Файл main.cpp содержащий main()

#include "userinfo.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

Userinfo w; //создание главного окна

w.show(); //отображение главного окна

return a.exec();

}

Файл Plane.h содержащий объявление класса Plane

#ifndef \_PLANE\_H\_

#define \_PLANE\_H\_

#include <QString>

#include <QTextStream>

class Plane //класс самолет

{

private:

QString model; //модель самолета

QString company; //название компании

public:

Plane(const QString& model = "", const QString& company = "") //конструктор

{

this->model = model;

this->company = company;

}

Plane(const Plane& obj) //конструктор копирования

{

this->model=obj.model;

this->company=obj.company;

}

~Plane(){} //деструктор

void setModel(const QString&); //установить модель

void setCompany(const QString&); //установить название компании

QString getModel(); //получить модель

QString getCompany(); //получить название компании

Plane& operator=(const Plane&); //оператор =

friend QTextStream& operator<<(QTextStream&, const Plane&); //оператор записи в файл

friend QTextStream& operator>>(QTextStream&, Plane&); //оператор чтения с файла

};

#endif

Файл Plane.cpp содержащий реализацию класса Plane

#include "plane.h"

void Plane::setModel(const QString& model)

{

this->model = model;

}

void Plane::setCompany(const QString& company)

{

this->company = company;

}

QString Plane::getModel()

{

return this->model;

}

QString Plane::getCompany()

{

return this->company;

}

Plane& Plane::operator=(const Plane& obj)

{

if(this != &obj)

{

this->model=obj.model;

this->company=obj.company;

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const Plane& obj)

{

out << obj.model <<"\n"<< obj.company <<"\n";

return out;

}

QTextStream& operator>>(QTextStream& in, Plane& obj)

{

obj.model = in.readLine();

obj.company = in.readLine();

return in;

}

Файл passengerPlane.h содержащий объявление класса PassengerPlane

#ifndef \_PASSENGERPLANE\_H\_

#define \_PASSENGERPLANE\_H\_

#include "plane.h"

struct spaceType //тип места

{

int economClass;

int businessClass;

int firstClass;

};

class PassengerPlane : public Plane //класс пассажирский самолет

{

private:

spaceType maxSpace; //максимальное кол-во мест

spaceType space; //количество занятых мест

int passengersCount; //количество пассажиров

public:

PassengerPlane(int econom = 0, int business = 0, int first = 0) //конструктор

{

this->maxSpace.economClass = econom;

this->maxSpace.businessClass = business;

this->maxSpace.firstClass = first;

this->space.economClass = econom;

this->space.businessClass = business;

this->space.firstClass = first;

}

PassengerPlane(const PassengerPlane& obj) //конструктор копирования

{

this->maxSpace.economClass = obj.maxSpace.economClass;

this->maxSpace.businessClass = obj.maxSpace.businessClass;

this->maxSpace.firstClass = obj.maxSpace.firstClass;

this->space.economClass = obj.space.economClass;

this->space.businessClass = obj.space.businessClass;

this->space.firstClass = obj.space.firstClass;

}

~PassengerPlane(){} //деструктор

void setPassengersCount(); //установить кол-во пассажиров

int getPassengersCount(); //получить кол-во пассажиров

void setMaxEconom(int); //установить максимум мест каждого типа

void setMaxBusiness(int);

void setMaxFirst(int);

int getMaxEconom(); //получение кол-ва мест каждого типа

int getMaxBusiness();

int getMaxFirst();

void setEconom(int); //установить кол-во занятых мест

void setBusiness(int);

void setFirst(int);

int getEconom(); //получить кол-во занятых мест

int getBusiness();

int getFirst();

bool isFool(); //проверка на заполненность

PassengerPlane& operator=(const PassengerPlane&); //оператор =

//оператор записи в файл

friend QTextStream& operator<<(QTextStream&, const PassengerPlane&);

//оператор чтения с файла

friend QTextStream& operator>>(QTextStream&, PassengerPlane&);

};

#endif

Файл passengerPlane.cpp содержащий реализацию класса PassengerPlane

#include "passengerPlane.h"

bool PassengerPlane::isFool()

{

return maxSpace.businessClass+maxSpace.economClass+maxSpace.firstClass == passengersCount;

}

void PassengerPlane::setMaxEconom(int count)

{

this->maxSpace.economClass = count;

}

void PassengerPlane::setMaxBusiness(int count)

{

this->maxSpace.businessClass = count;

}

void PassengerPlane::setMaxFirst(int count)

{

this->maxSpace.firstClass = count;

}

int PassengerPlane::getMaxEconom()

{

return this->maxSpace.economClass;

}

int PassengerPlane::getMaxBusiness()

{

return this->maxSpace.businessClass;

}

int PassengerPlane::getMaxFirst()

{

return this->maxSpace.firstClass;

}

void PassengerPlane::setEconom(int count)

{

this->space.economClass = count;

this->setPassengersCount();

}

void PassengerPlane::setBusiness(int count)

{

this->space.businessClass = count;

this->setPassengersCount();

}

void PassengerPlane::setFirst(int count)

{

this->space.firstClass = count;

this->setPassengersCount();

}

int PassengerPlane::getEconom()

{

return this->space.economClass;

}

int PassengerPlane::getBusiness()

{

return this->space.businessClass;

}

int PassengerPlane::getFirst()

{

return this->space.firstClass;

}

PassengerPlane& PassengerPlane::operator=(const PassengerPlane& obj)

{

if(this != &obj)

{

Plane::operator=(obj);

this->maxSpace.economClass = obj.maxSpace.economClass;

this->maxSpace.businessClass = obj.maxSpace.businessClass;

this->maxSpace.firstClass = obj.maxSpace.firstClass;

this->space.economClass = obj.space.economClass;

this->space.businessClass = obj.space.businessClass;

this->space.firstClass = obj.space.firstClass;

this->setPassengersCount();

}

return \*this; }

void PassengerPlane::setPassengersCount()

{

passengersCount = space.businessClass+space.economClass+space.firstClass;

}

int PassengerPlane::getPassengersCount()

{

return this->passengersCount;

}

QTextStream& operator<<(QTextStream& out, const PassengerPlane& obj)

{

out << dynamic\_cast<const Plane&>(obj) << obj.maxSpace.economClass << "\n" <<obj.maxSpace.businessClass << "\n" <<obj.maxSpace.firstClass

<< "\n" <<obj.space.economClass << "\n" << obj.space.businessClass << "\n" << obj.space.firstClass << "\n" ;

return out;

}

QTextStream& operator>>(QTextStream& in, PassengerPlane& obj)

{

in >> dynamic\_cast<Plane&>(obj);

obj.maxSpace.economClass = in.readLine().toInt();

obj.maxSpace.businessClass = in.readLine().toInt();

obj.maxSpace.firstClass = in.readLine().toInt();

obj.space.economClass = in.readLine().toInt();

obj.space.businessClass = in.readLine().toInt();

obj.space.firstClass = in.readLine().toInt();

obj.setPassengersCount();

return in;

}

Файл passengerFlight.h содержащий объявление класса PassengerFlight

#ifndef \_PASSENGERFLIGHT\_H\_

#define \_PASSENGERFLIGHT\_H\_

#include "passengerPlane.h"

#include "QDateTime"

class PassengerFlight : public PassengerPlane //класс пассижирский полет

{

private:

QDateTime date; //дата и время полета

QString location; //местоположение

public:

PassengerFlight(const QString& location = "") //констуктор

{

this->location=location;

}

PassengerFlight(const PassengerFlight& obj) //констуктор копирования

{

this->date=obj.date;

this->location=obj.location;

}

~PassengerFlight(){} //деструктор

void setDate(const QDateTime&); //установить дату и время

void setLocation(const QString&); //установить местоположение

QDateTime getDate(); //получить дату и время

QString getLocation(); //получить местоположение

PassengerFlight& operator=(const PassengerFlight&); //оператор =

//оператор записи в файл

friend QTextStream& operator<<(QTextStream&, const PassengerFlight&);

//оператор чтения с файла

friend QTextStream& operator>>(QTextStream&, PassengerFlight&);

};

#endif

Файл passengerFlight.cpp содержащий реализацию класса PassengerFlight

#include "passengerFlight.h"

void PassengerFlight::setDate(const QDateTime& date)

{

this->date = date;

}

void PassengerFlight::setLocation(const QString& location)

{

this->location = location;

}

QDateTime PassengerFlight::getDate()

{

return this->date;

}

QString PassengerFlight::getLocation()

{

return this->location;

}

PassengerFlight& PassengerFlight::operator=(const PassengerFlight& obj)

{

if (this != &obj)

{

PassengerPlane::operator=(obj);

this->date=obj.date;

this->location=obj.location;

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const PassengerFlight& obj)

{

out << dynamic\_cast<const PassengerPlane&>(obj)<< obj.date.toString(Qt::ISODate)

<< "\n" << obj.location<< "\n" ;

return out;

}

QTextStream& operator>>(QTextStream& in, PassengerFlight& obj)

{

in >> dynamic\_cast<PassengerPlane&>(obj);

QString data = in.readLine();

obj.date = QDateTime::fromString(data, Qt::ISODate);

obj.location = in.readLine();

return in;

}

Файл passengerDepartures.h содержащий объявление класса PassengerDepartures

#ifndef PASSENGERDEPARTURES\_H

#define PASSENGERDEPARTURES\_H

#include "passengerFlight.h"

#include "passenger.h"

#include <list>

class PassengerDepartures : public PassengerFlight //класс пассажирских вылетов

{

private:

int flyingTime; //время полета

int costEconom; //стоимость билета эконом класса

int costBusiness; //стоимость билета бизнес класса

int costFirst; //стоимость билета первого класса

std::list<Passenger> passengerList; //список пассажиров

public:

PassengerDepartures(int flyingTime = 0, int costEconom = 0,int costBusiness = 0,

int costFirst = 0) //конструктор

{

this->flyingTime=flyingTime;

this->costEconom=costEconom;

this->costBusiness=costBusiness;

this->costFirst=costFirst;

}

PassengerDepartures(const PassengerDepartures& obj) //конструктор копирования

{

this->flyingTime=obj.flyingTime;

this->costEconom=obj.costEconom;

this->costBusiness=obj.costBusiness;

this->costFirst=obj.costFirst;

this->passengerList=obj.passengerList;

}

~PassengerDepartures(){} //деструктор

void setFlyingTime(int); //установить время полета

void setCostEconom(int); //установить стоимость эконом

void setCostBusiness(int); //установить стоимость бизнес

void setCostFirst(int); //установить стоимость первого класса

int getFlyingTime(); //получить время полета

int getCostEconom(); //получить стоимость эконом

int getCostBusiness(); //получить стоимость бизнес

int getCostFirst(); //получить стоимость первого класса

PassengerDepartures& operator=(const PassengerDepartures&); //оператор =

std::list<Passenger>& getPassengerList(); //получить список пассажиров

//оператор записи в файл

friend QTextStream& operator<<(QTextStream&, const PassengerDepartures&);

//оператор чтения с файла

friend QTextStream& operator>>(QTextStream&, PassengerDepartures&);

};

#endif // PASSENGERDEPARTURES\_H

Файл passengerDepartures.cpp содержащий реализацию класса PassengerDepartures

#include "passengerDepartures.h"

void PassengerDepartures::setFlyingTime(int time)

{

this->flyingTime = time;

}

void PassengerDepartures::setCostEconom(int cost)

{

this->costEconom = cost;

}

void PassengerDepartures::setCostBusiness(int cost)

{

this->costBusiness = cost;

}

void PassengerDepartures::setCostFirst(int cost)

{

this->costFirst = cost;

}

int PassengerDepartures::getFlyingTime()

{

return this->flyingTime;

}

int PassengerDepartures::getCostEconom()

{

return this->costEconom;

}

int PassengerDepartures::getCostBusiness()

{

return this->costBusiness;

}

int PassengerDepartures::getCostFirst()

{

return this->costFirst;

}

std::list<Passenger>& PassengerDepartures::getPassengerList()

{

return this->passengerList;

}

PassengerDepartures& PassengerDepartures::operator=(const PassengerDepartures& obj)

{

if (this != &obj)

{

PassengerFlight::operator=(obj);

this->flyingTime=obj.flyingTime;

this->costEconom=obj.costEconom;

this->costBusiness=obj.costBusiness;

this->costFirst=obj.costFirst;

this->passengerList=obj.passengerList;

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const PassengerDepartures& obj)

{

out << dynamic\_cast<const PassengerFlight&>(obj) << obj.flyingTime << "\n" << obj.costEconom << "\n" << obj.costBusiness << "\n" << obj.costFirst << "\n";

return out;

}

QTextStream& operator>>(QTextStream& in, PassengerDepartures& obj)

{

in >> dynamic\_cast<PassengerFlight&>(obj);

obj.flyingTime = in.readLine().toInt();

obj.costEconom = in.readLine().toInt();

obj.costBusiness = in.readLine().toInt();

obj.costFirst = in.readLine().toInt();

return in;

}

Файл passengerArrivals.h содержащий объявление класса PassengerArrivals

#ifndef PASSENGERARRIVALS\_H

#define PASSENGERARRIVALS\_H

#include "passengerFlight.h"

class PassengerArrivals : public PassengerFlight //класс пассажирских прилетов

{

private:

int baggageTime; //время разгрузки багажа

bool delayed; //задержка

public:

PassengerArrivals(int baggageTime = 0, bool delayed = false){ //конструктор

this->baggageTime=baggageTime;

this->delayed=delayed;

}

PassengerArrivals(const PassengerArrivals& obj) //конструктор копирования

{

this->baggageTime=obj.baggageTime;

this->delayed=obj.delayed;

}

~PassengerArrivals(){} //деструктор

void setBaggageTime(int); //установить время разгрузки багажа

int getBaggageTime(); //получить время разгрузки

void setDelay(bool); //установить задержку

bool isDelayed(); //проверка на задержку

PassengerArrivals& operator=(const PassengerArrivals&); //оператор =

//оператор записи в файл

friend QTextStream& operator<<(QTextStream&, const PassengerArrivals&);

//оператор чтения с файла

friend QTextStream& operator>>(QTextStream&, PassengerArrivals&);

};

#endif // PASSENGERARRIVALS\_H

Файл passengerArrivals.cpp содержащий реализацию класса PassengerArrivals

#include "passengerArrivals.h"

void PassengerArrivals::setBaggageTime(int baggageTime)

{

this->baggageTime=baggageTime;

}

int PassengerArrivals::getBaggageTime()

{

return this->baggageTime;

}

void PassengerArrivals::setDelay(bool delayed)

{

this->delayed=delayed;

}

bool PassengerArrivals::isDelayed()

{

return this->delayed;

}

PassengerArrivals& PassengerArrivals::operator=(const PassengerArrivals& obj)

{

if (this != &obj)

{

PassengerFlight::operator=(obj);

this->baggageTime=obj.baggageTime;

this->delayed=obj.delayed;

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const PassengerArrivals& obj)

{

out << dynamic\_cast<const PassengerFlight&>(obj) << obj.baggageTime << "\n" << static\_cast<int>(obj.delayed) << "\n";

return out;

}

QTextStream& operator>>(QTextStream& in, PassengerArrivals& obj)

{

in >> dynamic\_cast<PassengerFlight&>(obj);

obj.baggageTime = in.readLine().toInt();;

int delayedValue;

delayedValue = in.readLine().toInt();;

obj.delayed = static\_cast<bool>(delayedValue);

return in;

}

Файл passenger.h содержащий объявление класса Passenger

#ifndef PASSENGER\_H

#define PASSENGER\_H

#include <QString>

#include <QTextStream>

class Passenger //класс пассажир

{

private:

QString name; //имя

QString surname; //фамилия

QString number; //контактные данные

int space; //тип места

public:

Passenger(const QString& name = "", const QString& surname = "",

const QString& number = "",int space = 0) //конструктор

{

this->name=name;

this->surname=surname;

this->number=number;

this->space=space;

}

Passenger(const Passenger& obj) //конструктор копирования

{

this->name=obj.name;

this->surname=obj.surname;

this->number=obj.number;

this->space=obj.space;

}

~Passenger(){} //деструктор

void setName(const QString&); //установить имя

void setSurname(const QString&); //установить фамилию

void setNumber(const QString&); //установить контактные данные

void setSpace(int); //установить тип места

QString getName(); //получить имя

QString getSurname(); //получить фамилию

QString getNumber(); //получить контактные данные

int getSpace(); //получить тип места

Passenger& operator=(const Passenger&); //оператор =

//оператор записи в файл

friend QTextStream& operator<<(QTextStream&, const Passenger&);

//оператор чтения с файла

friend QTextStream& operator>>(QTextStream&, Passenger&);

};

#endif // PASSENGER\_H

Файл passenger.cpp содержащий реализацию класса Passenger

#include "passenger.h"

void Passenger::setName(const QString& name)

{

this->name=name;

}

void Passenger::setSurname(const QString& surname)

{

this->surname=surname;

}

void Passenger::setNumber(const QString& number)

{

this->number=number;

}

void Passenger::setSpace(int space)

{

this->space=space;

}

QString Passenger::getName()

{

return this->name;

}

QString Passenger::getSurname()

{

return this->surname;

}

QString Passenger::getNumber()

{

return this->number;

}

int Passenger::getSpace()

{

return this->space;

}

Passenger& Passenger::operator=(const Passenger& obj)

{

if(this != &obj)

{

this->name=obj.name;

this->surname=obj.surname;

this->number=obj.number;

this->space=obj.space;

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const Passenger& obj)

{

out << obj.surname << "\n" << obj.name << "\n" << obj.number << "\n" << obj.space << "\n";

return out;

}

QTextStream& operator>>(QTextStream& in, Passenger& obj)

{

obj.surname = in.readLine();

obj.name = in.readLine();

obj.number = in.readLine();

obj.space = in.readLine().toInt();;

return in;

}

Файл cargoPlane.h содержащий объявление класса CargoPlane

#ifndef \_CARGOPLANE\_H\_

#define \_CARGOPLANE\_H\_

#include "Plane.h"

class CargoPlane: public Plane //класс грузовой самолет

{

private:

float cargoCapacity; //грузоподъемность

float currentWeigth; //текущий вес

float freeWeight; //свободный вес

int cargoCount; //количество грузов

public:

CargoPlane(float capacity = 0, float weight = 0, int cargoCount = 0)//конструктор

{

this->cargoCapacity = capacity;

this->currentWeigth = weight;

this->cargoCount=cargoCount;

setFreeWeight();

}

CargoPlane(const CargoPlane& obj) //конструктор копирования

{

this->cargoCapacity = obj.cargoCapacity;

this->currentWeigth=obj.currentWeigth;

this->cargoCount=obj.cargoCount;

this->setFreeWeight();

}

~CargoPlane(){} //деструктор

void setFreeWeight(); //установить свободный вес

void setCargoCapacity(float); //установить грузоподъемность

void setWeight(float); //установить текущий вес

void setCargoCount(int); //установить количество грузов

float getCargoCapacity(); //получить грузоподъемность

float getWeight(); //получить текущий вес

float getFreeWeight(); //получить свободный вес

int getCargoCount(); //получить количество грузов

CargoPlane& operator=(const CargoPlane&); //оператор =

//оператор записи в файл

friend QTextStream& operator<<(QTextStream&, const CargoPlane&);

//оператор чтения с файла

friend QTextStream& operator>>(QTextStream&, CargoPlane&);

};

#endif

Файл cargoPlane.cpp содержащий реализацию класса CargoPlane

#include "cargoPlane.h"

void CargoPlane::setFreeWeight()

{

freeWeight=cargoCapacity-currentWeigth;

}

void CargoPlane::setCargoCapacity(float weight)

{

this->cargoCapacity = weight;

this->setFreeWeight();

}

void CargoPlane::setWeight(float weight)

{

this->currentWeigth = weight;

this->setFreeWeight();

}

void CargoPlane::setCargoCount(int count)

{

this->cargoCount=count;

}

float CargoPlane::getCargoCapacity()

{

return this->cargoCapacity;

}

float CargoPlane::getWeight()

{

return this->currentWeigth;

}

float CargoPlane::getFreeWeight()

{

return this->freeWeight;

}

int CargoPlane::getCargoCount()

{

return this->cargoCount;

}

CargoPlane& CargoPlane::operator=(const CargoPlane& obj)

{

if(this != &obj)

{

Plane::operator=(obj);

this->cargoCapacity = obj.cargoCapacity;

this->currentWeigth=obj.currentWeigth;

this->cargoCount=obj.cargoCount;

this->setFreeWeight();

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const CargoPlane& obj)

{

out << dynamic\_cast<const Plane&>(obj) << obj.cargoCapacity << "\n"

<< obj.currentWeigth << "\n" << obj.cargoCount<< "\n" ;

return out;

}

QTextStream& operator>>(QTextStream& in, CargoPlane& obj)

{

in >> dynamic\_cast<Plane&>(obj);

obj.cargoCapacity = in.readLine().toFloat();

obj.currentWeigth = in.readLine().toFloat();

obj.cargoCount = in.readLine().toInt();

obj.setFreeWeight();

return in;

}

Файл cargoFlight.h содержащий объявление класса CargoFlight

#ifndef \_CARGOFLIGHT\_H\_

#define \_CARGOFLIGHT\_H\_

#include "cargoPlane.h"

#include <QDateTime>

class CargoFlight : public CargoPlane //класс грузовой полет

{

private:

QDateTime date; //дата и время

QString location; //местоположение

public:

CargoFlight(const QString& location="") //конструктор

{

this->location=location;

}

CargoFlight(const CargoFlight& obj) //конструктор копирования

{

this->location=obj.location;

this->date=obj.date;

}

~CargoFlight() {} //деструктор

void setDate(QDateTime); //установить дату и время

void setLocation(QString); //установить местоположение

QDateTime getDate(); //получить дату и время

QString getLocation(); //получить местоположение

CargoFlight& operator=(const CargoFlight&); //оператор =

//оператор записи в файл

friend QTextStream& operator<<(QTextStream&, const CargoFlight&);

//оператор чтения с файла

friend QTextStream& operator>>(QTextStream&, CargoFlight&);

};

#endif

Файл cargoFlight.cpp содержащий реализацию класса CargoFlight

#include "cargoFlight.h"

void CargoFlight::setDate(QDateTime date)

{

this->date = date;

}

void CargoFlight::setLocation(QString location)

{

this->location = location;

}

QDateTime CargoFlight::getDate()

{

return this->date;

}

QString CargoFlight::getLocation()

{

return this->location;

}

CargoFlight& CargoFlight::operator=(const CargoFlight& obj)

{

if(this != &obj)

{

CargoPlane::operator=(obj);

this->location=obj.location;

this->date=obj.date;

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const CargoFlight& obj)

{

out << dynamic\_cast<const CargoPlane&>(obj) << obj.date.toString(Qt::ISODate)<<"\n" << obj.location<< "\n" ;

return out;

}

QTextStream& operator>>(QTextStream& in, CargoFlight& obj)

{

in >> dynamic\_cast<CargoPlane&>(obj);

QString data = in.readLine();

obj.date = QDateTime::fromString(data, Qt::ISODate);

obj.location = in.readLine();

return in;

}

Файл cargoDepartures.h содержащий объявление класса CargoDepartures

#ifndef CARGODEPARTURES\_H

#define CARGODEPARTURES\_H

#include "cargoFlight.h"

#include "cargo.h"

#include <list>

class CargoDepartures : public CargoFlight //класс грузовох вылетов

{

private:

int flyingTime; //время полета

int cost; //стоимость

std::list<Cargo> cargoList; //список грузов

public:

CargoDepartures(int flyingTime = 0, int cost = 0) //конструктор

{

this->flyingTime=flyingTime;

this->cost = cost;

}

CargoDepartures(const CargoDepartures& obj) //конструктор копирования

{

this->flyingTime=obj.flyingTime;

this->cost = obj.cost;

this->cargoList=obj.cargoList;

}

~CargoDepartures(){} //деструктор

void setFlyingTime(int); //установить время полета

void setCost(int); //установить стоимость

int getFlyingTime(); //получить время полета

int getCost(); //получить стоимость

std::list<Cargo>& getCargoList(); //получить список грузов

CargoDepartures& operator=(const CargoDepartures&); //оператор =

//оператор записи в файл

friend QTextStream& operator<<(QTextStream&, const CargoDepartures&);

//оператор чтения с файла

friend QTextStream& operator>>(QTextStream&, CargoDepartures&);

};

#endif // CARGODEPARTURES\_H

Файл cargoDepartures.cpp содержащий реализацию класса CargoDepartures

#include "cargodepartures.h"

void CargoDepartures::setFlyingTime(int flyingTime)

{

this->flyingTime=flyingTime;

}

void CargoDepartures::setCost(int cost)

{

this->cost=cost;

}

int CargoDepartures::getFlyingTime()

{

return this->flyingTime;

}

int CargoDepartures::getCost()

{

return this->cost;

}

std::list<Cargo>& CargoDepartures::getCargoList()

{

return this->cargoList;

}

CargoDepartures& CargoDepartures::operator=(const CargoDepartures& obj)

{

if(this != &obj)

{

CargoFlight::operator=(obj);

this->flyingTime=obj.flyingTime;

this->cost=obj.cost;

this->cargoList=obj.cargoList;

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const CargoDepartures& obj)

{

out << dynamic\_cast<const CargoFlight&>(obj) << obj.flyingTime << "\n" << obj.cost <<"\n";

return out;

}

QTextStream& operator>>(QTextStream& in, CargoDepartures& obj)

{

in >> dynamic\_cast<CargoFlight&>(obj);

obj.flyingTime = in.readLine().toInt();

obj.cost = in.readLine().toInt();

return in;

}

Файл cargoArrivals.h содержащий объявление класса CargoArrivals

#ifndef CARGOARRIVALS\_H

#define CARGOARRIVALS\_H

#include "cargoFlight.h"

class CargoArrivals :public CargoFlight //класс грузовых прилетов

{

private:

int baggageTime; //время разгрузки

bool delayed; //задержка

public:

CargoArrivals(int baggageTime = 0, bool delayed = false) //конструктор

{

this->baggageTime=baggageTime;

this->delayed=delayed;

}

CargoArrivals(const CargoArrivals& obj) //конструктор копирования

{

this->baggageTime=obj.baggageTime;

this->delayed=obj.delayed;

}

~CargoArrivals(){} //деструктор

void setBaggageTime(int); //установить время разгрузки

int getBaggageTime(); //получить время разгрузки

void setDelay(bool); //установить задержку

bool isDelayed(); //проверка на задержку

CargoArrivals& operator=(const CargoArrivals&); //оператор =

//оператор записи в файл

friend QTextStream& operator<<(QTextStream&, const CargoArrivals&);

//оператор чтения с файла

friend QTextStream& operator>>(QTextStream&, CargoArrivals&);

};

#endif // CARGOARRIVALS\_H

Файл cargoArrivals.cpp содержащий реализацию класса CargoArrivals

#include "cargoarrivals.h"

void CargoArrivals::setBaggageTime(int baggageTime)

{

this->baggageTime=baggageTime;

}

int CargoArrivals::getBaggageTime()

{

return this->baggageTime;

}

void CargoArrivals::setDelay(bool delayed)

{

this->delayed=delayed;

}

bool CargoArrivals::isDelayed()

{

return this->delayed;

}

CargoArrivals& CargoArrivals::operator=(const CargoArrivals& obj)

{

if(this != &obj)

{

CargoFlight::operator=(obj);

this->baggageTime=obj.baggageTime;

this->delayed=obj.delayed;

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const CargoArrivals& obj)

{

out << dynamic\_cast<const CargoFlight&>(obj) << obj.baggageTime << "\n"

<< static\_cast<int>(obj.delayed) <<"\n";

return out;

}

QTextStream& operator>>(QTextStream& in, CargoArrivals& obj)

{

in >> dynamic\_cast<CargoFlight&>(obj);

obj.baggageTime = in.readLine().toInt();

int delayedValue = in.readLine().toInt();

obj.delayed = static\_cast<bool>(delayedValue);

return in;

}

Файл cargo.h содержащий объявление класса Cargo

#ifndef CARGO\_H

#define CARGO\_H

#include <QString>

#include <QTextStream>

class Cargo //класс груз

{

private:

QString name; //имя

QString surname; //фамилия

QString number; //контактные данные

QString cargoName; //название груза

float cargoWeight; //вес груза

public:

Cargo(const QString& name="", const QString& surname="", const QString& number="", const QString cargoName="", float cargoWeight=0) //конструктор

{

this->name=name;

this->surname=surname;

this->number=number;

this->cargoName=cargoName;

this->cargoWeight=cargoWeight;

}

Cargo(const Cargo& obj) //конструктор копирования

{

this->name=obj.name;

this->surname=obj.surname;

this->number=obj.number;

this->cargoName=obj.cargoName;

this->cargoWeight=obj.cargoWeight;

}

~Cargo(){} //деструктор

void setName(const QString&); //установить имя

void setSurname(const QString&); //установить фамиоию

void setNumber(const QString&); //установить контактные данные

void setCargoName(const QString&); //установить название груза

void setCargoWeight(float); //установить вес груза

QString getName(); //получить имя

QString getSurname(); //получить фамилию

QString getNumber(); //получить контактные данные

QString getCargoName(); //получить название груза

float getCargoWeight(); //получить вес груза

Cargo& operator=(const Cargo&); //оператор =

friend QTextStream& operator<<(QTextStream&, const Cargo&);//оператор записи в файл

friend QTextStream& operator>>(QTextStream&, Cargo&);//оператор чтения с файла

};

#endif // CARGO\_H

Файл cargo.cpp содержащий реализацию класса Cargo

#include "cargo.h"

void Cargo::setName(const QString& name)

{

this->name=name;

}

void Cargo::setSurname(const QString& surname)

{

this->surname=surname;

}

void Cargo::setNumber(const QString& number)

{

this->number=number;

}

void Cargo::setCargoName(const QString& cargoName)

{

this->cargoName=cargoName;

}

void Cargo::setCargoWeight(float cargoWeight)

{

this->cargoWeight=cargoWeight;

}

QString Cargo::getName()

{

return this->name;

}

QString Cargo::getSurname()

{

return this->surname;

}

QString Cargo::getNumber()

{

return this->number;

}

QString Cargo::getCargoName()

{

return this->cargoName;

}

float Cargo::getCargoWeight()

{

return this->cargoWeight;

}

Cargo& Cargo::operator=(const Cargo& obj)

{

if(this != &obj)

{

this->name=obj.name;

this->surname=obj.surname;

this->number=obj.number;

this->cargoName=obj.cargoName;

this->cargoWeight=obj.cargoWeight;

}

return \*this;

}

QTextStream& operator<<(QTextStream& out, const Cargo& obj)

{

out << obj.surname << "\n" << obj.name << "\n" << obj.number << "\n" << obj.cargoName << "\n" << obj.cargoWeight <<"\n";

return out;

}

QTextStream& operator>>(QTextStream& in, Cargo& obj)

{

obj.surname = in.readLine();

obj.name = in.readLine();

obj.number = in.readLine();

obj.cargoName = in.readLine();

obj.cargoWeight = in.readLine().toFloat();

return in;

}

Файл deque.h содержащий объявление класса Deque

#ifndef \_DEQUE\_H\_

#define \_DEQUE\_H\_

template <typename T>

struct Node //структура узла

{

T data;

Node<T>\* next;

Node<T>\* prev;

Node(const T& value)

{

this->data=value;

this->next=nullptr;

this->prev=nullptr;

}

};

template <typename T>

class Deque //класс двунаправленной очереди

{

private:

Node<T>\* head; //первый элемент

Node<T>\* tail; //последний элемент

int size; //размер

public:

Deque() //конструктор

{

this->head = nullptr;

this->tail = nullptr;

this->size = 0;

}

Deque(const Deque<T>& obj) //конструктор копирования

{

this->head = nullptr;

this->tail = nullptr;

this->size = 0;

for(auto iter = obj.cbegin(); iter != obj.cend(); iter++)

{

this->pushBack(\*iter);

}

}

~Deque() //деструктор

{

clear();

}

class Iterator //вложенный класс итератор

{

private:

Node<T>\* curr; //текущий узел

public:

Iterator(Node<T>\* node) : curr(node){} //конструктор

~Iterator() = default; //деструктор

Iterator& operator++(int); //оператор ++

Iterator& operator--(int); //оператор --

Iterator& operator+=(int); //оператор +=

T& operator\*(); //оператор \*

bool operator==(const Iterator&); //оператор ==

bool operator!=(const Iterator&); //оператор !=

};

Iterator begin(); //установить итератор на начало очереди

Iterator end(); //установить итератор на конец очереди

Iterator cbegin()const; //установить итератор на начало очереди

Iterator cend()const; //установить итератор на конец очереди

Iterator last(); //установить итератор на последний элемент очереди

bool empty(); //проверка на пустоту

int getSize(); //получение размера

void pushFront(const T&); //добавить в начало

void pushBack(const T&); //добавить в конец

void popFront(); //удалить с начала

void popBack(); //удалить с конца

void clear(); //очистить

void removeAt(int); //удалить по индексу

Deque<T>& operator=(const Deque<T>&); //оператор =

};

#endif

Файл deque.cpp содержащий реализацию класса deque

#ifndef DEQUE\_CPP

#define DEQUE\_CPP

#include "deque.h"

#include "contexception.h"

template <typename T>

bool Deque<T>::empty()

{

return head == nullptr;

}

template<typename T>

int Deque<T>::getSize()

{

return this->size;

}

template <typename T>

void Deque<T>::pushFront(const T& data)

{

Node<T>\* newNode = new Node<T>(data);

if(newNode==nullptr) throw ContException("Ошибка выделения памяти");

if (empty())

{

head = tail = newNode;

}

else

{

newNode->next = head;

head->prev = newNode;

head = newNode;

}

size++;

}

template <typename T>

void Deque<T>::pushBack(const T& data)

{

Node<T>\* newNode = new Node<T>(data);

if(newNode==nullptr) throw ContException("Ошибка выделения памяти");

if (empty())

{

head = tail = newNode;

}

else

{

newNode->prev = tail;

tail->next = newNode;

tail = newNode;

}

size++;

}

template <typename T>

void Deque<T>::popFront()

{

if (!empty())

{

Node<T>\* temp = head;

if (head == tail)

{

head = tail = nullptr;

}

else

{

head = head->next;

head->prev = nullptr;

}

delete temp;

size--;

}

}

template <typename T>

void Deque<T>::popBack()

{

if (!empty())

{

Node<T>\* temp = tail;

if (head == tail)

{

head = tail = nullptr;

}

else

{

tail = tail->prev;

tail->next = nullptr;

}

delete temp;

size--;

}

}

template<typename T>

void Deque<T>::clear()

{

while (!empty())

popFront();

}

template<typename T>

void Deque<T>::removeAt(int index)

{

if (index < 0 || index >= size)

{

throw ContException("Недопустимый индекс для удаления");

}

Node<T>\* temp = head;

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

{

temp=temp->next;

}

if(temp==head)

{

popFront();

}

else if(temp==tail)

{

popBack();

}

else

{

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

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

delete temp;

size--;

}

}

template<typename T>

typename Deque<T>::Iterator& Deque<T>::Iterator::operator++(int)

{

curr=curr->next;

return \*this;

}

template<typename T>

typename Deque<T>::Iterator& Deque<T>::Iterator::operator--(int)

{

curr=curr->prev;

return \*this;

}

template<typename T>

typename Deque<T>::Iterator& Deque<T>::Iterator::operator+=(int count)

{

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

{

curr=curr->next;

}

return \*this;

}

template<typename T>

T& Deque<T>::Iterator::operator\*()

{

return curr->data;

}

template<typename T>

bool Deque<T>::Iterator::operator==(const Deque<T>::Iterator& other)

{

return curr == other.curr;

}

template<typename T>

bool Deque<T>::Iterator::operator!=(const Deque<T>::Iterator& other)

{

return !(\*this == other);

}

template<typename T>

typename Deque<T>::Iterator Deque<T>::begin()

{

return Iterator(head);

}

template<typename T>

typename Deque<T>::Iterator Deque<T>::end()

{

return Iterator(nullptr);

}

template<typename T>

typename Deque<T>::Iterator Deque<T>::cbegin() const

{

return Iterator(head);

}

template<typename T>

typename Deque<T>::Iterator Deque<T>::cend() const

{

return Iterator(nullptr);

}

template<typename T>

typename Deque<T>::Iterator Deque<T>::last()

{

return Iterator(tail);

}

template<typename T>

Deque<T>& Deque<T>::operator=(const Deque<T>& obj)

{

if(this!= &obj)

{

this->clear();

for(auto iter = obj.cbegin(); iter != obj.cend(); iter++)

{

this->pushBack(\*iter);

}

}

return \*this;

}

#endif //DEQUE\_CPP

Файл algorithm.h содержащий объявление класса Algorithm

#ifndef ALGORITHM\_H

#define ALGORITHM\_H

#include "deque.cpp"

#include <QString>

#include <QTextStream>

#include <QFile>

template <typename T>

class Algorithm //класс алгоритм

{

public:

static int saveDeque(Deque<T>\*,const QString&); //сохранение контейнера в файл

static int loadDeque(Deque<T>\*,const QString&); //чтение в контейнер из файла

static void bubbleSort(Deque<T>\*); //сортировка контейнера

static Deque<T>\* search(typename Deque<T>::Iterator&,typename Deque<T>::Iterator&, const T&); //фильтрация контейнера

};

#endif // ALGORITHM\_H

Файл algorithm.cpp содержащий реализацию класса algorithm

#include "algorithm.h"

#include "fileexception.h"

template <typename T>

static void bubbleSort(Deque<T>& deque)

{

int n = deque.getSize();

if (n <= 1)

return;

typename Deque<T>::Iterator outer, inner, end, last;

end = deque.end();

last = deque.last();

for (outer = deque.begin(); outer != end; outer++)

{

for (inner = deque.begin(); inner != last; inner++)

{

typename Deque<T>::Iterator next = inner;

next++;

if (\*inner > \*next)

{

T temp = \*inner;

\*inner = \*next;

\*next = temp;

}

}

}

}

template <typename T>

static Deque<T>\* search(typename Deque<T>::Iterator& begin,typename Deque<T>::Iterator& end, const T& value)

{

Deque<T>\* temp = new Deque<T>;

for (auto iter = begin(); iter != end; iter++)

{

if (\*iter == value)

temp->pushBack(\*iter);

}

return temp;

}

template<typename T>

int Algorithm<T>::saveDeque(Deque<T>\* deque,const QString& filename)

{

QFile file(filename);

if (!file.open(QIODevice::WriteOnly | QIODevice::Text))

return -1;

QTextStream out(&file);

for (auto iter = deque->begin(); iter != deque->end(); iter++) {

out << \*iter;

}

if (out.status() != QTextStream::Ok) {

throw FileException("Ошибка записи в файл");

}

file.close();

return 1;

}

template<typename T>

int Algorithm<T>::loadDeque(Deque<T>\* deque,const QString& filename)

{

QFile file(filename);

if (!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

return -1;

}

QTextStream in(&file);

while(!in.atEnd())

{

T obj;

in >> obj;

deque->pushBack(obj);

}

if(in.status() != QTextStream::Ok){

throw FileException("Ошибка чтения файла");

}

file.close();

return 1;

}

Файл file.h содержащий объявление класса File

#ifndef FILE\_H

#define FILE\_H

#include "deque.cpp"

#include <List>

#include <QTextStream>

#include <QFile>

#include "passengerDepartures.h"

#include "cargodepartures.h"

class File

{

public:

//запись списка пассажиров в файл

static int savePassengers(Deque<PassengerDepartures>\*,const QString&);

//запись списка грузов в файл

static int saveCargo(Deque<CargoDepartures>\*,const QString&);

//чтение в список пассажиров из файла

static int loadPassengers(Deque<PassengerDepartures>\*,const QString&);

//чтение в список грузов из файла

static int loadCargo(Deque<CargoDepartures>\*,const QString&);

};

#endif // FILE\_H

Файл file.cpp содержащий реализацию класса file

#include "file.h"

#include "fileexception.h"

int File::savePassengers(Deque<PassengerDepartures>\* deque,const QString& filename)

{

QFile file(filename);

if (!file.open(QIODevice::WriteOnly | QIODevice::Text))

return -1;

QTextStream out(&file);

for (auto iter = deque->begin(); iter != deque->end(); iter++) {

for(const auto &item : (\*iter).getPassengerList())

{

out << item;

}

}

if (out.status() != QTextStream::Ok) {

throw FileException("Ошибка записи в файл");

}

file.close();

return 1;

}

int File::saveCargo(Deque<CargoDepartures>\* deque,const QString& filename)

{

QFile file(filename);

if (!file.open(QIODevice::WriteOnly | QIODevice::Text))

return -1;

QTextStream out(&file);

for (auto iter = deque->begin(); iter != deque->end(); iter++) {

for(const auto &item : (\*iter).getCargoList())

{

out << item;

}

}

if (out.status() != QTextStream::Ok) {

throw FileException("Ошибка записи в файл");

}

file.close();

return 1;

}

int File::loadPassengers(Deque<PassengerDepartures>\* deque,const QString& filename)

{

QFile file(filename);

if (!file.open(QIODevice::ReadOnly | QIODevice::Text))

return -1;

QTextStream in(&file);

for (auto iter = deque->begin(); iter != deque->end(); iter++) {

for(int i = 0; i< (\*iter).getPassengersCount();i++)

{

Passenger obj;

in >> obj;

(\*iter).getPassengerList().push\_back(obj);

}

}

if (in.status() != QTextStream::Ok) {

throw FileException("Ошибка чтения файла");

}

file.close();

return 1;

}

int File::loadCargo(Deque<CargoDepartures>\* deque,const QString& filename)

{

QFile file(filename);

if (!file.open(QIODevice::ReadOnly | QIODevice::Text))

return -1;

QTextStream in(&file);

for (auto iter = deque->begin(); iter != deque->end(); iter++) {

for(int i = 0; i< (\*iter).getCargoCount();i++)

{

Cargo obj;

in >> obj;

(\*iter).getCargoList().push\_back(obj);

}

}

if (in.status() != QTextStream::Ok) {

throw FileException("Ошибка чтения файла");

}

file.close();

return 1;

}

Файл exception.h содержащий реализацию класса Exception

#ifndef EXCEPTION\_H

#define EXCEPTION\_H

#include <QMessageBox>

class Exception

{

private:

int code; //код ошибки

QString msg; //текст ошибки

public:

Exception(int code, const QString& msg) : code(code),msg(msg){} //конструктор

void show() //вывод ошибки

{

QMessageBox::warning(nullptr,"Ошибка",msg+"\nКод ошибки:"+QString::number(code));

}

};

#endif // EXCEPTION\_H

Файл inputxception.h содержащий реализацию класса InputException

#ifndef INPUTEXCEPTION\_H

#define INPUTEXCEPTION\_H

#include "exception.h"

class InputException : public Exception

{

public:

InputException(const QString& msg) : Exception(1,msg){}

};

#endif // INPUTEXCEPTION\_H

Файл fileexception.h содержащий реализацию класса FileException

#ifndef FILEEXCEPTION\_H

#define FILEEXCEPTION\_H

#include "exception.h"

class FileException : public Exception

{

public:

FileException(const QString& msg) : Exception(2,msg){}

};

#endif // FILEEXCEPTION\_H

Файл contexexception.h содержащий реализацию класса ContException

#ifndef CONTEXCEPTION\_H

#define CONTEXCEPTION\_H

#include "exception.h"

class ContException : public Exception

{

public:

ContException(const QString& msg) : Exception(3,msg){}

};

#endif // CONTEXCEPTION\_H

Файл otherexception.h содержащий реализацию класса OtherException

#ifndef OTHEREXCEPTION\_H

#define OTHEREXCEPTION\_H

#include "exception.h"

class OtherException : public Exception

{

public:

OtherException(const QString& msg) : Exception(4,msg){}

};

#endif // OTHEREXCEPTION\_H

Файл checker.h содержащий объявления функций проверки данных

#ifndef CHECKER\_H

#define CHECKER\_H

#include "inputexception.h"

#include <QTableWidget>

#include <QDateTime>

#include "passengerDepartures.h"

#include "cargodepartures.h"

void checkPassDep(QTableWidget\* tableWidget); //проверка таблицы пассажирских вылетов

void checkPassArr(QTableWidget\* tableWidget); //проверка таблицы пассажирских прилетов

void checkCargoDep(QTableWidget\* tableWidget); //проверка таблицы грузовых вылетов

void checkCargoArr(QTableWidget\* tableWidget); //проверка таблицы грузовых прилетов

void checkPassenger(QTableWidget\* tableWidget,int row); //проверка таблицы пассажиров

void checkCargo(QTableWidget\* tableWidget,int row); //проверка таблицы грузов

//проверка на наличие свободных мест

void checkSpace(QTableWidget\* tableWidget,int row, PassengerDepartures&);

//проверка на наличие свободного веса

void checkWeight(QTableWidget\* tableWidget,int row, CargoDepartures&);

//проверка на правильность ввода грузоподъмности

void checkWeight(QTableWidget\* tableWidget, int row);

#endif // CHECKER\_H

Файл checker.cpp содержащий реализацию функций проверки данных

#include "checker.h"

void checkPassDep(QTableWidget\* tableWidget)

{

int i = 0;

bool isOk;

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите название самолета");

}

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите название авиакомпании");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите максимальное кол-во мест эконом-класса");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите максимальное кол-во мест эконом-класса");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите максимальное кол-во мест бизнес-класса");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите максимальное кол-во мест бизнес-класса");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите максимальное кол-во мест первого класса");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите максимальное кол-во мест первого класса");

}

if(!tableWidget->item(i,0)||!(QDate::fromString(tableWidget->item(i++,0)->text(), "dd/MM/yyyy")).isValid())

{

throw InputException("Введите дату в формате дд/мм/гггг");

}

if(!tableWidget->item(i,0)||!(QTime::fromString(tableWidget->item(i,0)->text(), "HH:mm")).isValid())

{

throw InputException("Введите время в формате чч:мм");

}

QDateTime inputDateTime(QDate::fromString(tableWidget->item(i-1,0)->text(), "dd/MM/yyyy"),QTime::fromString(tableWidget->item(i,0)->text(), "HH:mm"));

if(inputDateTime < QDateTime::currentDateTime())

{

throw InputException("Время введено раньше текущего времени");

}

i++;

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите направление полета");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите время полета");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите время полета");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите стоимость билета эконом-класса");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите стоимость билета эконом-класса");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите стоимость билета бизнес-класса");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите стоимость билета бизнес-класса");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите стоимость билета первого класса");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите стоимость билета первого класса");

}

}

void checkPassArr(QTableWidget\* tableWidget)

{

int i = 0;

bool isOk;

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите название самолета");

}

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите название авиакомпании");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите максимальное кол-во мест эконом-класса");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите максимальное кол-во мест эконом-класса");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите максимальное кол-во мест бизнес-класса");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите максимальное кол-во мест бизнес-класса");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите максимальное кол-во мест первого класса");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите максимальное кол-во мест первого класса");

}

if(!tableWidget->item(i,0)||!(QDate::fromString(tableWidget->item(i++,0)->text(), "dd/MM/yyyy")).isValid())

{

throw InputException("Введите дату в формате дд/мм/гггг");

}

if(!tableWidget->item(i,0)||!(QTime::fromString(tableWidget->item(i,0)->text(), "HH:mm")).isValid())

{

throw InputException("Введите время в формате чч:мм");

}

QDateTime inputDateTime(QDate::fromString(tableWidget->item(i-1,0)->text(), "dd/MM/yyyy"),QTime::fromString(tableWidget->item(i,0)->text(), "HH:mm"));

if(inputDateTime < QDateTime::currentDateTime())

{

throw InputException("Время введено раньше текущего времени");

}

i++;

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите местоположение начала полета");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите время разгрузки багажа");

}

else

{

tableWidget->item(i,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите время разгрузки багажа");

}

}

void checkCargoDep(QTableWidget\* tableWidget)

{

int i = 0;

bool isOk;

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите название самолета");

}

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите название авиакомпании");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите грузоподъемность самолета");

}

else

{

tableWidget->item(i++,0)->text().toFloat(&isOk);

if(!isOk) throw InputException("Введите грузоподъемность самолета");

}

if(!tableWidget->item(i,0)||!(QDate::fromString(tableWidget->item(i++,0)->text(), "dd/MM/yyyy")).isValid())

{

throw InputException("Введите дату в формате дд/мм/гггг");

}

if(!tableWidget->item(i,0)||!(QTime::fromString(tableWidget->item(i,0)->text(), "HH:mm")).isValid())

{

throw InputException("Введите время в формате чч:мм");

}

QDateTime inputDateTime(QDate::fromString(tableWidget->item(i-1,0)->text(), "dd/MM/yyyy"),QTime::fromString(tableWidget->item(i,0)->text(), "HH:mm"));

if(inputDateTime < QDateTime::currentDateTime())

{

throw InputException("Время введено раньше текущего времени");

}

i++;

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите направление полета");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите время полета");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите время полета");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите стоимость транспортировки груза");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите стоимость транспортировки груза");

}

}

void checkCargoArr(QTableWidget\* tableWidget)

{

int i = 0;

bool isOk;

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите название самолета");

}

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите название авиакомпании");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите грузоподъемность самолета");

}

else

{

tableWidget->item(i++,0)->text().toFloat(&isOk);

if(!isOk) throw InputException("Введите грузоподъемность самолета");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите вес груза");

}

else

{

tableWidget->item(i++,0)->text().toFloat(&isOk);

if(!isOk) throw InputException("Введите вес груза");

}

if(!tableWidget->item(i,0)||!(QDate::fromString(tableWidget->item(i++,0)->text(), "dd/MM/yyyy")).isValid())

{

throw InputException("Введите дату в формате дд/мм/гггг");

}

if(!tableWidget->item(i,0)||!(QTime::fromString(tableWidget->item(i,0)->text(), "HH:mm")).isValid())

{

throw InputException("Введите время в формате чч:мм");

}

QDateTime inputDateTime(QDate::fromString(tableWidget->item(i-1,0)->text(), "dd/MM/yyyy"),QTime::fromString(tableWidget->item(i,0)->text(), "HH:mm"));

if(inputDateTime < QDateTime::currentDateTime())

{

throw InputException("Время введено раньше текущего времени");

}

i++;

if(!tableWidget->item(i,0)||tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите местоположение начала полета");

}

if(!tableWidget->item(i,0))

{

throw InputException("Введите время разгрузки");

}

else

{

tableWidget->item(i++,0)->text().toInt(&isOk);

if(!isOk) throw InputException("Введите время разгрузки");

}

}

void checkPassenger(QTableWidget\* tableWidget, int row)

{

if(!tableWidget->item(row,0)||tableWidget->item(row,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите фамилию");

}

if(!tableWidget->item(row,1)||tableWidget->item(row,1)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите имя");

}

if(!tableWidget->item(row,2)||tableWidget->item(row,2)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите контактные данные");

}

}

void checkCargo(QTableWidget\* tableWidget, int row)

{

if(!tableWidget->item(row,0)||tableWidget->item(row,0)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите фамилию");

}

if(!tableWidget->item(row,1)||tableWidget->item(row,1)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите имя");

}

if(!tableWidget->item(row,2)||tableWidget->item(row,2)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите контактные данные");

}

if(!tableWidget->item(row,3)||tableWidget->item(row,3)-> data(Qt::DisplayRole).toString().trimmed().isEmpty())

{

throw InputException("Введите контактные данные");

}

if(!tableWidget->item(row,4))

{

throw InputException("Введите вес груза");

}

else

{

bool isOk;

tableWidget->item(row,4)->text().toFloat(&isOk);

if(!isOk) throw InputException("Введите вес груза");

}

}

void checkSpace(QTableWidget\* tableWidget,int row, PassengerDepartures& passdep)

{

if(tableWidget->item(row++,0)->text().toInt()<passdep.getEconom())

throw InputException("Мест эконом-класса забронировано больше, чем указано максимальное кол-во мест");

if(tableWidget->item(row++,0)->text().toInt()<passdep.getBusiness())

throw InputException("Мест бизнес-класса забронировано больше, чем указано максимальное кол-во мест");

if(tableWidget->item(row++,0)->text().toInt()<passdep.getFirst())

throw InputException("Мест первого класса забронировано больше, чем указано максимальное кол-во мест");

}

void checkWeight(QTableWidget\* tableWidget,int row,CargoDepartures& cargodep)

{

if(tableWidget->item(row++,0)->text().toFloat()<cargodep.getWeight())

throw InputException("Вес груза больше чем указанная грузоподъемность");

}

void checkWeight(QTableWidget\* tableWidget,int row)

{

if(tableWidget->item(row,0)->text().toFloat()<tableWidget->item(row+1,0)-> text().toFloat())

throw InputException("Вес груза больше чем указанная грузоподъемность");

}

Файл userinfo.h содержащий объявление класса Userinfo

#ifndef USERINFO\_H

#define USERINFO\_H

#include "passengerArrivals.h"

#include "passengerDepartures.h"

#include "cargodepartures.h"

#include "cargoarrivals.h"

#include "addflightwindow.h"

#include "editflightwindow.h"

#include <QMainWindow>

#include <list>

#include <stack>

#include <QFile>

#include "file.h"

#include "algorithm.cpp"

#include "otherexception.h"

namespace Ui {

class Userinfo;

}

class Userinfo : public QmainWindow //класс окна расписания

{

Q\_OBJECT

public:

explicit Userinfo(QWidget \*parent = nullptr); //конструктор

~Userinfo(); //деструктор

void save(); //сохранение всех данных

void load(); //загрузка всех данных

void checkTime(); //проверка времени

private slots:

void on\_passDepButton\_clicked(); //кнопка пассажирских вылетов

void on\_passArrButton\_clicked(); //кнопка пассажирских прилетов

void on\_cargoDepButton\_clicked(); //кнопка пассажирских вылетов

void on\_cargoArrButton\_clicked(); //кнопка пассажирских прилетов

void on\_addFlightButton\_clicked(); //кнопка добавления полета

void on\_deleteFlightButton\_clicked(); //кнопка удаления полета

void on\_editButton\_clicked(); //кнопка изменения полета

void on\_undoButton\_clicked(); //кнопка отмены последнего действия

private:

Ui::Userinfo \*ui; //указатель на интерфейс

AddFlightWindow\* addflightwindow; //окно добавления

EditFlightWindow\* editflightwindow; //окно изменения

Deque<PassengerDepartures>\* passdep; //пассажирский вылеты

Deque<PassengerArrivals>\* passarr; //пассажирские прилеты

Deque<CargoDepartures>\* cargodep; //грузовые вылеты

Deque<CargoArrivals>\* cargoarr; //грузовые прилеты

//стек пассажирских вылетов

std::stack<Deque<PassengerDepartures>>\* passdepStack;

//стек пассажирских прилетов

std::stack<Deque<PassengerArrivals>>\* passarrStack;

//стек грузовых вылетов

std::stack< Deque<CargoDepartures>>\* cargodepStack;

//стек грузовых прилетов

std::stack< Deque<CargoArrivals>>\* cargoarrStack;

//стек типов рейса

std::stack<int>\* typeStack;

//тип рейса

int type;

};

#endif // USERINFO\_H

Файл userinfo.cpp содержащий реализацию класса Userinfo

#include "userinfo.h"

#include "ui\_userinfo.h"

Userinfo::Userinfo(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::Userinfo)

{

ui->setupUi(this);

ui->tableWidget->hideColumn(0);

this->passarr = new Deque<PassengerArrivals>;

this->passdep = new Deque<PassengerDepartures>;

this->cargodep = new Deque<CargoDepartures>;

this->cargoarr = new Deque<CargoArrivals>;

this->passdepStack = new std::stack<Deque<PassengerDepartures>>;

this->passarrStack = new std::stack<Deque<PassengerArrivals>>;

this->cargodepStack = new std::stack<Deque<CargoDepartures>>;

this->cargoarrStack = new std::stack<Deque<CargoArrivals>>;

this->typeStack = new std::stack<int>;

this->on\_passDepButton\_clicked();

this->addflightwindow = new AddFlightWindow(passdep,passarr,cargodep,cargoarr,passdepStack,passarrStack,cargodepStack,cargoarrStack,typeStack);

this->editflightwindow = new EditFlightWindow(passdep,passarr,cargodep,cargoarr,passdepStack,passarrStack,cargodepStack,cargoarrStack,typeStack);

this->load();

}

Userinfo::~Userinfo()

{

this->save();

this->addflightwindow->close();

this->editflightwindow->close();

delete this->passarr;

delete this->passdep;

delete this->cargoarr;

delete this->cargodep;

delete this->addflightwindow;

delete this->editflightwindow;

delete this->passdepStack;

delete this->passarrStack;

delete this->cargodepStack;

delete this->cargoarrStack;

delete this->typeStack;

delete ui;

}

void Userinfo::save()

{

Algorithm<PassengerDepartures>::saveDeque(passdep,"passengerdepartures.txt");

Algorithm<PassengerArrivals>::saveDeque(passarr,"passengerarrivals.txt");

Algorithm<CargoDepartures>::saveDeque(cargodep,"cargodepartures.txt");

Algorithm<CargoArrivals>::saveDeque(cargoarr,"cargoarrivals.txt");

File::savePassengers(passdep,"passengers.txt");

File::saveCargo(cargodep,"cargo.txt");

}

void Userinfo::load()

{

Algorithm<PassengerDepartures>::loadDeque(passdep,"passengerdepartures.txt");

Algorithm<PassengerArrivals>::loadDeque(passarr,"passengerarrivals.txt");

Algorithm<CargoDepartures>::loadDeque(cargodep,"cargodepartures.txt");

Algorithm<CargoArrivals>::loadDeque(cargoarr,"cargoarrivals.txt");

File::loadPassengers(passdep,"passengers.txt");

File::loadCargo(cargodep,"cargo.txt");

}

void Userinfo::on\_passDepButton\_clicked()

{

checkTime();

ui->tableWidget->setSortingEnabled(false);

ui->tableWidget->clear();

ui->tableWidget->setColumnCount(7);

ui->tableWidget->setRowCount(0);

ui->tableWidget->setHorizontalHeaderLabels(QStringList()<<"id"<<"Авиакомпания" <<"Самолет"<<"Направление" <<"Дата вылета"<<"Время вылета"<<"Свободные места");

auto iter = passdep->begin();

for(int i=0;i<passdep->getSize();i++,iter++)

{

ui->tableWidget->setRowCount(ui->tableWidget->rowCount()+1);

ui->tableWidget->setItem(i,0,new QTableWidgetItem(QString::number(i)));

ui->tableWidget->setItem(i,1,new QTableWidgetItem((\*iter).getCompany()));

ui->tableWidget->setItem(i,2,new QTableWidgetItem((\*iter).getModel()));

ui->tableWidget->setItem(i,3,new QTableWidgetItem((\*iter).getLocation()));

ui->tableWidget->setItem(i,4,new QTableWidgetItem((\*iter).getDate().toString("dd.MM.yyyy")));

ui->tableWidget->setItem(i,5,new QTableWidgetItem((\*iter).getDate().toString("hh:mm")));

if((\*iter).isFool())

{

ui->tableWidget->setItem(i,6,new QTableWidgetItem("Свободных мест нет"));

}

else

{

ui->tableWidget->setItem(i,6,new QTableWidgetItem(QString::number((\*iter).getMaxEconom()+(\*iter).getMaxBusiness()+

(\*iter).getMaxFirst()-(\*iter).getPassengersCount())));

}

}

ui->tableWidget->setSortingEnabled(true);

this->type=0;

}

void Userinfo::on\_passArrButton\_clicked()

{

checkTime();

ui->tableWidget->setSortingEnabled(false);

ui->tableWidget->clear();

ui->tableWidget->setColumnCount(7);

ui->tableWidget->setRowCount(0);

ui->tableWidget->setHorizontalHeaderLabels(QStringList()<<"id"<<"Авиакомпания" <<"Самолет"<<"Прибывает из"<<"Дата прилета"<<"Время прилета"<<"Статус");

auto iter = passarr->begin();

for(int i=0;i<passarr->getSize();i++,iter++)

{

ui->tableWidget->setRowCount(ui->tableWidget->rowCount()+1);

ui->tableWidget->setItem(i,0,new QTableWidgetItem(QString::number(i)));

ui->tableWidget->setItem(i,1,new QTableWidgetItem((\*iter).getCompany()));

ui->tableWidget->setItem(i,2,new QTableWidgetItem((\*iter).getModel()));

ui->tableWidget->setItem(i,3,new QTableWidgetItem((\*iter).getLocation()));

ui->tableWidget->setItem(i,4,new QTableWidgetItem((\*iter).getDate().toString("dd.MM.yyyy")));

ui->tableWidget->setItem(i,5,new QTableWidgetItem((\*iter).getDate().toString("hh:mm")));

if((\*iter).isDelayed())

{

ui->tableWidget->setItem(i,6,new QTableWidgetItem("Задерживается"));

}

}

ui->tableWidget->setSortingEnabled(true);

this->type=1;

}

void Userinfo::on\_cargoDepButton\_clicked()

{

checkTime();

ui->tableWidget->setSortingEnabled(false);

ui->tableWidget->clear();

ui->tableWidget->setColumnCount(7);

ui->tableWidget->setRowCount(0);

ui->tableWidget->setHorizontalHeaderLabels(QStringList()<<"id"<<"Авиакомпания" <<"Самолет"<<"Направление"<<"Дата вылета"<<"Время вылета"<<"Доступный груз, кг");

auto iter = cargodep->begin();

for(int i=0;i<cargodep->getSize();i++,iter++)

{

ui->tableWidget->setRowCount(ui->tableWidget->rowCount()+1);

ui->tableWidget->setItem(i,0,new QTableWidgetItem(QString::number(i)));

ui->tableWidget->setItem(i,1,new QTableWidgetItem((\*iter).getCompany()));

ui->tableWidget->setItem(i,2,new QTableWidgetItem((\*iter).getModel()));

ui->tableWidget->setItem(i,3,new QTableWidgetItem((\*iter).getLocation()));

ui->tableWidget->setItem(i,4,new QTableWidgetItem((\*iter).getDate().toString("dd.MM.yyyy")));

ui->tableWidget->setItem(i,5,new QTableWidgetItem((\*iter).getDate().toString("hh:mm")));

ui->tableWidget->setItem(i,6,new QTableWidgetItem(QString::number((\*iter).getFreeWeight())));

}

ui->tableWidget->setSortingEnabled(true);

this->type=2;

}

void Userinfo::on\_cargoArrButton\_clicked()

{

checkTime();

ui->tableWidget->setSortingEnabled(false);

ui->tableWidget->clear();

ui->tableWidget->setColumnCount(7);

ui->tableWidget->setRowCount(0);

ui->tableWidget->setHorizontalHeaderLabels(QStringList()<<"id" <<"Авиакомпания"<<"Самолет"<<"Прибывает из"<<"Дата прилета"<<"Время прилета"<<"Статус");

auto iter = cargoarr->begin();

for(int i=0;i<cargoarr->getSize();i++,iter++)

{

ui->tableWidget->setRowCount(ui->tableWidget->rowCount()+1);

ui->tableWidget->setItem(i,0,new QTableWidgetItem(QString::number(i)));

ui->tableWidget->setItem(i,1,new QTableWidgetItem((\*iter).getCompany()));

ui->tableWidget->setItem(i,2,new QTableWidgetItem((\*iter).getModel()));

ui->tableWidget->setItem(i,3,new QTableWidgetItem((\*iter).getLocation()));

ui->tableWidget->setItem(i,4,new QTableWidgetItem((\*iter).getDate().toString("dd.MM.yyyy")));

ui->tableWidget->setItem(i,5,new QTableWidgetItem((\*iter).getDate().toString("hh:mm")));

if((\*iter).isDelayed())

{

ui->tableWidget->setItem(i,6,new QTableWidgetItem("Задерживается"));

}

}

ui->tableWidget->setSortingEnabled(true);

this->type=3;

}

void Userinfo::on\_addFlightButton\_clicked()

{

addflightwindow->close();

editflightwindow->close();

addflightwindow->setType(this->type);

addflightwindow->setTable();

addflightwindow->show();

}

void Userinfo::on\_deleteFlightButton\_clicked()

{

addflightwindow->close();

editflightwindow->close();

int i = ui->tableWidget->currentRow();

if(i >=0 )

{

switch(type)

{

case(0):

{

passdepStack->push(\*passdep);

typeStack->push(type);

passdep->removeAt(ui->tableWidget->item(i,0)->text().toInt());

on\_passDepButton\_clicked();

return;

}

case(1):

{

passarrStack->push(\*passarr);

typeStack->push(type);

passarr->removeAt(ui->tableWidget->item(i,0)->text().toInt());

on\_passArrButton\_clicked();

return;

}

case(2):

{

cargodepStack->push(\*cargodep);

typeStack->push(type);

cargodep->removeAt(ui->tableWidget->item(i,0)->text().toInt());

on\_cargoDepButton\_clicked();

return;

}

case(3):

{

cargoarrStack->push(\*cargoarr);

typeStack->push(type);

cargoarr->removeAt(ui->tableWidget->item(i,0)->text().toInt());

on\_cargoArrButton\_clicked();

return;

}

}

}

}

void Userinfo::on\_editButton\_clicked()

{

addflightwindow->close();

editflightwindow->close();

int i = ui->tableWidget->currentRow();

if(i >=0 )

{

editflightwindow->setValue(type,ui->tableWidget->item(i,0)->text().toInt());

editflightwindow->show();

editflightwindow->setTable();

}

}

void Userinfo::on\_undoButton\_clicked()

{

addflightwindow->close();

editflightwindow->close();

try {

if(typeStack->empty()) throw OtherException("Отмена невозможна");

int stackType = typeStack->top();

typeStack->pop();

switch(stackType)

{

case(0):

{

(\*passdep) = passdepStack->top();

passdepStack->pop();

break;

}

case(1):

{

\*passarr= passarrStack->top();

passarrStack->pop();

break;

}

case(2):

{

\*cargodep = cargodepStack->top();

cargodepStack->pop();

break;

}

case(3):

{

\*cargoarr = cargoarrStack->top();

cargoarrStack->pop();

break;

}

}

} catch (OtherException& exp) {

exp.show();

return;

}

}

void Userinfo::checkTime()

{

int i = 0;

for(auto iter = passdep->begin();iter !=passdep->end(); iter++,i++)

{

if((\*iter).getDate()<QDateTime::currentDateTime())

passdep->removeAt(i);

}

for(auto iter = passarr->begin();iter !=passarr->end(); iter++)

{

if((\*iter).getDate()<QDateTime::currentDateTime())

(\*iter).setDelay(true);

}

i=0;

for(auto iter = cargodep->begin();iter !=cargodep->end(); iter++,i++)

{

if((\*iter).getDate()<QDateTime::currentDateTime())

cargodep->removeAt(i);

}

for(auto iter = cargoarr->begin();iter !=cargoarr->end(); iter++)

{

if((\*iter).getDate()<QDateTime::currentDateTime())

(\*iter).setDelay(true);

}

}

Файл addflightwindow.h содержащий объявление класса AddFlightWindow

#ifndef ADDFLIGHTWINDOW\_H

#define ADDFLIGHTWINDOW\_H

#include "deque.cpp"

#include "passengerArrivals.h"

#include "passengerDepartures.h"

#include "cargodepartures.h"

#include "cargoarrivals.h"

#include <QMainWindow>

#include <stack>

namespace Ui {

class AddFlightWindow;

}

class AddFlightWindow : public QmainWindow //класс окна добавления рейса

{

Q\_OBJECT

public:

explicit AddFlightWindow(Deque<PassengerDepartures>\* passdepinfo,Deque<PassengerArrivals>\* passarrinfo,Deque<CargoDepartures>\* cargodepinfo,

Deque<CargoArrivals>\* cargoarrinfo,std::stack<Deque<PassengerDepartures>>\* passdepStack, std::stack<Deque<PassengerArrivals>>\* passarrStack,

std::stack< Deque<CargoDepartures>>\* cargodepStack,std::stack< Deque<CargoArrivals>>\* cargoarrStack,std::stack<int>\* typeStack, QWidget \*parent = nullptr); //конструктор

~AddFlightWindow(); //деструктор

void setType(int); //установить тип рейса

void setPassDepTable(); //отображение таблицы для пассажирских вылетов

void setPassArrTable(); //отображение таблицы для пассажирских прилетов

void setCargoDepTable(); //отображение таблицы для грузовых вылетов

void setCargoArrTable(); //отображение таблицы для грузовых прилетов

void setTable(); //установить тип таблицы

void addPassDep(); //добавление пассажирского вылета

void addPassArr(); //добавление пассажирского прилета

void addCargoDep(); //добавление грузового вылета

void addCargoArr(); //добавление грузового прилета

private slots:

void on\_addButton\_clicked(); //кнопка добавления

void on\_cancelButton\_clicked(); //кнопка отмены

private:

Ui::AddFlightWindow \*ui; //указатель на интерфейс

Deque<PassengerDepartures>\* passdep; //пассажирский вылеты

Deque<PassengerArrivals>\* passarr; //пассажирские прилеты

Deque<CargoDepartures>\* cargodep; //грузовые вылеты

Deque<CargoArrivals>\* cargoarr; //грузовые прилеты

//стек пассажирских вылетов

std::stack<Deque<PassengerDepartures>>\* passdepStack;

//стек пассажирских прилетов

std::stack<Deque<PassengerArrivals>>\* passarrStack;

//стек грузовых вылетов

std::stack< Deque<CargoDepartures>>\* cargodepStack;

//стек грузовых прилетов

std::stack< Deque<CargoArrivals>>\* cargoarrStack;

std::stack<int>\* typeStack; //стек типов рейса

int type; //тип рейса

};

#endif // ADDFLIGHTWINDOW\_H

Файл addflight.cpp содержащий реализацию класса AddFlightWindow

#include "addflightwindow.h"

#include "ui\_addflightwindow.h"

#include <QCheckBox>

#include "checker.h"

AddFlightWindow::AddFlightWindow(Deque<PassengerDepartures>\* passdepinfo,Deque<PassengerArrivals>\* passarrinfo,Deque<CargoDepartures>\* cargodepinfo,

Deque<CargoArrivals>\* cargoarrinfo,std::stack<Deque<PassengerDepartures>>\* passdepStack, std::stack<Deque<PassengerArrivals>>\* passarrStack,

std::stack<Deque<CargoDepartures>>\* cargodepStack,std::stack<Deque<CargoArrivals>>\* cargoarrStack,std::stack<int>\* typeStack,QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::AddFlightWindow)

{

this->passdepinfo=passdepinfo;

this->passarrinfo=passarrinfo;

this->cargodepinfo=cargodepinfo;

this->cargoarrinfo=cargoarrinfo;

this->passdepStack=passdepStack;

this->passarrStack=passarrStack;

this->cargodepStack=cargodepStack;

this->cargoarrStack=cargoarrStack;

this->typeStack=typeStack;

this->type=0;

ui->setupUi(this);

}

AddFlightWindow::~AddFlightWindow()

{

delete ui;

}

void AddFlightWindow::setPassDepTable()

{

ui->tableWidget->clear();

ui->tableWidget->setRowCount(12);

ui->tableWidget->setVerticalHeaderLabels(QStringList()<<"Самолет" <<"Авиакомпания"<<"Мест эконом-класса"<<"Мест бизнес-класса"

<<"Мест первого-класса"<<"Дата вылета"<<"Время вылета"<<"Направление"<<"Время полета"

<<"Стоимость эконом-класса"<<"Стоимость бизнес-класса"<<"Стоимость первого-класса");

}

void AddFlightWindow::setPassArrTable()

{

ui->tableWidget->clear();

ui->tableWidget->setRowCount(10);

ui->tableWidget->setVerticalHeaderLabels(QStringList()<<"Самолет" <<"Авиакомпания"<<"Мест эконом-класса"<<"Мест бизнес-класса"<<"Мест первого-класса" <<"Дата прилета"<<"Время прилета"<<"Откуда"<<"Время выдачи багажа"<<"Задерживается");

ui->tableWidget->setCellWidget(9,0,new QCheckBox());

}

void AddFlightWindow::setCargoDepTable()

{

ui->tableWidget->clear();

ui->tableWidget->setRowCount(8);

ui->tableWidget->setVerticalHeaderLabels(QStringList()<<"Самолет"

<<"Авиакомпания"<<"Грузоподъемность"<<"Дата вылета"<<"Время вылета"

<<"Направление"<<"Время полета"<<"Стоимость");

}

void AddFlightWindow::setCargoArrTable()

{

ui->tableWidget->clear();

ui->tableWidget->setRowCount(9);

ui->tableWidget->setVerticalHeaderLabels(QStringList()<<"Самолет" <<"Авиакомпания"<<"Грузоподъемность"<<"Масса груза"<<"Дата прилета"

<<"Время прилета"<<"Откуда"<<"Время разгрузки"<<"Задерживается");

ui->tableWidget->setCellWidget(8,0,new QCheckBox());

}

void AddFlightWindow::setTable()

{

switch(type)

{

case(0):

{

setPassDepTable();

break;

}

case(1):

{

setPassArrTable();

break;

}

case(2):

{

setCargoDepTable();

break;

}

case(3):

{

setCargoArrTable();

break;

}

}

}

void AddFlightWindow::setType(int type)

{

this->type=type;

}

void AddFlightWindow::addPassDep()

{

try {

checkPassDep(ui->tableWidget);

int i=0;

PassengerDepartures passdep;

passdep.setModel(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passdep.setCompany(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passdep.setMaxEconom(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setMaxBusiness(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setMaxFirst(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setDate(QDateTime(QDate::fromString(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed(),"dd/MM/yyyy"),

QTime::fromString(ui->tableWidget->item(i+1,0)-> data(Qt::DisplayRole).toString().trimmed(),"HH:mm")));

i+=2;

passdep.setLocation(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passdep.setFlyingTime(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setCostEconom(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setCostBusiness(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setCostFirst(ui->tableWidget->item(i++,0)->text().toInt());

passdepStack->push(\*passdepinfo);

typeStack->push(this->type);

passdepinfo->pushBack(passdep);

} catch (Exception& exp) {

exp.show();

}

}

void AddFlightWindow::addPassArr()

{

try {

checkPassArr(ui->tableWidget);

int i=0;

PassengerArrivals passarr;

passarr.setModel(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passarr.setCompany(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passarr.setMaxEconom(ui->tableWidget->item(i++,0)->text().toInt());

passarr.setMaxBusiness(ui->tableWidget->item(i++,0)->text().toInt());

passarr.setMaxFirst(ui->tableWidget->item(i++,0)->text().toInt());

passarr.setDate(QDateTime(QDate::fromString(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed(),"dd/MM/yyyy"),

QTime::fromString(ui->tableWidget->item(i+1,0)-> data(Qt::DisplayRole).toString().trimmed(),"HH:mm")));

i+=2;

passarr.setLocation(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passarr.setBaggageTime(ui->tableWidget->item(i++,0)->text().toInt());

passarr.setDelay((qobject\_cast<QCheckBox\*>(ui->tableWidget->cellWidget(i,0)))-> isChecked());

passarrStack->push(\*passarrinfo);

typeStack->push(this->type);

passarrinfo->pushBack(passarr);

} catch (Exception& exp) {

exp.show();

}

}

void AddFlightWindow::addCargoDep()

{

try {

checkCargoDep(ui->tableWidget);

int i=0;

CargoDepartures cargodep;

cargodep.setModel(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargodep.setCompany(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargodep.setCargoCapacity(ui->tableWidget->item(i++,0)->text().toFloat());

cargodep.setDate(QDateTime(QDate::fromString(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed(),"dd/MM/yyyy"),

QTime::fromString(ui->tableWidget->item(i+1,0)-> data(Qt::DisplayRole).toString().trimmed(),"HH:mm")));

i+=2;

cargodep.setLocation(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargodep.setFlyingTime(ui->tableWidget->item(i++,0)->text().toInt());

cargodep.setCost(ui->tableWidget->item(i++,0)->text().toInt());

cargodepStack->push(\*cargodepinfo);

typeStack->push(this->type);

cargodepinfo->pushBack(cargodep);

} catch (Exception& exp) {

exp.show();

}

}

void AddFlightWindow::addCargoArr()

{

try {

checkCargoArr(ui->tableWidget);

checkWeight(ui->tableWidget,2);

int i = 0;

CargoArrivals cargoarr;

cargoarr.setModel(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargoarr.setCompany(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargoarr.setCargoCapacity(ui->tableWidget->item(i++,0)->text().toFloat());

cargoarr.setWeight(ui->tableWidget->item(i++,0)->text().toFloat());

cargoarr.setDate(QDateTime(QDate::fromString(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed(),"dd/MM/yyyy"),

QTime::fromString(ui->tableWidget->item(i+1,0)-> data(Qt::DisplayRole).toString().trimmed(),"HH:mm")));

i+=2;

cargoarr.setLocation(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargoarr.setBaggageTime(ui->tableWidget->item(i++,0)->text().toInt());

cargoarr.setDelay((qobject\_cast<QCheckBox\*>(ui->tableWidget->cellWidget(i,0)))-> isChecked());

cargoarrStack->push(\*cargoarrinfo);

typeStack->push(this->type);

cargoarrinfo->pushBack(cargoarr);

} catch (Exception& exp) {

exp.show();

}

}

void AddFlightWindow::on\_addButton\_clicked()

{

switch(type)

{

case(0):

{

addPassDep();

break;

}

case(1):

{

addPassArr();

break;

}

case(2):

{

addCargoDep();

break;

}

case(3):

{

addCargoArr();

break;

}

}

}

void AddFlightWindow::on\_cancelButton\_clicked()

{

this->close();

}

Файл editflightwindow.h содержащий объявление класса EditFlightWindow

#ifndef EDITFLIGHTWINDOW\_H

#define EDITFLIGHTWINDOW\_H

#include "deque.cpp"

#include "passengerArrivals.h"

#include "passengerDepartures.h"

#include "cargodepartures.h"

#include "cargoarrivals.h"

#include "passengerwindow.h"

#include <QMainWindow>

#include <stack>

namespace Ui {

class EditFlightWindow;

}

class EditFlightWindow : public QmainWindow //класс окна изменения рейса

{

Q\_OBJECT

public:

explicit EditFlightWindow(Deque<PassengerDepartures>\* passdepinfo,Deque<PassengerArrivals>\* passarrinfo,Deque<CargoDepartures>\* cargodepinfo,

Deque<CargoArrivals>\* cargoarrinfo,std::stack<Deque<PassengerDepartures>>\* passdepStack, std::stack<Deque<PassengerArrivals>>\* passarrStack,

std::stack< Deque<CargoDepartures>>\* cargodepStack,std::stack< Deque<CargoArrivals>>\* cargoarrStack,std::stack<int>\* typeStack, QWidget \*parent = nullptr); //конструктор

~EditFlightWindow(); //деструктор

void setValue(int,int); //установить значения

void setPassDepTable(); //заполнение таблицы пассажирских вылетов

void setPassArrTable(); //заполнение таблицы пассажирских прилетов

void setCargoDepTable(); //заполнение таблицы грузовых вылетов

void setCargoArrTable(); //заполнение таблицы грузовых прилетов

void setTable(); //выбор типа таблицы

void editPassDep(); //изменение пассажирского вылета

void editPassArr(); //изменение пассажирского прилета

void editCargoDep(); //изменение грузового вылета

void editCargoArr(); //изменение грузового прилета

private slots:

void on\_editButton\_clicked(); //кнопка изменения

void on\_cancelButton\_clicked(); //кнопка отмены

void on\_passengerButton\_clicked(); //кнопка бронирования билетов

private:

Ui::EditFlightWindow \*ui; //указатель на интерфейс

Deque<PassengerDepartures>\* passdep; //пассажирский вылеты

Deque<PassengerArrivals>\* passarr; //пассажирские прилеты

Deque<CargoDepartures>\* cargodep; //грузовые вылеты

Deque<CargoArrivals>\* cargoarr; //грузовые прилеты

//стек пассажирских вылетов

std::stack<Deque<PassengerDepartures>>\* passdepStack;

//стек пассажирских прилетов

std::stack<Deque<PassengerArrivals>>\* passarrStack;

//стек грузовых вылетов

std::stack< Deque<CargoDepartures>>\* cargodepStack;

//стек грузовых прилетов

std::stack< Deque<CargoArrivals>>\* cargoarrStack;

std::stack<int>\* typeStack; //стек типов рейса

int type; //тип рейса

int index; //индекс рейса

PassengerWindow \* passWind; //окно бронирования билетов

};

#endif // EDITFLIGHTWINDOW\_H

Файл editflight.cpp содержащий реализацию класса EditFlightWindow

#include "editflightwindow.h"

#include "ui\_editflightwindow.h"

#include <QCheckBox>

EditFlightWindow::EditFlightWindow(Deque<PassengerDepartures>\* passdepinfo,Deque<PassengerArrivals>\* passarrinfo,Deque<CargoDepartures>\* cargodepinfo,

Deque<CargoArrivals>\* cargoarrinfo,std::stack<Deque<PassengerDepartures>>\* passdepStack, std::stack<Deque<PassengerArrivals>>\* passarrStack,

std::stack<Deque<CargoDepartures>>\* cargodepStack,std::stack<Deque<CargoArrivals>>\* cargoarrStack,std::stack<int>\* typeStack,QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::EditFlightWindow)

{

this->passdepinfo=passdepinfo;

this->passarrinfo=passarrinfo;

this->cargodepinfo=cargodepinfo;

this->cargoarrinfo=cargoarrinfo;

this->passdepStack=passdepStack;

this->passarrStack=passarrStack;

this->cargodepStack=cargodepStack;

this->cargoarrStack=cargoarrStack;

this->typeStack=typeStack;

this->type=0;

this->index=0;

this->passWind = new PassengerWindow(passdepinfo,cargodepinfo,passdepStack,cargodepStack,typeStack);

ui->setupUi(this);

}

EditFlightWindow::~EditFlightWindow()

{

passWind->close();

delete ui;

delete passWind;

}

void EditFlightWindow::setPassDepTable()

{

int i = 0;

ui->tableWidget->clear();

ui->tableWidget->setRowCount(12);

ui->tableWidget->setVerticalHeaderLabels(QStringList()<<"Самолет"<<"Авиакомпания" <<"Мест эконом-класса"<<"Мест бизнес-класса"

<<"Мест первого-класса"<<"Дата вылета"<<"Время вылета"<<"Направление"<<"Время полета"

<<"Стоимость эконом-класса"<<"Стоимость бизнес-класса"<<"Стоимость первого-класса");

PassengerDepartures & passdep = \*(passdepinfo->begin()+=index);

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passdep.getModel()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passdep.getCompany()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passdep.getMaxEconom())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passdep.getMaxBusiness())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passdep.getMaxFirst())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passdep.getDate().date().toString("dd/MM/yyyy")));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passdep.getDate().time().toString("HH:mm")));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passdep.getLocation()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passdep.getFlyingTime())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passdep.getCostEconom())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passdep.getCostBusiness())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passdep.getCostFirst())));

}

void EditFlightWindow::setPassArrTable()

{

int i = 0;

ui->tableWidget->clear();

ui->tableWidget->setRowCount(10);

ui->tableWidget->setVerticalHeaderLabels(QStringList()<<"Самолет"<<"Авиакомпания"<<"Мест эконом-класса"<<"Мест бизнес-класса"

<<"Мест первого-класса"<<"Дата прилета"<<"Время прилета"<<"Откуда"<<"Время выдачи багажа"<<"Задерживается");

PassengerArrivals & passarr = \*((passarrinfo->begin())+=index);

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passarr.getModel()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passarr.getCompany()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passarr.getMaxEconom())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passarr.getMaxBusiness())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passarr.getMaxFirst())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passarr.getDate().date().toString("dd/MM/yyyy")));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passarr.getDate().time().toString("HH:mm")));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(passarr.getLocation()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(passarr.getBaggageTime())));

ui->tableWidget->setCellWidget(i,0,new QCheckBox());

qobject\_cast<QCheckBox\*>(ui->tableWidget->cellWidget(i++,0))->setChecked(passarr.isDelayed());

}

void EditFlightWindow::setCargoDepTable()

{

int i = 0;

ui->tableWidget->clear();

ui->tableWidget->setRowCount(8);

ui->tableWidget->setVerticalHeaderLabels(QStringList()<<"Самолет" <<"Авиакомпания"<<"Грузоподъемность"<<"Дата вылета"<<"Время вылета"

<<"Направление"<<"Время полета"<<"Стоимость");

CargoDepartures & cargodep = \*(cargodepinfo->begin()+=index);

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargodep.getModel()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargodep.getCompany()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(cargodep.getCargoCapacity())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargodep.getDate().date().toString("dd/MM/yyyy")));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargodep.getDate().time().toString("HH:mm")));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargodep.getLocation()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(cargodep.getFlyingTime())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(cargodep.getCost())));

}

void EditFlightWindow::setCargoArrTable()

{

int i = 0;

ui->tableWidget->clear();

ui->tableWidget->setRowCount(9);

ui->tableWidget->setVerticalHeaderLabels(QStringList()<<"Самолет"<<"Авиакомпания" <<"Грузоподъемность"<<"Масса груза"<<"Дата прилета"

<<"Время прилета"<<"Откуда"<<"Время разгрузки"<<"Задерживается");

CargoArrivals & cargoarr = \*(cargoarrinfo->begin()+=index);

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargoarr.getModel()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargoarr.getCompany()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(cargoarr.getCargoCapacity())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(cargoarr.getWeight())));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargoarr.getDate().date().toString("dd/MM/yyyy")));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargoarr.getDate().time().toString("HH:mm")));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(cargoarr.getLocation()));

ui->tableWidget->setItem(i++,0,new QTableWidgetItem(QString::number(cargoarr.getBaggageTime())));

ui->tableWidget->setCellWidget(i,0,new QCheckBox());

qobject\_cast<QCheckBox\*>(ui->tableWidget->cellWidget(i++,0))-> setChecked(cargoarr.isDelayed());

}

void EditFlightWindow::setTable()

{

switch(type)

{

case(0):

{

setPassDepTable();

break;

}

case(1):

{

setPassArrTable();

break;

}

case(2):

{

setCargoDepTable();

break;

}

case(3):

{

setCargoArrTable();

break;

}

}

}

void EditFlightWindow::setValue(int type, int index)

{

this->type=type;

this->index=index;

switch(type)

{

case 0:

{

ui->passengerButton->show();

ui->passengerButton->setText("Бронирование билетов");

break;

}

case 2:

{

ui->passengerButton->show();

ui->passengerButton->setText("Бронирование грузовых перевозок");

break;

}

default:

{

ui->passengerButton->hide();

break;

}

}

}

void EditFlightWindow::editPassDep()

{

try {

PassengerDepartures & passdep = \*(passdepinfo->begin()+=index);

checkPassDep(ui->tableWidget);

checkSpace(ui->tableWidget,2,passdep);

passdepStack->push(\*passdepinfo);

typeStack->push(this->type);

int i=0;

passdep.setModel(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passdep.setCompany(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passdep.setMaxEconom(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setMaxBusiness(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setMaxFirst(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setDate(QDateTime(QDate::fromString(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed(),"dd/MM/yyyy"),

QTime::fromString(ui->tableWidget->item(i+1,0)-> data(Qt::DisplayRole).toString().trimmed(),"HH:mm")));

i+=2;

passdep.setLocation(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passdep.setFlyingTime(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setCostEconom(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setCostBusiness(ui->tableWidget->item(i++,0)->text().toInt());

passdep.setCostFirst(ui->tableWidget->item(i++,0)->text().toInt());

} catch (Exception & exp) {

exp.show();

}

}

void EditFlightWindow::editPassArr()

{

try {

checkPassArr(ui->tableWidget);

passarrStack->push(\*passarrinfo);

typeStack->push(this->type);

int i=0;

PassengerArrivals & passarr = \*(passarrinfo->begin()+=index);

passarr.setModel(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passarr.setCompany(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passarr.setMaxEconom(ui->tableWidget->item(i++,0)->text().toInt());

passarr.setMaxBusiness(ui->tableWidget->item(i++,0)->text().toInt());

passarr.setMaxFirst(ui->tableWidget->item(i++,0)->text().toInt());

passarr.setDate(QDateTime(QDate::fromString(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed(),"dd/MM/yyyy"),

QTime::fromString(ui->tableWidget->item(i+1,0)-> data(Qt::DisplayRole).toString().trimmed(),"HH:mm")));

i+=2;

passarr.setLocation(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

passarr.setBaggageTime(ui->tableWidget->item(i++,0)->text().toInt());

passarr.setDelay((qobject\_cast<QCheckBox\*>(ui->tableWidget->cellWidget(i,0)))-> isChecked());

} catch (Exception & exp) {

exp.show();

}

}

void EditFlightWindow::editCargoDep()

{

try {

CargoDepartures & cargodep = \*(cargodepinfo->begin()+=index);

checkCargoDep(ui->tableWidget);

checkWeight(ui->tableWidget,2,cargodep);

cargodepStack->push(\*cargodepinfo);

typeStack->push(this->type);

int i=0;

cargodep.setModel(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargodep.setCompany(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargodep.setCargoCapacity(ui->tableWidget->item(i++,0)->text().toFloat());

cargodep.setDate(QDateTime(QDate::fromString(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed(),"dd/MM/yyyy"),

QTime::fromString(ui->tableWidget->item(i+1,0)-> data(Qt::DisplayRole).toString().trimmed(),"HH:mm")));

i+=2;

cargodep.setLocation(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargodep.setFlyingTime(ui->tableWidget->item(i++,0)->text().toInt());

cargodep.setCost(ui->tableWidget->item(i++,0)->text().toInt());

} catch (Exception & exp) {

exp.show();

}

}

void EditFlightWindow::editCargoArr()

{

try {

checkCargoArr(ui->tableWidget);

cargoarrStack->push(\*cargoarrinfo);

typeStack->push(this->type);

int i = 0;

CargoArrivals & cargoarr = \*(cargoarrinfo->begin()+=index);

cargoarr.setModel(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargoarr.setCompany(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargoarr.setCargoCapacity(ui->tableWidget->item(i++,0)->text().toFloat());

cargoarr.setWeight(ui->tableWidget->item(i++,0)->text().toFloat());

cargoarr.setDate(QDateTime(QDate::fromString(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed(),"dd/MM/yyyy"),

QTime::fromString(ui->tableWidget->item(i+1,0)-> data(Qt::DisplayRole).toString().trimmed(),"HH:mm")));

i+=2;

cargoarr.setLocation(ui->tableWidget->item(i++,0)-> data(Qt::DisplayRole).toString().trimmed());

cargoarr.setBaggageTime(ui->tableWidget->item(i++,0)->text().toInt());

cargoarr.setDelay((qobject\_cast<QCheckBox\*>(ui->tableWidget->cellWidget(i,0)))-> isChecked());

} catch (Exception & exp) {

exp.show();

}

}

void EditFlightWindow::on\_editButton\_clicked()

{

switch(type)

{

case(0):

{

editPassDep();

break;

}

case(1):

{

editPassArr();

break;

}

case(2):

{

editCargoDep();

break;

}

case(3):

{

editCargoArr();

break;

}

}

}

void EditFlightWindow::on\_cancelButton\_clicked()

{

this->close();

}

void EditFlightWindow::on\_passengerButton\_clicked()

{

passWind->setValue(type,index);

passWind->setTable();

passWind->show();

}

Файл passengerwindow.h содержащий объявление класса PassengerWindow

#ifndef PASSENGERWINDOW\_H

#define PASSENGERWINDOW\_H

#include "deque.cpp"

#include "passengerDepartures.h"

#include "cargodepartures.h"

#include <QMainWindow>

#include "checker.h"

#include <stack>

#include "otherexception.h"

namespace Ui {

class PassengerWindow;

}

class PassengerWindow : public QmainWindow //класс окна бронирования билетов

{

Q\_OBJECT

public:

explicit PassengerWindow(Deque<PassengerDepartures> \*passdepinfo = nullptr, Deque<CargoDepartures> \*cargodepinfo = nullptr,

std::stack<Deque<PassengerDepartures>>\* passdepStack = nullptr, std::stack<Deque<CargoDepartures>>\* cargodepStack = nullptr,

std::stack<int>\* typeStack = nullptr, QWidget \*parent = nullptr); //конструктор

~PassengerWindow(); //деструктор

void setValue(int,int); //установить значения

void setTable(); //установить тип таблицы

void showPassenger(); //заполнение таблицы пассажиров

void showCargo(); //заполнение таблицы грузов

void editPassenger(); //изменение пассажиров

void editCargo(); //изменение грузов

void updateStatusBar(); //изменение панели состояния

private slots:

void on\_addButton\_clicked(); //кнопка добавления

void on\_dellButton\_clicked(); //кнопка удаления

void on\_editButton\_clicked(); //кнопка изменения

private:

Ui::PassengerWindow \*ui; //указатель на интерфейс

int type; //тип

int index; //индекс

Deque<PassengerDepartures> \*passdepinfo; //пассажирские вылеты

Deque<CargoDepartures> \*cargodepinfo; //грузовые вылеты

//стек пассажирских вылетов

std::stack<Deque<PassengerDepartures>>\* passdepStack;

//стек грузовых вылетов

std::stack< Deque<CargoDepartures>>\* cargodepStack;

std::stack<int>\* typeStack; //стек типов рейса

};

#endif // PASSENGERWINDOW\_H

Файл passengerwindow.cpp содержащий реализацию класса PassengerWindow

#include "passengerwindow.h"

#include "ui\_passengerwindow.h"

#include <QComboBox>

#include <QDebug>

PassengerWindow::PassengerWindow(Deque<PassengerDepartures> \*passdepinfo, Deque<CargoDepartures> \*cargodepinfo,std::stack<Deque<PassengerDepartures>>\* passdepStack,

std::stack< Deque<CargoDepartures>>\* cargodepStack, std::stack<int>\* typeStack, QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::PassengerWindow)

{

this->type=0;

this->index=0;

this->passdepinfo=passdepinfo;

this->cargodepinfo=cargodepinfo;

this->passdepStack=passdepStack;

this->cargodepStack=cargodepStack;

this->typeStack=typeStack;

ui->setupUi(this);

}

PassengerWindow::~PassengerWindow()

{

delete ui;

}

void PassengerWindow::setValue(int type, int index)

{

this->type=type;

this->index=index;

}

void PassengerWindow::updateStatusBar()

{

if(type==0)

{

PassengerDepartures & a = \*(passdepinfo->begin()+=index);

QString str = QString("Мест Эконом: %1/%2, Бизнес: %3/%4, Первый: %5/%6").arg(a.getEconom()).arg(a.getMaxEconom()).arg(a.getBusiness()).arg(

a.getMaxBusiness()).arg(a.getFirst()).arg(a.getMaxFirst());

ui->statusbar->showMessage(str);

}

else {

CargoDepartures & a = \*(cargodepinfo->begin()+=index);

QString str = QString("Занятый вес: %1/%2,

свободно: %3").arg(a.getWeight()).arg(a.getCargoCapacity()).arg(a.getFreeWeight());

ui->statusbar->showMessage(str);

}

}

void PassengerWindow::showPassenger()

{

ui->tableWidget->setColumnCount(4);

ui->tableWidget->setHorizontalHeaderLabels(QStringList() << "Фамилия" << "Имя"

<< "Контактные данные" << "Тип места");

std::list<Passenger> & passList = (\*(passdepinfo-> begin()+=index)).getPassengerList();

int count = passList.size();

ui->tableWidget->setRowCount(count);

auto iter = passList.begin();

for(int i = 0;i<count;i++,iter++)

{

ui->tableWidget->setItem(i,0,new QTableWidgetItem((\*iter).getSurname()));

ui->tableWidget->setItem(i,1,new QTableWidgetItem((\*iter).getName()));

ui->tableWidget->setItem(i,2,new QTableWidgetItem((\*iter).getNumber()));

QComboBox \* comboBox = new QComboBox;

comboBox->addItems(QStringList()<<"Эконом"<<"Бизнес"<<"Первый");

comboBox->setCurrentIndex((\*iter).getSpace());

ui->tableWidget->setCellWidget(i,3, comboBox);

}

}

void PassengerWindow::showCargo()

{

ui->tableWidget->setColumnCount(5);

ui->tableWidget->setHorizontalHeaderLabels(QStringList() << "Фамилия" << "Имя"

<< "Контактные данные" << "Груз" << "Вес груза");

std::list<Cargo> & cargoList = (\*(cargodepinfo->begin()+=index)).getCargoList();

int count = cargoList.size();

ui->tableWidget->setRowCount(count);

auto iter = cargoList.begin();

for(int i = 0;i<count;i++,iter++)

{

ui->tableWidget->setItem(i,0,new QTableWidgetItem((\*iter).getSurname()));

ui->tableWidget->setItem(i,1,new QTableWidgetItem((\*iter).getName()));

ui->tableWidget->setItem(i,2,new QTableWidgetItem((\*iter).getNumber()));

ui->tableWidget->setItem(i,3,new QTableWidgetItem((\*iter).getCargoName()));

ui->tableWidget->setItem(i,4,new QTableWidgetItem(QString::number((\*iter).getCargoWeight())));

}

}

void PassengerWindow::setTable()

{

ui->tableWidget->clear();

updateStatusBar();

if(type==0)

{

showPassenger();

}

else {

showCargo();

}

}

void PassengerWindow::on\_addButton\_clicked()

{

ui->tableWidget->setRowCount(ui->tableWidget->rowCount()+1);

if(type == 0)

{

QComboBox \* comboBox = new QComboBox;

comboBox->addItems(QStringList()<<"Эконом"<<"Бизнес"<<"Первый");

ui->tableWidget->setCellWidget(ui->tableWidget->rowCount()-1,3, comboBox);

}

}

void PassengerWindow::on\_dellButton\_clicked()

{

ui->tableWidget->removeRow(ui->tableWidget->currentRow());

}

void PassengerWindow::on\_editButton\_clicked()

{

if(type==0)

{

PassengerDepartures & passdep = \*(passdepinfo->begin()+=index);

std::list<Passenger> tempList;

PassengerDepartures temp;

try

{

for(int i=0;i<ui->tableWidget->rowCount();i++)

{

checkPassenger(ui->tableWidget,i);

Passenger pass;

pass.setSurname(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed());

pass.setName(ui->tableWidget->item(i,1)-> data(Qt::DisplayRole).toString().trimmed());

pass.setNumber(ui->tableWidget->item(i,2)-> data(Qt::DisplayRole).toString().trimmed());

pass.setSpace(qobject\_cast<QComboBox\*>(ui->tableWidget-> cellWidget(i,3))->currentIndex());

if(pass.getSpace()==0)

{

if(passdep.getMaxEconom()>temp.getEconom()) temp.setEconom(temp.getEconom()+1);

else throw OtherException("Все места эконом-класса заняты");

}

else if(pass.getSpace()==1)

{

if(passdep.getMaxBusiness()>temp.getBusiness()) temp.setBusiness(temp.getBusiness()+1);

else throw OtherException("Все места бизнес-класса заняты");

}

else if(pass.getSpace()==2)

{

if(passdep.getMaxFirst()>temp.getFirst()) temp.setFirst(temp.getFirst()+1);

else throw OtherException("Все места первого класса заняты");

}

tempList.push\_back(pass);

}

} catch (Exception & exp) {

exp.show();

return;

}

passdepStack->push(\*passdepinfo);

typeStack->push(type);

passdep.setEconom(temp.getEconom());

passdep.setBusiness(temp.getBusiness());

passdep.setFirst(temp.getFirst());

passdep.getPassengerList() = tempList;

}

else{

CargoDepartures & cargodep = \*(cargodepinfo->begin()+=index);

std::list<Cargo> tempList;

CargoDepartures temp;

try

{

for(int i=0;i<ui->tableWidget->rowCount();i++)

{

checkCargo(ui->tableWidget,i);

Cargo cargo;

cargo.setSurname(ui->tableWidget->item(i,0)-> data(Qt::DisplayRole).toString().trimmed());

cargo.setName(ui->tableWidget->item(i,1)-> data(Qt::DisplayRole).toString().trimmed());

cargo.setNumber(ui->tableWidget->item(i,2)-> data(Qt::DisplayRole).toString().trimmed());

cargo.setCargoName(ui->tableWidget->item(i,3)-> data(Qt::DisplayRole).toString().trimmed());

cargo.setCargoWeight((ui->tableWidget->item(i,4)->text()).toFloat());

if(cargo.getCargoWeight()>cargodep.getFreeWeight()) throw OtherException("Превышена грузоподъемность");

temp.setWeight(temp.getWeight()+cargo.getCargoWeight());

temp.setCargoCount(temp.getCargoCount()+1);

tempList.push\_back(cargo);

}

} catch (Exception& exp) {

exp.show();

return;

}

cargodepStack->push(\*cargodepinfo);

typeStack->push(type);

cargodep.setWeight(temp.getWeight());

cargodep.getCargoList() = tempList;

cargodep.setCargoCount(temp.getCargoCount());

}

updateStatusBar();

}