Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математикии программирования

РАБОТА ПРОВЕРЕНА
Рецензент, руководитель IT-отдела
Компании RBT, Попова Г.С.
_____(Попова Г. С.)
«12» июня 2016 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____А. А. Замышляева
«20» 07 2016 г.

Разработка модуля планирования нагрузки по кафедре

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–231000.2016.123.ПЗ ВКР

Руководитель работы, к.т.н.
доцент
_____/Т.Ю. Оленчикова
«12» 07 2016 г.
Автор работы
Студент группы ММиКН-474
_____/ А. Д. Федотенков
«12» 07 2016 г.
Нормоконтролер, к.ф.-м.н.
доцент
_____/С.У. Турлакова
«12» 07 2016 г.

Челябинск 2016

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра прикладной математики
Направление подготовки Прикладная математика и информатика

УТВЕРЖДАЮ
Заведующий кафедрой
_____ (Прокудин)
_____ 2015 г.

З А Д А Н И Е

на выпускную квалификационную работу студентки
Федотенкова Андрея Дмитриевича
Группа ММиКН-474

1. **Тема работы** Разработка модуля планирования нагрузки по кафедре.

   утверждена приказом по университету от « 15 » 04 2016 г. № 661

2. **Срок сдачи студентом законченной работы « 1» июня 2016 г.**

3. **Исходные данные к работе**
   3.1. MS SQL Server.
   3.2. Языки программирования: C++.

4. **Перечень вопросов, подлежащих разработке**
   4.1. Постановка задачи.
   4.2. Разработка структуры базы данных.
   4.3. Разработка интерфейса.
   4.4. Разработка информационного программы «Планировщик».
   4.5. Тестирование и отладка программного продукта.
   4.6. Разработка программной документации (описание программы, текст программы, руководство пользователя)

5. **Иллюстративный материал** (плакаты, альбомы, раздаточный материал, макеты, электронные носители и др.)
   5.1. Схема бизнес-процесса – 1 л.
   5.2. Схема базы данных и ее описание– 1 л.
   5.3. Общая схема информационной системы - 1 л.
   5.4. Мультимедийная презентация – 10 слайдов.
6. **Дата выдачи задания** «3» ноября 2015 г.

Руководитель _____ /Т.Ю. Оленчикова/

(подпись)

Задание принял к исполнению _____ /А.Д. Федотенков/

(подпись)

7. Календарный план

Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Отметка о выполнении руководителя
1. Постановка задачи	3.11.15 – 15.11.15	
3. Проектирование базы данных	16.11.15 – 06.01.16	
4. Разработка интерфейса	07.01.16 – 15.02.16	
5. Написание программы	16.02.16 – 16.04.16	
6. Тестирование и отладка программного продукта.	17.04.16 – 15.05.16	
7. Разработка программной документации	10.05.16 – 15.05.16	
8. Проверка работы руководителем, исправление замечаний	16.05.16 – 20.05.16	
9. Нормоконтроль	21.05.16 – 26.05.16	
10. Подготовка иллюстративного материала и доклада	23.05.16 – 28.05.16	
11. Рецензирование, представление зав. кафедрой	30.05.16 – 01.06.16	

Заведующий кафедрой _____ /Л.А. Прокудина/

(подпись)

Руководитель работы _____ /Т.Ю. Оленчикова/

(подпись)

Студент _____ /А.Д. Федотенков/

(подпись)

СОДЕРЖАНИЕ

АННОТАЦИЯ

Федотенков А.Д. Разработка модуля планирования нагрузки по кафедре.– Челябинск: ЮУрГУ, ММиКН-474, 54 с., 28 ил., 14 табл., библиогр. список – 14 наим., 2 прил., 1 л. плакатов ф. А1.

В выпускной квалификационной работе разработано приложения для ученого секретаря и преподавателей кафедры, позволяющее распределять учебную нагрузку по преподавателям кафедры. Рассмотрены существующие решения, выделены их недостатки.

Программная реализация осуществлена на языках C++ и MySQL.

В приложениях приведена программная документация на разработанный программный продукт.

ВВЕДЕНИЕ

Каждую четверть в школе, каждый семестр в университете составляют расписание занятий. Эта задача является одной из самых нужных задач, которую решают при планировании учебного процесса в каждом учебном заведении. Во-первых, это связано с тем, что без расписания занятий невозможно функционирование учебного заведения. Иными словами, расписание учебных занятий должно быть составлено своевременно и «качественно», т. е. отвечать ряду требований и критериев. В качестве таких критериев могут выступать критерии: отражающие экономическую эффективность использования имеющихся ресурсов образовательной системы, комфортность учебы студентов и работы ППС, ограничения по времени обучения и т. д.

Целью работы является разработка программы для автоматизации распределения нагрузки.

Составление расписания – это родственная задача распределения нагрузки между преподавателями. Важно правильно распределить нагрузку между преподавателями, чтобы учебный процесс был наиболее эффективным для учащихся.

Для преподавателя важно, чтобы нагрузка была правильно рассчитана, чтобы преподаватель вел «свою» дисциплину и имел необходимое и достаточное количество часов. Преподаватель не должен быть перегружен. Он должен иметь возможность и время уделить внимание каждой группе на занятие и каждому студенту на консультации. Поэтому группы должны быть грамотно разделены на подгруппы на практических занятиях и объединены на лекционных занятиях.

Количество часов занятий у преподавателя должно соответствовать его занимаем должность, научной степенью и ставкой.

Задачараспределения нагрузки относится к задаче целочисленного программирования, сложность решения которой растет экспоненциально с ростом числа и возможных значений варьируемых переменных (такие задачи относятся к классу NP-трудных задач). Кроме того, для нее характерно наличие большого объема различной по своему составу исходной информации и большого числа трудноформализуемых требований.

Выше перечисленные сложности препятствуют автоматизации процедуры распределения нагрузки, несмотря на наличие широкого спектра методов целочисленного программирования: методов полного перебора, метода ветвей и границ, метода раскраски графов, эвристических методов.

Так, в группе работ для автоматизации процедуры распределения нагрузки предлагаются подходы, основанные на так называемых точных (классических) методах и алгоритмах целочисленного программирования.

Недостатком данных методов является громоздкость и сложность получаемой математической модели задачи составления расписания, резкий рост временных затрат с ростом объемов исходной информации на поиск

решения в силу NP-сложного характера задачи составления расписания в ее классической постановке. Кроме того, в данных работах слабо учитываются структурные особенности объектов расписания и распределения нагрузки (преподаватели, группы, аудитории, дисциплины, временные интервалы занятий), между которыми существуют сильные связи, обусловленные спецификой организации учебного процесса. Так, например, преподаватели ведут занятия в строго определенных группах и в строго определенное время. Более того, занятия в группах ведутся по строго определенным дисциплинам в соответствии с учебным планом. Отметим, что часть из указанных объектов имеет иерархическую структуру. Так, например, учебные группы могут объединяться в потоки, включающие в себя группы одного курса одной специальности. Поэтому предлагаемые в данных работах подходы оказываются малоэффективными при составлении расписания занятий для образовательных систем массового обучения.

На данный момент в высшем учебном заведении можно использовать информационное пространство, включающее компоненты по различным направлениям: кадровый учет сотрудников, учет студентов, служба безопасности, рейтинг преподавателей, контроль успеваемости студентов и другие.

В настоящее время использование информационных систем в высших образовательных учреждениях не является редкостью. Спектр их применения широк и варьируется от автоматизации отдельно взятых рабочих мест до полной автоматизации деятельности ВУЗа.

Вне зависимости от объекта автоматизации, будь то преподавательский состав или администрация университета, в образовательном учреждении такие системы внедряют, преследуя конечную цель - повышение качества образования.

ВУЗ, как и любое предприятие, непременно проходит процесс автоматизации и, несмотря на то, что понятие образовательной деятельности едино для всех образовательных учреждений, в каждом ВУЗе этот процесс проходит по-разному. Значительное влияние на процессы автоматизации оказывает как наличие денежных средств, так и готовность использования предлагаемых рынком информационных услуг программных продуктов.

В связи с функционированием вузов в рамках единого информационного пространства, использование сторонних программных продуктов делается невозможным ввиду специфики работы имеющихся систем или же по причине дороговизны внедрения, влекущего значительную доработку как имеющихся, так и приобретаемых информационных систем.

С целью автоматизации планирования было разработано решение, упрощающее процесс создания электронного расписания на основе анализа имеющихся учебных планов специальностей, позволяющий анализировать структуру нагрузки, а также планировать структурную доработку и некоторую унификацию имеющихся учебных планов.

1 СОВРЕМЕННЫЕ ТЕХНОЛОГИИ СОСТАВЛЕНИЯ РАСПИСАНИЯ И ПЛАНИРОВАНИЯ НАГРУЗКИ

1.1. Анализ требований к составлению нагрузки по кафедре

Целью работы является разработка программы для автоматизации распределения нагрузки.

Задача распределения нагрузки преподавателя решается ежегодно и состоит в том, чтобы распределить нагрузку учебных планов данного учебного года между преподавателями кафедры в соответствии с их квалификацией, занимаемой должностью и ставкой. Нагрузку распределяет ученый секретарь кафедры.

Функциональные требования к ПО:

1. Исходными данными являются учебные планы специальностей в формате .xlsx (Приложение 1), нормативные документы ВУЗа по нагрузке, состав студентов, состав преподавателей и предметы, которые они хотели бы вести.
2. Выходные данные должны быть представлены в формате .xlsx и соответствовать двум типам документов:
   - Годовой учебный план по кафедре с указанием фамилии, имени, отчества преподавателей, обеспечивающих учебный процесс по данным дисциплинам.
   - Для каждого преподавателя расчет годовой нагрузки с указанием дисциплины.
3. Функциональные характеристики.
   - 3.1. Программа должна обеспечивать сбор данных формирования годового учебного плана на данный период с файлов учебных планов специальностей.
   - 3.2. Программа должна обеспечивать и поддерживать в актуальном состоянии информацию о преподавателях кафедры, их предпочтений относительно читаемых дисциплин, а также нормативную документацию ВУЗа по нагрузке.
   - 3.3. Предоставлять пользователю удаленный доступ к серверу кафедры.
   - 3.4. Автоматизированное распределение нагрузки с возможностью «ручной» коррекции.
   - 3.5. Формирование и печать выходных документов.
   - 3.6. Ученый секретарь имеет право объединять нагрузку в потоки, группы, назначать преподавателю конкретную дисциплину.
   - 3.7. Возможность для преподавателя просматривать свою нагрузку, комментировать, вводить и изменять предпочтения до утверждения годового учебного плана.
4. Требования к надежности.
   - 4.1. Конфиденциальность персональных данных преподавателей и учебных планов.

4.2. Возможность одновременной работы с системой до 50 человек.
4.3. Разграничение на уровне СУБД прав ученого секретаря и преподавателей кафедры.
4.4. Программа должна работать 24 часа 6 дней в неделю.

1.2 Существующие подходы решения задачи к составлению расписания.

1.2.1 Генетические алгоритмы

Оптимизация является важнейшим этапом решения задач идентификации. Основные трудности применения классических методов оптимизации нелинейных функций связаны с проблемами локального экстремума (рисунок 1.1) и «проклятия размерности» (рисунок 1.2).

Попытки преодоления указанных проблем привели к созданию теории генетических алгоритмов, которые выращивают оптимальное решение путем

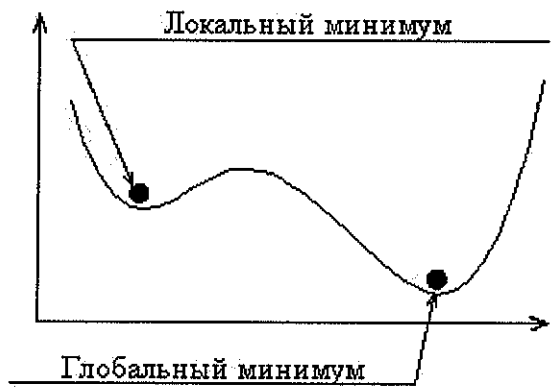


Рисунок 1.1 – Проблема локального экстремума

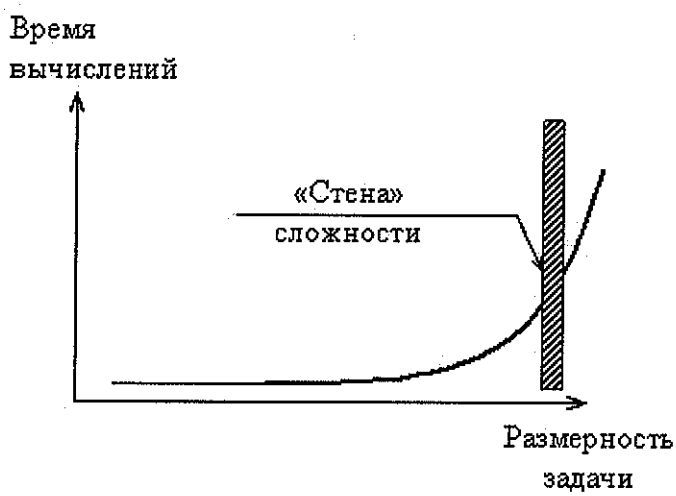


Рисунок 1.2 – Проблема «проклятия размерности»

скрещивания исходных вариантов с последующей селекцией по некоторому критерию. Генетический алгоритм (ГА) можно рассматривать как одну из

разновидностей случайного поиска, которая основана на механизмах, напоминающих естественный отбор и размножение [8].

В отличие от существующих методик, ГА начинает работу с некоторого случайного набора исходных решений, который называется популяцией. Каждый элемент из популяции называется хромосомой и представляет некоторое решение проблемы в первом приближении. Хромосома представляет собой строку символов некоторой природы, не обязательно бинарных. Хромосомы эволюционируют на протяжении множества итераций, носящих название поколений (или генераций). В ходе каждой итерации хромосома оценивается с использованием некоторой меры соответствия, которую мы будем называть функцией соответствия. Для создания следующего поколения новые хромосомы, называемые отпрысками, формируются либо путем скрещивания двух хромосом - родителей из текущей популяции, либо путем случайного изменения (мутации) одной хромосомы. Новая популяция формируется путем (а) выбора согласно функции соответствия некоторых родителей и отпрысков и (б) удаления оставшихся для того, чтобы сохранять постоянным размер популяции.

Хромосомы с большей функцией соответствия имеют больше шансов быть выбранными (выжить). После нескольких итераций алгоритм сходится к лучшей хромосоме, которая является либо оптимальным, либо близким к оптимальному решением. Пусть P(t) и C(t) являются родителями и отпрысками из текущей генерации t.

Процедура: Генетический алгоритм

```
begin
 t:=0 ;
 Задать_начальное_значение P(t);
 Оценить P(t) с помощью функции соответствия;
 while (нет условия_завершения) do
 Скрещивать P(t) чтобы получить C(t);
  Оценить C(t) с помощью функции соответствия;
  Выбрать P(t+1) из P(t) и C(t) ;
  t:=t+1 ;
 end
end.
```

Таким образом, используются два вида операций:

1. Генетические операции: скрещивание и мутация;
2. Эволюционная операция: выбор.

Генетические операции напоминают процесс наследования генов при создании нового отпрыска в каждой генерации. Эволюционная операция, осуществляющая переход от одной популяции к следующей, напоминает процесс Дарвиновской эволюции.

Операция скрещивания. Скрещивание является главной генетической операцией. Эта операция выполняется над двумя хромосомами- родителями и создает отпрыск путем комбинирования особенностей обоих родителей.

Приведем простейший пример скрещивания. В начале выберем некоторую случайную точку после этого создадим хромосому-отпрыск путем комбинирования сегмента первого родителя, стоящего слева от выбранной точки скрещивания, с сегментом второго родителя, стоящего по правую сторону от точки скрещивания, как это показано на рисунке 1.3.

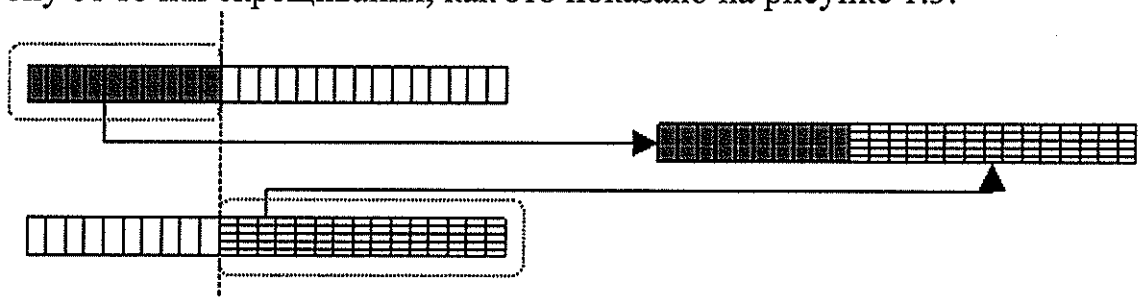


Рисунок 1.3 – Операция скрещивания

Этот метод работает очень хорошо, если хромосомы представляют собой битовые строчки. Кроме того производительность всего генетического алгоритма в первую очередь зависит от производительности используемой операции скрещивания.

Доля производимых на каждой итерации отпрысков называется коэффициентом скрещивания (P_c). Произведение (P_c) * размер_популяции показывает количество отпрысков. Большое значение этого коэффициента позволяет исследовать больше областей пространства поиска (или пространства решений) и уменьшает шанс попадания в локальный минимум. Но если значение P_c слишком велико, то это приведет к большим затратам времени вычислений на исследование бесперспективных областей.

Операция мутации. Мутация - это фоновая операция, производящая случайное изменение в различных хромосомах. Наипростейший вариант мутации состоит в случайном изменении одного или более генов. В ГА мутация играет важную роль для (а) восстановления генов, выпавших из популяции в ходе операции выбора, так что они могут быть опробованы в новых комбинациях, (б) формирования генов, которые не были представлены в исходной популяции. Интенсивность мутаций определяется коэффициентом мутаций (P_m). Он представляет собой долю генов, подвергающихся мутации на данной итерации, в расчете на их общее число. Слишком малое значение этого коэффициента приводит к тому, что многие гены, которые могли бы быть полезными, никогда не будут рассмотрены. В то же время слишком большое значение коэффициента (P_m) приведет к большим случайным возмущениям. Отпрыски перестанут быть похожими на родителей и алгоритм потеряет возможность обучаться, сохраняя наследственные признаки .

Поиск является одним из наиболее универсальных методов нахождения решения для случаев, когда априори не известна последовательность шагов, ведущая к оптимуму.

Существуют две поисковые стратегии: эксплуатация наилучшего решения и исследование пространства решений. Градиентный метод является примером

стратегии, которая выбирает наилучшее решение для возможного улучшения, игнорируя в то же время исследование всего пространства поиска. Случайный поиск является примером стратегии, которая, наоборот, исследует пространство решений, игнорируя исследование перспективных областей поискового пространства. Генетический алгоритм представляет собой класс поисковых методов общего назначения, которые комбинируют элементы обоих стратегий. Использование этих методов позволяет удерживать приемлемый баланс между исследованием и эксплуатацией наилучшего решения. В начале работы генетического алгоритма популяция случайна и имеет разнообразные элементы. Поэтому оператор скрещивания осуществляет обширное исследование пространства решений. С ростом значения функции соответствия получаемых решений оператор скрещивания обеспечивает исследование окрестностей каждого из них. Другими словами, тип поисковой стратегии (эксплуатация наилучшего решения или исследование области решений) для оператора скрещивания определяется разнообразием популяции, а не самим этим оператором.

В общем, алгоритм решения оптимизационных проблем представляет собой последовательность вычислительных шагов, которые асимптотически сходятся к оптимальному решению. Большинство классических методов оптимизации генерируют детерминированную последовательность вычислений, основанную на градиенте или производной целевой функции более высокого порядка. Эти методы применяются к одной исходной точке поискового пространства. Затем решение постепенно улучшается в направлении наискорейшего роста или убывания целевой функции. При таком поточечном подходе существует опасность попадания в локальный оптимум.

Генетический алгоритм осуществляет одновременный поиск по многим направлениям путем использования популяции возможных решений. Переход от одной популяции к другой позволяет избежать попадания в локальный оптимум. Популяция претерпевает нечто наподобие эволюции: в каждом поколении относительно хорошие решения репродуцируются, в то время как относительно плохие отмирают. ГА используют вероятностные правила для определения репродуцируемой или уничтожаемой хромосомы, чтобы направить поиск к областям вероятного улучшения целевой функции.

Существуют два главных преимущества генетических алгоритмов перед классическими оптимизационными методиками:

1. ГА не имеет значительных математических требований к видам целевых функций и ограничений. Исследователь не должен упрощать модель объекта, теряя ее адекватность, и искусственно добиваясь возможности применения доступных математических методов. При этом могут использоваться самые разнообразные целевые функции и виды ограничений (линейные и нелинейные), определенные на дискретных, непрерывных и смешанных универсальных множествах.

2. При использовании классических пошаговых методик глобальный оптимум может быть найден только в том случае, когда проблема обладает

свойством выпуклости. В то же время эволюционные операции генетических алгоритмов позволяют эффективно отыскивать глобальный оптимум.

Жадные алгоритмы

Для многих оптимизационных задач есть более простые и быстрые алгоритмы, чем динамическое программирование. В этой главе мы рассматриваем задачи, которые можно решать с помощью жадных алгоритмов (greedyalgorithms). Такой алгоритм делает на каждом шаге локально оптимальный выбор, - в надежде, что итоговое решение также окажется оптимальным. Это не всегда так – но для многих задач такие алгоритмы действительно дают оптимум. Наш первый пример – простая, но не вполне тривиальная задача о выборе заявок. Далее мы обсуждаем, для каких задач годятся жадные алгоритмы[14].

Не для всех задач жадный алгоритм дает оптимальное решение, но для нашей дает. Убедимся в этом. Проблема в том, что жадные алгоритмы дают локальные экстремумы, поэтому для многокритериальных задач оптимизации они плохо подходят.

Теорема 1. *Алгоритм* дает *набор из наибольшего возможного количества совместных заявок.*

Доказательство. Прежде всего докажем, что существует оптимальное решение задачи о выборе заявок, содержащее заявку номер1 (с самым ранним временем окончания). В самом деле, если в каком-то оптимальном множестве заявок заявка номер 1 не содержится, то можно заменить в нем заявку с самым ранним временем окончания не заявку номер 1, что не повредит совместности заявок (ибо заявка номер 1 кончается еще раньше, чем прежняя, и не с чем пересечься не может) и не изменит их общего количества. Стало быть, можно искать оптимальное решение, начинающееся с жадного выбора.

После того, как мы договорились рассматривать только наборы, содержащие заявку номер 1, все несовместные с ней заявки можно выкинуть, и задача сводится к выбору оптимального набора заявок из множества оставшихся заявок (совместных с заявкой номер 1). Другими словами, мы свели задачу к аналогичной задаче с меньшим числом заявок. Рассуждая по индукции, получаем, что, делая на каждом шаге жадный выбор, мы придем к оптимальному решению.

Как узнать, даст ли жадный алгоритм оптимум применительно к данной задаче? Общих рецептов тут нет, но существует две особенности, характерные для задач, решаемых жадными алгоритмами. Это принцип жадного выбора и свойство оптимальности для подзадач.

Говорят, что к оптимизационной задаче применим принцип жадного выбора, если последовательность локально оптимальных (жадных) выборов дает глобально оптимальное решение. Различие между жадными алгоритмами и динамическим программированием можно пояснить так: на каждом шаге жадный алгоритм берет "самый жирный кусок", а потом уже пытается сделать

наилучший выбор среди оставшихся, каковы бы они ни были; алгоритм динамического программирования принимает решение, просчитав заранее последствия для всех вариантов.

Как доказать, что жадный алгоритм дает оптимальное решение? Это не всегда тривиально, но в типичном случае такое доказательство следует схеме, использованной в доказательстве теоремы 1. Сначала мы доказываем, что жадный выбор на первом шаге не закрывает пути к оптимальному решению: для всякого решения есть другое, согласованное с жадным выбором и не худшее первого. Затем показывается, что подзадача, возникающая после жадного выбора на первом шаге, аналогична исходной, и рассуждение завершается по индукции.

Говоря иными словами, решаемые с помощью жадных алгоритмов задачи обладают свойством оптимальности для подзадач: оптимальное решение всей задачи содержит в себе оптимальные решения подзадач. (С этим свойством мы уже встречались, говоря о динамическом программировании). Например, при доказательстве теоремы 1 мы видели, что если A – оптимальный набор заявок, содержащий заявку номер 1, то A'=A \{1} – оптимальный набор заявок для меньшего множества заявок S', состоящего из тех заявок, для которых s_i ³f_1 .

И жадные алгоритмы, и динамическое программирование основываются на свойстве оптимальности для подзадач, поэтому может возникнуть искушение применить динамическое программирование в ситуации, где хватило бы жадного алгоритма, или, напротив, применить жадный алгоритм к задаче, в которой он не даст оптимума. Мы проиллюстрируем возможные ловушки на примере двух вариантов классической оптимизационной задачи.

Дискретная задача о рюкзаке состоит в следующем. Пусть вор пробрался на склад, на котором хранится n вещей. Вещь номер i стоит v_i долларов и весит w_i килограммов (v_i и w_i - целые числа). Вор хочет украсть товара на максимальную сумму, причем максимальный вес, который он может унести в рюкзаке, равен W (число W тоже целое). Что он должен положить в рюкзак?

Непрерывная задача о рюкзаке (fractionalknapsackproblem) отличается от дискретной тем, что вор может дробить краденые товары на части и укладывать в рюкзак эти части, а не обязательно вещи целиком (если в дискретной задаче вор имеет дело с золотыми слитками, то в непрерывной – с золотым песком).

Обе задачи о рюкзаке обладают свойством оптимальности для подзадач. В самом деле, рассмотрим дискретную задачу. Вынув вещь номер j из оптимально загруженного рюкзака, получим решение задачи о рюкзаке с максимальным весом $W - w_j$ и набором из n -1 вещи (все вещи, кроме j -й). Аналогичное рассуждение подходит и для непрерывной задачи: вынув из оптимально загруженного рюкзака, в котором лежит w килограммов товара номер j , весь этот товар, получим оптимальное решение непрерывной задачи, в которой максимальный вес равен W-w (вместо W), а количество j -го товара равно w_j - w (вместо w_j).

Хотя две задачи о рюкзаке и похожи, жадный алгоритм дает оптимум в непрерывной задаче о рюкзаке и не дает в дискретной. В самом деле, решение

непрерывной задачи о рюкзаке с помощью жадного алгоритма выглядит так. Вычислим цены (в расчете на килограмм) всех товаров (цена товара номер i равна v_i / w_i). Сначала вор берет по максимуму самого дорогого товара; если весь этот товар кончился, а рюкзак не заполнен, вор берет следующий по цене товар, затем следующий, и так далее, пока не наберет вес W. Поскольку товары надо предварительно отсортировать по ценам, на что уйдет время O($nlogn$), время работы описанного алгоритма будет O($n\ logn$).

Пример. Грузоподъемность рюкзака 50 кг, на складе имеются три вещи, весящие 10, 20 и 30 кг и стоящие 60, 100 и 120 долларов соответственно. Цена их в расчете на единицу веса равна 6,5 и 4. Жадный алгоритм для начала положит в рюкзак вещь номер 1; однако оптимальное решение включает предметы номер 2 и 3.

Для непрерывной задачи с теми же исходными данными жадный алгоритм, предписывающий начать с товара номер 1, дает оптимальное решение. В дискретной задаче такая стратегия не срабатывает: положив в рюкзак предмет номер 1, вор лишается возможности заполнить рюкзак «под завязку», а пустое место в рюкзаке снижает цену наворованного в расчете на единицу веса. Здесь, чтобы решить, класть ли данную вещь в рюкзак, надо сравнить решения двух подзадач: когда данная вещь заведомо лежит в рюкзаке и когда этой вещи в рюкзаке заведомо нет. Тем самым дискретная задача о рюкзаке порождает множество перекрывающихся подзадач – типичный признак того, что может пригодиться динамическое программирование. И действительно, к дискретной задаче о рюкзаке оно применимо.

В дискретной задаче о рюкзаке жадная стратегия может не сработать. Вор должен выбрать две вещи из трех с тем, чтобы их суммарный вес не превысил 50кг. Оптимальный выбор – вторая и третья вещи; если положить в рюкзак первую, то выбор оптимальным не будет, хотя именно она дороже всех в расчете на единицу веса. Для непрерывной задачи о рюкзаке с теми же исходными данными выбор товаров в порядке убывания цены на единицу веса будет оптимален.

Муравьиный алгоритм

Муравьи относятся к социальным насекомым, образующим коллективы. Коллективная система способна решать сложные динамические задачи по выполнению совместной работы, которая не могла бы выполняться каждым элементом системы в отдельности в разнообразных средах без внешнего управления, контроля или координации. [8]

В таких случаях говорят о роевом интеллекте (Swarmintelligence), как о замысловатых способах кооперативного поведения, то есть стратегии выживания. Одним из подтверждений оптимальности поведения муравьиных колоний является тот факт, что сеть гнёзд суперколоний близка к минимальному основному дереву графа их муравейников. Основу поведения

муравьиной колонии составляет самоорганизация, обеспечивающая достижения общих целей колонии на основе низкоуровневого взаимодействия.

Колония не имеет централизованного управления, и её особенностями являются обмен локальной информацией только между отдельными особями (прямой обмен – пища, визуальные и химические контакты) и наличие непрямого обмена, который и используется в муравьиных алгоритмах. Таким образом, в общем случае рассматриваются слепые муравьи, не способные чувствовать близость пищи. Непрямой обмен – стигмержи (stigmergy), представляет собой разнесённое во времени взаимодействие, при котором одна особь изменяет некоторую область окружающей среды, а другие используют эту информацию позже, когда в неё попадают. Биологи установили, что такое отложенное взаимодействие происходит через специальное химическое вещество – феромон (pheromone), секрет специальных желёз, откладываемый при перемещении муравья. Концентрация феромона на пути определяет предпочтительность движения по нему. Адаптивность поведения реализуется испарением феромона, который в природе воспринимается муравьями в течение нескольких суток. Мы можем провести некоторую аналогию между распределением феромона в окружающем колонию пространстве, и «глобальной» памятью муравейника, носящей динамический характер. Муравьиные алгоритмы представляют собой вероятностную жадную эвристику, где вероятности устанавливаются, исходя из информации о качестве решения, полученной из предыдущих решений. Они могут использоваться как для статических, так и для динамических комбинаторных оптимизационных задач. Сходимость гарантирована, то есть в любом случае мы получим оптимальное решение, однако скорость сходимости неизвестна.

Идея муравьиного алгоритма – моделирование поведения муравьёв, связанного с их способностью быстро находить кратчайший путь от муравейника к источнику пищи и адаптироваться к изменяющимся условиям, находя новый кратчайший путь. При своём движении муравей метит путь феромоном, и эта информация используется другими муравьями для выбора пути. Это элементарное правило поведения и определяет способность муравьёв находить новый путь, если старый оказывается недоступным [12].

Рассмотрим случай, показанный на рисунке, когда на оптимальном доселе пути возникает преграда. В этом случае необходимо определение нового оптимального пути. Дойдя до преграды, муравьи с равной вероятностью будут обходить её справа и слева. То же самое будет происходить и на обратной стороне преграды. Однако, те муравьи, которые случайно выберут кратчайший путь, будут быстрее его проходить, и за несколько передвижений он будет более обогащён феромоном. Поскольку движение муравьёв определяется концентрацией феромона, то следующие будут предпочитать именно этот путь, продолжая обогащать его феромоном до тех пор, пока этот путь по какой-либо причине не станет недоступен.

Очевидная положительная обратная связь быстро приведёт к тому, что кратчайший путь станет единственным маршрутом движения большинства

муравьёв. Моделирование испарения феромона – отрицательной обратной связи – гарантирует нам, что найденное локально оптимальное решение не будет единственным – муравьи будут искать и другие пути. Если мы моделируем процесс такого поведения на некотором графе, рёбра которого представляют собой возможные пути перемещения муравьёв, в течение определённого времени, то наиболее обогащённый феромоном путь по рёбрам этого графа и будет являться решением задачи, полученным с помощью муравьиного алгоритма.

Любой муравьиный алгоритм, независимо от модификаций, представим в следующем виде.

Пока (условия выхода не выполнены)

1. Создаём муравьёв
2. Ищем локальные решения
3. Обновляем феромон
4. Дополнительные действия {опционально}

Теперь рассмотрим каждый шаг в цикле более подробно.

1. Создаём муравьёв
   - Стартовая точка, куда помещается муравей, зависит от ограничений, накладываемых условиями задачи. Потому что для каждой задачи способ размещение муравьёв является определяющим. Либо все они помещаются в одну точку, либо в разные с повторениями, либо без повторений.
   - На этом же этапе задаётся начальный уровень феромона. Он инициализируется небольшим положительным числом для того, чтобы на начальном шаге вероятности перехода в следующую вершину не были нулевыми.
2. Ищем решения
   - Вероятность перехода из вершины i в вершину j определяется по следующей формуле:

$$p_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}(\frac{1}{d_{ij}})^{\beta}}{\sum_{j \in allowed\ nodes} \tau_{ij}(t)^{\alpha}(\frac{1}{d_{ij}})^{\beta}}$$

где - τ_{ij} уровень феромона,– d_{ij} эвристическое расстояние, а α,β – константные параметры. При α = 0 выбор ближайшего города наиболее вероятен, то есть алгоритм становится жадным. При β = 0 выбор происходит только на основании феромона, что приводит к субоптимальным решениям. Поэтому необходим компромисс между этими величинами, который находится экспериментально.

3. Обновляем феромон

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k \epsilon Colony\ that\ used\ edge\ (i,j)} \frac{Q}{L_k(t)}$$

• Уровень феромона обновляется в соответствии с приведённой формулой. Где ρ – интенсивность испарения, $L_k(t)$ – цена текущего решения для k-ого муравья, а Q – параметр, имеющий значение порядка цены оптимального решения, то есть $\frac{Q}{L_k(t)}$ феромон, откладываемый k-ым муравьём, использующим ребро (i,j).

Для того чтобы построить подходящий муравьиный алгоритм для решения какой-либо задачи, нужно:

1. Представить задачу в виде набора компонент и переходов или набором неориентированных взвешенных графов, на которых муравьи могут строить решения.
2. Определить значение следа феромона
3. Определить эвристику поведения муравья, когда строим решение
4. Если возможно, то реализовать эффективный локальный поиск
5. Выбрать специфический ACO алгоритм и применить для решения задачи
6. Настроить параметр ACO алгоритма.

Также определяющими являются

- Количество муравьёв.
- Баланс между изучением и использованием.
- Сочетание с жадными эвристиками или локальным поиском.
- Момент, когда обновляется феромон.

1.3 Существующие системы для составления расписания и распределения нагрузки, их возможности, достоинства и недостатки

1.3.1"Ректор-ВУЗ"

Данная версия программы используется для составления расписания занятий в университете и распределение нагрузки преподавателей (рисунок 1.4-1.6)

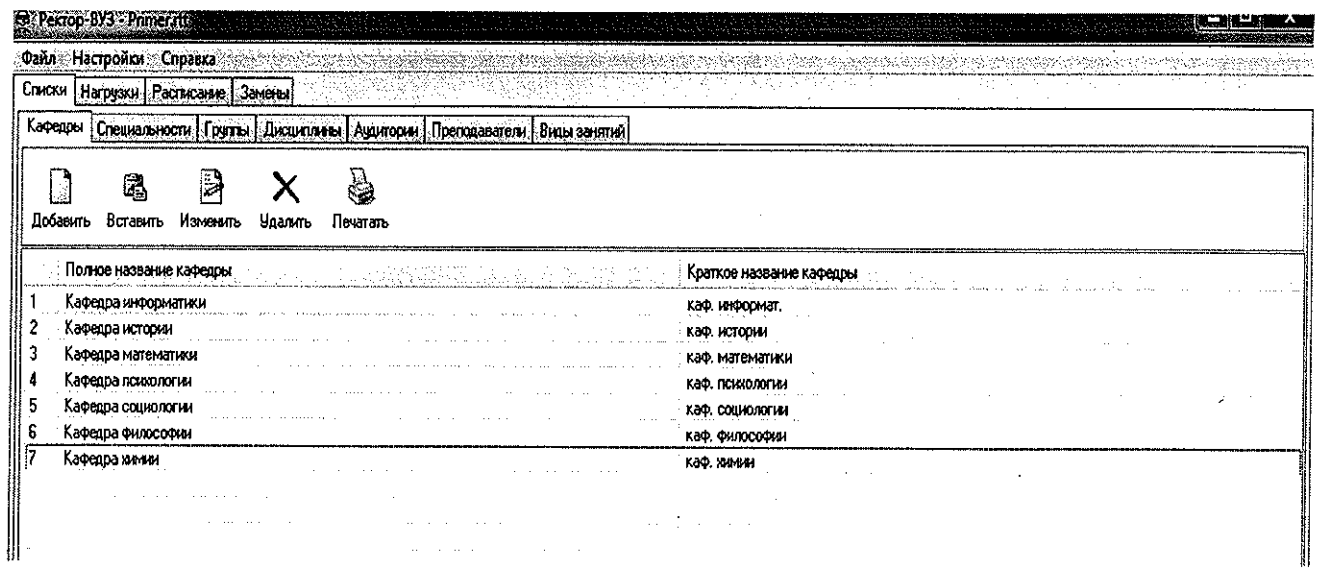


Рисунок 1.4 – Главное окно «Ректор-ВУЗ»

Меню программы: «Списки», «Нагрузки», «Расписание», «Замены»

Вкладка «Списки» позволяет вводить, редактировать и выводить на печать списки специальностей, дисциплин, групп, аудиторий и преподавателей.

Вкладка «Нагрузки» позволяет вводить, редактировать и выводить на печать учебные планы по каждой специальности, нагрузку преподавателя, график распределения часов по неделям, отчеты по нагрузке преподавателей.

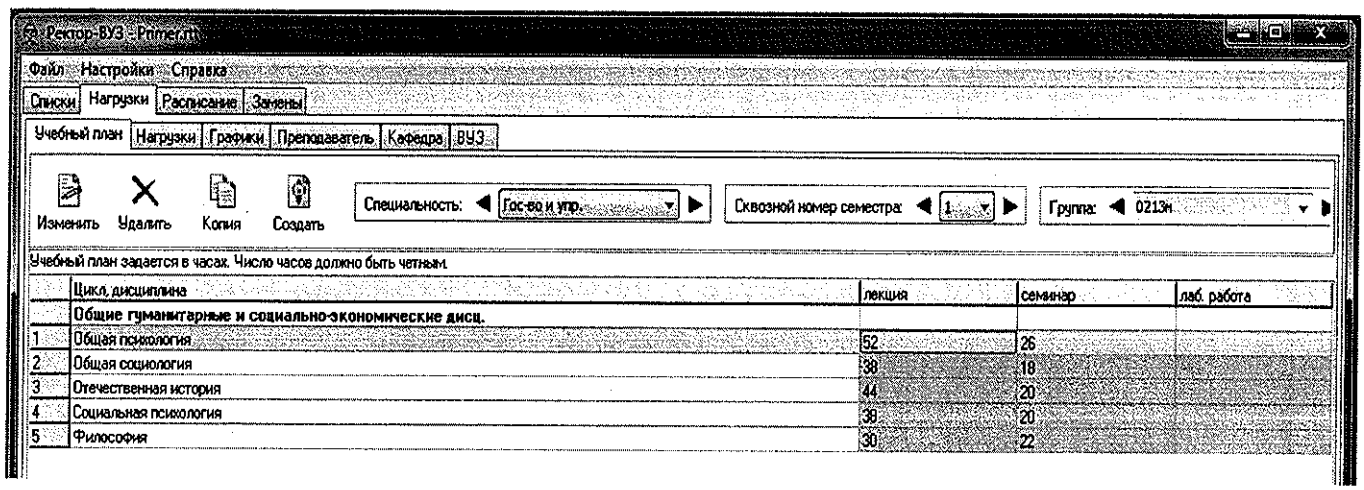


Рисунок 1.5 – Окно «Нагрузка» программы «Ректор-ВУЗ»

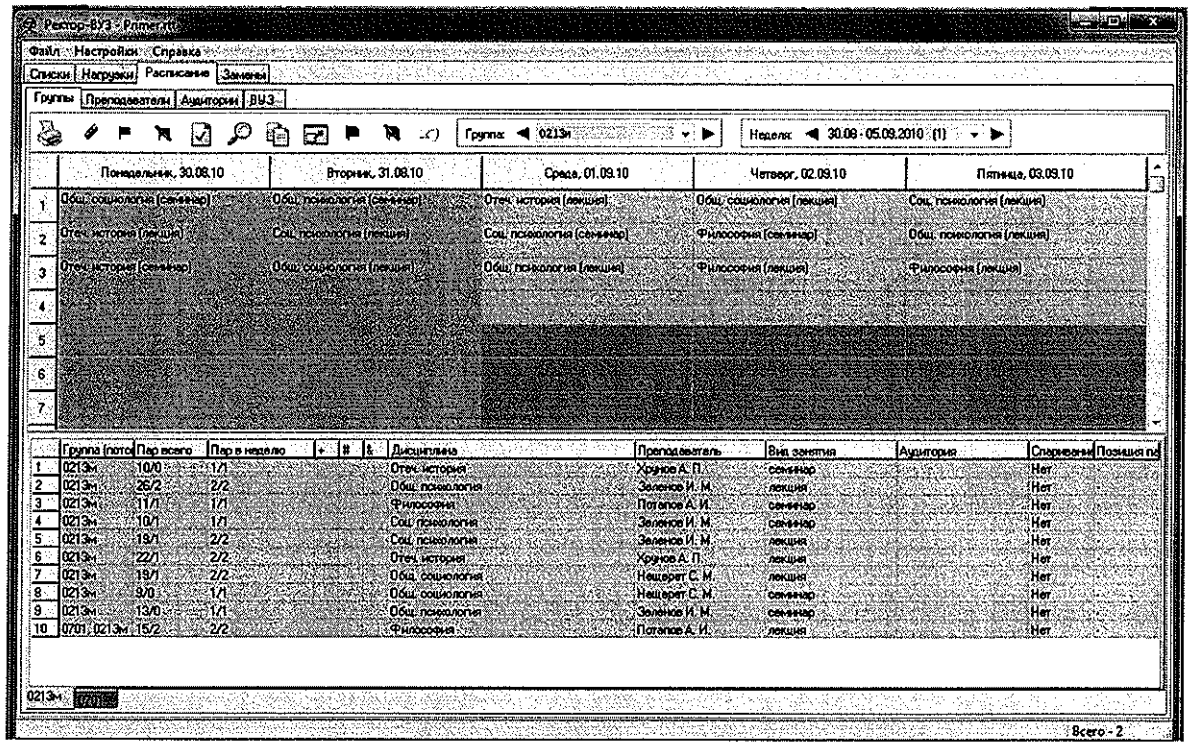


Рисунок 1.6 – Окно расписания программы «Ректор-ВУЗ»

Вкладка «Расписание» используется для составления расписания по преподавателям, группам, аудиториям и целому ВУЗу. В программе существует несколько типов составления расписания: ручной, автоматический и комбинированный. В автоматическом режиме программа отслеживает требования к расписанию и придерживается им. В ручном режиме программа подсказывает пользователю несколько возможных вариантов составления расписания группы или преподавателя, следит за образованием «окон» в расписании и количеством мест в аудитории.

Сформированное расписание и нагрузку можно сохранить в форматах «.doc», «.docx», «.xls», «.xslx»и «.pdf».

Вкладка «Замены» позволяет оперировать заменами преподавателей.

Минусы:

- сложный и интуитивно непонятный графический интерфейс;
- отсутствие регистрации пользователей с разными права доступа;
- высокая цена покупки и поддержки программного продукта.

1.3.2 Экспресс-расписание ВУЗ

На рисунках 1.7-1.8 представлен интерфейс программы «Экспресс-расписание ВУЗ».

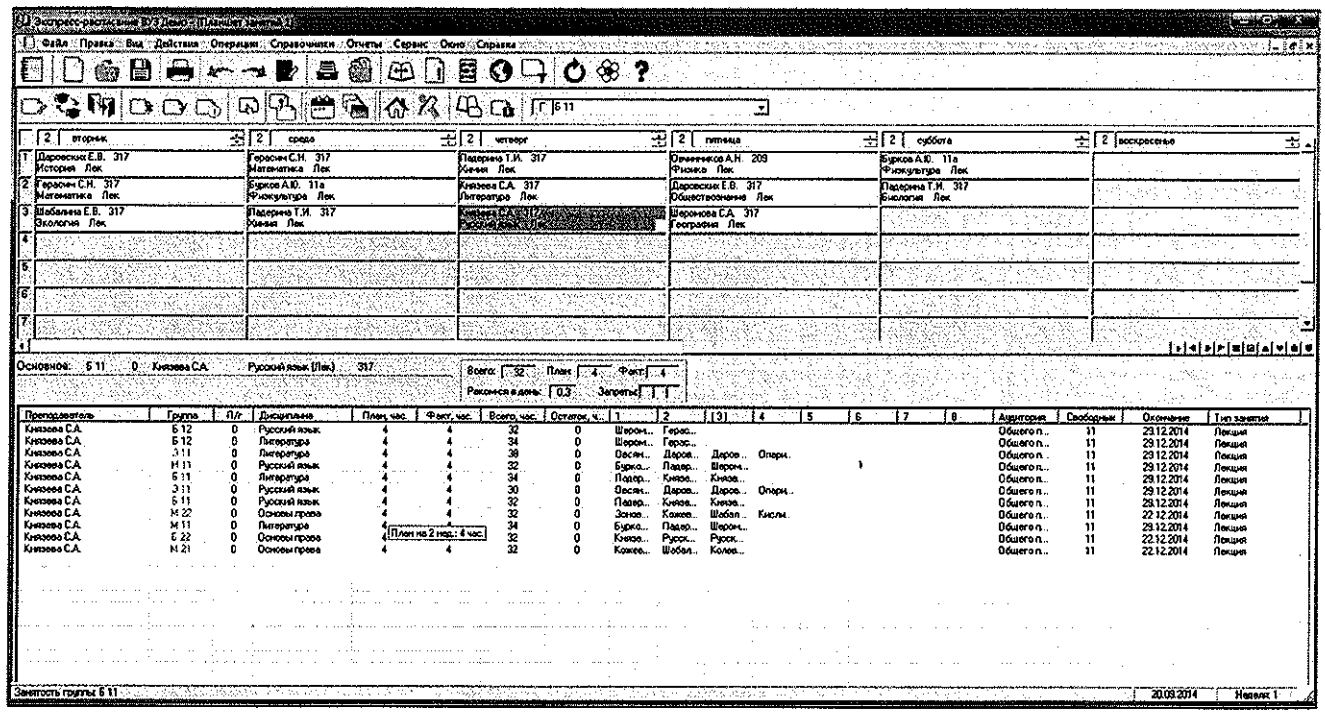


Рисунок 1.7 – Главное окно программы «Экспресс-расписание ВУЗ»

Программа автоматически составляет основное расписание, позволяет вести ежедневные изменения расписания, учет выполненных часов, формирует разнообразные отчеты.

В программе можно просмотреть журнал замен занятий, лист проблем и журнал неназначенных занятий, распечатать расписание и нагрузку.

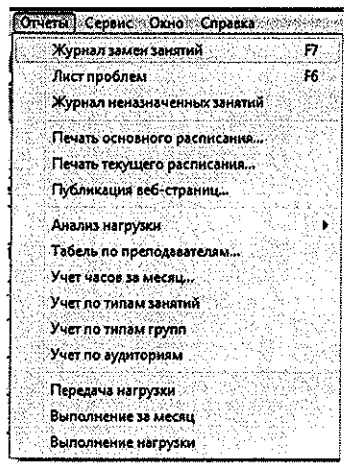


Рисунок 1.8 – Вкладка меню «Отчеты»

Программа составляет отчеты по нагрузке в разрезе типов занятий, групп, аудиторий и преподавателей, просмотреть прогресс выполнения нагрузки преподавателей за определенный период.

При автоматическом составлении основного расписания учитываются запреты, то есть графики преподавателей совместителей, методические дни и т.п. Полученное расписание можно корректировать вручную.

У этого программного продукта минусы совпадают с минусами продукта, описанного выше. Но было отмечено несколько полезных функций:

- Учитываются непроведенные занятия, по причине болезни, командировки преподавателя;
- Составление различных отчетов по нагрузке преподавателей.

1.3.3 Система «Составление расписания»

Данная система составления расписания позволяет составлять расписание и распределять нагрузку по преподавателям в автоматическом режиме. Программа анализирует данные преподавателей, групп, учебные планы и перебирает множество вариантов расписания (Рисунок 1.9).

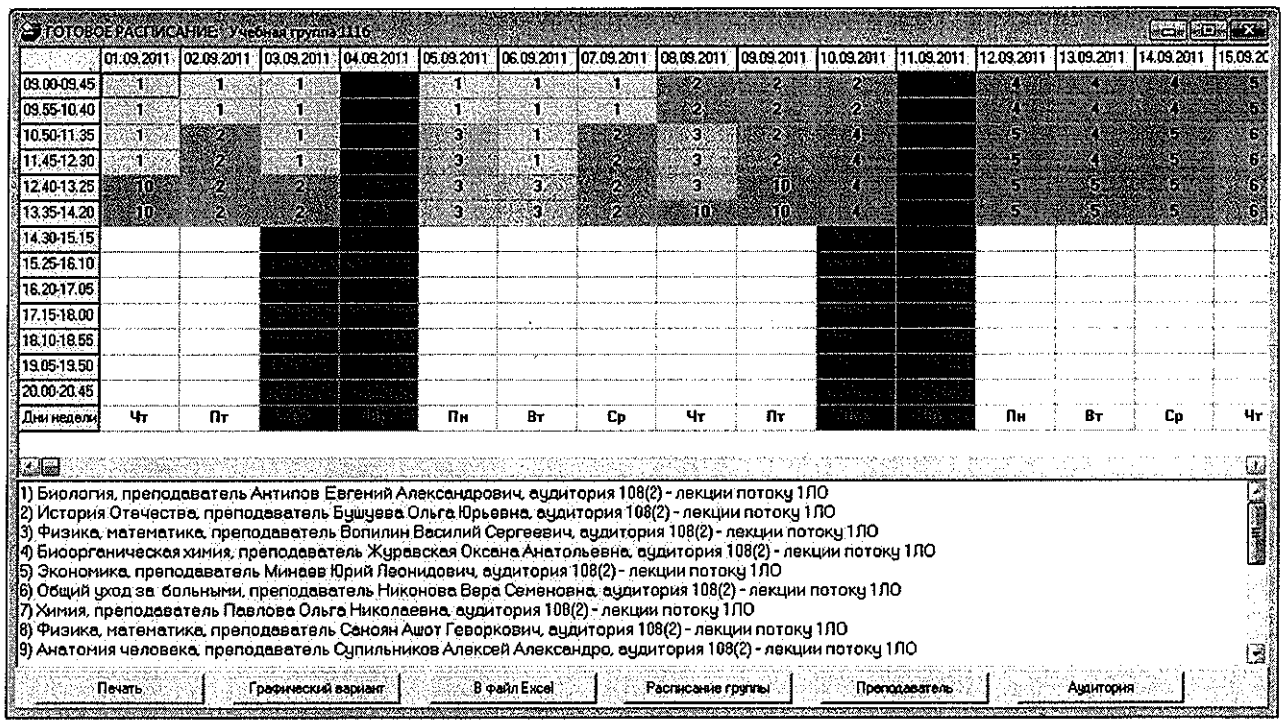


Рисунок 1.9 – Окно расписания

Существует возможность объединять группы в потоке, производить замены преподавателей, изменять время проведения занятий.

Можно вывести на печать расписание по аудиториям, преподавателям и группам.

Программа может работать на сервере.

У данного продукта все также присутствуют выше перечисленные минусы, за исключением интерфейса: в этой программе он довольно понятен и удобен.

1.4 Постановка задачи и цели работы

В результате проведения анализа существующего программного обеспечения было выявлено несколько недостатков:

- Все программные продукты имеют только платную лицензию;
- В программы нельзя загружать свои данные из файлов формата .xlsx;
- В программах нет разделения пользователей: администратор и преподаватель.

Цель работы: разработать программу для автономного распределения нагрузки по кафедре.

Для достижения цели необходимо решить следующие задачи:

1. разработка и реализовать базу данных;
2. разработать архитектуру приложения, удовлетворяющего функциональным требованиям;
3. разработать интерфейс пользователей;
4. разработать алгоритм автоматизированного распределения нагрузки;
5. реализовать и отладить приложение.

1.5 Выводы

Проанализировав существующие программы для составления расписания и распределения нагрузки, была составлена сводная таблица (Таблица 1) по функционалу этих программ и выявлены функции, которые будут реализованы в приложении.

Таблица 1

Функции существующего ПО

Функции	Ректор-ВУЗ	Экспресс расписание ВУЗ	Система расписания	Наш программный продукт
Загрузка учебных планов из файла	-	-	-	+
Объединение групп в потоки	+	-	+	+
Просмотр и печать нагрузки преподавателей	+	+	+	+
Регистрация с разделением прав пользователей	-	+	-	+
Ручная корректировка	+	+	+	+
Принудительное назначение и замена	+	+	+	+
Сохранение учебных планов в excel.	+	-	+	+
Комментарии пользователями	-	-	-	+
Приоритет дисциплины у преподавателя	-	-	-	+
Добавление новой информации о преподавателях, группах, дисциплинах.	+	+	+	+
Платная лицензия и обслуживание	+	+	+	-

Для удовлетворения функциональных требований к программе, достаточно разработать Desktop-приложение, так как:

- количество пользователей не будет превышать 50 человек;
- работа приложения будет осуществляться в пределах кафедры;
- все пользователи приложения имеют доступ к серверу кафедры;
- desktop-приложение решает требование к безопасности о сохранение данных о преподавателях, студентах и готовых учебных планах.

Решено разработать Desktop-приложение в среде BorlandC++ Builder с использованием СУБД MS SQL Server.

2 РАЗРАБОТКА АРХИТЕКТУРЫ МОДУЛЯ СОСТАВЛЕНИЯ РАСПИСАНИЯ

Диаграмма вариантов использования показывает взаимодействие между вариантами использования и действующими лицами. Она отражает требования к системе с точки зрения пользователя (Рисунок 2.1).

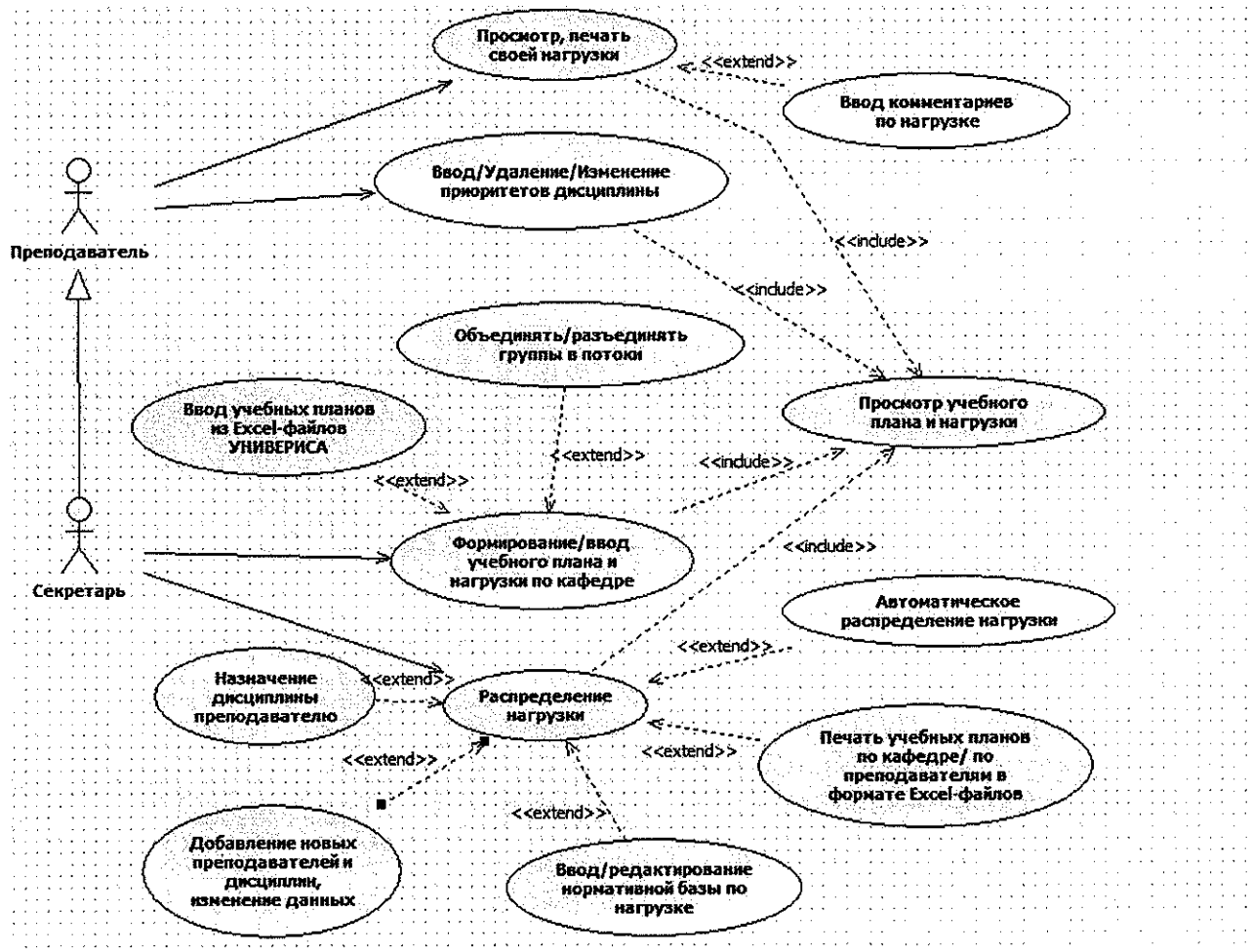


Рисунок 2.1 – Диаграмма Use-Case

В приложении существует два типа учетной записи с разными правами доступа: преподаватель и ученый секретарь.

Преподаватель имеет доступ только к своей нагрузке и может воспользоваться следующим функционалом:

1) просмотр и печать распределенной нагрузки на текущий семестр, учебный год;
2) преподаватель может добавлять приоритет к дисциплинам, которые он хотел бы преподавать больше и меньше, в каких он наиболее или наименее компетентен. Преподаватель может удалить или изменить свой приоритет до распределения нагрузки и начала учебного года;

3) преподаватель может оставлять комментарии ученому секретарю (администратору приложения).

Преподаватель – это пользователь системы, поэтому у него нет доступа к изменению, корректировки и удалению учебной нагрузки.

Администратором системы является ученый секретарь, который распределяет нагрузку по всем преподавателям кафедры.

Он имеет следующие функции:

1) загрузка учебных планов в систему и формирование нагрузки из этих учебных планов.
2) корректировка сформированной учебной нагрузки.
3) ручной ввод нагрузки, которая была предоставлена другими кафедрами университета, на которых преподаватели кафедры проводят занятия. Эта информация предоставляется учеными секретарями других кафедр в бумажном виде, поэтому администратору системы нужно вбивать эту нагрузку вручную.
4) объединение групп в потоки. Если у нескольких групп совпадает дисциплина и количество лекционных часов по этой дисциплине, то ученый секретарь может объединить эти группы в поток.
5) ученый секретарь может назначить преподавателю дисциплину, чтобы уменьшить нагрузку одному преподавателю или увеличить другому преподавателю.
6) ученый секретарь может добавлять/удалять/изменять данные о преподавателях (личные данные, ставка, ученая степень), дисциплины, учебные планы.
7) печать готовой учебной нагрузки.

Пример потока событий варианта использования «Добавление/изменение/удаление приоритета» преподавателем:

1. Вариант использования начинается, когда пользователь запускает приложение.
2. Система выводит приветствие и предлагает пользователю ввести логин и пароль.
3. Пользователь вводит логин и пароль.
4. Система проверяет введенный логин и пароль. Если логин или пароль не верны, то выполняется альтернативный поток событий А1.
5. Система выводит на экран учебный план преподавателя.
6. Пользователь выбирает учебный год и семестр. Нажимает кнопку печать.
7. Система открывает окно «Печать» с настройками печати.
8. Пользователь выбирает настройки печать и нажимает кнопку «ОК». Если документ не был напечатан, то выполняется альтернативный поток событий А2.

9. Вариант использования завершается (рисунок 2.2).

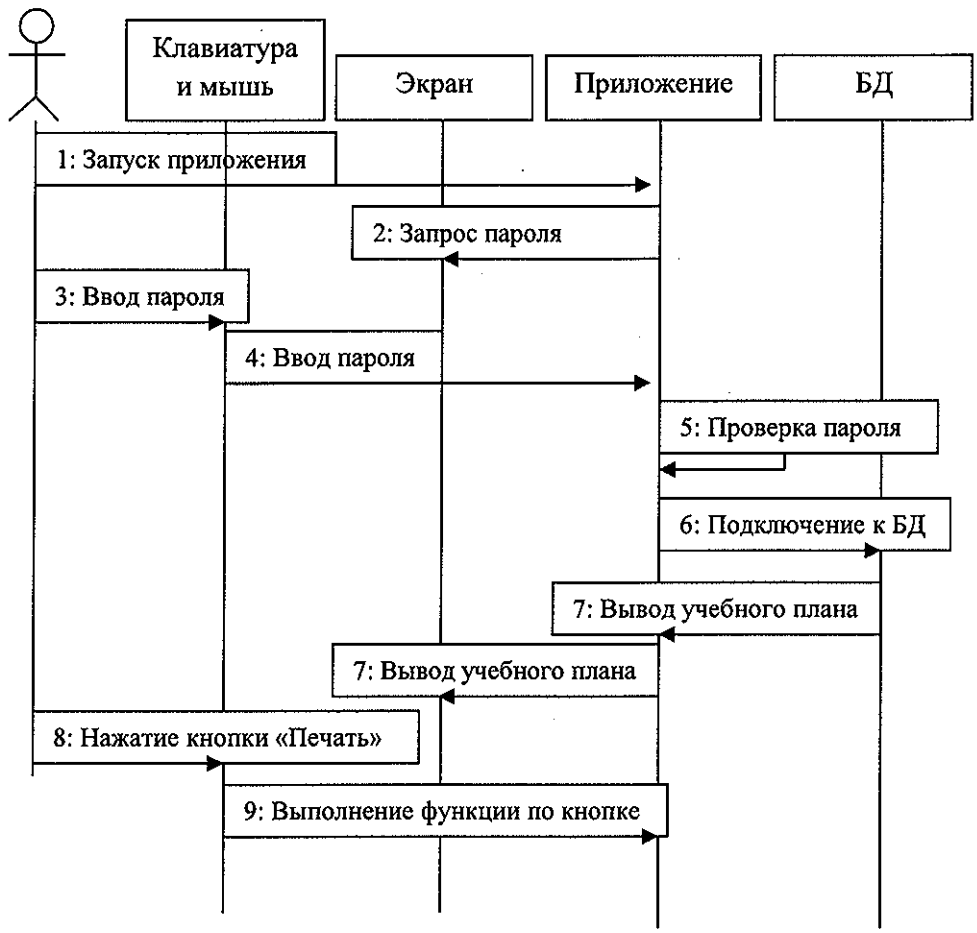


Рисунок 2.2 – Диаграмма последовательности

Альтернативный поток событий А1.

1. Система информирует пользователя, что логин или пароль введены неправильно и предлагает повторить попытку.
2. Альтернативный поток событий А1 завершается.

Альтернативный поток событий А2.

1. Система информирует пользователя, что документ не был напечатан и предлагает проверить настройки принтера и повторить попытку.
2. Альтернативный поток событий А2 завершается.

3 РАЗРАБОТКА АЛГОРИТМА РАСПРЕДЕЛЕНИЯ НАГРУЗКИ

3.1. Основной алгоритм программы

На рисунке 3.1 представлен основный алгоритм программы.

Рисунок 3.1 - Общая схема алгоритма работы программы

Алгоритм работы процедуры авторизации приведен на рисунке 3.2.

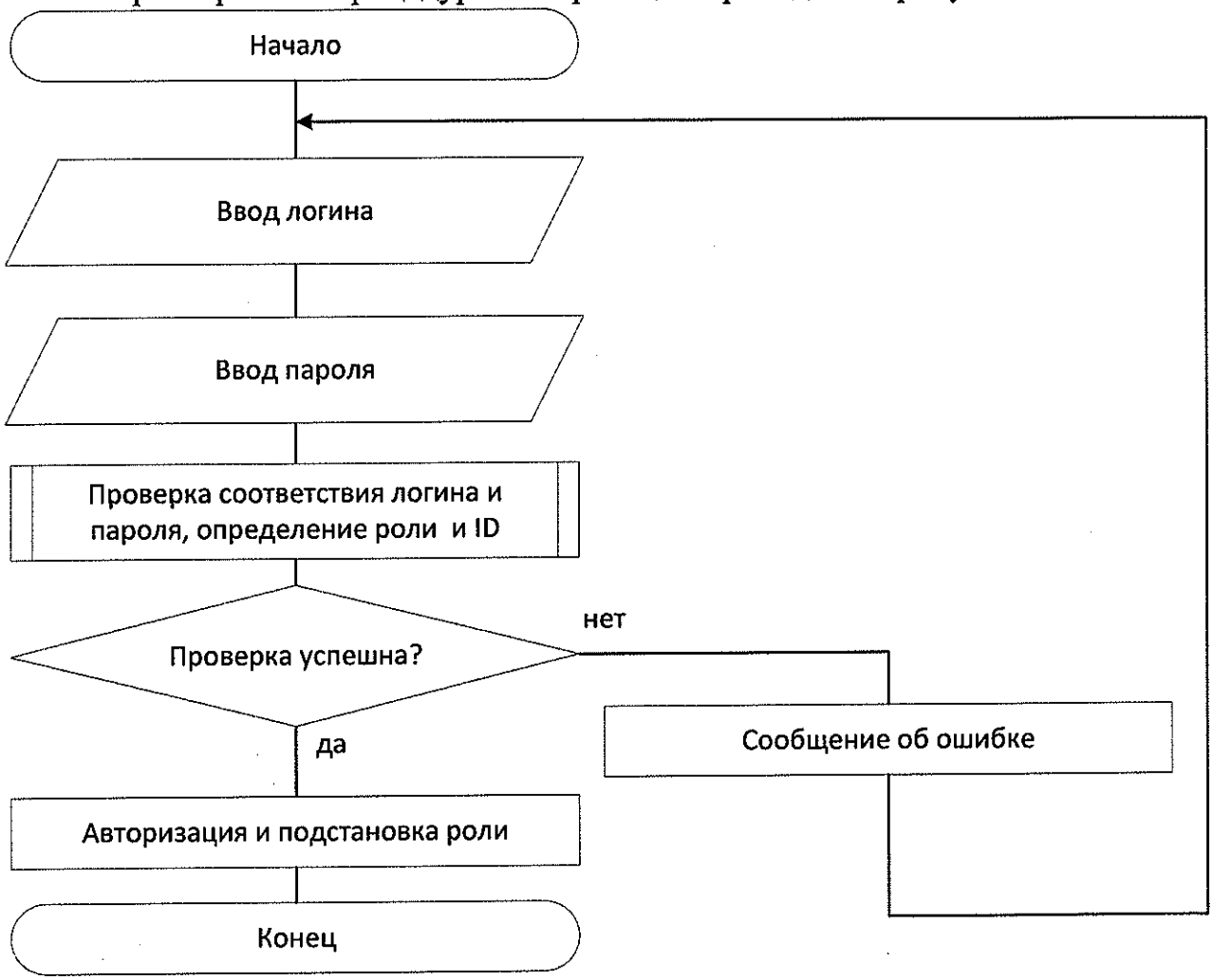


Рисунок 3.2 - Алгоритм процедуры авторизации

При запуске программы пользователю необходимо ввести логин и пароль. Программа проверяет совпадение введенных логина и пароля пользователя, зарегистрированных в базе и, в случае успеха, возвращает ID_Пользователя и его роль. В зависимости от роли активируется та или иная форма (модуль) программы.

Схема алгоритма добавления нового учебного плана представлена на рисунке 3.3.

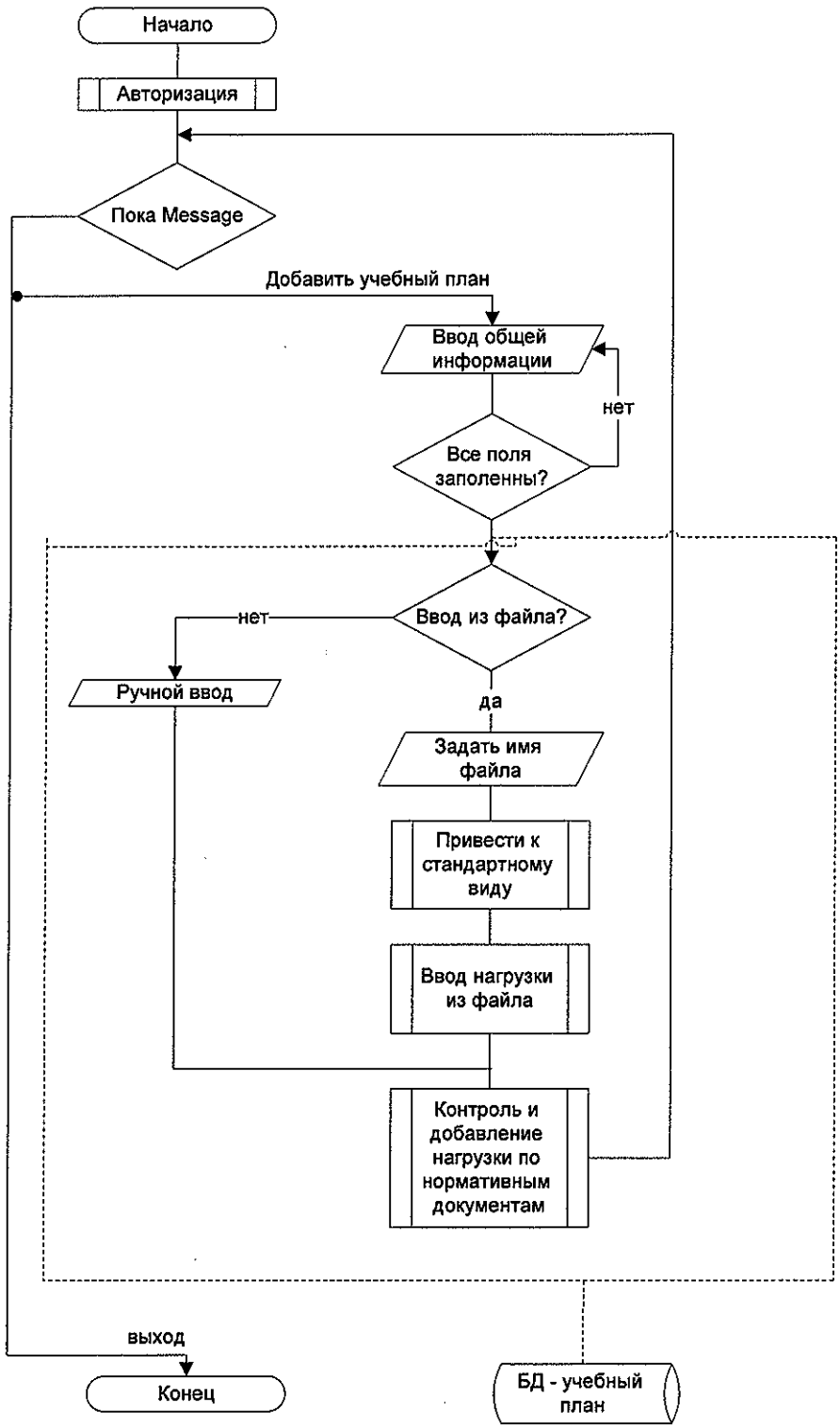


Рисунок 3.3 – Алгоритм добавление учебного плана

3.2 Алгоритм распределения нагрузки

3.2.1 Математическая модель

Дан двудольный граф предпочтений дисциплин преподавателями G_f=(T,P,D) (рисунок 3.4а), где T-множество вершин, соответствующих преподавателям, P-множество вершин, соответствующих предметам, D-множество дуг соответствия преподавателей, читаемым дисциплинам.

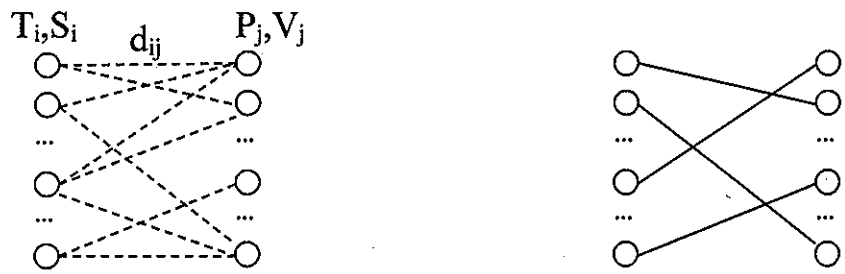
Каждой дуге присваивается «вес» d_{ij}-предпочтение дисциплины (1-наивысший приоритет, …, 10-могу преподавать).

С каждой вершиной $t_i \in T$ связана величина Si-максимальная допустимая нагрузка с учетом должности и ставки

С каждой вершиной $p_j \in P$ связана величина Vi-нагрузка по дисциплине.

Введем коэффициент относительной нагрузки:

$$K_0 = \frac{\sum_{p_j \in P} V_i}{\sum_{t_i \in T} S_i} \tag{3.1}$$



а) б)

Рисунок 3.4 – Графы распределений нагрузки
(а-исходный граф предпочтений G_f; б-результирующий граф нагрузки G_n).

Ставится задача преобразовать граф G_f к графу G_n, минимизируя функционал

$$Q = Kd \cdot \sum_i \sum_j dij + Kr \cdot \sum_i \left(\frac{\sum_{j,dij \in D_n} V_j}{S_i} - K_0\right)^2 \tag{3.2}$$

при ограничениях: $\forall j$ существует в точности одна дуга dij.

Первое слагаемое Q обеспечивает учет предпочтений преподавателей к предметам, второе слагаемое – «равномерное» распределение нагрузки между преподавателями с учетом их ставки и квалификации.

Согласующие коэффициенты Kd и Kr подбираются экспериментально.

3.2.2 Муравьиный алгоритм решения задачи

Для численного решения задачи (3.2) построим «муравьиный» алгоритм.

1. Начальное распределение феромона на дугах – равномерное $\tau_{ij} = \frac{1}{|D|}$

2. Создание муравьев и поиск решения

Создадим M муравьев. Каждый муравей строит свой результирующий граф по следующему алгоритму.

3. Поиск решения

3.1 Обозначим G^k_m -частичное решение, полученное m-ым муравьем на k-ом шаге.

Тогда $G^k_f=G_f\backslash G^k_m$ – граф к распределению, который получается из G_f удалением вершин P_j и дуг dij, для всех j принадлежащими G^k_m.

Если $P^k\neq\varnothing$ и для всех $Pj \in P^k$ существует хотя бы одно ребро dij, то переход к п.3.2, иначе – к п. 3.5.

3.2 Помешаем муравья в вершину T_n графа G^k_m с вероятностью, обратно пропорциональной числу ребер, связанных с данной вершиной:

$$p_n = \sum_j d_{nj} / \sum_i \sum_j d_{ij} \tag{3.3}$$

3.3 Определяем вероятности добавления ребра $d_{nj} \in G_k$ в G_{k+1} по следующей формуле:

$$p_j = \frac{\tau_{nj}{}^{\alpha}\left(\frac{1}{\sum_i d_{ij}}\right)^{\beta}}{\sum_n(\tau_{nj}{}^{\alpha}\left(\frac{1}{\sum_i d_{ij}}\right)^{\beta})}, \tag{3.4}$$

где τ_{ij} - концентрация феромона на дуге dij;

α, β – регулируемые параметры, подбираются экспериментально

3.4 Добавляем к графу G^k_m ребро dnj. Переход к п.3.1

3.5 Если $P^k=\varnothing$, то решение найдено, определяем качество найденного m-ым муравьем решения Q_mпо формуле (2), иначе – решение отсутствует, и данный муравей не участвует в дальнейшем распределении феромона.

Пп. 3.1-3.5 повторяются для всех муравьев, в результате будут найдены L графов G_m с качеством Q_m. Запоминаем $Q_n=\min Q_m$ и соответствующий ему граф G_n.

4. Обновление концентрации феромона.

Концентрацию феромона будем изменять по следующей формуле:

$$\tau_{ij}(k+1) = (1-\rho)\tau_{ij}(k) + \sum_{m,dij\in Gm}\frac{L}{Q_m}, \tag{3.5}$$

где ρ – скорость испарения феромона, $0<\rho< 1$;

L – число найденных решениий;

m – номер муравья;

i – номер вершины;

G_m–граф решения, найденной m-ым муравьем.

5. Условие останова.

Пункты 1 – 3 повторяются пока Q_n на i-ой итерации меньше Q_n на (i-1)-ой итерации. В качестве ответа можно взять любое их G_n.

Так как на каждом шаге алгоритма Q_n может только уменьшаться, то решение стремится точному решению.

Временная сложность алгоритма равна $O(nm)$, где n-число вершин графа, m-число муравьев.

Схема данного алгоритма приведена на рисунке 3.5.

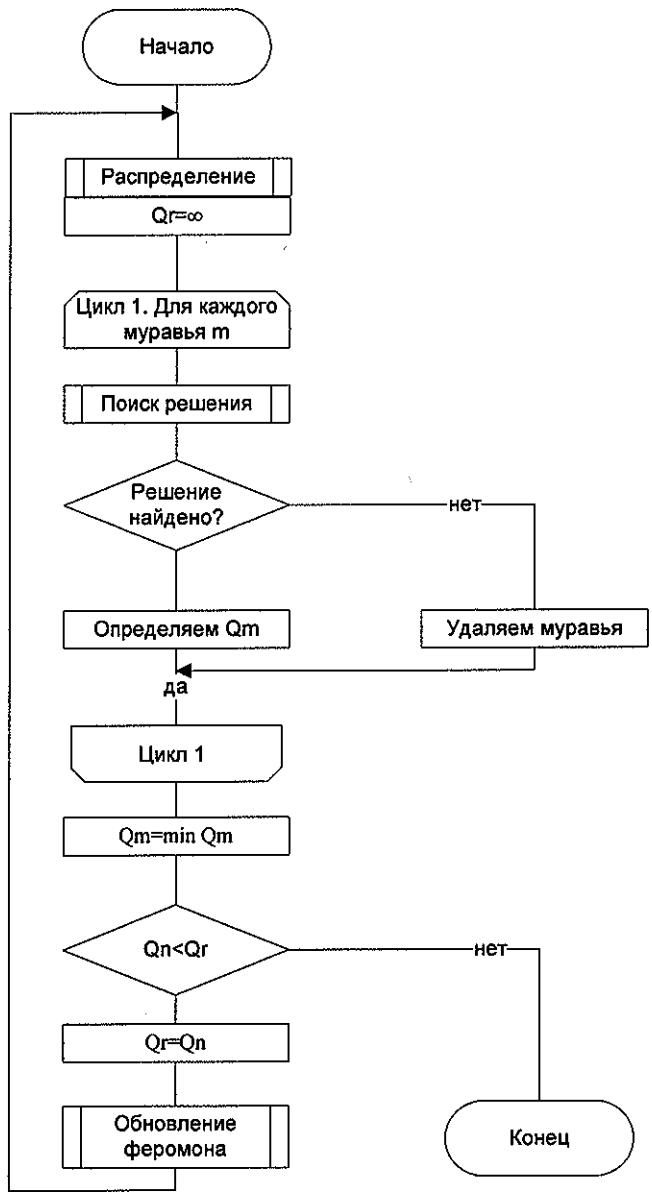


Рисунок 3.5 – Схема муравьиного алгоритма

3.3 Выводы

В данной главе рассмотрены алгоритмы реализации расчета фактической учебной нагрузки преподавателей.

Приведены основной алгоритм программы, а также алгоритмы основных модулей: «Алгоритм модуля авторизации» и «Алгоритм добавления учебного плана» и «Муравьиный алгоритм»

Авторизация предназначен для идентификации пользователя и его роли при входе в программу. Полученный идентификатор пользователя используется основными модулями системы для разграничения прав доступа к персональным данным.

Алгоритм добавления учебного плана обеспечивает ввод учебного плана из системы «Универис», которого еще нет в базе данных. Факультет, шифр направления подготовки, форма обучения (очная/заочная) и степень (бакалавр/магистр) вводиться пользователем вручную. Информация о количестве часов, зачетных единиц и контрольных мероприятиях загружается из .xlsx файла.

4 РАЗРАБОТКА БАЗЫ ДАННЫХ

4. 1 ER-диаграмма

В предметной области расписания учебного заведения можно выделить следующие объекты (сущности), их атрибуты и связи:

1. Факультеты
   - 1.1. Шифр факультета.
   - 1.2. Название факультета.
2. Образовательная программа
   - 2.1. Факультет.
   - 2.2. Шифр направления подготовки.
   - 2.3. Год приема.
   - 2.4. Форма обучения (очная, заочная).
3. Семестр
   - 3.1. Образовательная программа.
   - 3.2. Номер семестра.
   - 3.3. Число недель.
4. План образовательной программы (на весь курс обучения)
   - 4.1. Шифр дисциплины.
   - 4.2. Дисциплина.
   - 4.3. Количество лекционных часов.
   - 4.4. Количество практических часов.
   - 4.5. Количество лабораторных часов.
5. Контроль (за семестр)
   - 5.1. План образовательной программы.
   - 5.2. Номер семестра.
   - 5.3. Наличие экзамена в текущем семестре.
   - 5.4. Наличие зачета в текущем семестре.
   - 5.5. Наличие дифференциального зачета в текущем семестре.
   - 5.6. Наличие курсового проекта в текущем семестре.
   - 5.7. Наличие курсовой работы в текущем семестре.
   - 5.8. Количество аудиторных часов в текущем семестре.
   - 5.9. Количество зачетныхедениц в текущем семестре.
   - 5.10. Количество лекционных часов в текущем семестре.
   - 5.11. Количество практических часов в текущем семестре.
   - 5.12. Количество лабораторных часов в текущем семестре.
6. Год приема
   - 6.1. Год
7. Дисциплина
   - 7.1. Название дисциплины
8. Приоритет дисциплины у преподавателя
   - 8.1. Приоритет
   - 8.2. Преподаватель
   - 8.3. Дисциплина

9. Кафедра
    9.1. Название кафедры
10. Специальность
    10.1. Название специальности
11. Преподаватель
    11.1. Фамилия, имя, отчество преподавателя
    11.2. Кафедра, на которой работает преподаватель.
    11.3. Должность
12. Группа
    12.1. Номер группы
    12.2. Число учащихся в группе
13. Тип проводимого занятия
    - Лекция
    - Практика
    - Лабораторная работа
    - Курсовая работа
    - Курсовой проект
    - Экзамен
    - Зачет
    - Консультация
    - Консультация к экзамену
    - Руководство ВКР бакалавра
    - Руководство ВКР магистра
    - Руководство практикой
14. Поток
    14.1. Преподаватель, который ведет поток
    14.2. Группы в потоке

Объекты и их атрибуты представлены на рисунке 4.1.

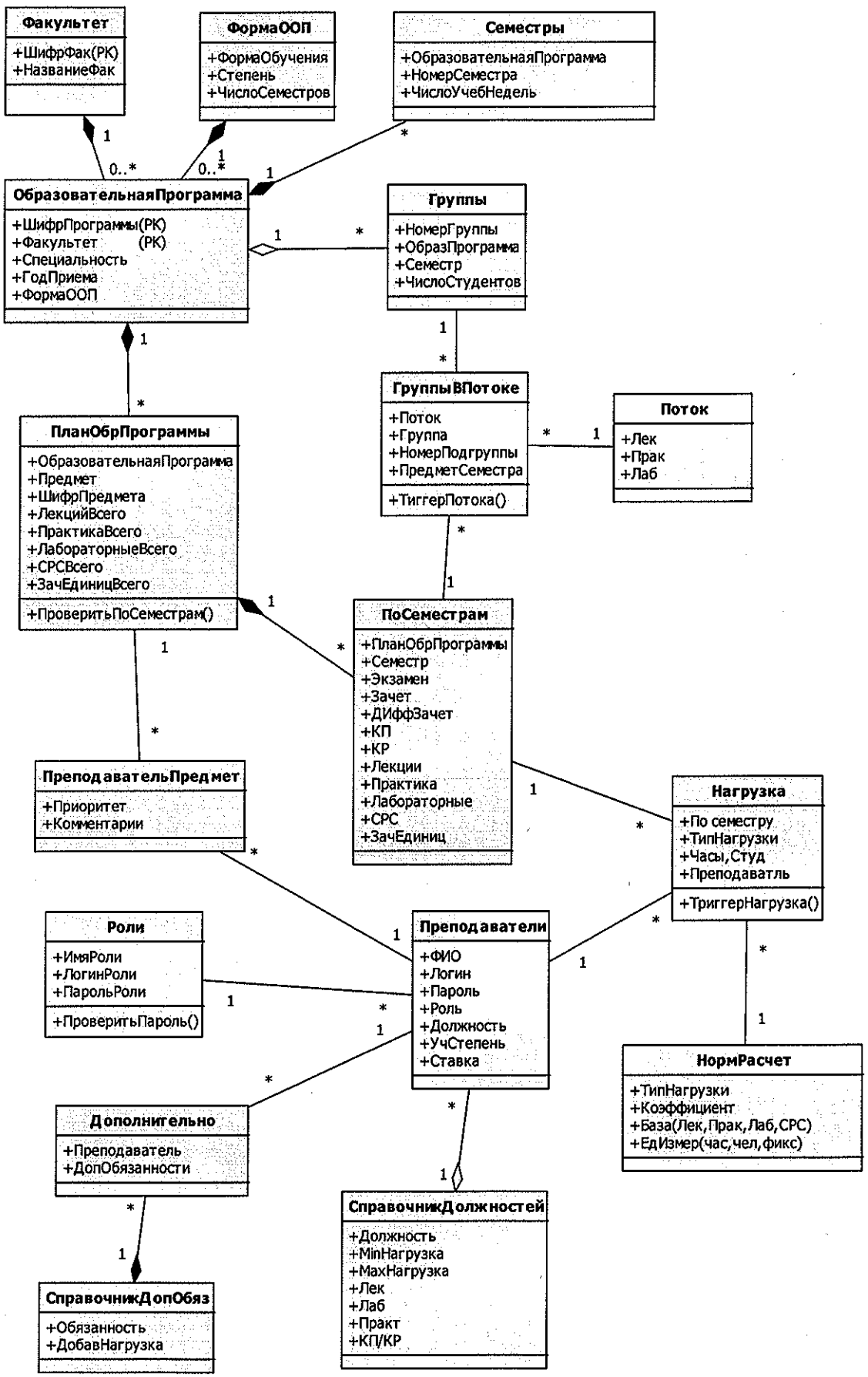


Рисунок 4.1 – Модель предметной области

Разработана схема базы данных, содержащая следующие таблицы:

- Fuculty – факультет.
- EducationalProgram – образовательная программа.
- Semestr – семестр.
- PlanOOP – учебный план на год
- Control – содержит информацию об экзаменах, зачетах, курсовых работах за семестр.
- Years – год приема
- Distiplane – дисциплина
- Tip – тип занятия
- Stream – поток
- Groups – группа
- Possiblediscipline–приоритет дисциплины
- Teacher – преподаватель
- Department – кафедра
- Specialty – специальность

Таблицы базы данных представлены на рисунке 4.2.

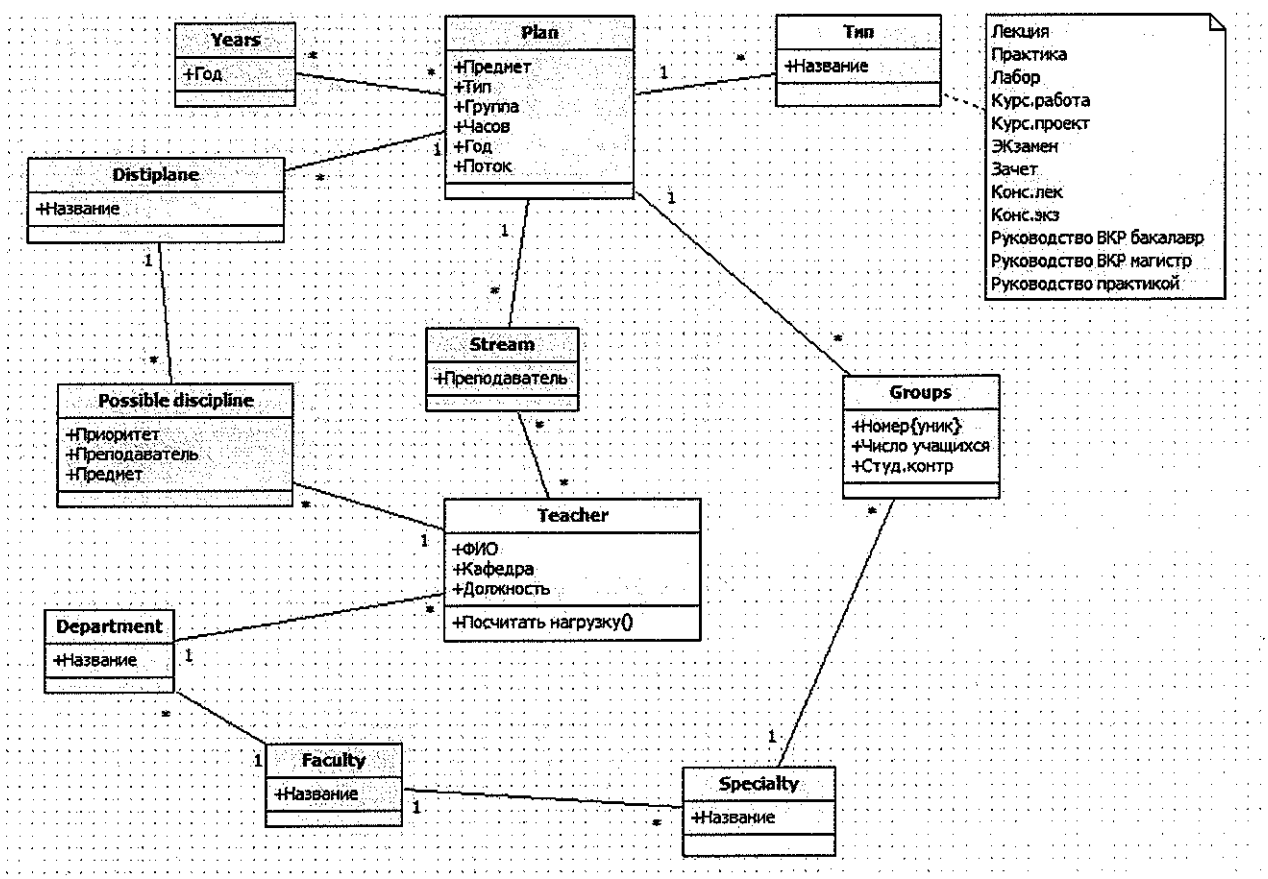


Рисунок 4.2 – ER-диаграмма

4.2. Логическое проектирование базы данных.

Подробное описание базы данных представлено в таблицах 2-14.

В таблице Facultetсодержится список факультетов Южно-Уральского государственного университета.

Таблица 2

Имя	Тип данных	Размерность	Примечание
ShifrFac	nvarchar	8	Шифрфакультета, Primary key, not null
NameFac	Nvarchar	50	Название факультета, not null

В таблице Educationalprogramпредставлена образовательная программа на весь период обучения.

Таблица 3

Имя	Тип данных	Размерность	Примечание
OOP_ID	bigint	11	Идентификатор образовательной программы, primarykey, notnull
Facultet	Nvarchar	8	Шифр факультета, который относиться к образовательной программе, notnull
ShifrProgramm	Nvarchar	70	Шифр образовательной программы, notnull
Year	Smallint	11	Год приема, not null
FormOfTrening	Bigint	11	Формаобучения (очная, заочная), notnull

В таблицеSemestrсодержится информация о семестре.

Таблица 4

Имя	Тип данных	Размерность	Примечание
OOP	bigint	11	
NumberSemestr	Int	11	Номерсеместра, primary key, not null
Weeks	Int	11	Число недель в семестре

В таблице PlanOOP содержится информация об учебном плане в разрезе дисциплины на весь период обучения.

Таблица 5

Имя	Тип данных	Размерность	Примечание
Plan_ID	Bigint	11	Идентификатор учебного плана, primarykey, notnull
ShifrDiscipline	Nvarchar	16	Шифр дисциплины, not null
Discipline	Nvarchar	50	Название дисциплины, not null
Lectures	Int	11	Количество лекционных часов
Practical	Int	11	Количество практических часов
Lab	Int	11	Количество лабораторных часов
OOP_ID	BigInt	11	Идентификатор образовательной программы, к которой относиться учебный план, notnull

В таблице Control содержится информация о контрольных мероприятиях по дисциплине в текущем семестре.

Таблица 6

Имя	Тип данных	Размерность	Примечание
Check_ID	Bigint	11	Идентификатор контрольных мероприятий, primarykey, notnull
PlanOOP	Bigint	11	Идентификатор учебного плана, к которому относиться текущий семестр и контрольные мероприятия, notnull
NumberControl	Int	11	Номер семестра, not null
Ekzamen	Nchar	10	Экзамен (да, нет)
Zachet	Nchar	10	Зачет (да, нет)
Diff_zachet	Nhar	10	Дифференциальный зачем (да, нет)
KR	Nchar	10	Курсовая работа (да, нет)
KP	Nchar	10	Курсовой проект (да, нет)
Hour	Int	11	Аудиторные часы
Zachet_Ed	Int	11	Зачетные единицы

В таблице Yearscодержится год поступления.

Таблица 7

Имя	Тип данных	Размерность	Примечание
Years_ID	bigint	11	Идентификаторгода, primary key, not null
Year	Int	11	Год поступления, not null
Plan_ID	Int	11	Идентификатор учебного плана, notnull

В таблице Disciplanecодержится список дисциплин.

Таблица 8

Имя	Тип данных	Размерность	Примечание
Disciplane_ID	bigint	11	Идентификатордисциплины, primary key, not null
Name	Nvarchar	50	Наименование дисциплины, not null
Plan_ID	Bigint	11	Идентификатор учебного плана, notnull

В таблице Groupscодержится информация о группах.

Таблица 9

Имя	Тип данных	Размерность	Примечание
Number	Nvarchar	20	Уникальный номер группы, notnull
Control_ID	Bigint	11	Идентификатор к контрольным мероприятиям группы, notnull
Groups_ID	Bigint	11	Идентификатор группы, not null

В таблице Specialitycодержится список специальностей на кафедре.

Таблица 10

Имя	Тип данных	Размерность	Примечание
Name	Nvarchar	30	Наименование специальности, not null
Speciality_ID	Bigint	11	Идентификатор специальности, primarykey, notnull
Groups_ID	Bigint	11	Идентификатор группы, not null
Faculty_ID	Bigint	11	Идентификатор факультета

В таблице Teachercодержится список преподавателей.

Таблица 11

Имя	Тип данных	Размерность	Примечание
Name	Nvarchar	50	Фамилия, имя, отчество преподавателя, notnull
Department	Nvarchar	30	Кафедра, на которой работает преподаватель, notnull
Dolgnost	Nvarchar	30	Должность преподавателя, not null
Teacher_ID	Bigint	11	Идентификатор преподавателя, primarykey, notnull

В таблице PossibleDisciplinecодержится список преподавателей.

Таблица 12

Имя	Тип данных	Размерность	Примечание
Prioritet	Int	11	Приоритет дисциплины, not null
Teacher	Nvarchar	30	Преподаватель, not null
Discipline	Nvarchar	30	Дисциплина преподавателя, not null
Possible_ID	Bigint	11	Идентификатор приоритета, primarykey, notnull

В таблице Departmentcодержится список кафедр.

Таблица 13

Имя	Тип данных	Размерность	Примечание
Name	Nvarchar	50	Наименование кафедры, not null
Department_ID	Bigint	11	Идентификаторкафедры, primary key, not null
Teacher_ID	Bigint	11	Идентификатор преподавателя, not null

В таблицеStreamсодержится список потоков.

Таблица 14

Имя	Тип данных	Размерность	Примечание
Teacher	Nvarchar	50	Преподаватель, который ведет поток, notnull
Stream_ID	Bigint	11	Идентификаторпотока, primary key, not null
Teacher_ID	Bigint	11	Идентификатор преподавателя, not null
Plan_ID	Bigint	11	Идентификатор учебного плана, notnull

4.3. Выводы

В результате разработки были определены сущности и их атрибуты, на основании которых составлена модель базы данных, созданная в форматеSQL–сервера MySQL.

5 РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

5.1 Интерфейс администратора

При запуске приложения открывается окно с авторизацией пользователя (рисунок 5.1). В программе нет регистрации, так как все пользователи (преподаватели) будут внесены администратором системы заранее.

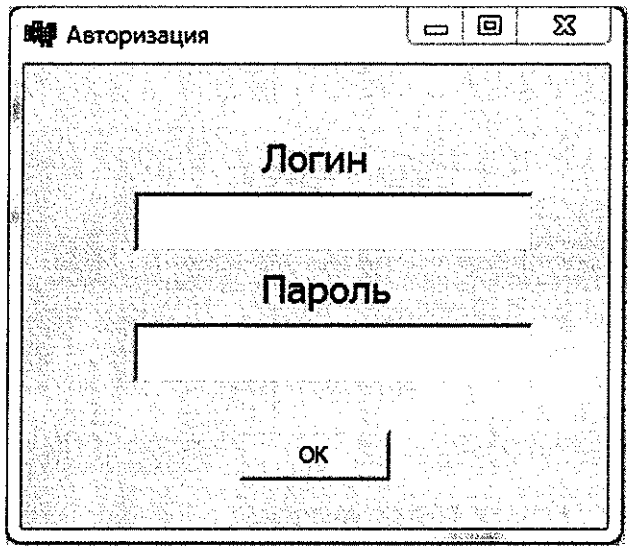


Рис 5.1 – Авторизация пользователя

Рассмотрим интерфейс администратора (рисунок 5.2).

После авторизации открывается главное окно с пустой таблицей, так как учебный план еще не выбран.

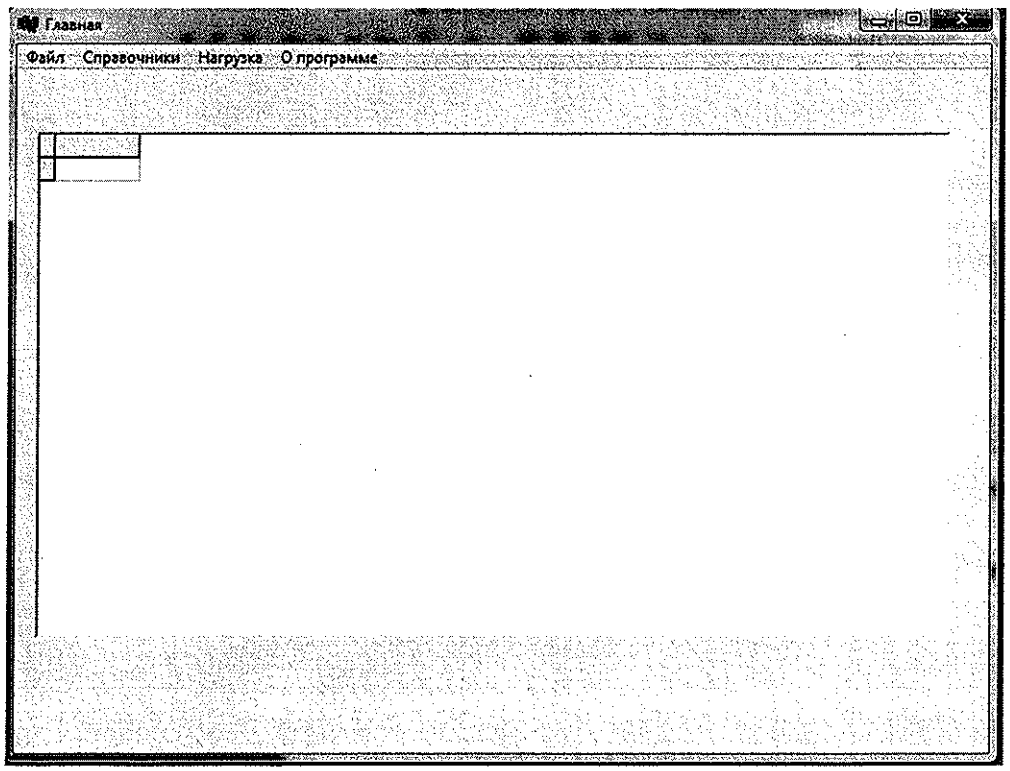


Рисунок 5.2 – Главное окно администратора

Рассмотрим меню (рисунок 5.3-5.5).

Меню «Файл» содержит вкладки: «Загрузить файл», «Выгрузить файл», «Печать», «Выход».

Меню «Справочники» содержит вкладки: «Факультет», «Преподаватель», «Кафедра», «Специальность».

Меню «Нагрузка» содержит вкладки: «Рассчитать нагрузку», «Корректировать нагрузку».

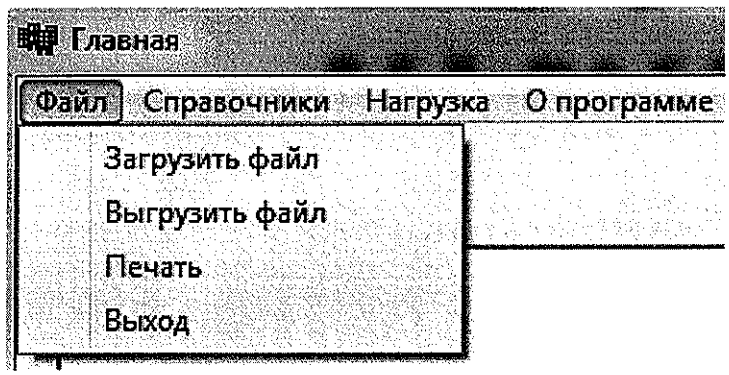


Рисунок 5.3 – Меню «Файл»

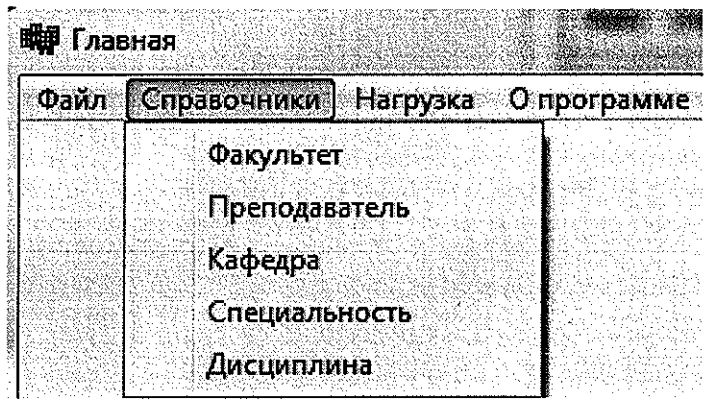


Рисунок 5.4 – Меню «Справочники»

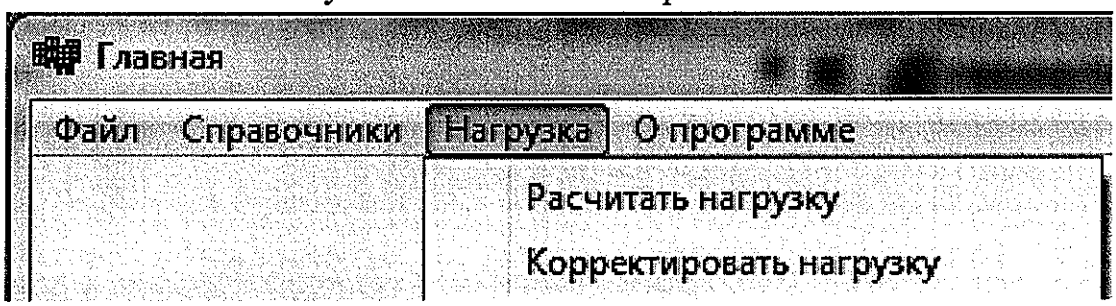


Рисунок 5.5 – Меню «Нагрузка»

У администратора все пункты меню доступны, а у пользователя будут ограничения.

В справочниках содержится справочная информация: списки факультетов, преподавателей, дисциплин, кафедр и специальностей, которая будет

использоваться при работе с учебными планами и расчетом нагрузки в ручном режиме.

Рассмотрим окно справочника «Дисциплины» (рисунок 5.6).

Администратор может не только просматривать справочники, но и редактировать их: добавлять, изменять и удалять записи.

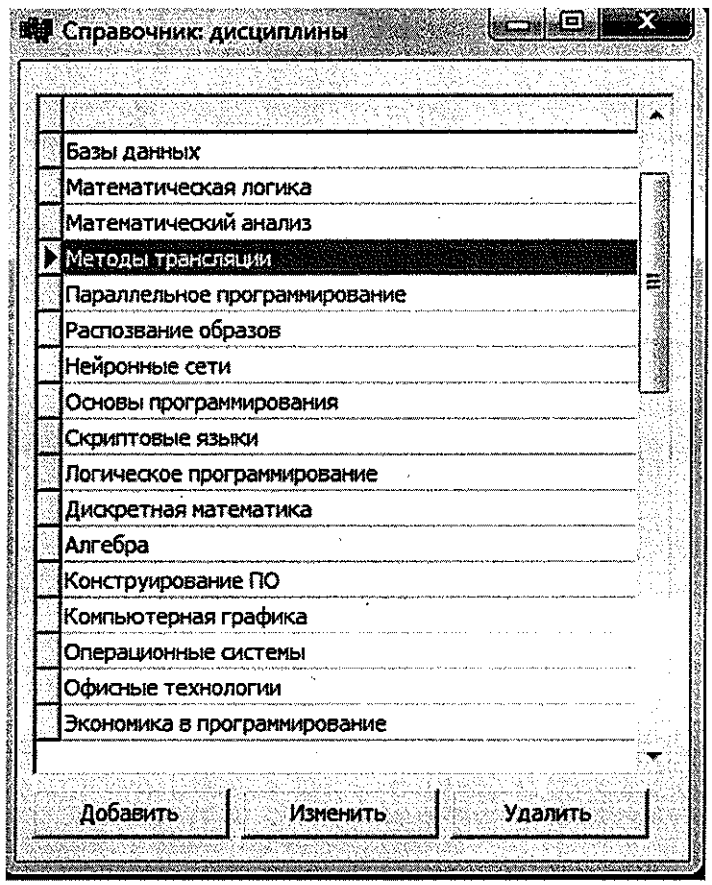


Рисунок 5.6 – Справочник «Дисциплины»

Например, рассмотрим форму добавления преподавателя в справочник (рисунок 5.7). Фамилию, имя и отчество администратор вводит вручную, должность и кафедра заполняются из предложенных вариантов.

При нажатие на кнопку «Изменить» (рисунок 5.6), вызывается идентичная форма. Все компоненты этой формы будут уже заполнены текущими данными. Администратор меняет их и нажимает на кнопку «ОК».

При нажатии на кнопку «Удалить», выбранная запись удаляется.

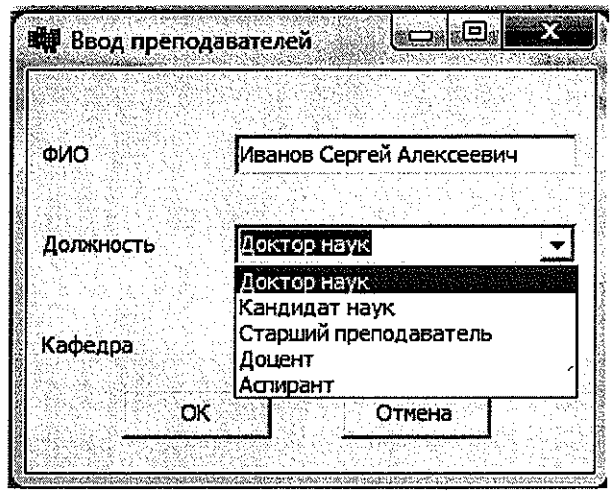


Рисунок 5.7 – Добавление преподавателя в справочник

Перейдем к добавлению нового учебного плана.

В меню «Файл» выбираем «Загрузить файл». Вызывается окно (рисунок 5.8) на котором заносится вручную шифр направления подготовки и год приема, факультет выбирается из справочника, форма обучения и степень из предложенных вариантов. После нажатия на кнопку «ОК» откроется диалоговое окно выбора файла, которое нужно загрузить в программу.

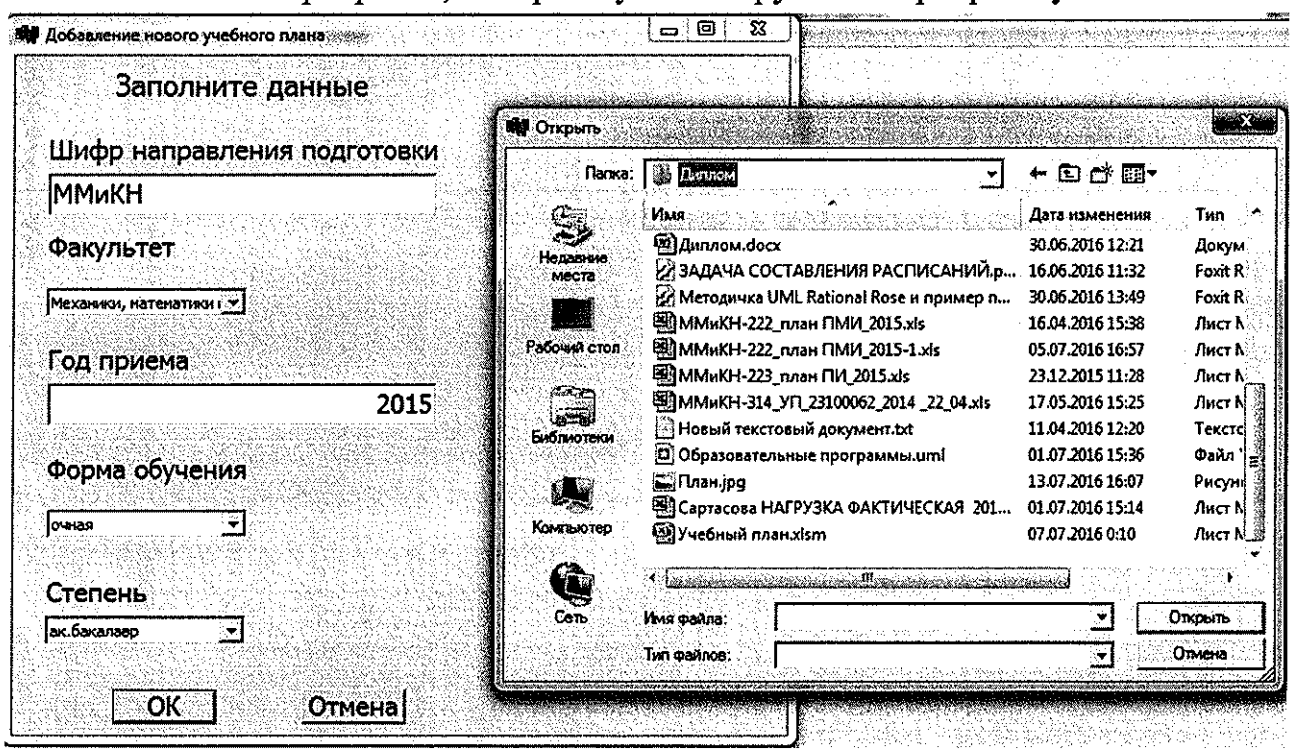
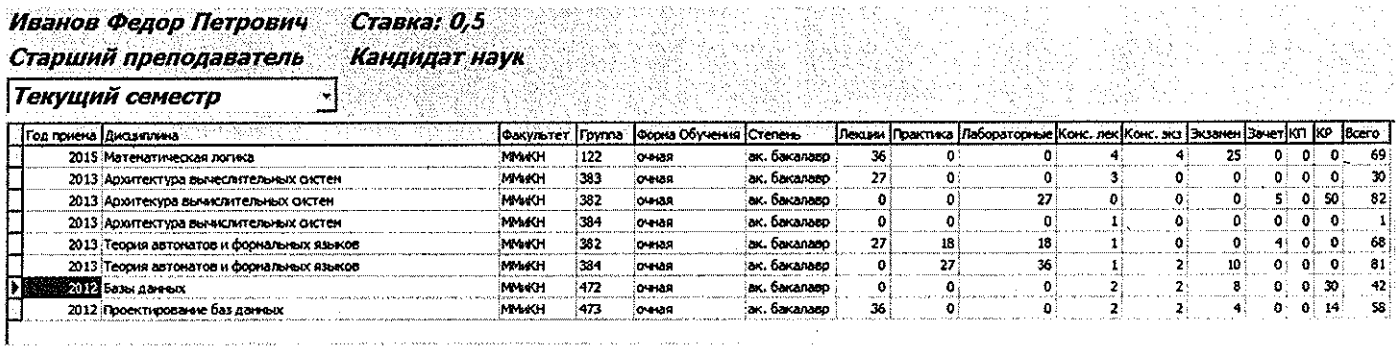


Рисунок 5.8 – Добавление нового учебного плана

5.2 Интерфейс пользователя (преподавателя)

Главное окно пользователя (рисунок 5.9) содержит таблицу с нагрузкой преподавателя на текущий семестр. Так же преподаватель может просмотреть свой учебный план на следующий семестр и предыдущие учебные года.



Год приема	Дисциплина	Факультет	Группа	Форма Обучения	Степень	Лекции	Практика	Лабораторные	Конс. лек	Конс. экз	Экзамен	Зачет	КП	КР	Всего
2015	Математическая логика	ММиКН	122	очная	ак. бакалавр	36	0	0	4	4	25	0	0	0	69
2013	Архитектура вычеслительных систем	ММиКН	383	очная	ак. бакалавр	27	0	0	3	0	0	0	0	0	30
2013	Архитекура вычислительных систем	ММиКН	382	очная	ак. бакалавр	0	0	27	0	0	0	5	0	50	82
2013	Архитектура вычислительных систем	ММиКН	384	очная	ак. бакалавр	0	0	0	1	0	0	0	0	0	1
2013	Теория автоматов и формальных языков	ММиКН	382	очная	ак. бакалавр	27	18	18	1	0	0	4	0	0	68
2013	Теория автоматов и формальных языков	ММиКН	384	очная	ак. бакалавр	0	27	36	1	2	10	0	0	0	81
2012	Базы данных	ММиКН	472	очная	ак. бакалавр	0	0	0	2	2	8	0	0	30	42
2012	Проектирование баз данных	ММиКН	473	очная	ак. бакалавр	36	0	0	2	2	4	0	0	14	58

Рисунок 5.9 - Окно нагрузки преподавателя

На рисунке 5.10 представлена форма добавления, изменения и удаления приоритета дисциплины.

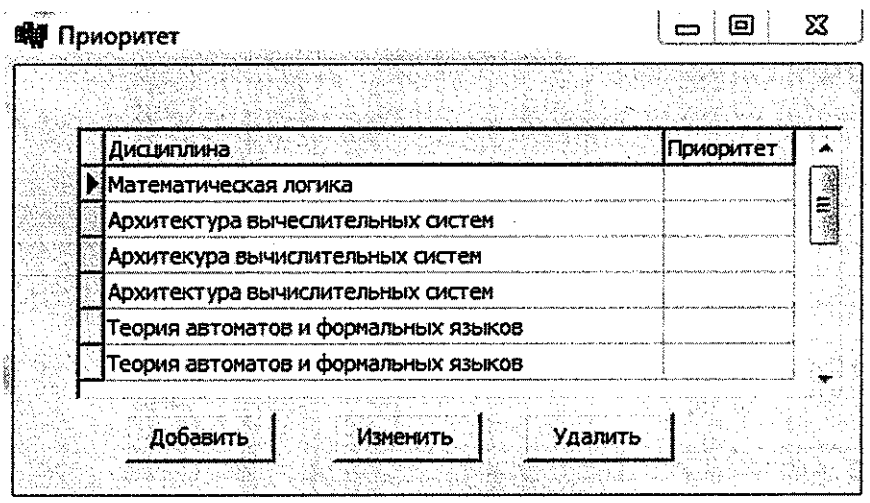


Рисунок 5.10 - Приоритет

ЗАКЛЮЧЕНИЕ

Ежегодное распределение нагрузки преподавателя представляет собой достаточно трудоемкий процесс, требующий большой аккуратности. После заполнения возникает вопрос о соответствии отчетной нагрузки и фактической, так как существует вероятность ошибки. Поэтому автоматизация составления отчетов преподавателей по выполненной нагрузке является актуальной. В качестве входных данных для расчета нагрузки были взяты учебные планы из системы «УНИВЕРИС».

Проведенный анализ программ автоматизации учебного процесса: «Система составления расписания», «Ректор-ВУЗ», «Экспресс-расписание ВУЗ» показал, что основными недостатками этих программных продуктов является высокая стоимость, отсутствие расчета фактической учебной нагрузки преподавателя, ограниченный доступ преподавателей к этим программам.

На основе анализа программного обеспечения с похожей функциональностью и аналогичными технологиями осуществлен выбор средств и технологий для реализации поставленной задачи.

Предполагается, что разрабатываемая программа должна содержать следующие функциональные требование:

- удобный интерфейс;
- таблица с выбором типа занятия и расчетом фактической учебной нагрузки по конкретно выбранному типу;
- выполняется суммарный расчет по семестрам и учебному году;
- таблица расписания, куда будут заноситься данные о типе занятия;
- выбор семестра, в который будет заноситься информация;

Был проведен анализ предметной области и разработана реляционная база данных. Разработаны алгоритмы и реализовано приложение для ввода учебных планов в базу данных и расчета фактической нагрузки. Спроектированный интерфейс программы для расчета фактической учебной нагрузки преподавателей разработан с помощью использования паттернов проектирования приложений, предложенных в книге Тидвелла. Произведено тестирование данной программы с целью выявления ошибок или ситуаций, в которых программа ведет себя некорректно. В результате тестирования ошибок выявлено не было.

Недостатком данной программы является то:

1. Что она не позволяет производить автоматический контроль ввода не аудиторной нагрузки, что можно рекомендовать в качестве дальнейшего развития данной программы.
2. В ситуации, когда по одной дисциплине лекцию ведет один преподаватель, а практику – другой, нет возможности отслеживать с кем преподаватель работает «в паре».

Данные недостатки можно устранить при дальнейшем развитии программы.

Несмотря на недостаток, разработанное программное обеспечение позволило свести количество ошибок при заполнении таблиц к минимуму, и за

счет использования централизованной базы данных позволило получить суммарные отчеты со статистикой по выполненной учебной нагрузке всех преподавателей кафедры.

ЛИТЕРАТУРА

1. Архангельский, А. Язык C++ в C++ Builder / А. Я. Архангельский. – М.:Бином, 2008. – 942 с.
2. Буч, Г. UML / Г. Буч, А. Якобсон, Д. Рамбо – СПб.: Питер, 2006.
3. Бен-Ган, И. Microsoft SQL Server 2008. Основы T-SQL / И. Бен-Ган. – СПб.: БХВ-Петербург, 2009. – 430 с.
4. Виейра, Р. Программирование баз данных Microsoft SQL Server 2008 для профессионалов: пер с англ. / Р. Виейра. – М.: Издательский дом «Вильямс», 2010. – 816 с.
5. Все о работе с SQL [Электронный ресурс]. – Режим доступа: http://sql.ru/, свободный.
6. Гарбер, Г. З. Основы программирования на VisualBasic и VBA в Excel 2007 / Г. З. Гарбен – М.: Соло-Пресс, 2008. – 189 с.
7. Жилинский, А. Самоучитель Microsoft SQL Server 2008 / А. Жилинский. - СПб.: БХВ-Петербург, 2009. – 421 с.
8. Ковелев, М. М. Дискретная оптимизация. Целочисленное программирование / М. М. Ковалев. – М.:Либроком, 2011 – 191 с.
9. Купер, А. Алан Купер об интерфейсе. Основы проектирования взаимодействия: Пер. с англ. / А. Купер, Р. Рейман, Д.М. Кронин. — СПб.: Символ-Плюс, 2010. – 688 с.
10. Лафоре, Р. Объектно-ориентированное программирование C++ / Р. Лафоре. – СПб.: Питер, 2011. – 923 с.
11. Рутковская, Д. Нейронный сети, генетические алгоритмы и нечеткие системы / Д. Рутковская – М.: Горячая линия – Телеком, 2013. – 383 с.
12. Сухарев, А. Г. Методы оптимизации: учебник для вузов по естественным научным направлениям и специальностям / А. Г. Сухарев. – М.:Юрайт, 2015. – 367 с.
13. Страуструп, Б. Язык программирования C++ / Б. Страуструп. – М.:Бином-Пресс, 2008. – 1098 с.
14. Татаренко, А. Метод решения задачи оптимизации структуры распределенной вычислительной сети вуза на основе модифицированного генетического алгоритма / А. С. Татаренко // Открытое образование. – 2011. - №1. –С. 51-57

ПРИЛОЖЕНИЕ 1
ООП

Цикл № п/п	Название дисциплины	Кафедра	Кафедра: экз.	Кафедра: зач.	Кафедра: диф. зач.	Кафедра: К.П.	Кафедра: К.Р.	ГОС,ЗЕ	[illegible]	Всего	Объем работы студентов, час: Всего ауд.	Из них: Лекции	Из них: Практика	Из них: Лаб. раб.	[illegible]	I курс, 1: 18	I курс, 1: ЗЕ	I курс, 2: 18	I курс, 2: ЗЕ	II курс, 3: 18	II курс, 3: ЗЕ	II курс, 4: 18	II курс, 4: ЗЕ	III курс, 5: 18	III курс, 5: ЗЕ	III курс, 6: 18	III курс, 6: ЗЕ	IV курс, 7: 18	IV курс, 7: ЗЕ	IV курс, 8: 12	IV курс, 8: ЗЕ
Б.1.09	Математические основы компьютерной графики	Прима	3						3	108	72	36		36	36					4	3,0										
Б.1.11	Математические основы	Прима		1					2	72	54	18	18	18	18	3	2,0														
Б.1.12	Математическая логика	Прима	1						4	144	72	36	36		72	4	4,0														
Б.1.13	Методы оптимизации	Прима	6						4	144	72	36		36	72											4	4,0				
Б.1.14	Исследование операций	Прима	7						3	108	72	36	36		36													4	3,0		
Б.1.16	Дискретная математика	Прима		2					3	108	72	36	36		36			4	3,0												
Б.1.17	Дискретная оптимизация	Прима	7						4	144	72	36	36		72													4	4,0		
Б.1.19	Графическое моделирование	Прима		4					3	108	54	18		36	54							3	3,0								
Б.1.21	Основы программирования	Прима	1						5	180	108	36	36	36	72	6	5,0														
Б.1.22	Языки программирования	Прима	2				2		5	180	108	36	36	36	72			6	5,0												
Б.1.23	Операционные системы и сети	Прима		6			6		3	108	72	36		36	36											4	3,0				
Б.1.24	Базы данных	Прима		5			5		3	108	72	36		36	36									4	3,0						
Б.1.25	Проектирование баз данных	Прима	6				6		4	144	72	36		36	72											4	4,0				
Б.1.27	Системный анализ	Прима			8				3	108	72	24	24	24	36															6	3,0
Б.1.28	Математическая статистика	Прима			6				3	108	54	18	36		54											3	3,0				
В.1.07	Компьютерные сети	Прима	6						3	108	72	36		36	36											4	3,0				
В.1.08	Теория переключательных функций и автоматов	Прима	3						4	144	72	36	36		72					4	4,0										
В.1.09	Функциональное и логическое программирование	Прима		7					3	108	54	36		18	54													3	3,0		
В.1.10	Алгоритмы и структуры данных	Прима		3			3		3	108	72	36	18	18	36					4	3,0										
В.1.11	Объектно-ориентированное программирование	Прима			4		4		3	108	72	36		36	36							4	3,0								
В.1.12	Компьютерная графика	Прима			7	7			4	144	72	36		36	72													4	4,0		
В.1.14	Архитектура ЭВМ	Прима		4					2	72	54	36	18		18							3	2,0								
В.1.15	Основы теории управления	Прима		7					3	108	54	36	18		54													3	3,0		
В.1.16	Визуальное программирование	Прима		5		5			3	108	54	18		36	54									3	3,0						
В.1.17	Технологии разработки программного обеспечения	Прима	5						4	144	72	36	18	[illegible]	72									4	4,0						

ПРИЛОЖЕНИЕ 2
Учебный план

№	Форма об	Дисциплина	Институт	Специальность	Группа	Семестр	Студ.	Студ. поток	Подгр.	Подгр.	Подгр. поток	Лек.	Лек. поток	Практ.	Практ. поток	Лаб.	Лаб. поток	Экзам	Зачет	Курс.	Курс. проект	Контр.	Конс.	Конс.	Доп.	Всего	Всего	Всего	Кредит	Студ.	Номер потока	Из уч. планов	ФИО преподавателя
6	очная	Дополнительные главы математического анализа	ММиИН	09.03.04	214	3	23	0	1	0	1	36	36	36	36	0		12				6	2	2		93	0	93		25	147294	12683	Гераськин А. В.
7	очная	Офисные технологии	ММиИН	01.03.02	212	3	65	5	3	3	1	18	18	18	54	18	54		13				3			142	11	131		60	147286	12682;12684	Сурнин А. А.
8	очная	Теория переключательных функций	ММиИН	01.03.02	212	3	25	4	1	0	1	27	27	45	45	0			5	50			1			128	20	108		21	152475	12682	Оленичкова Т. Ю.
16	очная	Архитектура вычислительных систем	ММиИН	010400	382	5	28	1	2	1	0	1	27	27	0		27	27		6	56		3			116	2	114		27	147309	11234;11236	Оленичкова Т. Ю.
17	очная	Архитектура вычислительных систем	ММиИН	231000	384	5	37	0	1	1	0	0	27		0		27	27		3			1			32	0	32		17	147309	11235	Оленичкова Т. Ю.
18	очная	Защита информации и криптография	ММиИН	231000	384	5	37	0	1	1	1	1	27	27	27	27	18	18	9				1	2		84	0	84		17	147311	11235	Елсаков Б. М.
19	очная	Конструирование программного обеспечения	ММиИН	231000	384	5	47	0	1	1	1	1	18	18	18	18	18	18		3			1			56	0	56		17	147305	11235	Демидов А. К.
20	очная	Математические основы защиты информации	ММиИН	231300	383	5	8	0	1	1	1	1	27	27	27	27	18	18		2			1			75	0	75		8	147314	11236	Елсаков Б. М.
21	очная	Теория автоматов и формальных языков	ММиИН	010400	382	5	20	1	1	1	1	1	27	27	18	18	18	18		4			1			68	3	65		19	147310	11234	Дрозин Д. А.
22	очная	Теория автоматов и формальных языков	ММиИН	231000	384	5	17	0	1	1	0	1	27		27	27	36	36	9				4	1	2	79	0	79		17	147310	11235	Дрозин Д. А.
23	очная	Теория вероятностей и математическая статистика	ММиИН	010400	382	5	20	1	1	1	1	1	36	36	27	27	9	9	10				5	2	2	91	5	86		19	147318	11234	Бекетин
24	очная	Теория вероятностей и математическая статистика	ММиИН	231000	384	5	17	0	1	1	1	0	36		27	27	27	27	9				4	2	2	71	0	71		17	147318	11235	Бекетин
25	очная	Теория вероятностей, математическая статистика и	ММиИН	231300	383	5	8	0	1	1	1	1	27	27	27	27	18	18		2			1			77	0	77		8	147315	11236	Бекетин
26	очная	Численные методы	ММиИН	010400	382	5	20	1	1	1	1	1	27	27	27	27	18	18		4			5	1		82	4	78		19	147306	11234	Васильев Ю. С.
27	очная	Численные методы	ММиИН	231000	384	5	17	0	1	1	1	1	36	36	18	18	18	16		3			4	2		81	0	81		17	147316	11235	Васильев Ю. С.
29	очная	Базы данных	ММиИН	010400	472	7	18	1	1	1	1	1	36	36	36	36	18	18	9		36			2	2	139	8	131		17	147350	9847	Катергин М. Ю.

ПРИЛОЖЕНИЕ 3
КОД ПРОГРАММЫ

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
#include <comobj.hpp>
#include <OleCtnrs.hpp>

#include "unit2.h"
#include "unit3.h"
#include "unit4.h"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N2Click(TObject *Sender)
{
        Form1->Glavnaya->Insert();
        Form2->ShowModal();
        Variant Excel;
        Variant Book;
        Variant Sheet;

  Excel=CreateOleObject("Excel.Application");
Excel.OlePropertySet("Visible",true);
        Book=Excel.OlePropertyGet("Workbooks").OlePropertyGet("Open",
"H:\Диплом\new\Plan_MM.xlsx"); //открываемкнигу, указаveёимя
  Sheet=Book.OlePropertyGet("Worksheets", 2);

this->Edit1->Text=Sheet.OlePropertyGet("Range", "AD11");

Excel.OleProcedure("Quit");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N6Click(TObject *Sender)
```

```
{
Form1->Facultet->Insert();
Form3->ShowModal();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N7Click(TObject *Sender)
{
        Form4->ShowModal();
        Form4->ADOConnection2->Connected=true;
        Form4->ADOQuery1->Active=true;
}


#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

#include "Unit1.h"
#include "unit3.h"
#include "unit4.h"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
longint id;
try {
        Form1->Glavnaya->Post();
        ADOQuery1->SQL->Clear();
        ADOQuery1->SQL->Text=("SELECT max(OOP_ID)as OOPID FROM
[ОбразовательнаяПрограмма]");
        ADOQuery1->Active=true;
        ADOQuery1->ExecSQL();
       AnsiString h=ADOQuery1->Fields->FieldByNumber(1)->AsString  ;
       id=ADOQuery1->FieldByName("OOPID")->AsInteger;
        ADOQuery1->Active=false;
```

```
        if (OpenDialog1->Execute()) {

AnsiString Name=OpenDialog1->FileName;
                if (!AddPlanOOP(id, Name))   {
                        Form1->ADOQuery1->SQL->Text=("DELETE FROM
[Образовательнаяпрограмма] WHERE OOP_ID=:param");
                        Form1->ADOQuery1->Parameters->ParamByName("param")->Value=id;
                        Form1->ADOQuery1->ExecSQL();
                        Form1->Glavnaya->Edit();                    return;
                }
                else {
                        Application-
>MessageBoxA("Данныеуспешнодобавлены","Вводучебнойпрограммы",MB_OK);
                        Form1->Glavnaya->Edit(); return;
                 }
                 }
        else {
                Application->MessageBoxA("Нажмите Ok и выберите файл для ввода учебной
программы или \nнажмите Отмена - выход",
                                "Файл не выбран", MB_OK);
                Form1->Glavnaya->Edit(); return;
                }



        AnsiStringsfac;
        sfac=DBLookupComboBox3->Text;

        for (inti=1; i<= Form1->Glavnaya->RecordCount; i++) {
                Form1->Glavnaya->RecNo=i; Form1->DBNavigator1->B
               if (Form1->Glavnaya->FieldByName("НазваниеФак")->AsString!=sfac) {


                }
          }
          Form1->DBFacultet->Refresh();   */
         }

catch (const Error& e) {
        Application->MessageBox("Неизвестнаяошибка", "Ошибка", MB_OK);
        Form1->Glavnaya->Cancel();
}
Close();
}
//---------------------------------------------------------------------------
```

void __fastcall TForm2::Button2Click(TObject *Sender)


```
{
Close();
}
#include <vcl.h>
#pragma hdrstop

#include "Unit4.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
boolAddPlanOOP(long int id, AnsiString Name){
        //return false;
        Variant var_Excel, var_Book, er;
        Variant personal_path="H:\Диплом\Учебныйплан.xlsm";
        var_Excel.OleProcedure("Run","Учебныйплан.xlsm!Macros");
                Form4->ADOConnection2->Connected=true;
                Form4->ADOQuery1->Active=true;
                intkolstr=ADOQuery1->RecordCount;
                for (inti = 1; i<=kolstr; i++) {
                        ADOQuery1->RecNo=i;
                        AnsiString s=ADOQuery1->FieldByName("Номер")->AsString;
                        AnsiString s1=ADOQuery1->FieldByName("Дисциплина")->AsString;
                        int s2=ADOQuery1->FieldByName("Лекции")->AsInteger;
                        int s3=ADOQuery1->FieldByName("Практика")->AsInteger;
                        int s4=ADOQuery1->FieldByName("Лаб# работа")->AsInteger;

                        Form1->ADOQuery1->Close();
                        Form1->ADOQuery1->SQL->Clear();
                        Form1->ADOQuery1->SQL->Text=("INSERT INTO ПланООП (ООП,
Дисциплина, Лекции, Практические, Лабораторные) VALUES (:param, :param1, :param2,
:param3, :param4)");
                        Form1->ADOQuery1->Parameters->ParamByName("param")->Value=s;
                        Form1->ADOQuery1->Parameters->ParamByName("param1")->Value=s1;
                        Form1->ADOQuery1->Parameters->ParamByName("param2")->Value=s2;
                        Form1->ADOQuery1->Parameters->ParamByName("param3")->Value=s3;
                        Form1->ADOQuery1->Parameters->ParamByName("param4")->Value=s4;
                        Form1->ADOQuery1->ExecSQL();
                        ADOQuery1->ExecSQL();
                }
}
```

```
_fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm4::FormHide(TObject *Sender)
{
         ADOConnection2->Connected=false;

}
//---------------------------------------------------------------------------
void __fastcall TForm4::FormShow(TObject *Sender)
{       ADOConnection2->ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Mode=Share Deny None;JetOLEDB:System database="";Jet OLEDB:Registry
Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine Type=35;Jet OLEDB:Database
Locking Mode=0;Jet OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Global Bulk
Transactions=1;Jet OLEDB:New Database Password="";Jet OLEDB:Create System
Database=False;JetOLEDB:Encrypt Database=False;JetOLEDB:Don't Copy Locale on
Compact=False;JetOLEDB:Compact Without Replica Repair=False;Jet OLEDB:SFP=False;Data
Source=F:\Plan1.xls;" ;
        //ADOConnection2->Connected=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm4::Button1Click(TObject *Sender)
{
        Form4->ADOConnection2->Connected=true;
        Form4->ADOQuery1->Active=true;
        intkolstr=ADOQuery1->RecordCount;
                for (inti = 1; i<=kolstr; i++) {
                        ADOQuery1->RecNo=i;
                        AnsiString s=ADOQuery1->FieldByName("Номер")->AsString;
                        AnsiString s1=ADOQuery1->FieldByName("Дисциплина")->AsString;
                        int s2=ADOQuery1->FieldByName("Лекции")->AsInteger;
                        int s3=ADOQuery1->FieldByName("Практика")->AsInteger;
                        int s4=ADOQuery1->FieldByName("Лаб# работа")->AsInteger;

                        Form1->ADOQuery1->Close();
                        Form1->ADOQuery1->SQL->Clear();
                        Form1->ADOQuery1->SQL->Text=("INSERT INTO ПланООП (ООП,
Дисциплина, Лекции, Практические, Лабораторные) VALUES (:param, :param1, :param2,
:param3, :param4)");
                        Form1->ADOQuery1->Parameters->ParamByName("param")->Value=s;
                        Form1->ADOQuery1->Parameters->ParamByName("param1")->Value=s1;
```

```
                Form1->ADOQuery1->Parameters->ParamByName("param2")->Value=s2;
                        Form1->ADOQuery1->Parameters->ParamByName("param3")->Value=s3;
                        Form1->ADOQuery1->Parameters->ParamByName("param4")->Value=s4;
                        Form1->ADOQuery1->ExecSQL();
                        ADOQuery1->ExecSQL();
                }
                for (inti = 1; i<=kolstr; i++) {
                        ADOQuery1->RecNo=i;
                        AnsiString s=ADOQuery1->FieldByName("Номер")->AsString;
                        int ekz1=ADOQuery1->FieldByName("Экзамен 1")->AsInteger;
                        int ekz2=ADOQuery1->FieldByName("Экзамен 2")->AsString;
                        AnsiString ekz3=ADOQuery1->FieldByName("Экзамен 3")->AsString;
                        AnsiString ekz4=ADOQuery1->FieldByName("Экзамен 4")->AsString;
                        AnsiString zach1=ADOQuery1->FieldByName("Зачет 1")->AsString;
                        AnsiString zach2=ADOQuery1->FieldByName("Зачет 2")->AsString;
                        AnsiString zach3=ADOQuery1->FieldByName("Зачет 3")->AsString;
                        AnsiString zach4=ADOQuery1->FieldByName("Зачет 4")->AsString;
                        AnsiString zach5=ADOQuery1->FieldByName("Зачет 5")->AsString;
                        AnsiString zach6=ADOQuery1->FieldByName("Зачет 6")->AsString;
                        AnsiString dzach1=ADOQuery1->FieldByName("Диф# Зачет 1")-
>AsString;
                        AnsiString dzach2=ADOQuery1->FieldByName("Диф# Зачет 2")-
>AsString;
                        AnsiString dzach3=ADOQuery1->FieldByName("Диф# Зачет 3")-
>AsString;
                        AnsiString dzach4=ADOQuery1->FieldByName("Диф# Зачет 4")-
>AsString;
                        AnsiString kp1=ADOQuery1->FieldByName("КП 1")->AsString;;
                        AnsiString kp2=ADOQuery1->FieldByName("КП 2")->AsString;;
                        AnsiString kr1=ADOQuery1->FieldByName("КР 1")->AsString;;
                        AnsiString kr2=ADOQuery1->FieldByName("КР 2")->AsString;;
                        bool flag=false;
                        int k=1;

                        Form1->ADOQuery1->SQL->Text=("INSERT INTO Контроль (ПланООП,
Номерсеместра, Экзамен) VALUES (:param, :semestr, :ekzamen)");
                        if (ekz1==1) {
                                Form1->ADOQuery1->Parameters->ParamByName("param")-
>Value=s;
                                Form1->ADOQuery1->Parameters->ParamByName("ekzamen")-
>Value="да";
                                Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
```

```
                    flag=true;
                        }
                    if (ekz2==1) {
                        Form1->ADOQuery1->Parameters->ParamByName("param")-
>Value=s;
                        Form1->ADOQuery1->Parameters->ParamByName("ekzamen")-
>Value="да";
                        Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
                        flag=true;
                        }
                    if (ekz3==1) {
                        Form1->ADOQuery1->Parameters->ParamByName("param")-
>Value=s;
                        Form1->ADOQuery1->Parameters->ParamByName("ekzamen")-
>Value="да";
                        Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
                        flag=true;
                        }
                    if (ekz4==1) {
                        Form1->ADOQuery1->Parameters->ParamByName("param")-
>Value=s;
                        Form1->ADOQuery1->Parameters->ParamByName("ekzamen")-
>Value="да";
                        Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
                        flag=true;
                        }
                    if (zach1==1) {
                        if (flag==false) {
                          Form1->ADOQuery1->Parameters->ParamByName("param")-
>Value=s;
                          Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
                        flag=true;
                        }
                        Form1->ADOQuery1->Parameters->ParamByName("zachet")-
>Value="да";
                        }
                    if (zach2==1) {
```

```
                                                    if (flag==false) {
        Form1->ADOQuery1->Parameters->ParamByName("param")->Value=s;
                                    Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
                                    flag=true;
                                    }
                                    Form1->ADOQuery1->Parameters->ParamByName("zachet")-
>Value="да";
                                    }
                                if (zach3==1) {
                                    if (flag==false) {
                                      Form1->ADOQuery1->Parameters->ParamByName("param")-
>Value=s;
                                      Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
                                    flag=true;
                                    }
                                    Form1->ADOQuery1->Parameters->ParamByName("zachet")-
>Value="да";
                                    }
                                if (zach4==1) {
                                    if (flag==false) {
                                      Form1->ADOQuery1->Parameters->ParamByName("param")-
>Value=s;
                                      Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
                                    flag=true;
                                    }
                                    Form1->ADOQuery1->Parameters->ParamByName("zachet")-
>Value="да";
                                    }
                                if (zach5==1) {
                                    if (flag==false) {
                                      Form1->ADOQuery1->Parameters->ParamByName("param")-
>Value=s;
                                      Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
                                    flag=true;
                                    }
                                    Form1->ADOQuery1->Parameters->ParamByName("zachet")-
>Value="да";
                                    }
                                if (zach6==1) {
                                    if (flag==false) {
```

```
                Form1->ADOQuery1->Parameters->ParamByName("param")->Value=s;
                        Form1->ADOQuery1->Parameters->ParamByName("semestr")-
>Value=k;
                        flag=true;
                        }
                        Form1->ADOQuery1->Parameters->ParamByName("zachet")-
>Value="да";
                        }
                    Form1->ADOQuery1->ExecSQL();
            }
            Form1->ADOQuery1->Active=true;
}
//---------------------------------------------------------------------------

void __fastcall TForm4::FormCreate(TObject *Sender)
{
      //Form4->ADOConnection2->Connected=true;
      //Form4->ADOQuery1->Active=true;
}
```

Макрос приведения листа Excel к стандартному виду.

```
Sub Macros()
' Macros Макрос
For i = 2 To 250
  If Cells(i, 4).Value <> "Прима" Then
    Rows(i).Delete
    i = i - 1
    k = k + 1
    If k = 250 Then
      Exit For
    End If
  End If
Next i
Columns("A:BK").Select
Selection.EntireColumn.Hidden = False
Columns(1).Delete
Columns(22).Delete
Columns(22).Delete
Columns(22).Delete
Columns(22).Delete
Columns(22).Delete
Columns(22).Delete
Columns(22).Delete
```

```
Columns(22).Delete
Cells(1, 1).Value = "Номер"
Cells(1, 2).Value = "Дисциплина"
Cells(1, 3).Value = "Кафедра"
Cells(1, 4).Value = "Экзамен 1"
Cells(1, 5).Value = "Экзамен 2"
Cells(1, 6).Value = "Экзамен 3"
Cells(1, 7).Value = "Экзамен 4"
Cells(1, 8).Value = "Зачет 1"
Cells(1, 9).Value = "Зачет 2"
Cells(1, 10).Value = "Зачет 3"
Cells(1, 11).Value = "Зачет 4"
Cells(1, 12).Value = "Зачет 5"
Cells(1, 13).Value = "Зачет 6"
Cells(1, 14).Value = "Диф. зачет 1"
Cells(1, 15).Value = "Диф. зачет 2"
Cells(1, 16).Value = "Диф. зачет 3"
Cells(1, 17).Value = "Диф. зачет 4"
Cells(1, 18).Value = "КП 1"
Cells(1, 19).Value = "КП 2"
Cells(1, 20).Value = "КР 1"
Cells(1, 21).Value = "КР 2"
Cells(1, 22).Value = "Лекции"
Cells(1, 23).Value = "Практика"
Cells(1, 24).Value = "Лаб. работа"
Columns(25).Delete
Columns(25).Delete
Columns(25).Delete
Cells(1, 25).Value = "1-Неделя"
Cells(1, 26).Value = "1-ЗЕ"
Cells(1, 27).Value = "2-Неделя"
Cells(1, 28).Value = "2-ЗЕ"
Cells(1, 29).Value = "3-Неделя"
Cells(1, 30).Value = "3-ЗЕ"
Cells(1, 31).Value = "4-Неделя"
Cells(1, 32).Value = "4-ЗЕ"
Cells(1, 33).Value = "5-Неделя"
Cells(1, 34).Value = "5-ЗЕ"
Cells(1, 35).Value = "6-Неделя"
Cells(1, 36).Value = "6-ЗЕ"
Cells(1, 37).Value = "7-Неделя"
Cells(1, 38).Value = "7-ЗЕ"
Cells(1, 39).Value = "8-Неделя"
Cells(1, 40).Value = "8-ЗЕ"
```

```
For i = 1 To 12
   Columns(41).Delete
Next i
End Sub
```