверки введённой стоимости

int Check::checkprice()

{

int var;

cin >> var;

while (cin.fail() || var < 0 || cin.get() != '\n')

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите еще раз:\n ";

cin >> var;

}

return var;

}

//функция для проверки и заполнения дат поездки

void Check::fillandcheckdate(unsigned short int\* m)

{

unsigned short int date = 0;

for (int i = 0; i < 6; i++)

{

if (i == 0) {

cout << "Введите дату начала тура\n";

}

else if (i == 3) {

cout << "Введите дату окончания тура\n";

}

if (i == 0 || i == 3)

{

cout << "Введите день\n";

cin >> date;

if (m[1] == 2) //если февраль

{

if (m[2] % 4)//проверка на високосный год

{

while (cin.fail() || date < 1 || date>28)

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите попытку!\n\n";

cout << "Введите день(от 1 до 28)\n";

cin >> date;

}

}

else

{

while (cin.fail() || date < 1 || date>29)

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите попытку!\n\n";

cout << "Введите день (от 1 до 29)\n"; cin >> date;

}

}

}

//если январь, март, май, июль, август, октябрь, декабрь

else if (m[1] == 1 || m[1] == 3 || m[1] == 5 || m[1] == 7 || m[1] == 8 || m[1] == 10 || m[1] == 12)

{

while (cin.fail() || date < 1 || date>31) {

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите попытку!\n\n";

cout << "Введите день(от 1 до 31)\n";

cin >> date;

}

}

//если апрель, июнь, сентябрь, ноябрь

else

{

while (cin.fail() || date < 1 || date>30) {

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите попытку!\n\n";

cout << "Введите день(от 1 до 30)\n";

cin >> date;

}

}

m[i] = date;

}

if (i == 1 || i == 4)

{

cout << "Введите месяц (от 1 до 12)\n";

cin >> date;

while (cin.fail() || date < 1 || date>12) {

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите попытку!\n\n";

cout << "Введите месяц отъезда(от 1 до 12)\n";

cin >> date;

}

m[i] = date;

}

if (i == 5 || i == 2)

{

cout << "Введите год (от 2021 до 2023):\n";

cin >> date;

while (cin.fail() || date < 2021 || date>2023) {

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите попытку!\n\n";

cout << "Введите год(от 1900 до 2021)\n";

cin >> date;

}

m[i] = date;

}

}

}

//проверка введённого номера операции

int Check::checkf()

{

cout << "\nНомер операции: ";

int var;

cin >> var;

while (cin.fail() || var < 0 || var >2 || cin.get() != '\n')

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail(), '\n');

cout << "Неверный ввод, повторите еще раз:\n ";

cin >> var;

}

return var;

}

/\*MPO1.cpp\*/

#include <iostream>

#include <windows.h>

#include "Check.h"

#include "Class.h"

#include <string>

#include <fstream>

#include <iostream>

#include <time.h>

#include <cstdlib>

using namespace std;

void savetimeanddate();

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

system("color F1 ");

Dlist list;

Check check;

string resort, apart, food,code;

unsigned short int h, price, x = 0, z=0, y=0,l;

unsigned short int date[6];

bool result;

savetimeanddate();

list.load();

do

{

cout << "\n\tГлавное меню:\n";

cout << "1. Поиск тура;\n2. Панель мастера обновления ПО;\n0. Выход из программы;";

x = check.checkc();

system("cls");

switch (x)

{

case 1:

do {

cout << "\n\tМеню:\n";

cout << "1. Вывести список всех предложений;\n2. Сортировать туры;\n3. Найти тур;\n0. Перейти в главное меню;";

z = check.checkc();

system("cls");

switch (z) {

case 1:

if (list.free()) {

cout << "Купить тур можно в нашем агенстве по адресу г. Санкт-Петербург ул. Цветочная д. 2 \nПо всем вопросам звоните на горячую линию: 8(800)555-35-35;\n";

list.show();

}

else {

cout << "На данный момент туров по стране нет\n";

}

break;

case 2:

if (list.free())

{

cout << "Что вы хотите сделать?\n\n";

cout << "1. Сортировать по возрастанию цены;\n2. Сортировать по убыванию цены;\n0. Перейти в меню;";

l = check.checkf();

list.sorting(l);

list.save();

}

else

{

cout << "Список предложений пуст!\n\n";

}

l = 0;

break;

case 3:

if (list.free())

{

cout << "\n1. Найти предложение по индивидуальному коду;\n" << "2. Найти элемент по названию курорта;\n";

l = check.checkf();

system("cls");

if (l == 1) {

cout << "Введите индивидуальный код предложения:\n";

cin >> code;

system("cls");

cout << "\nКупить тур можно в нашем агенстве по адресу г. Санкт-Петербург ул. Цветочная д. 2 \nПо всем вопросам звоните на горячую линию: 8(800)555-35-35;\n\n";

cout << "\nПо вашему запросу найдено:\n";

list.findelem(l, code);

}

else {

cout << "Введите название курорта:\n";

resort = check.checkname();

system("cls");

cout << "\nКупить тур можно в нашем агенстве по адресу г. Санкт-Петербург ул. Цветочная д. 2 \nПо всем вопросам звоните на горячую линию: 8(800)555-35-35;\n\n";

cout << "\nПо вашему запросу найдено:\n";

list.findelem(l, resort);

}

}

else

{

cout << "Список предложений пуст!\n\n";

}

l = 0;

break;

case 0:

break;

default:

break;

}

} while (z);

break;

case 2:

do {

cout << "\n\tМеню:\n";

cout << "1. Вывести список всех предложений;\n2. Добавить предложение;\n3. Изменить предложение;\n4. Удалить предложение;\n5. Удалить все предложения;\n0. Перейти в главное меню;";

y = check.checkc();

system("cls");

switch (y) {

case 1:

if (list.free()) {

list.show();

}

else{

cout << "Список предложений пуст!Заполните его.\n";

}

break;

case 2:

code=list.forcode();

check.fillandcheckdate(date);

cout << "Введите название курорта: ";

resort = check.checkname();

cout << "Введите тип и название апартаментов: ";

apart = check.checkname();

cout << "Введите кол-во человек: ";

h = check.chec();

cout << "Введите информацию о питании: ";

food = check.checkname();

cout << "Введите стоимость: ";

price = check.checkprice();

list.add(code, date, resort, apart, h, food, price);

cout << "Информация успешно добавлена.\n\n";

list.del();

list.save();

break;

case 3:

if (list.free())

{

cout << "Введите индивидуальный код предложения:\n";

cin >> code;

list.findelem(1, code);

result = list.editing(code);

if (result)

{

list.del();

list.save();

system("cls");

cout << "\nПредложение успешно отредактировано!\n\n";

}

else

{

system("cls");

cout << "\nТакого предложения нет!\n";

}

}

else

{

cout << "Список предложений пуст!Заполните его.\n\n";

}

break;

case 4:

if (list.free())

{

cout << "Введите индивидуальный код предложения:\n";

cin >> code;

list.findelem(1, code);

l = list.remove(code);

if (l == 1)

{

system("cls");

cout << "\nПредложение успешно удалёно!\n\n";

list.del();

list.save();

}

else if (l== 2)

{

cout << "\nТакого предложения нет!\n\n";

}

}

else

{

cout << "Список предложений пуст!\n\n";

}

l = 0;

break;

case 5:

if (list.free())

{

cout << "Вы точно хотите удалить все предложения?\nПосле удаления все текущие данные будут потеряны!\n\n";

cout << "1. Да;\n2. Нет;\n";

l = check.checkf();

if (l == 1)

{

list.clear();

list.del();

system("cls");

cout << "Данные удалены\n\n";

}

}

else

{

cout << "Список предложений пуст!\n\n";

}

break;

case 0:

break;

default:

break;

}

} while (y);

break;

case 0:

break;

default:

break;

}

} while (x);

}

void savetimeanddate()

{

ofstream fout("fortimeanddate.txt");

if (fout.is\_open()) {

struct tm newtime;

time\_t now = time(0);

localtime\_s(&newtime, &now);

fout << "Дата: ";

if (newtime.tm\_mday < 10) {

fout << "0" << newtime.tm\_mday << ".";

}

else {

fout << newtime.tm\_mday << ".";

}

if (1 + newtime.tm\_mon) {

fout << "0" << 1 + newtime.tm\_mon << ".";

}

else {

fout << 1 + newtime.tm\_mon << ".";

}

fout << 1900 + newtime.tm\_year;

fout << " Время: ";

if (1 + newtime.tm\_hour < 10) {

fout << "0" << newtime.tm\_hour << ":";

}

else {

fout << newtime.tm\_hour << ":";

}

if (1 + newtime.tm\_min < 10) {

fout << "0" << 1 + newtime.tm\_min << ":";

}

else {

fout << 1 + newtime.tm\_min << ":";

}

if (1 + newtime.tm\_sec < 10) {

fout << "0" << 1 + newtime.tm\_sec;

}

else {

fout << 1 + newtime.tm\_sec;

}

}

else {

cout << "Сообщение для мастера ПО: проблема с файлом";

}

}