СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc138742156)

[1. ИЗУЧЕНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc138742157)

[2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА 7](#_Toc138742158)

[2.1. Постановка задачи 7](#_Toc138742159)

[2.2. Проектирование разрабатываемого программного продукта 7](#_Toc138742160)

[2.2.1. Требования к языкам программирования в среде разработки. Обоснование выбора основных средств разработки 7](#_Toc138742161)

[2.2.2. Построение диаграмм 8](#_Toc138742162)

[2.3. Реализация разрабатываемого программного продукта 9](#_Toc138742163)

[2.3.1. Методы решения задач. Алгоритм работы программы 9](#_Toc138742164)

[3. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 11](#_Toc138742165)

[3.1. Функциональное тестирование 11](#_Toc138742166)

[ЗАКЛЮЧЕНИЕ 13](#_Toc138742167)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 14](#_Toc138742168)

[ПРИЛОЖЕНИЕ А 15](#_Toc138742169)

[ПРИЛОЖЕНИЕ Б 22](#_Toc138742170)

# 

# ВВЕДЕНИЕ

В современном обществе образование является важнейшей составляющей жизни каждого человека. При этом одной из ключевых задач образовательных учреждений является составление качественного учебного расписания, учитывающего множество параметров, таких как экономическая эффективность, удобство для студентов и преподавателей, ограничения времени и многое другое.

Однако задача составления расписания занятий является NP-трудной, что затрудняет ее автоматизацию. Ранее были предложены точные методы целочисленного программирования, однако они имеют ряд недостатков, таких как сложность математической модели и экспоненциальный рост временных затрат. В последнее время все чаще используются эвристические методы, в том числе генетические алгоритмы.

Цель данной курсовой работы - разработать генетический алгоритм для составления учебного расписания, учитывающий структурные особенности объектов расписания и имеющий адаптированные генетические операции. В работе будут рассмотрены существующие подходы к составлению расписания занятий, а также проанализированы особенности генетических алгоритмов и их применение в данной задаче.

# ИЗУЧЕНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Составление расписания занятий является одной из важных задач, решаемых при планировании учебного процесса. В первую очередь, это связано с тем, что без расписания занятий невозможно функционирование любого учебного заведения. Иными словами, расписание учебных занятий должно быть составлено своевременно и отвечать ряду требований и критериев, таких как экономическая эффективность использования ресурсов образовательной системы, комфортность учебы студентов и работы ППС, ограничения по времени обучения и т. д.

Задача составления расписания относится к задачам целочисленного программирования, сложность решения которых 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.

Указанные сложности препятствуют автоматизации процедуры составления расписания, несмотря на наличие широкого спектра методов целочисленного программирования, таких как методы полного перебора, метод ветвей и границ, метод раскраски графов, эвристические методы. Так, в группе работ для автоматизации процедуры составления расписания занятий 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 (преподаватели, группы, аудитории, дисциплины, временные интервалы занятий), между которыми существуют сильные связи, обусловленные спецификой организации учебного процесса.

В последнее время все чаще для решения большеразмерных задач целочисленного программирования, в том числе задачи составления расписания, используют различные эвристические методы. К числу таких методов относятся так называемые генетические алгоритмы, которые и являются предметом исследования в настоящей статье.

С учетом всего вышесказанного можно сказать, что разработка генетических алгоритмов оптимизации, учитывающих структурные особенности расписания учебных занятий, является актуальной задачей. В настоящей работе для составления расписания занятий предлагается композиционный генетический алгоритм составления расписания учебных занятий, который учитывает особенности объектов расписания и позволяет получать более оптимальные решения, чем классические методы целочисленного программирования.

# ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА

## Постановка задачи

Цель курсовой работы - разработать алгоритм для составления учебного расписания, используя генетические алгоритмы.

Для решения данной задачи необходимо выполнить следующие задачи:

1) Изучить существующие методы составления учебного расписания и выбрать подходящий метод для решения задачи.

2) Разработать генетический алгоритм, который будет использоваться для оптимизации задачи составления учебного расписания.

3) Сделать выводы и предложить возможные направления дальнейших исследований в данной области.

Результатом курсовой работы должен быть рабочий алгоритм для составления учебного расписания, который может быть использован в образовательных учреждениях для оптимизации учебного процесса.

## Проектирование разрабатываемого программного продукта

### Требования к языкам программирования в среде разработки. Обоснование выбора основных средств разработки

При выборе языка программирования для разработки программы, важно учитывать не только его синтаксис и возможности, но и возможности интеграции с другими инструментами, необходимыми для создания приложения. Для разработки программы "Составление учебного расписания с использованием генетических алгоритмов" был выбран язык C#, который хорошо подходит для создания приложений Windows и имеет широкий набор библиотек для работы с различными типами данных.

Для разработки программы использовалась среда разработки Microsoft Visual Studio, которая предоставляет широкий набор инструментов для создания приложений на языке C#. В частности, в Visual Studio есть интегрированная среда разработки (IDE), которая позволяет разрабатывать, отлаживать и тестировать приложение в одной среде. Также в Visual Studio есть интегрированные средства для работы с базами данных, создания пользовательских интерфейсов и отладки кода.

В целом, выбор языка C# и среды разработки Microsoft Visual Studio для создания программы "Составление учебного расписания с использованием генетических алгоритмов" обоснован, так как они предоставляют все необходимые инструменты и возможности для успешной разработки приложения.

### Построение диаграмм

Для наглядного представления функциональности программы была спроектирована диаграмма вариантов использования (рисунок 1).

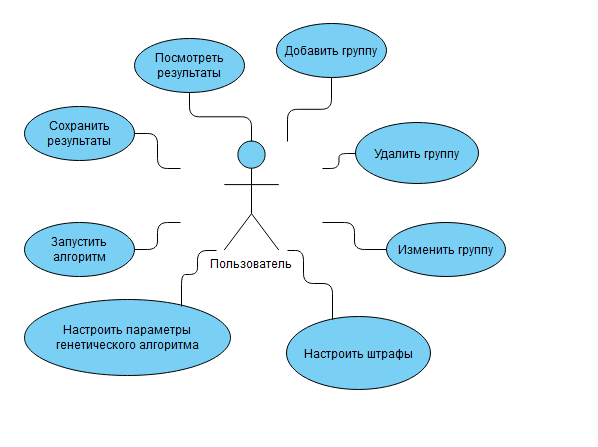
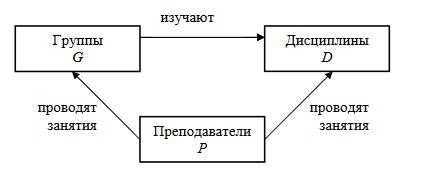


Рисунок 1 – Диаграмма вариантов использования

Ради лучшего понимания связей между сущностями алгоритма приведена структура объекта «занятие» (рисунок 2).

Рисунок 2 – Структура объекта «занятие»

В завершении, чтобы представить работу базового генетического алгоритма, необходимо рассмотреть блок-схему имитации эволюционного процесса (рисунок 3).

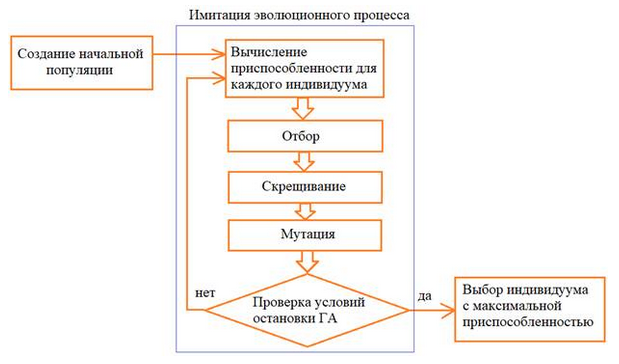


Рисунок 3 – Блок-схема имитации эволюционного процесса

## Реализация разрабатываемого программного продукта

### Методы решения задач. Алгоритм работы программы

Перед запуском алгоритма пользователю необходимо заполнить список исходных данных, то есть группы, которые включают в себя предметы, преподавателей и кабинеты.

Следом, он должен настроить параметры алгоритма (количество популяции, поколений, вероятность кроссинговера, вероятность мутации и другие), затем необходимо настроить систему штрафов (штраф за неполные пары, несоответствие предметов преподавателя и так далее). От данных значений зависит насколько правильно алгоритм исполнит свою задачу.

После приведенных действий пользователь может запустить алгоритм и проанализировать его работу, смотря на график зависимости минимальной, средней и максимальной приспособленности от поколения. И в зависимости от конечного результата либо изменить гиперпараметры, чтобы улучшить следующий результат, либо сохранить расписание для последующего его использования.

# 

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

## Функциональное тестирование

Функциональное тестирование является одним из ключевых этапов тестирования приложения "Генетический алгоритм составления расписания". Оно проводится для проверки корректности работы приложения в соответствии с его функциональными требованиями

Результаты тестирования отображены в таблице:

Таблица 1- Результаты функционального тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тестовый сценарий | Ожидаемый результат | Фактический результат | Статус |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Добавить группу | Группа добавлена в общий список | Группа добавлена в общий список | ✓ |
| 2. | Удаление группы | Группа была успешно удалена из общего списка | Группа была успешно удалена из общего списка | ✓ |
| 3 | Изменение группы | Изменения были успешно сохранены | Изменения были успешно сохранены | ✓ |
| 4 | Настройка параметров алгоритма | Все параметры были успешно сохранены | Все параметры были успешно сохранены | ✓ |
| 5 | Указание штрафов | Все штрафы были успешно сохранены | Все штрафы были успешно сохранены | ✓ |
| 6 | Запуск алгоритма | Алгоритм успешно запустился и прошел указанное количество поколений | Алгоритм успешно запустился и прошел указанное количество поколений | ✓ |
| 7 | Сохранение расписания группы | Расписание группы было успешно сохранено в .xlsx формате | Расписание группы было успешно сохранено в .xlsx формате | ✓ |

В результате функционального тестирования было установлено, что программный продукт соответствует функциональным требованиям и обеспечивает необходимый функционал для успешной работы с пользователем.

# ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была решена задача составления учебного расписания с использованием генетических алгоритмов. Введение в работе показало, что составление расписания является сложной задачей, которая может быть оптимизирована с помощью генетических алгоритмов.

Для решения поставленной задачи были выполнены следующие задачи: изучены существующие методы составления учебного расписания и выбран подходящий метод для решения задачи; разработан генетический алгоритм, который был использован для оптимизации задачи составления учебного расписания; сделаны выводы и предложены возможные направления дальнейших исследований в данной области.

Разработанный генетический алгоритм показал высокую эффективность в задаче составления учебного расписания. Он позволяет учитывать множество параметров, таких как экономическая эффективность, удобство для студентов и преподавателей, ограничения времени и многое другое. Кроме того, генетический алгоритм является гибким и адаптивным, что позволяет его использование в различных условиях.

В заключение можно отметить, что разработанный алгоритм может быть использован в образовательных учреждениях для оптимизации учебного процесса. Он позволит сократить временные затраты на составление расписания, улучшить его качество и повысить удобство для студентов и преподавателей. Также возможно дальнейшее улучшение алгоритма, например, путем учета дополнительных параметров и ограничений.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Документация по языку программирования C# и платформе .NET [Электронный ресурс]. - <https://learn.microsoft.com/en-us/dotnet/csharp/>
2. "Genetic Algorithm-based Timetabling System"
3. "A hybrid genetic algorithm for course timetabling problem"
4. "A Genetic Algorithm for Course Timetabling Problem"

# 

# ПРИЛОЖЕНИЕ А

ЛИСТИНГ ПРОГРАММЫ

public class GA

{

public int days;

public int hours;

public List<List<string>> teacher;

public List<List<List<string>>> teacherSubject;

public List<List<List<int>>> teacherSubjectNum;

public List<List<string>> cabinets;

public List<List<string>> subjects;

public List<List<int>> subjectNeeds;

public List<Individual> population;

public Individual hof;

public int groups;

public int group\_size;

public int individual\_size;

public int population\_size;

public int max\_generation;

public int tournament\_size;

public double p\_crossover;

double p\_mutation;

Random random;

public Fines fines;

public GA(Settings settings, Fines fines, Groups groups\_list)

{

this.days = settings.days;

this.hours = settings.hours;

this.groups = settings.groups;

this.population\_size = settings.population\_size;

this.max\_generation = settings.max\_generation;

this.tournament\_size = settings.tournament\_size;

this.p\_crossover = settings.p\_crossover;

this.p\_mutation = settings.p\_mutation;

this.fines = fines;

group\_size = hours \* days \* 3;

individual\_size = group\_size \* groups;

teacher = groups\_list.teacher;

teacherSubject = groups\_list.teacherSubject;

teacherSubjectNum = groups\_list.teacherSubjectNum;

cabinets = groups\_list.cabinets;

subjects = groups\_list.subjects;

subjectNeeds = groups\_list.subjectNeeds;

population = new List<Individual>();

random = new Random();

}

public void start(Train output)

{

population = populationCreator();

List<double> avg = new List<double>();

List<double> min = new List<double>();

List<double> max = new List<double>();

List<double> generations = new List<double>();

for (int gen = 0; gen < max\_generation; gen++)

{

population = selTournament();

population = cxTwoPoints();

population = mutUniformInt();

avg.Add(population.Average(ind => ind.Fitness()));

min.Add(population.Min(ind => ind.Fitness()));

max.Add(population.Max(ind => ind.Fitness()));

generations.Add(gen);

Hof();

output.RefreshListBox(

avg[avg.Count() - 1],

generations[generations.Count() - 1],

min[min.Count() - 1],

max[max.Count() - 1]);

output.Refresh();

output.DrawGraph(generations, avg, max, min);

}

}

public void Hof()

{

var select = new List<Individual>(population);

select.Add(hof);

hof = select.Min();

}

public int[] individualCreator()

{

int[] individual = new int[individual\_size];

int group\_id = 0;

int caret = 0;

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

{

if (i % group\_size == 0 && i != 0)

group\_id += 1;

if (caret == 3)

caret = 0;

if (caret == 0)

individual[i] = random.Next(0, subjects[group\_id].Count());

else if (caret == 1)

individual[i] = random.Next(0, teacher[group\_id].Count());

else if (caret == 2)

individual[i] = random.Next(0, cabinets[group\_id].Count());

caret += 1;

}

return individual;

}

public List<Individual> populationCreator()

{

List<Individual> population = new List<Individual>();

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

population.Add(new Individual(individualCreator(), this));

return population;

}

public List<Individual> selTournament()

{

List<Individual> choosen = new List<Individual>();

List<Individual> new\_population = new List<Individual>();

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

{

choosen.Clear();

for (int j = 0; j < tournament\_size; j++)

{

choosen.Add(population[random.Next(0, population\_size)]);

}

new\_population.Add(choosen.Min());

}

return new\_population;

}

public List<Individual> cxTwoPoints()

{

List<Individual> new\_population = new List<Individual>();

int[] ind1, ind2;

List<int> new\_ind1 = new List<int>();

List<int> new\_ind2 = new List<int>();

while (new\_population.Count() <= population\_size)

{

new\_ind1.Clear();

new\_ind2.Clear();

ind1 = population[random.Next(0, population\_size)].individual;

ind2 = population[random.Next(0, population\_size)].individual;

if (random.NextDouble() < p\_crossover)

{

int cxPoint1 = random.Next(0, individual\_size);

int cxPoint2 = random.Next(0, individual\_size - 1);

if (cxPoint2 >= cxPoint1)

cxPoint2 += 1;

else

(cxPoint1, cxPoint2) = (cxPoint2, cxPoint1);

new\_ind1.AddRange(ind1.ToList().GetRange(0, cxPoint1));

new\_ind1.AddRange(ind2.ToList().GetRange(cxPoint1, cxPoint2 - cxPoint1));

new\_ind1.AddRange(ind1.ToList().GetRange(cxPoint2, individual\_size - cxPoint2));

new\_ind2.AddRange(ind2.ToList().GetRange(0, cxPoint1));

new\_ind2.AddRange(ind1.ToList().GetRange(cxPoint1, cxPoint2 - cxPoint1));

new\_ind2.AddRange(ind2.ToList().GetRange(cxPoint2, individual\_size - cxPoint2));

new\_population.Add(new Individual(new\_ind1.ToArray(), this));

new\_population.Add(new Individual(new\_ind2.ToArray(), this));

}

}

return new\_population;

}

public List<Individual> mutUniformInt()

{

List<Individual> new\_population = new List<Individual>();

Individual new\_ind;

int len = 0;

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

{

new\_ind = population[i];

if (random.NextDouble() < p\_mutation)

{

int group\_id = 0;

int caret = 0;

for (int j = 0; j < individual\_size; j++)

{

if (j % group\_size == 0 && j != 0)

group\_id += 1;

if (caret == 3)

caret = 0;

if (caret == 0)

len = subjects[group\_id].Count();

else if (caret == 1)

len = teacher[group\_id].Count();

else if (caret == 2)

len = cabinets[group\_id].Count();

if (random.NextDouble() < 1.0 / individual\_size)

{

new\_ind.individual[j] = random.Next(0, len);

}

caret += 1;

}

}

new\_population.Add(new\_ind);

}

return new\_population;

}

} {

throw new NotImplementedException();

}

}

}

# 

# ПРИЛОЖЕНИЕ Б

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Руководство пользователя для проекта "Генетический алгоритм составления расписания":

Главное меню:

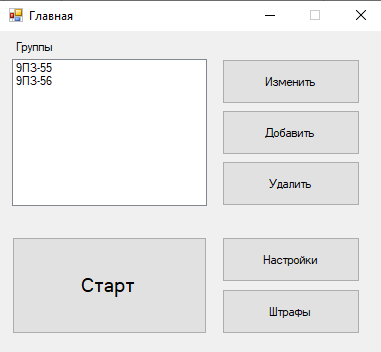


Рисунок Б.1 – Пример работы программы 1

Меню групп:

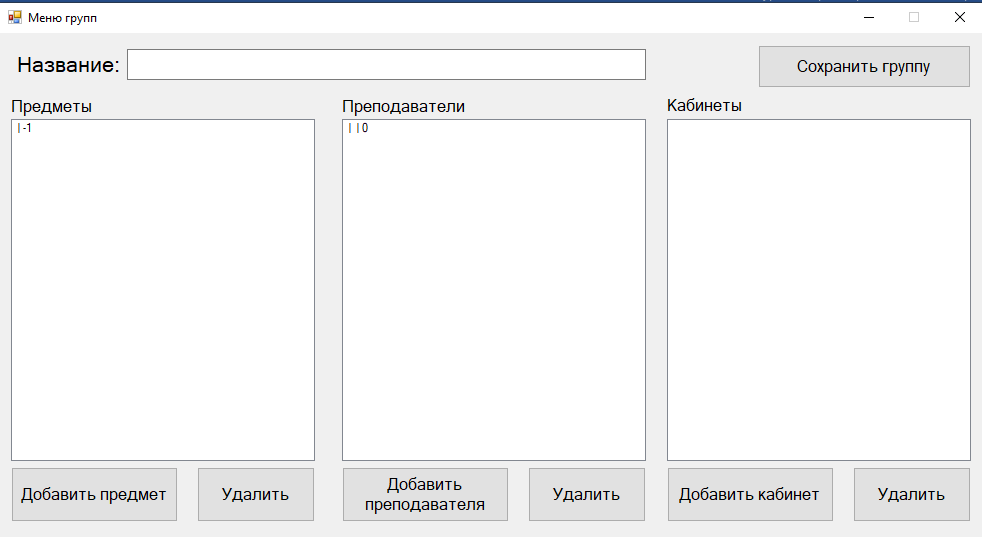


Рисунок Б.2 – Пример работы программы 2

Добавление предмета:

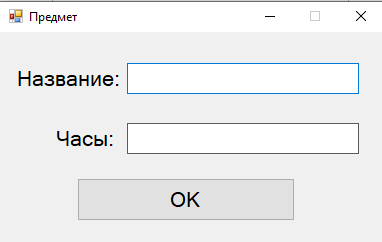


Рисунок Б.3 - Пример работы программы 3

Добавление преподавателя:

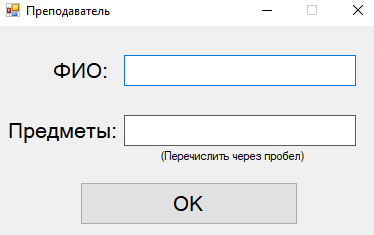


Рисунок Б.4 - Пример работы программы 4

Добавление кабинета:

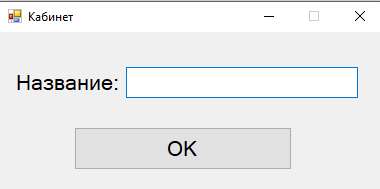


Рисунок Б.5 - Пример работы программы 5

Настройки алгоритма:

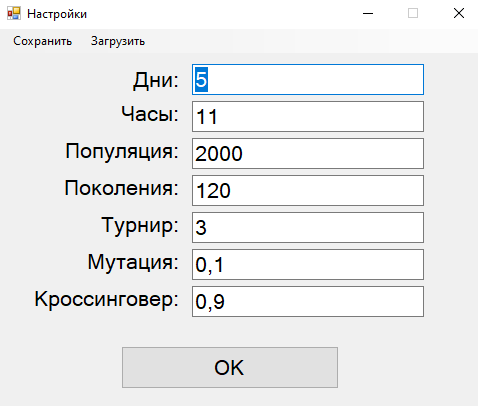


Рисунок Б.6 - Пример работы программы 6

Настройка штрафов:

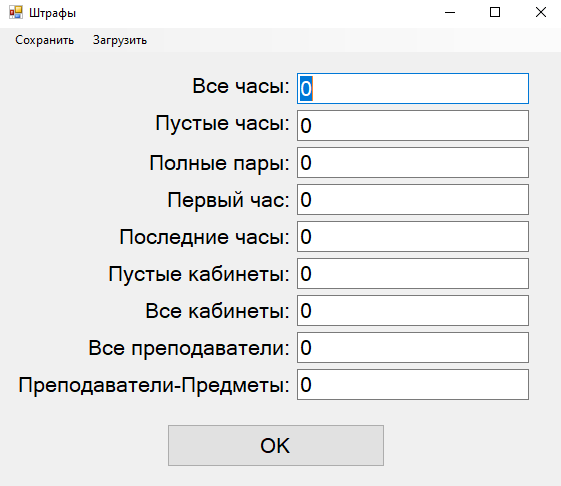


Рисунок Б.7 - Пример работы программы 7

Генетический алгоритм:

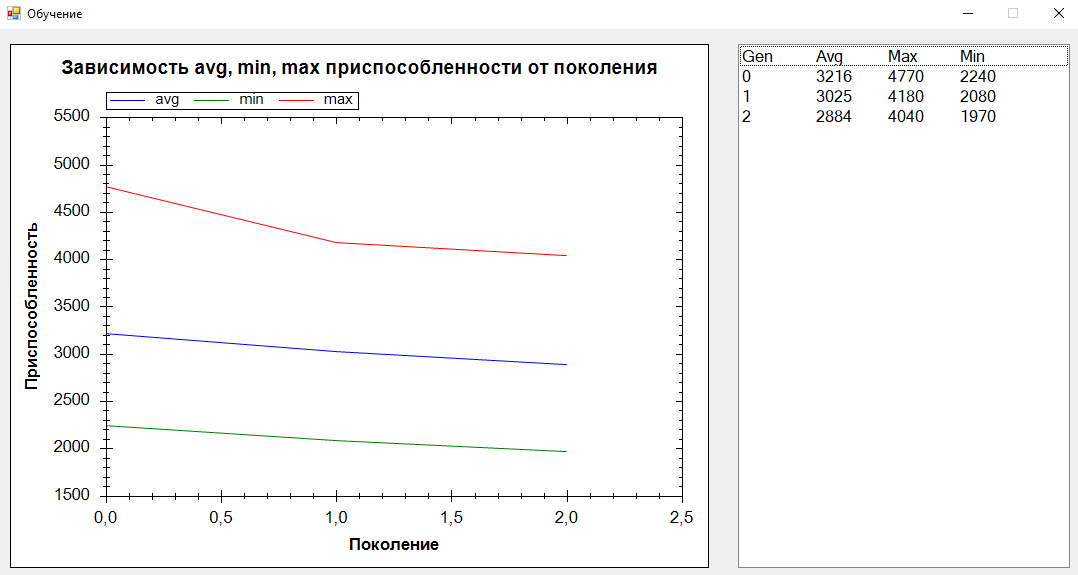


Рисунок Б.8 - Пример работы программы 8

Расписание групп:

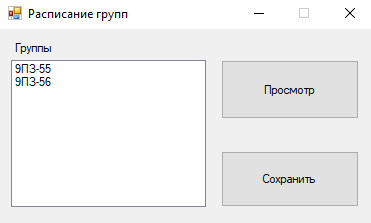


Рисунок Б.9 - Пример работы программы 9

Расписание группы:

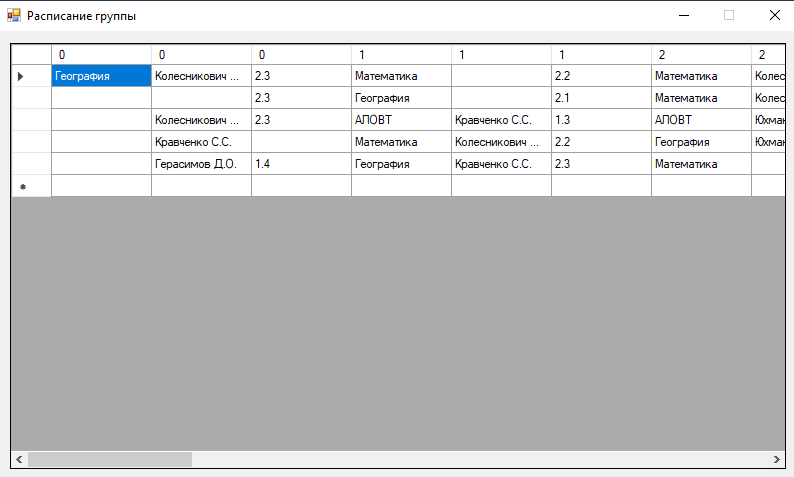


Рисунок Б.10 - Пример работы программы 10