**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(национальный исследовательский университет)»

**Факультет Информационные технологии и прикладная математика Кафедра 806\_\_\_**

**Направление подготовки 02.04.02 ФИИТ Группа М8О-207М-19\_\_\_\_\_\_\_\_**

**Квалификация (степень) магистр \_\_\_**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**МАГИСТРА**

**На тему:** Разработка модели оптимального размещения средств наземного обслуживания по местам стоянки воздушных судов

**Автор квалификационной работы** Четвериков Виталий Анатольевич (\_\_\_\_\_\_\_\_\_\_\_)

(Фамилия, имя, отчество)

**Научный руководитель** Булакина Мария Борисовна (\_\_\_\_\_\_\_\_\_\_\_)

(Фамилия, имя, отчество)

**Рецензент** Ильченко Татьяна Евгеньевна (\_\_\_\_\_\_\_\_\_\_\_)

(Фамилия, имя, отчество)

**К з а щ и т е д о п у с т и т ь**

**Зав. кафедрой** Крылов Сергей Сергеевич (\_\_\_\_\_\_\_\_\_\_\_)

(Фамилия, имя, отчество)

“ ”\_\_\_\_\_\_\_ 2021г.

Москва 2021 г.

Оглавление

[Введение 3](#_Toc72363386)

[Актуальность темы 4](#_Toc72363387)

[Цели и задачи работы 5](#_Toc72363388)

[Теоретическая часть 6](#_Toc72363389)

[Формализованная постановка задачи 10](#_Toc72363390)

[Алгоритмы решения 13](#_Toc72363391)

[Схема прилёт – обслуживание – вылет 14](#_Toc72363392)

[Расписание 15](#_Toc72363393)

[Воздушное судно 15](#_Toc72363394)

[Место стоянки 15](#_Toc72363395)

[Технологический график обслуживания 16](#_Toc72363396)

[Операция ТГО 16](#_Toc72363397)

[Средства наземного обслуживания 17](#_Toc72363398)

[Связь типов СНО с операциями 21](#_Toc72363399)

[Практическая часть 22](#_Toc72363400)

[Общий алгоритм 22](#_Toc72363401)

[Загрузка исходных данных 23](#_Toc72363402)

[Группировка МС в блоки 24](#_Toc72363403)

[Расчёт прогнозируемого количества комплектов СНО 26](#_Toc72363404)

[Отчёт и его формирование 26](#_Toc72363405)

[Заключение 28](#_Toc72363406)

[Источники 29](#_Toc72363407)

[Сокращения 29](#_Toc72363408)

[Приложения 31](#_Toc72363409)

# Введение

Международный аэропорт Шереметьево — крупнейший российский аэропорт по объемам пассажирских и грузовых перевозок, взлетно-посадочных операций, площади аэровокзального комплекса и мощности карго-комплекса. Маршрутная сеть аэропорта составляет свыше 200 направлений. По итогам 2018 года этот аэропорт признан самым пунктуальным аэропортом среди крупнейших аэропортов мира.

В процессе обслуживания воздушных судов (далее – ВС) в Международном аэропорту Шереметьево используется несколько видов специального оборудования – средств наземного обслуживания: стремянки нескольких типов (в зависимости от количества ступеней и высоты воздушного судна), конуса безопасности и упорные колодки (далее – СНО). Для своевременного обслуживания все СНО должны находиться или непосредственно на месте стоянки воздушного судна (далее – МС), или на ближайшей к нему площадке хранения СНО. Площадки организованы таким образом, чтобы от них можно было максимально оперативно доставить СНО непосредственно на МС – по группам стоянок.

Требования к конкретному виду СНО или их количеству, в частности, для стремянок и конусов безопасности, зависят от типа (иногда – конкретной модели) воздушного судна и расписания полётов. К началу обслуживания – времени прибытия рейса по расписанию – все СНО должны быть доступны либо непосредственно на МС, либо на ближайшей к нему площадке хранения СНО.

## Актуальность темы

Любой сбой в работе аэропорта обходится очень дорого, либо для авиакомпаний затронутых этим сбоем, либо для самого аэропорта в зависимости от того по чьей вине произошёл сбой. Это ключевая причина, которая на наш взгляд является обоснованием необходимости выполнения данной работы. Для обслуживания мест стоянок во время ТГО воздушных судов необходимо наличие в доступности СНО определённых типов. В связи с этим проблема своевременной поставки СНО для участия в операциях ТГО является актуальной. На текущий момент расчёты необходимого СНО ведутся в программе MS Excel, и являются не оптимальными, т к СНО в процессе использования могут быть применены в ТГО различных рейсов на одном и том же МС, и их количество, которое должно храниться в заданный период времени для МС можно существенно уменьшить. В связи с этим возникла потребность в данной работе в которой рассматривается оптимизация размещения СНО по местам стоянок.

## Цели и задачи работы

**Цель работы:** разработать систему расчёта необходимого количества СНО на каждом МС или на каждой площадке хранения СНО в зависимости от расписания с учётом распределения воздушных судов по стоянкам.

**Задачи:**

1. обеспечить возможность загрузки расписания полётов с указанием:

* даты, времени рейса по расписанию за произвольный отрезок времени (одни сутки, трое или десять суток);
* выполняющего рейс воздушного судна (типа ВС);
* планируемого места стоянки;

1. рассчитать на основании этих данных необходимое количество средств наземного обслуживания по типам на каждое МС или на группу стоянок;
2. проанализировать эффективность варианта размещения СНО на каждое МС или на группу мест стоянок, если это группа мест стоянок выбрать оптимальное количество стоянок в группе.

**Ожидаемые результаты работы:** расчёт с указанием требуемого количества СНО и анализ эффективности той или иной модели расстановки СНО в зависимости от расписания. Более эффективной будет признана та модель, при которой требуется меньшее количество СНО на заявленный период (на основании загруженного расписания).

# Теоретическая часть

Более подробно о том, как происходит обработка ВС в аэропорту, разберём на примере рейса прилёта. В заданное расписанием время самолёт приземляется в аэропорту далее он своим ходом или при помощи буксировщика отправляется к своему месту стоянки (далее - МС). По прибытии на МС, начинается обслуживание самолёта согласно технологическому графику обслуживания (далее – ТГО). В ТГО перечислены все операции, которые необходимо произвести с ВС. Длительность ТГО составляет в общих случаях от 30 до 60 минут и зависит от нескольких параметров, как - то фюзеляжность самолёта (широкий или узкий фюзеляж, далее ШФ и УФ), кода авиакомпании (далее - АК), и типа рейса прилёт, вылет или разворот. Сами операции типовые их длительность зависит от АК, типа используемого СНО и модели самолёта. Для примера приведён ТГО прилета см. рис 1.

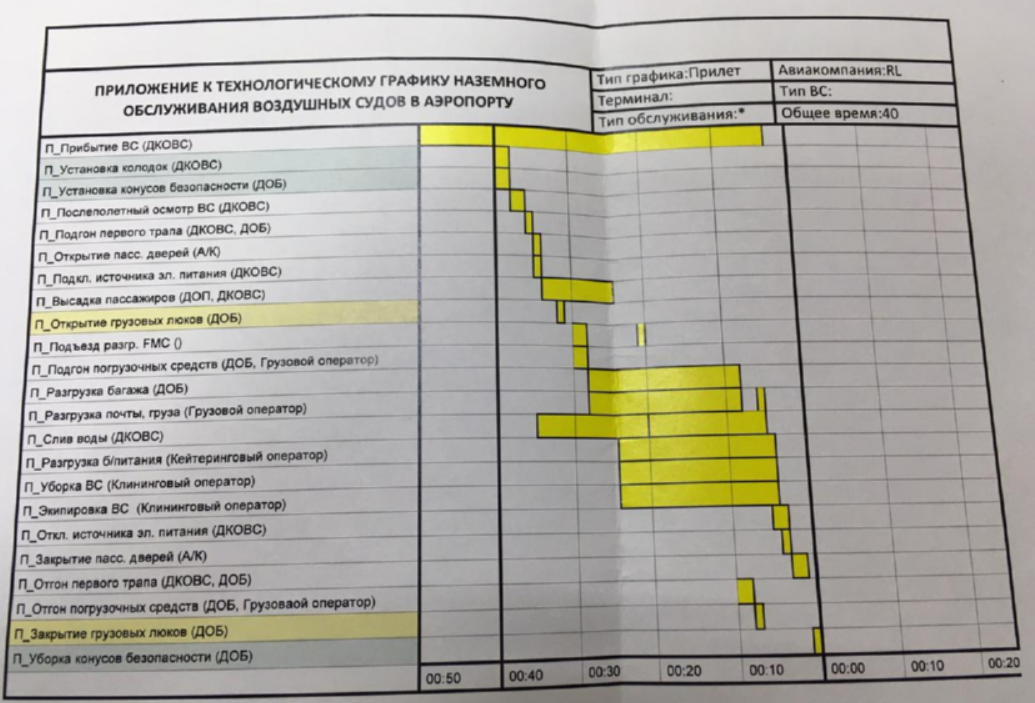


Рис. 1 Пример ТГО

Для каждой операции в ТГО, необходимо наличие в доступности на МС СНО определённого типа. Разберем подробнее операции, которые использовались для расчётов в данной работе:

Для операции «Установка колодок», требуется наличие комплекта упорных колодок с учётом типа ВС и уклона местности, где расположено МС. Данный комплект колодок будет задействован до операции «Уборка колодок». Также следует отметить, что в отдельных случаях если не производится перебуксировка ВС или перегон его на удалённую МС, для типа ТГО «прилёт», комплект колонок остаётся задействованным до вылета данного ВС, то есть до завершения операции «Уборка колодок» из ТГО типа «Вылет» следующего рейса.

Для операции «Установка конусов безопасности» требуется наличие комплекта конусов безопасности с учётом типа ВС, комплект конусов безопасности будет задействован до окончания операции «Уборка конусов безопасности».

Для операций «Открытие/закрытие грузовых люков» требуется наличие стремянки, тип стремянки зависит от фюзеляжности самолёта

Оптимизация заключается в двух формах

* МС разбиваются на блоки, для удобства обеспечения СНО, и его централизованного хранения, поэтому необходимо вычислить оптимальное количество МС в блоке.
* Внутри блоков операции по технологическому обслуживанию на МС могут пересекаться а могут не пересекаться, во втором случае можно оптимизировать это таким образом, что один и тот же комплект СНО в разные промежутки времени будет использоваться на разных МС внутри блока, что приведёт к уменьшению количества хранимых МС на блок в заданный период времени.

Рассмотрим более подробно схему работы. Авиакомпании отправляют запросы в аэропорт с указанием типа ВС и планируемого времени прибытия/вылета. На основании поступивших данных аэропорт формирует расписание рейсов, с указанием времени, МС, вида ВС, типа ТГО. Модуль расчёта представляет собой программный интерфейс в виде десктопного приложения. Пользователю предоставляется возможность загрузить карту с МС, справочник ТГО, расписание рейсов, справочники типов ВС, и видов СНО, а также задать количество МС в блоке и промежуток времени, для которого будет производиться расчёт. После того как входные данные подготовлены и загружены, начинается расчет: по карте считываются координаты МС и согласно заданному количеству МС в блоке с условием максимальной близости барицентров (среднее арифметическое положений всех точек фигуры) МС формируются блоки стоянок. Затем для каждого блока МС определяются соответствующие строки расписания на заданный промежуток времени, в соответствии с этими строками формируется представление о загрузке МС внутри каждого блока. Далее к каждой строке расписания связанной с конкретным МС, привязывается график ТГО. График ТГО представляет собой интервал времени в котором описываются все операции производимые с ВС, начиная от прибытия на МС и заканчивая убытием. Также для каждой операции в рамках ТГО указывается временной промежуток и используемый тип СНО. По графикам ТГО и типу ВС из связанной строки расписания выявляются задействованные типы СНО и их необходимое количество для обслуживания целого блока в заданный промежуток времени. Эта операция производится для каждого блока, на выходе получается потребное количество СНО на блок. После чего вычисляется совокупное количество комплектов СНО необходимое на обеспечение аэропорта в заданный промежуток времени. При варьировании количества МС в блоке появляется возможность оптимально разместить средства наземного обслуживания по местам стоянок, что и является целью расчётов. В результате формируется отчёт с указанием требуемого количества СНО и анализом эффективности той или иной модели расстановки СНО. Более эффективной признаётся та модель, при которой требуется меньшее количество СНО на заданный промежуток времени (на основании загруженного расписания).

## Формализованная постановка задачи

Пусть имеется Y – множество всех мест стоянок.

, где – точка на плоскости.

Общее количество МС равно N. Также известно, что всё это множество стоянок должно разделяться на блоки, по k штук в блоке. Но поскольку N может нацело не делится на k, то имеются остаточные l штук. Из этого формируется следующее соотношение: N = kq+l, где q - число блоков МС, k – количество МС в блоке, l - число МС в остаточном блоке.

Также имеется дискретное время t, где , где T – максимальное время.

Предполагаем, что блоки можно конструировать не произвольным образом, а они должны удовлетворять некоторым геометрическим ограничениям, которые будут задаваться с помощью некоторой функции g, которая для каждого подмножества показывает возможно ли существование такого блока, .

Есть функция , которая действует из {0,1,…,T} , где

p – количество возможных типов оборудования.Эта функция для каждого момента времени и каждого отражает потребность в СНО. Функция Z задаётся таблично и не может быть смоделирована какой-либо случайной функцией, потому что она не случайна. Это потребность, которая задаётся на основе сложных факторов, не учитывающихся в данной задаче.

В дальнейшем вместо будет использовано другое обозначение, которое возникнет из допустимых стратегий.

В задаче множество допустимых стратегий – это множество непересекающихся разбиений множества Y на подмножества, удовлетворяющие количественным и геометрическим ограничениям.

Опишем это множество:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

где

Скобка Айверсона […] — функция, возвращающая 1 для истинного высказывания, и 0, если аргумент ложный;

– число стратегий, разбиения МС на блоки, зависит от общего числа МС, а также от K – множества возможных количеств МС в блоке, задаётся в ограничениях задачи.

- стратегия (i-й вариант разбиения Y на блоки), см. рис 2.

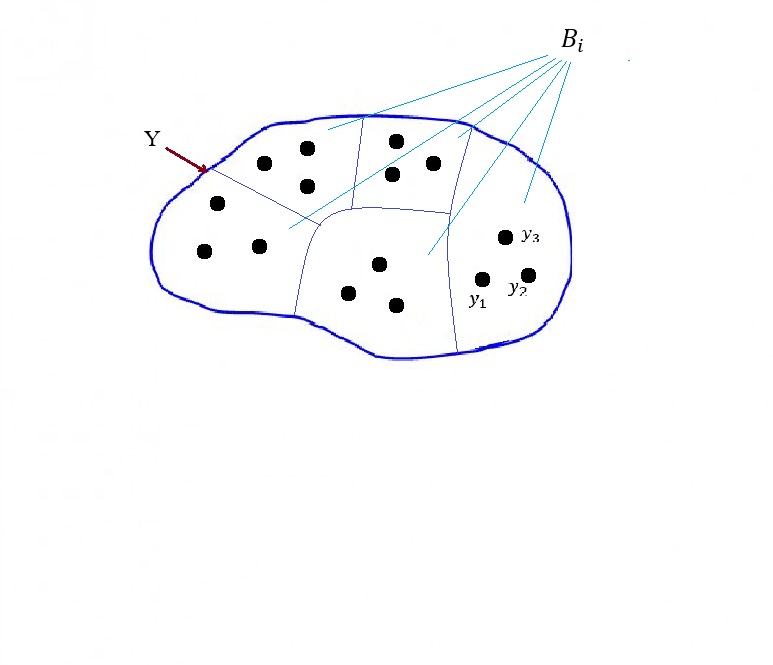


Рис. 2 пример разбиения на множества МС на блоки

– блок МС,

Проиллюстрируем на примере фрагмента схемы аэропорта, см. рис 3.

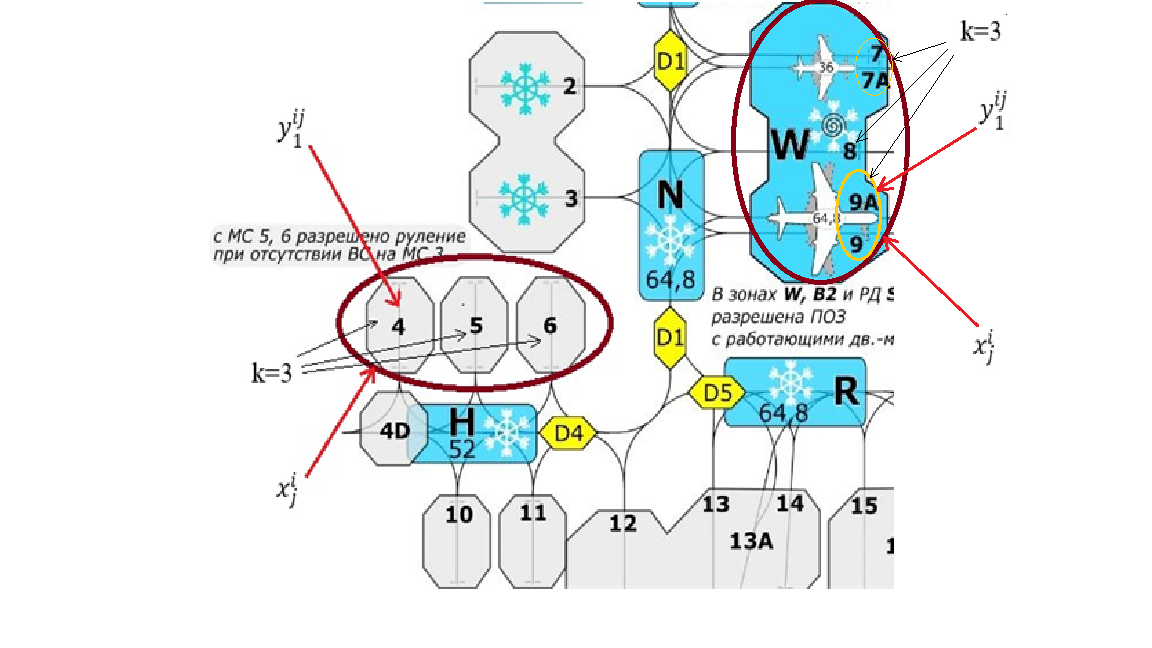


Рис. 3 фрагмент схемы аэропорта, с примером блоков МС

Для того, чтобы найти в работе был предложен следующий алгоритм построения подходящего разбиения.

Алгоритм для разбиения на блоки:

Алгоритм 1

**Вход**: Множество МС - Y, количество МС в блоке - k, геометрические ограничения.

**Выход**: Стратегия, разбиения МС на блоки - .

**пока** Y

X = новый блок стоянок

**для** всех

**если** ((количество y в X < k )

и (X удовлетворяет геометрическим ограничениям) и (yB) и (yX))

**то** в X добавить y

конец **если**

конец **для**

в добавить X

конец **пока**

Тогда оптимизируемый функционал принимает следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

где первое слагаемое отвечает за потребность во всех блоках объёма k, а второе слагаемое отвечает за потребность в блоке состоящем из l оставшихся МС.

Для решения задачи необходимо минимизировать функционал

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

Это задача комбинаторной оптимизации, так как в качестве ответа идёт разбиение исходного множества на непересекающиеся подмножества. В данной задаче количество разбиений можно вычислить только алгоритмически, но оно не зависит от функции расписания, поэтому для характерных значений K и N, может быть рассчитано заранее и хранится в памяти как допустимая стратегия.

## Алгоритмы решения

Данную модель решать не алгоритмически невозможно, в связи с этим для её решения можно применять прямой перебор, либо метаэвристические алгоритмы оптимизации, например генетический алгоритм.

Применяемые в данной работе методы:

* прямой перебор;
* генетический алгоритм.

## Схема прилёт – обслуживание – вылет

В среднем разница во времени между посадкой и взлетом у самолета составляет 30–90 минут. Вот какие этапы он проходит за этот промежуток:

Оставив взлетно-посадочную полосу, самолет отправляется на место стоянки. Авиакомпании стараются выбирать их подальше от телетрапов (подвижная конструкция для прохода пассажиров в самолёт и выхода их же из самолёта. Один конец телетрапа прикреплён к зданию аэропорта, а другой стыкуется к припаркованному самолёту), чтобы для высадки и посадки можно было использовать два разных трапа вместо одного.

Далее, в рамках ТГО, производятся следующие операции:

* Фиксация колес шасси при помощи упорных колодок;
* Ограничение зоны безопасности специальными конусами;
* Осмотр борта с целью выявления возможных повреждений;
* Выгрузка багажа;

Одновременно самолет заправляют топливом, перед чем его обязательно заземляют. Если этого не сделать, статическое электричество может спровоцировать опасную ситуацию: например, выход из строя электроники во время полета или даже взрыв паров топлива.

По необходимости чистят туалеты и заправляют самолет водой. Однако, на коротких рейсах запасов обычно хватает.

Параллельно идет уборка салона. Ее выполняют либо сотрудники клининговой службы аэропорта, либо бортпроводники. Обычно компании выбирают второе: это дешевле.

Последний этап — еще одна проверка исправности двигателей, посадка пассажиров и получение у диспетчера разрешения на взлет.

## Расписание

Расписание составляется обычно на сезон. Подразделяют осенне-зимний и весенне-летний периоды. Расписание представляет собой документ, содержащий данные о рейсах за определённый период времени. В рейс входят данные о том какое ВС его совершает, какая авиакомпания обслуживает, на какое МС прибывает либо с какого убывает, обозначены дата и время прилёта/вылета ВС, а также информацию о ТГО, согласно которому обслуживается воздушное судно.

## Воздушное судно

Летательный аппарат, поддерживаемый в атмосфере за счет взаимодействия с воздухом, отличного от взаимодействия с воздухом, отраженным от поверхности земли или воды. В рамках данной работы под воздушным судном понимается самолёт из заранее заданного списка. Одна из важнейших частей самолёта фюзеляж — корпус летательного аппарата. Рассматриваются в работе два типа фюзеляжей:

- широкий;

- узкий.

Узкофюзеляжный самолет — это авиалайнер, где сиденья расположены по бокам единственного прохода, позволяющий разместить до 6 человек в ряду в салоне меньше 4 м ширины.

Широкофюзеляжный самолет — это более крупный авиалайнер, обычно имеющий несколько проходов и диаметр фюзеляжа более 5 м, позволяя посадить не менее семи человек в ряд.

## Место стоянки

Это часть перрона или аэродрома, отведённая для расположения воздушного судна (ВС) на время разгрузки, загрузки, хранения или технического обслуживания. Обычно имеет порядковый номер, размечена условными границами для расположения ВС, вспомогательного оборудования и транспортных средств, осевыми линиями для заруливания ВС и остановки их в зависимости от типа. Часто оборудована наземными источником электропитания и кондиционером (или подогревателем). Аэродромные сооружения включают в себя грунтовые элементы летного поля, грунтовые основания, аэродромные покрытия, водоотводные и дренажные системы, а также специальные площадки и конструкции.

## Технологический график обслуживания

Важным документом в деятельности аэропорта и авиакомпании при наземном обслуживании являются технологические графики обслуживания воздушных судов (ТГО ВС). ТГО включает в себя все наземные процессы от заруливания ВС на место стоянки (далее — МС) до отправления, включая установку колодок, посадку и высадку пассажиров, погрузку и выгрузку багажа, груза, почты, заправку топливом и прочие предполетные процедуры, выполнение которых необходимо для подготовки ВС к вылету. Технологические графики обслуживания бывают различных типов, их три «прилёт», «вылет» и «разворот». Для максимально эффективного взаимодействия участников процесса необходимо совместно разрабатывать ТГО под каждую авиакомпанию отдельно. Для авиакомпании очень важным является, чтобы не в ущерб безопасности полётов время обслуживания ВС рейса было минимальным, так как оно в итоге влияет на среднесуточный налёт ВС. Минимальное время обслуживания у рейсов ООО "Авиакомпания "Победа". В 2018 г. эта компания признана лучшей по показателю среднесуточного налёта на самолётах типа Boeing 737. Максимальное значение достигло до 16,5 ч в день - мировой рекорд среди авиакомпаний.

### Операция ТГО

В рамках ТГО выполняются различные процессы называемые технологическими операциями. Каждая такая операция имеет строгие временные рамки в которые она должна быть произведена. Из совокупного количества времени затрачиваемого на операции внутри ТГО складывается общая продолжительность графика технологического обслуживания, которая также является строго регламентированной.

## Средства наземного обслуживания

Средства наземного обслуживания – это оборудование, которое необходимо при обслуживании воздушного судна после или перед рейсом. Перечень типов СНО достаточно велик, но в данной работе будут рассматриваться 4 базовых типа:

* Упорные колодки;
* Буксировочные водила;
* Конуса безопасности;
* Стремянки

Рассмотрим подробнее каждый тип.

Упорные колодки – специальные устройства, которые предназначены для использования как фиксирующие противооткатные изделия для самолетов и вертолетов во время стоянки на аэродромах и в аэропортах. На практике тип упорных колодок зависит от типа ВС и соответственно размеров колеса шасси, а также от уклона поверхности места стоянки ВС. Зимой упорные колодки ошиповываются. В рамках данной работы используется один базовый тип упорных колодок, применяемый для всех типов ВС, единственное в зависимости от типа ВС изменяется количество упорных колодок потребное для обслуживания, так широкофюзеляжные самолёты требуют большее количество колодок.

см. рис 4.



Рис. 4 упорная колодка

Буксировочные водила – устройства необходимые при буксировке ВС, могут использоваться как для прилетающих судов, так и для вылетающих. Буксировка не входит в график технологического обслуживания, поэтому временем использования водила для прилета считается 10 минут до начала ТГО а для вылета 10 минут после окончания ТГО.

см. рис 5-6.



Рис. 5 буксировочное водило в действии



Рис. 6 буксировочное водило

Конусы безопасности – в соответствии с руководством ИАТА по наземному обслуживанию, являются предупредительным знаком для водителей, чтобы поддержать безопасные расстояния между транспортным средством и самолетом. Они защищают поверхности воздушного судна от столкновения с наземным оборудованием. К самолету не должны приближаться наземные транспортные средства, пока все конусы безопасности не расставлены. Установка конусов осуществляется после того, как самолет произвел полную остановку, двигатели выключены и вращение роторов двигателей прекращены, проблесковые маяки выключены, установлены упорные колонки, а уборка конусов производится после прекращения действий наземного транспорта и оборудования до отправления воздушного судна. Количество и порядок установки конусов безопасности зависит от политики авиакомпании и конструктивных особенностей воздушного судна. В данной работе порядок установки и тип конусов считается неизменным, изменяется количество конусов в зависимости от типа обслуживаемого воздушного судна.

см. рис 7.



Рис. 7 конуса безопасности

Стремянки – устройства предназначенные для использования при проведении работ по наземному обслуживанию летательных аппаратов, а также другой техники на высоте от 0,6 до 6,0 метров от поверхности земли до плоскости платформы, в зависимости от типа изделия. Виды используемых стремянок имеют разное количество ступеней, а также зависят от характеристик обслуживаемого ВС.

см. рис 8.



Рис. 8 стремянка авиационная

Таким образом были рассмотрены типы СНО, задействованные в данной работе.

## Связь типов СНО с операциями

Типы СНО:

* Стремянки - операции «Открытие грузовых люков», «Закрытие грузовых люков»;
* Конуса безопасности – операции «Установка конусов безопасности», «Уборка конусов безопасности»;
* Упорные колодки – операции «Установка колодок», «Уборка колодок».
* Трапы – если стоянка не контактная (такая стоянка к которой можно подвести телетрап), то для загрузки/разгрузки пассажиров подгоняем/отгоняем трапы. Для широкофюзеляжных самолётов это два трапа, для узкофюзеляжных самолётов один трап;
* Буксировочные водила – необходимы для буксировки с прилёта и на вылет, в работе для упрощения считается - 10 минут от прибытия и +10 минут к отправлению, как время, когда нужны водила по типам (каждый тип ВС - свой тип водила).

МС считается контактным если его номер входит в диапазоны 10-60 для ЮТК и 113-131 для СТК.

Важное дополнение конуса безопасности и упорные колодки **не убираются при прилёте**, в связи с этим возможны два случая первый, когда они убираются для перебуксировки ВС, в этом случаем считается что окончание использования конусов безопасности и упорных колодок это время окончания ТГО, во втором случае ищется следующий рейс с этого МС если это ТГО с типом вылет и код АК совпадает с кодом АК прилёта, тогда и только тогда мы считаем временем окончания использования конусов безопасности и упорных колодок окончание операции уборки конусов/колодок.

# Практическая часть

Программный модуль реализован в Visual Studio 2017 Community Edition, на языке программирования C#. Позволяет получать прогноз потребности в СНО за период выбранный пользователем.

Интерфейс модуля см. рис 9.

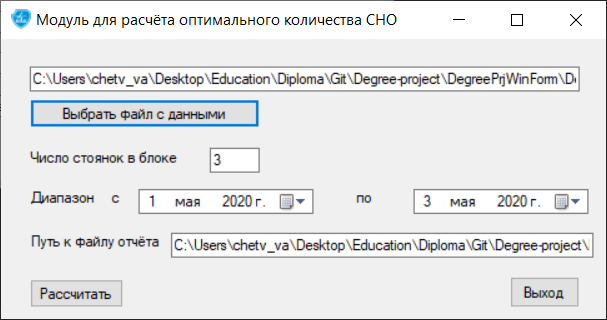


Рис. 9

Имеется возможность задать:

* путь к файлу с данными (приходит от заказчика);
* количество МС, в блоке (по умолчанию выбирается оптимальное количество из всех возможных вариантов);
* диапазон расписания, для которого будет произведено моделирование;
* путь по которому будет сохранён файл сформированного отчета.

## Общий алгоритм

* Загрузка исходных данных;
* Группировка МС в блоки;
* Расчёт потребного количества СНО для обеспечения МС;
* Формирование отчёта.

Рассмотрим каждый из этапов более подробно:

## Загрузка исходных данных

Для начала расчётов необходимо загрузить исходные данные, предоставляются заказчиком примеры файлов можно найти в приложениях к работе. Загружается из файла Excel информация о расписании и о воздушных судах, из файлов Xml загружается информация о местах стоянок и их координатах, а также загружаются типовые ТГО и все связанные с ними операции, в каждой операции указан тип СНО, который задействован и количество комплектов этого типа на операцию. Для работы с Excel используется открытая версия библиотеки EPPlus, для работы с Xml стандартная библиотека System.Xml.XDocument. В приложениях представлены фрагменты справочников СНО, расписания и примеры Xml МС.

ТГО генерируются в тестовом режиме, авиакомпаний 41 штука, типов ВС всего два ШФ и УФ, типов ТГО берётся также два «Прилет», «Вылет». Соответственно общее количество ТГО будет равно 41\*2\*2=164 штуки, им присваиваются уникальные идентификационные номера, для использования в дальнейших расчётах.

Данные загружаются в следующую структуру классов

см. рис 10.

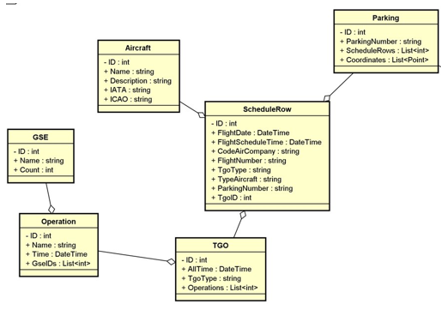


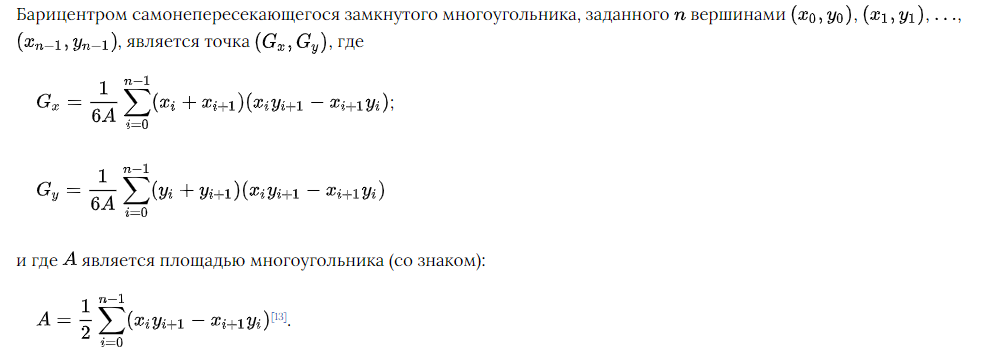
Рис. 10 схема классов для инициализации входных данных

(схема немного устарела дополнить)

Листинг кода классов, представлен в приложениях к данной работе. После того как данные загружены, производится группировка МС в блоки.

## Группировка МС в блоки

МС группируются по блокам следующим образом, считываются координаты МС. Из них формируется самонепересекающийся замкнутый многоугольник, для него вычисляется геометрический центр (барицентр)



(заменить на формулы)

После вычислений согласно заданному количеству МС в блоке с условием максимальной близости геометрических барицентров координат МС формируются блоки стоянок.

см. рис 11.

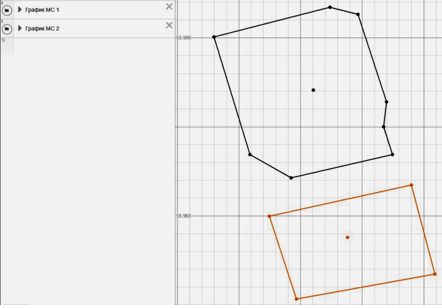


Рис. 11 визуализация МС, как совокупности точек на плоскости

После группировки блоки подгружаются в существующую структуру классов.

см. рис 12.

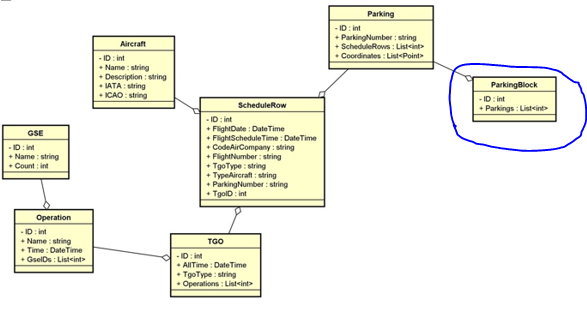


Рис. 12 схема классов с дополненными блоками стоянок

После того как данные загружены, и все множество МС разбито на блоки, начинается непосредственно расчёт.

## Расчёт прогнозируемого количества комплектов СНО

В каждом блоке производится расчёт необходимого количества СНО по МС. У каждой операции в рамках ТГО есть дата начала операции и дата окончания операции имеется задача оценить востребованность в СНО каждого типа в конкретный момент времени внутри блока. Все операции ТГО внутри блока в каждый момент времени сравниваются между собой и максимальное значение за каждый временной промежуток фиксируется, в итоге формируется максимальное количество комплектов СНО по каждому типу на блок. После того как потребность в СНО по типам в блоках МС рассчитана. Считается общее количество комплектов на все стоянки по каждому типу СНО за заданный период времени.

## Отчёт и его формирование

На основании расчётов формируется отчет о потребности в СНО за период, отчёт содержит информацию по потребности в различных типах СНО по блокам, также МС входящие в блоки, а также суммарную прогнозируемую потребность в СНО. Отчёт формируется и сохраняется в файл MS Excel также посредством библиотеки EPPlus.

# Заключение

Разработана модель расчета оптимального размещения средств наземного обслуживания по местам стоянки воздушных судов.

Решена проблема оптимального размещения средств наземного обслуживания по местам стоянок воздушных судов.

По результатам моделирования сформирован отчёт, прогнозирующий потребность в СНО на заданный период времени.

Разработан программный модуль, который в дальнейшем можно использовать для формирования подобных отчётов у заказчика, специалистами с его стороны. Такие отчёты будут полезны для компании - оператора по наземному обслуживанию воздушных судов, при планировании закупок СНО, а также уменьшит вероятность сбоя при выполнении ТГО, так как необходимые средства наземного обслуживания будут, своевременно доставлены при выполнении технологических операций.

В перспективе предполагается интеграция программного модуля с базы данных PostgreSQL, для хранения и своевременного обновления данных необходимых для расчётов.

Апробация работы планируется в ООО "ШЕРЕМЕТЬЕВО ХЭНДЛИНГ". Шереметьево Хэндлинг сегодня — это ключевой оператор по наземному обслуживанию воздушных судов Международного аэропорта Шереметьево.

# Источники

1. Грешилов А.А. Прикладные задачи математического программирования: учебное пособие. - 2-е изд. - М.: Логос, 2006. - 288 с.
2. Черноруцкий И. Г. Методы оптимизации. Компьютерные технологии. — СПб.: БХВ-Петербург, 2011. — 384 с..
3. Руководство ИАТА по наземному обслуживанию (IGOM), 9-е издание/ Material No.: 9390-09 ISBN 978-92-9229-784-8 c 2019 International Air Transport Association. Все права защищены. Монреаль - Женева.
4. СНиП 32-03-96: Аэродромы
5. Руднева, М. А. О технологическом графике обслуживания воздушного судна авиакомпании / М. А. Руднева. — Текст : непосредственный // Молодой ученый. — 2020. — № 39 (329). — С. 9-12.
6. Комбинаторика, Виленкин Н.Я., Виленкин А.Н., Виленкин П.А., 2006.
7. Рихтер Дж. "CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#", 2019.

# Сокращения

* МС – места стоянки;
* СНО – средства наземного обслуживания;
* ВС – воздушное судно;
* ТГО – технологический график обслуживания;
* СТК – северный терминальный комплекс;
* ЮТК – южный терминальный комплекс;
* ОЗП – осенне-зимний период;
* ВЛП – весенне-летний период;
* Блок МС – совокупность, состоящая из одного или нескольких МС;
* УФ – узкофюзеляжный;
* ШФ – широкофюзеляжный;
* АК – авиакомпания.

# Приложения

Приложение 1 листинг кода программного модуля

Реализация класса воздушное судно **Aircraft.cs**

/// <summary>

/// Воздушное судно

/// </summary>

**public** **class** **Aircraft**

{

/// <summary>

/// Идентификатор объекта.

/// </summary>

**public** **string** Id { **get**; **set**; }

/// <summary>

/// Обозначение IATA.

/// </summary>

**public** **string** IATA { **get**; **set**; }

/// <summary>

/// Обозначение ICAO.

/// </summary>

**public** **string** ICAO { **get**; **set**; }

/// <summary>

/// Обозначение по-русски.

/// </summary>

**public** **string** RUS { **get**; **set**; }

/// <summary>

/// Название.

/// </summary>

**public** **string** Name { **get**; **set**; }

}

Реализация класса ТГО **TGO.cs**

/// <summary>

/// Объект типа ТГО

/// </summary>

**public** **class** **TGO**

{

/// <summary>

/// Общее время выполнения ТГО

/// </summary>

**public** **string** TotalTime { **get**; **set**; }

/// <summary>

/// Код авиакомпании

/// </summary>

**public** **string** AircompanyCode { **get**; **set**; }

/// <summary>

/// Тип самолёта ШФ/УФ

/// </summary>

**public** AircraftBodyType AircraftBodyType { **get**; **set**; }

/// <summary>

/// Тип ТГО

/// </summary>

**public** TgoType Type { **get**; **set**; }

/// <summary>

/// Список СНО используемых в данном ТГО

/// </summary>

**public** List<GSE> GseList = **new** List<GSE>();

/// <summary>

/// Список операций производимых в данном ТГО

/// </summary>

**public** List<Operation> Operations = **new** List<Operation>();

/// <summary>

/// Общее время

/// </summary>

/// <returns></returns>

**public** DateTime **GetTotalTime**()

{

**var** splittedTime = TotalTime.Split(':'); //default format mm:ss

**var** time = TimeSpan.FromMinutes(**int**.Parse(splittedTime[**1**]));

**return** DateTime.MinValue + time;

}

}

Реализация класса строка расписания **ScheduleRow.cs**

/// <summary>

/// Строка расписания

/// </summary>

**public** **class** **ScheduleRow**

{

/// <summary>

/// Дата рейса.

/// </summary>

**public** **string** FlightDate { **get**; **set**; }

/// <summary>

/// Время по расписанию.

/// </summary>

**public** **string** FlightScheduleTime { **get**; **set**; }

/// <summary>

/// Код авиакомпании.

/// </summary>

**public** **string** CodeAirCompany { **get**; **set**; }

/// <summary>

/// Номер рейса.

/// </summary>

**public** **string** FlightNumber { **get**; **set**; }

/// <summary>

/// Флаг прилет/вылет.

/// </summary>

**public** **string** Type { **get**; **set**; }

/// <summary>

/// Тип ВС IATA.

/// </summary>

**public** **string** TypePlane { **get**; **set**; }

/// <summary>

/// Стоянка ВС.

/// </summary>

**public** **string** ParkingPlane { **get**; **set**; }

/// <summary>

/// Стоянка сектор.

/// </summary>

**public** **string** ParkingSector { **get**; **set**; }

/// <summary>

/// Название авиакомпании.

/// </summary>

**public** **string** AirCompanyName { **get**; **set**; }

/// <summary>

/// ТГО для воздушного судна по расписанию

/// </summary>

**public** TGO LinkedTGO { **get**; **set**; }

/// <summary>

/// Есть ли пересечения с другими ТГО внутри блока (false по умолчанию)

/// </summary>

**public** **bool** IsCrossed { **get**; **set**; } = **false**;

/// <summary>

/// Есть ли пересечения с другими ТГО внутри блока (false по умолчанию)

/// </summary>

**public** **bool** IsBlockUsed { **get**; **set**; } = **false**;

/// <summary>

/// Есть ли пересечения с другими ТГО внутри блока (false по умолчанию)

/// </summary>

**public** **bool** IsTowBarUsed { **get**; **set**; } = **false**;

/// <summary>

/// Есть ли пересечения с другими ТГО внутри блока (false по умолчанию)

/// </summary>

**public** **bool** IsLadderUsed { **get**; **set**; } = **false**;

/// <summary>

/// Есть ли пересечения по конусам безопасности

/// </summary>

**public** **bool** IsMarkerConeUsed { **get**; **set**; } = **false**;

/// <summary>

/// Начало техобслуживания по графику

/// </summary>

**public** DateTime StartTGO { **get**; **set**; }

/// <summary>

/// Окончание тех обслуживания по графику

/// </summary>

**public** DateTime EndTGO { **get**; **set**; }

/// <summary>

/// Получение типа ТГО

/// </summary>

/// <returns></returns>

**public** TgoType **GetTgoType**()

{

**if** (Type.Contains("D"))

**return** TgoType.departure;

**if** (Type.Contains("A"))

**return** TgoType.arrival;

**return** TgoType.reverse;

}

/// <summary>

/// Получение времени начала ТГО

/// </summary>

/// <returns></returns>

**public** DateTime **GetStartDate**()

{

**var** date = Convert.ToDateTime(FlightDate);

**var** time = Convert.ToDateTime(FlightScheduleTime);

StartTGO = date.Add(time.TimeOfDay);

**return** StartTGO;

}

/// <summary>

/// Есть ли пересечения в операциях по определённому типу сно

/// </summary>

/// <param name="gseType"></param>

/// <returns></returns>

**public** **bool** **IsUsedGseByType**(GseType gseType)

{

**switch** (gseType)

{

**case** GseType.block:

**return** IsBlockUsed;

**case** GseType.ladder:

**return** IsLadderUsed;

**case** GseType.towBar:

**return** IsTowBarUsed;

**case** GseType.markerCone:

**return** IsMarkerConeUsed;

**default**:

**throw** **new** **ArgumentException**("Gse type is undefined!");

}

}

}

Реализация класса Блок МС **ParkingBlock.cs**

/// <summary>

/// Блок стоянок

/// </summary>

**public** **class** **ParkingBlock**

{

/// <summary>

/// Идентификатор объекта.

/// </summary>

**public** **int** Id { **get**; **set**; }

/// <summary>

/// Есть ли рейсы по расписанию для стоянок внутри блока

/// </summary>

**public** **bool** IsFilled { **get**; **set**; }

/// <summary>

/// Места стоянки входящие в блок.

/// </summary>

**public** List<Parking> Parkings { **get**; **set**; }

/// <summary>

/// Количество СНО типа упорные колодки на блок

/// </summary>

**public** **int** BlockGseCount { **get**; **set**; }

/// <summary>

/// Количество СНО типа стремянки на блок

/// </summary>

**public** **int** LadderGseCount { **get**; **set**; }

/// <summary>

/// Количество СНО типа конуса безопасности на блок

/// </summary>

**public** **int** MarkerConeGseCount { **get**; **set**; }

/// <summary>

/// Количество СНО типа буксировочное водило на блок

/// </summary>

**public** **int** TowBarGseCount { **get**; **set**; }

/// <summary>

/// Получение количества СНО на блок по типу

/// </summary>

/// <param name="type"></param>

/// <returns></returns>

**public** **int** **GetGseCountByType**(GseType type)

{

**var** gseCount = **0**;

// Если ни одной нет операции с этим GSE то возвращаем 0

**if** (!IsExistSingleByType(type)) **return** **0**;

gseCount++;

**foreach** (**var** parking **in** Parkings)

{

**foreach** (**var** row **in** parking.LinkedScheduleRows)

{

**if** (row.IsUsedGseByType(type))

{

gseCount++;

}

}

}

**if** (gseCount > Parkings.Count)

gseCount = Parkings.Count;

**return** gseCount;

}

/// <summary>

/// Существует ли хоть одно использование СНО данного типа в этом блоке

/// </summary>

/// <param name="gseType"></param>

/// <returns></returns>

**private** **bool** **IsExistSingleByType**(GseType gseType)

{

**switch** (gseType)

{

**case** GseType.block:

**return** **IsExistBlock**();

**case** GseType.ladder:

**return** **IsExistLadder**();

**case** GseType.towBar:

**return** **IsExistTowBar**();

**case** GseType.markerCone:

**return** **IsExistMarkerCone**();

**default**:

**return** **false**;

}

}

/// <summary>

/// Существуют ли хоть одни колодки на блок

/// </summary>

/// <returns></returns>

**private** **bool** **IsExistBlock**()

{

**foreach** (**var** parking **in** Parkings)

{

**foreach** (**var** row **in** parking.LinkedScheduleRows)

{

**foreach** (**var** operation **in** row.LinkedTGO.Operations)

{

**if** (**string**.Equals(operation.Name, "Установка колодок")) **return** **true**;

}

}

}

**return** **false**;

}

/// <summary>

/// Существует ли хоть одна стремянка на блок

/// </summary>

/// <returns></returns>

**private** **bool** **IsExistLadder**()

{

**foreach** (**var** parking **in** Parkings)

{

**foreach** (**var** row **in** parking.LinkedScheduleRows)

{

**foreach** (**var** operation **in** row.LinkedTGO.Operations)

{

**if** (**string**.Equals(operation.Name, "Открытие грузовых люков")) **return** **true**;

}

}

}

**return** **false**;

}

/// <summary>

/// Существует ли хоть одно буксировочное водило на блок

/// </summary>

/// <returns></returns>

**private** **bool** **IsExistTowBar**()

{

**foreach** (**var** parking **in** Parkings)

{

**foreach** (**var** row **in** parking.LinkedScheduleRows)

{

**foreach** (**var** operation **in** row.LinkedTGO.Operations)

{

**if** (**string**.Equals(operation.Name, "Прибытие ВС")) **return** **true**;

}

}

}

**return** **false**;

}

/// <summary>

/// существует ли хоть один комплект конусов безопасности на блок

/// </summary>

/// <returns></returns>

**private** **bool** **IsExistMarkerCone**()

{

**foreach** (**var** parking **in** Parkings)

{

**foreach** (**var** row **in** parking.LinkedScheduleRows)

{

**foreach** (**var** operation **in** row.LinkedTGO.Operations)

{

**if** (**string**.Equals(operation.Name, "Установка конусов безопасности")) **return** **true**;

}

}

}

**return** **false**;

}

/// <summary>

/// Возвращает номера МС в блоке через запятую

/// </summary>

/// <returns></returns>

**internal** **string** **GetParkingsByComma**()

{

**var** str = "";

**foreach** (**var** parking **in** Parkings)

{

str += parking.Number + ",";

}

**return** str.TrimEnd(',');

}

}

Реализация класса место стоянки воздушного судна **Parking.cs**

/// <summary>

/// Место стоянки воздушного судна

/// </summary>

public class Parking

{

/// <summary>

/// Идентификатор объекта.

/// </summary>

public string Id { get; set; }

/// <summary>

/// Обозначение стоянки.

/// </summary>

public string Number { get; set; }

/// <summary>

/// Координаты стоянки.

/// </summary>

public List<Coordinate> Coordinates = new List<Coordinate>();

/// <summary>

/// Признак своим ходом туда добирается ВС или нет.

/// </summary>

public bool IsPropelled;

/// <summary>

/// Связанные Строки расписания

/// </summary>

public List<ScheduleRow> LinkedScheduleRows = new List<ScheduleRow>();

/// <summary>

/// Среднее по X коодрдинате

/// </summary>

/// <returns>число с плавающей точкой</returns>

public double MiddleX()

{

double sum = 0;

foreach (var c in Coordinates)

{

sum += c.X;

}

return sum / Coordinates.Count;

}

/// <summary>

/// Среднее по Y коодрдинате

/// </summary>

/// <returns>число с плавающей точкой</returns>

public double MiddleY()

{

double sum = 0;

foreach (var c in Coordinates)

{

sum += c.Y;

}

return sum / Coordinates.Count;

}

/// <summary>

/// Является ли МС контактным

/// </summary>

/// <returns></returns>

public bool IsContact()

{

var utk = new string[] {

"10", "11", "12", "13", "13A", "14", "15", "15A", "16", "17", "17A", "18", "19", "19A",

"20", "21", "21A", "22", "23", "24", "25", "26", "27", "27A", "28", "29", "29A",

"30", "31", "32", "32A", "33", "34", "34A", "35", "36", "37", "37A", "38", "39",

"40", "41", "42", "42A", "42C", "43", "43A", "43C", "44",

"44A", "44C", "45", "45A", "46", "47", "48", "48A", "49",

"50", "51", "52", "53", "53A", "54", "55", "56", "57", "57A", "58", "59","60"

};

var stk = new string[] {

"113", "114", "115", "116", "117", "118", "119",

"120", "120A", "121", "122", "123", "124", "125", "126",

"127", "128", "128A", "129", "130", "130A", "131"

};

return Array.Exists(utk, t => t == Number) || Array.Exists(stk, t => t == Number);

}

}

Реализация класса операции ТГО **Operation.cs**

/// <summary>

/// Объект операции

/// </summary>

**public** **class** **Operation**

{

/// <summary>

/// Наименование операции

/// </summary>

**public** **string** Name { **get**; **set**; }

/// <summary>

/// Тип СНО

/// </summary>

**public** **string** GseType { **get**; **set**; }

/// <summary>

/// Время начала операции

/// </summary>

**public** **string** StartTime { **get**; **set**; }

/// <summary>

/// Время окончания операции

/// </summary>

**public** **string** EndTime { **get**; **set**; }

}

Реализация класса отвечающего за формирование отчёта **ReportService.cs**

public static class ReportService

{

#region Excel

/// <summary>

/// Вывод отчёта в файл Excel

/// </summary>

/// <param name="objectManager"></param>

public static void WriteReportInExcel(ObjectManager objectManager)

{

// If you use EPPlus in a noncommercial context

// according to the Polyform Noncommercial license:

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

// Creating an instance of ExcelPackage

var package = new ExcelPackage();

// add sheet

var sheet = package.Workbook.Worksheets.Add("Отчёт");

AddReportHeader(sheet, objectManager.FromDate, objectManager.ToDate);

AddTableHeader(sheet);

var row = 5;

row = AddReportBody(objectManager, sheet, row);

row++;

AddReportSummary(objectManager, sheet, row);

SetStyles(sheet, row);

var fileName = objectManager.GetReportFileName();

SaveInExcel(package, fileName);

//Close Excel package

package.Dispose();

}

/// <summary>

/// Форматирование и настройка параметров листа Excel

/// </summary>

/// <param name="sheet">Рабочая страница</param>

/// <param name="row">Индекс строки</param>

private static void SetStyles(ExcelWorksheet sheet, int row)

{

// Setting the properties of the header table row

sheet.Row(4).Height = 20;

sheet.Row(4).Style.HorizontalAlignment = ExcelHorizontalAlignment.Center;

sheet.Row(4).Style.VerticalAlignment = ExcelVerticalAlignment.Center;

// покраска шапки

sheet.Cells[4, 1, 4, 4].Style.Fill.PatternType = ExcelFillStyle.Solid;

sheet.Cells[4, 1, 4, 4].Style.Fill.BackgroundColor.SetColor(System.Drawing.Color.LightGray);

sheet.Cells[4, 1, 4, 4].Style.Border.BorderAround(ExcelBorderStyle.Thin);

sheet.Column(2).Width = 15;

sheet.Column(3).AutoFit();

sheet.Column(4).AutoFit();

}

/// <summary>

/// Заголовок отчёта

/// </summary>

/// <param name="sheet"></param>

/// <param name="fromDate"></param>

/// <param name="toDate"></param>

private static void AddReportHeader(ExcelWorksheet sheet, DateTime fromDate, DateTime toDate)

{

sheet.Cells[2, 1].Value = $"Прогноз потребности в СНО на период с 00:00 {fromDate.ToShortDateString()} по 23:59 {toDate.ToShortDateString()}";

}

/// <summary>

/// Шапка таблицы отчёта

/// </summary>

/// <param name="sheet"></param>

private static void AddTableHeader(ExcelWorksheet sheet)

{

// Header of the Excel sheet

sheet.Cells[4, 1].Value = "Блок";

sheet.Cells[4, 2].Value = "МС";

sheet.Cells[4, 3].Value = "Тип СНО";

sheet.Cells[4, 4].Value = "Количество(шт)";

}

/// <summary>

/// Отрисовка тела отчёта

/// </summary>

/// <param name="objectManager"></param>

/// <param name="sheet"></param>

/// <param name="row"></param>

/// <returns></returns>

private static int AddReportBody(ObjectManager objectManager, ExcelWorksheet sheet, int row)

{

var startRow = row;

// TODO Каждый блок тонким контуром

foreach (var block in objectManager.ParkingBlocks)

{

sheet.Cells[row, 1].Value = block.Id;

sheet.Cells[row, 2].Value = block.GetParkingsByComma();

sheet.Cells[row, 1, row + 3, 1].Merge = true;

sheet.Cells[row, 2, row + 3, 2].Merge = true;

// Выравнивание

sheet.Cells[row, 1, row + 3, 2].Style.HorizontalAlignment = ExcelHorizontalAlignment.Center;

sheet.Cells[row, 1, row + 3, 2].Style.VerticalAlignment = ExcelVerticalAlignment.Center;

sheet.Cells[row, 1, row + 3, 2].AutoFitColumns();

for (int i = 1; i < 5; i++)

{

var typeGse = "";

var countGse = 0;

if (i == 1)

{

typeGse = "стремянки";

countGse = block.LadderGseCount;

}

if (i == 2)

{

typeGse = "упорные колодки";

countGse = block.BlockGseCount;

}

if (i == 3)

{

typeGse = "конуса безопасности";

countGse = block.MarkerConeGseCount;

}

if (i == 4)

{

typeGse = "буксировочные водила";

countGse = block.TowBarGseCount;

}

sheet.Cells[row, 3].Value = typeGse;

sheet.Cells[row, 4].Value = countGse;

row++;

}

}

sheet.Cells[startRow, 1, row - 1, 4].Style.Border.BorderAround(ExcelBorderStyle.Thin);

return row;

}

/// <summary>

/// Вывод результатов отчёта

/// </summary>

/// <param name="objectManager"></param>

/// <param name="sheet"></param>

/// <param name="row"></param>

private static void AddReportSummary(ObjectManager objectManager, ExcelWorksheet sheet, int row)

{

var startIndex = row;

sheet.Cells[row, 1].Value = "Всего на период";

sheet.Cells[row, 1, row, 2].Merge = true;

sheet.Cells[row, 1, row, 2].Style.Fill.PatternType = ExcelFillStyle.Solid;

sheet.Cells[row, 1, row, 2].Style.Fill.BackgroundColor.SetColor(System.Drawing.Color.Yellow);

for (int i = 1; i < 5; i++)

{

var typeGse = "";

var countGse = 0;

if (i == 1)

{

typeGse = "стремянки";

countGse = objectManager.GetGseCountByType(GseType.ladder);

}

if (i == 2)

{

typeGse = "упорные колодки";

countGse = objectManager.GetGseCountByType(GseType.block);

}

if (i == 3)

{

typeGse = "конуса безопасности";

countGse = objectManager.GetGseCountByType(GseType.markerCone);

}

if (i == 4)

{

typeGse = "буксировочные водила";

countGse = objectManager.GetGseCountByType(GseType.towBar);

}

sheet.Cells[row, 3].Value = typeGse;

sheet.Cells[row, 4].Value = countGse;

row++;

}

var usedRange = sheet.Cells[startIndex, 1, row - 1, 4];

usedRange.Style.Border.BorderAround(ExcelBorderStyle.Thick);

}

/// <summary>

/// Сохранение объекта в excel файл

/// </summary>

/// <param name="excel"></param>

private static void SaveInExcel(ExcelPackage excel, string fileName)

{

// file name with .xlsx extension

var fileReportPath = GetPathToExcel() + fileName +".xlsx";

if (File.Exists(fileReportPath))

File.Delete(fileReportPath);

// Create excel file on physical disk

var objFileStream = File.Create(fileReportPath);

objFileStream.Close();

// Write content to excel file

File.WriteAllBytes(fileReportPath, excel.GetAsByteArray());

}

/// <summary>

/// Получение пути к расположению отчёта

/// </summary>

/// <returns></returns>

public static string GetPathToExcel()

{

var path = System.Reflection.Assembly.GetExecutingAssembly().Location;

path = path.Substring(0, path.IndexOf("bin"));

return path + @"Source\";

}

}

Реализация класса отвечающего за хранение объектов ObjectManager**.cs**

/// <summary>

/// Менеджер существующих объектов

/// </summary>

public class ObjectManager

{

/// <summary>

/// Начало диапазона моделирования

/// </summary>

public DateTime FromDate;

/// <summary>

/// Окончание диапазона моделирования

/// </summary>

public DateTime ToDate;

/// <summary>

/// Количество стоянок в блоке

/// </summary>

public int CountParkingInBlock { get; set; }

/// <summary>

/// Строки расписания

/// </summary>

public List<ScheduleRow> ScheduleRows;

/// <summary>

/// Места стоянок

/// </summary>

public List<Parking> Parkings;

/// <summary>

/// Воздушные суда

/// </summary>

public List<Aircraft> Aircrafts;

/// <summary>

/// Места стоянок

/// </summary>

public List<ParkingBlock> ParkingBlocks;

/// <summary>

/// Все ТГО

/// </summary>

public List<TGO> TgoObjects;

/// <summary>

/// Конструктор класса

/// </summary>

public ObjectManager(DateTime from, DateTime to, int countParking)

{

FromDate = from;

ToDate = to;

CountParkingInBlock = countParking;

ScheduleRows = new List<ScheduleRow>();

Parkings = new List<Parking>();

Aircrafts = new List<Aircraft>();

ParkingBlocks = new List<ParkingBlock>();

TgoObjects = new List<TGO>();

}

/// <summary>

/// Формирование имени отчёта

/// </summary>

/// <returns></returns>

internal string GetReportFileName()

{

// processing

var year = DateTime.Now.ToString("yy");

var month = DateTime.Now.ToString("MM");

var day = DateTime.Now.ToString("dd");

var sDay = FromDate.ToString("dd");

var sMonth = FromDate.ToString("MM");

var endDay = ToDate.ToString("dd");

var endMonth = ToDate.ToString("MM");

var fileName = year + month + day + " ШХ - Прогноз потребности в СНО на " +

sDay + sMonth + " - " + endDay + endMonth + " - " + CountParkingInBlock.ToString() + "МС";

return fileName;

}

/// <summary>

/// Получить строки расписания в заданном диапазоне

/// </summary>

/// <returns></returns>

internal List<ScheduleRow> GetScheduleRows()

{

return ScheduleRows.Where(t => (FromDate <= Convert.ToDateTime(t.FlightDate))&& (ToDate >= Convert.ToDateTime(t.FlightDate))).ToList();

}

/// <summary>

/// Расчёт суммы всего СНО в зависимости от типа

/// </summary>

/// <param name="type">Тип СНО</param>

/// <returns>Количество СНО заданного типа</returns>

public int GetGseCountByType(GseType type)

{

var count = 0;

foreach (var block in ParkingBlocks)

{

count += block.GetGseCountByType(type);

}

return count;

}

}

Реализация метода с вычислениями в классе MainForm**.cs**

**private** **void** **ComputeButton\_Click**(**object** sender, EventArgs e)

{

// initialize

\_objectManager = **new** ObjectManager(dateTimePickerFrom.Value, dateTimePickerTo.Value, Convert.ToInt32(textBoxParkingsCount.Text));

//load

ExcelService.LoadData(textBoxWorkPath.Text, \_objectManager);

XmlService.LoadParkingCoordinates(\_objectManager);

XmlService.LoadTgoObjects(\_objectManager);

// linking

LinkingService.LinkScheduleRowsToParkings(\_objectManager);

LinkingService.LinkTgoToScheduleRows(\_objectManager);

// processing

// first step generate blocks

ParkingBlockService.FillParkingBlocks(\_objectManager, Convert.ToInt32(textBoxParkingsCount.Text));

// second handle blocks return gse count by types

**try**

{

// Compute

ProcessingService.HandleBlocks(\_objectManager);

// Write report

ReportService.WriteReportInExcel(\_objectManager);

}

**catch** (Exception ex)

{

MessageBox.Show("Exception" + ex.Message);

}

MessageBox.Show("Отчёт успешно сформирован!");

}

Приложение 2 примеры расписания и справочников

Пример XML файла одного МС 1.xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<geozoneType id="2137" ver="2">

<uuid>dc550c07-4e32-4da1-ad14-07bbc01759a8</uuid>

<name>1</name>

<type>Polygon</type>

<color>#f8fda2</color>

<transparency>50</transparency>

<allowed-speed>0.0</allowed-speed>

<use-as-address>false</use-as-address>

<enable-speed-monitoring>false</enable-speed-monitoring>

<status>1</status>

<geozone-root-id>1</geozone-root-id>

<geometry type="1">

<point x="37.397418" y="55.965221"/>

<point x="37.398239" y="55.96535"/>

<point x="37.398169" y="55.965512"/>

<point x="37.398196" y="55.96562"/>

<point x="37.397956" y="55.966134"/>

<point x="37.397752" y="55.96617"/>

<point x="37.397326" y="55.9661"/>

<point x="37.39702" y="55.966056"/>

<point x="37.396795" y="55.966004"/>

<point x="37.39687" y="55.965815"/>

<point x="37.396967" y="55.965608"/>

<point x="37.397091" y="55.96535"/>

<point x="37.397418" y="55.965221"/>

</geometry>

</geozoneType>

Пример XML файла одного ТГО TGO.xml

<?xml version="1.0" encoding="utf-8"?>

<TGOs>

<TGO id="1">

<airCompany>N4</airCompany>

<totalTime>00:60</totalTime>

<typePlane>ШФ</typePlane>

<type>Вылет</type>

<operations>

<operation>

<name>Прибытие ВС</name>

<gseType>Водила</gseType>

<startTime>00:00</startTime>

<endTime>00:00</endTime>

</operation>

<operation>

<name>Установка колодок</name>

<gseType>Колодки</gseType>

<startTime>00:00</startTime>

<endTime>00:05</endTime>

</operation>

<operation>

<name>Установка конусов безопасности</name>

<gseType>Конуса безопасности</gseType>

<startTime>00:05</startTime>

<endTime>00:10</endTime>

</operation>

<operation>

<name>Подгон первого трапа</name>

<gseType>Трапы</gseType>

<startTime>00:04</startTime>

<endTime>00:06</endTime>

</operation>

<operation>

<name>Открытие грузовых люков</name>

<gseType>Стремянки</gseType>

<startTime>00:08</startTime>

<endTime>00:10</endTime>

</operation>

<operation>

<name>Закрытие грузовых люков</name>

<gseType>Стремянки</gseType>

<startTime>00:42</startTime>

<endTime>00:45</endTime>

</operation>

<operation>

<name>Отгон первого трапа</name>

<gseType>Трапы</gseType>

<startTime>00:45</startTime>

<endTime>00:48</endTime>

</operation>

<operation>

<name>Уборка колодок</name>

<gseType>Колодки</gseType>

<startTime>00:40</startTime>

<endTime>00:45</endTime>

</operation>

<operation>

<name>Уборка конусов безопасности</name>

<gseType>Конуса безопасности</gseType>

<startTime>00:45</startTime>

<endTime>00:50</endTime>

</operation>

<operation>

<name>Отправление ВС</name>

<gseType>Водила</gseType>

<startTime>00:00</startTime>

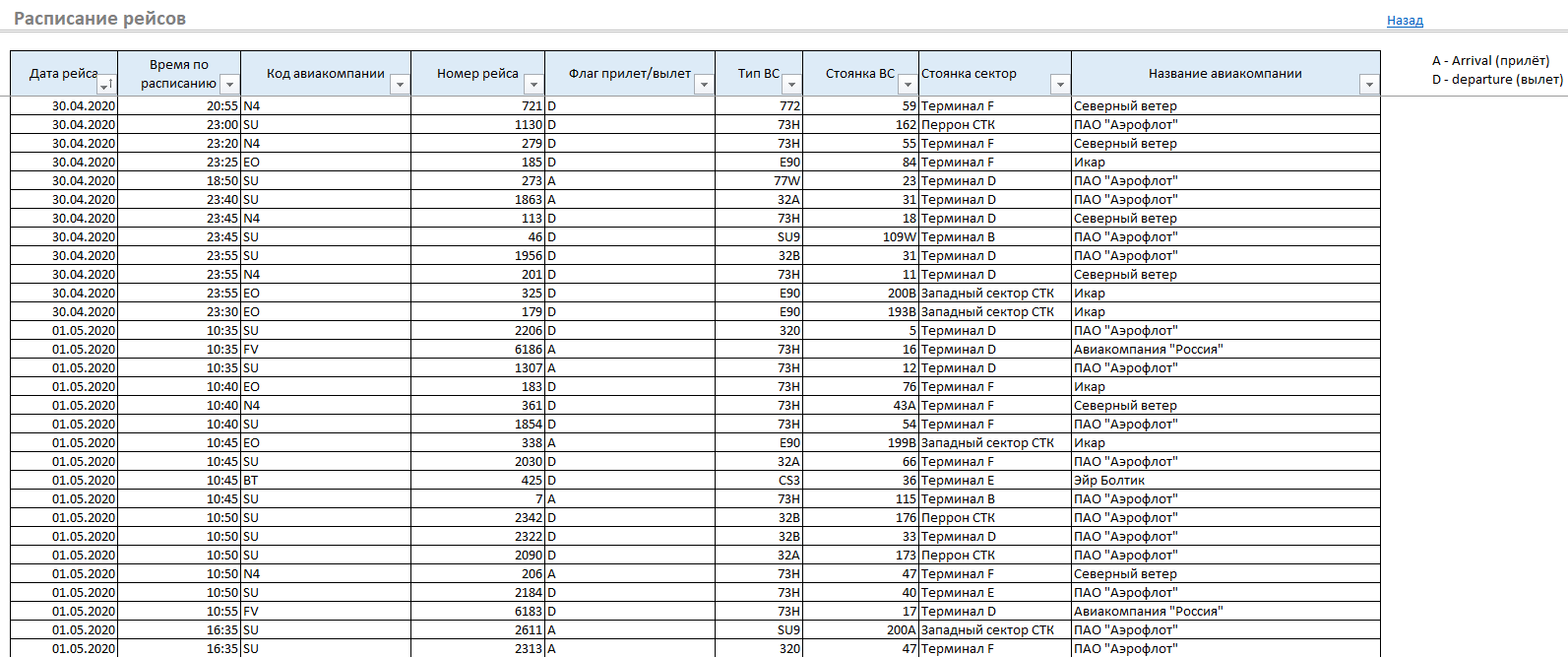
<endTime>00:00</endTime>

</operation>

</operations>

</TGO>

Фрагмент расписания из файла Excel



Фрагмент справочника типы ВС в Excel

