Лабораторная работа № 2.  
Active server pages (ASP) .Net

**Цель лабораторной работы**

Закрепление теоретических знаний по основам разработки приложений на C# с помощью технологии ASP.NET.

**Постановка задачи**

ASP.NET — это платформа для разработки веб-приложений, созданная компанией Microsoft. Она позволяет разработчикам создавать динамические веб-сайты, веб-приложения и веб-сервисы. ASP.NET поддерживает языки .NET, такие как C#, и предоставляет мощные инструменты и библиотеки для упрощения разработки.

**Основные компоненты ASP.NET**

1. **ASP.NET Web Forms**: Позволяет создавать веб-приложения с использованием событийно-ориентированной модели, аналогичной Windows-приложениям. Подходит для быстрого создания приложений с минимальным количеством кода.
2. **ASP.NET MVC**: Архитектурный паттерн Model-View-Controller, который разделяет приложение на три основных компонента:
   * **Model**: Представляет данные и бизнес-логику.
   * **View**: Отвечает за отображение данных пользователю.
   * **Controller**: Обрабатывает пользовательские запросы, взаимодействует с моделью и выбирает представление для отображения.
3. **ASP.NET Web API**: Позволяет создавать RESTful сервисы, которые могут быть использованы для взаимодействия с клиентскими приложениями, такими как мобильные приложения и одностраничные приложения (SPA).
4. **ASP.NET Core**: Современная версия ASP.NET, которая является кросс-платформенной, высокопроизводительной и более легковесной. Поддерживает создание как веб-приложений, так и API.

**Пример реализации на C#**

Рассмотрим пример расчёта суммы ряда.

**Шаг 1: Установка**

Необходимо открыть Visual Studio Installer, выбрать «ASP.NET и разработка веб-приложений» (рисунок 1) и дождаться окончания установки.

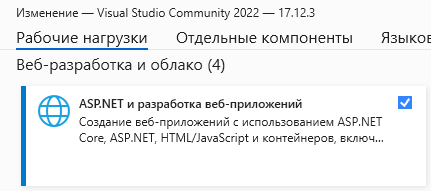


Рисунок 1 – Установка

**Шаг 2: Создание проекта**

При создании проекта необходимо выбрать «Пустой шаблон ASP.NET Core (Майкрософт)» (рисунок 2).

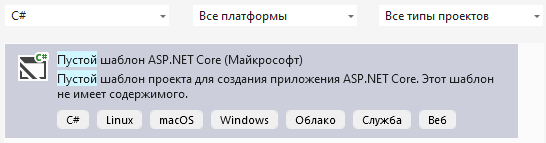


Рисунок 2 – Создание проекта

**Шаг 3: Иерархия классов и папок в решении**

Создать папки и классы согласно рисунку 3.

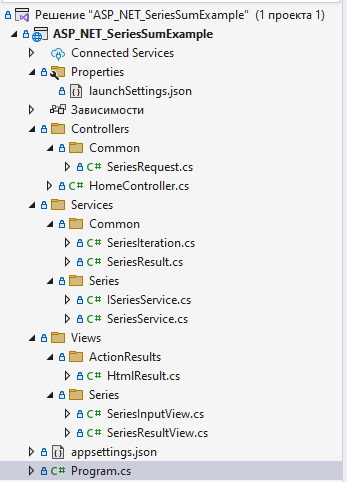


Рисунок 3 – Иерархия

**Шаг 4: Интерфейс сервиса расчёта суммы ряда**

using ASP\_NET\_SeriesSumExample.Services.Common;

namespace ASP\_NET\_SeriesSumExample.Services.Series;

public interface ISeriesService

{

SeriesResult Calculate(Func<long, double> function, long n0, long N);

}

**Шаг 5: Сервис расчёта суммы ряда**

using ASP\_NET\_SeriesSumExample.Services.Common;

namespace ASP\_NET\_SeriesSumExample.Services.Series;

public class SeriesService : ISeriesService

{

public SeriesResult Calculate(Func<long, double> function, long n0, long N)

{

var iterations = new List<SeriesIteration>();

var sum = 0.0;

for (var i = n0; i < n0 + N; i++)

{

var element = function(i);

sum += element;

iterations.Add(new SeriesIteration(i, element, sum));

}

return new SeriesResult(iterations);

}

}

**Шаг 6: Результат итерации расчёта суммы ряда**

namespace ASP\_NET\_SeriesSumExample.Services.Common;

public record SeriesIteration(long N, double Element, double CurrentSum);

**Шаг 7: Результат расчёта суммы ряда**

using System.Collections;

namespace ASP\_NET\_SeriesSumExample.Services.Common;

public record SeriesResult(IReadOnlyCollection<SeriesIteration> Iterations) : IEnumerable<SeriesIteration>

{

public double TotalSum => Iterations.Last().CurrentSum;

public IEnumerator<SeriesIteration> GetEnumerator()

{

return Iterations.GetEnumerator();

}

IEnumerator IEnumerable.GetEnumerator()

{

return Iterations.GetEnumerator();

}

}

**Шаг 8: Класс результата в формате страницы HTML**

using Microsoft.AspNetCore.Mvc;

namespace ASP\_NET\_SeriesSumExample.Views.ActionResults;

public class HtmlResult : IActionResult

{

private readonly string body;

public HtmlResult(string body)

{

this.body = body;

}

public string GetHTML()

{

return @$"

<!DOCTYPE html>

<html>

<head>

<title>Лабораторная работа 1 Иванов И.И.</title>

<meta charset=utf-8 />

</head>

<body>{body}</body>

</html>";

}

public async Task ExecuteResultAsync(ActionContext context)

{

var fullHtmlCode = GetHTML();

await context.HttpContext.Response.WriteAsync(fullHtmlCode);

}

public override string ToString()

{

return GetHTML();

}

}

**Шаг 9: Класс для отображения формы ввода параметров ряда в виде страницы HTML**

using ASP\_NET\_SeriesSumExample.Views.ActionResults;

using Microsoft.AspNetCore.Mvc;

namespace ASP\_NET\_SeriesSumExample.Views.Series;

public class SeriesInputView : IActionResult

{

public async Task ExecuteResultAsync(ActionContext context)

{

string content = @$"

<h1>Лабораторная работа 1</h1>

<h2>Работу выполнил Иванов И.И.</h2>

<form method='post'>

{GetSeriesHTML()}

<label>Номер первого члена ряда n0:</label><br>

<input name='request.n0' /><br>

<label>Количество членов ряда N:</label><br>

<input name='request.N' /><br>

<input type='submit' value='Send'/>

</form>";

var result = new HtmlResult(content);

await context.HttpContext.Response.WriteAsync(result.GetHTML());

}

/// <summary>

/// Математическая формула ряда в формате html

/// </summary>

/// <returns></returns>

private string GetSeriesHTML()

{

return """

<p>

<math display="block">

<mrow>

<munderover>

<mo>∑</mo>

<mrow>

<mi>n</mi>

<mo>=</mo>

<mn>n0</mn>

</mrow>

<mrow>

<mn>n0 + N</mn>

</mrow>

</munderover>

<mfrac>

<mn>1</mn>

<msup>

<mi>n</mi>

<mn>2</mn>

</msup>

</mfrac>

</mrow>

</math>

</p>

""";

}

}

**Шаг 10: Класс для отображения результата расчета суммы ряда в виде страницы HTML**

using System.Text;

using ASP\_NET\_SeriesSumExample.Controllers.Common;

using ASP\_NET\_SeriesSumExample.Services.Common;

using ASP\_NET\_SeriesSumExample.Views.ActionResults;

using Microsoft.AspNetCore.Mvc;

namespace ASP\_NET\_SeriesSumExample.Views.Series;

public class SeriesResultView : IActionResult

{

private readonly SeriesRequest \_request;

private readonly SeriesResult \_result;

public SeriesResultView(SeriesRequest request, SeriesResult result)

{

\_request = request;

\_result = result;

}

public async Task ExecuteResultAsync(ActionContext context)

{

var sb = new StringBuilder();

sb.Append(@$"Сумма первых N={\_request.N} членов ряда равна: {\_result.TotalSum:F8}<br>");

sb.Append("Расчёты представлены в таблице ниже<br>");

sb.Append("""

<table>

<tr>

<th>Итерация</th>

<th>Член ряда</th>

<th>Сумма</th>

</tr>

""");

foreach (var (iteration, element, sum) in \_result)

{

sb.Append($"""

<tr>

<td>{iteration}</td>

<td>{element}</td>

<td>{sum}</td>

</tr>

""");

}

sb.Append("</table>");

var html = new HtmlResult(sb.ToString());

await context.HttpContext.Response.WriteAsync(html.GetHTML());

}

}

**Шаг 11: Класс запроса к контроллеру**

namespace ASP\_NET\_SeriesSumExample.Controllers.Common;

public record class SeriesRequest(long n0, long N);

**Шаг 12: Контроллер**

using ASP\_NET\_SeriesSumExample.Controllers.Common;

using ASP\_NET\_SeriesSumExample.Services.Series;

using ASP\_NET\_SeriesSumExample.Views.Series;

using Microsoft.AspNetCore.Mvc;

namespace ASP\_NET\_SeriesSumExample.Controllers;

public class HomeController : Controller

{

[HttpGet]

public IActionResult Index()

{

return new SeriesInputView();

}

[HttpPost]

public IActionResult Index([FromServices] ISeriesService seriesService, SeriesRequest request)

{

var seriesResult = seriesService.Calculate((n) => 1.0 / (n \* n), request.n0, request.N);

return new SeriesResultView(request, seriesResult);

}

}

**Шаг 13: Точка входа в программу**

using ASP\_NET\_SeriesSumExample.Services.Series;

namespace ASP\_NET\_SeriesSumExample;

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddTransient<ISeriesService, SeriesService>();

builder.Services.AddControllers();

var app = builder.Build();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

}

}

**Задание на лабораторную работу**

1. Разработать ASP.NET приложение для расчёта суммы ряда и интеграла.
2. Программа должна предусматривать:

2.1. обращение по 2 адресам:

а) **/Series/Sum** – для расчёта суммы ряда;

б) **/Integral/Integrate** – для интегрирования;

2.2. ввод параметров:

а) количество членов ряда или точность;

б) пределов интегрирования, количества разбиений области интегрирования.

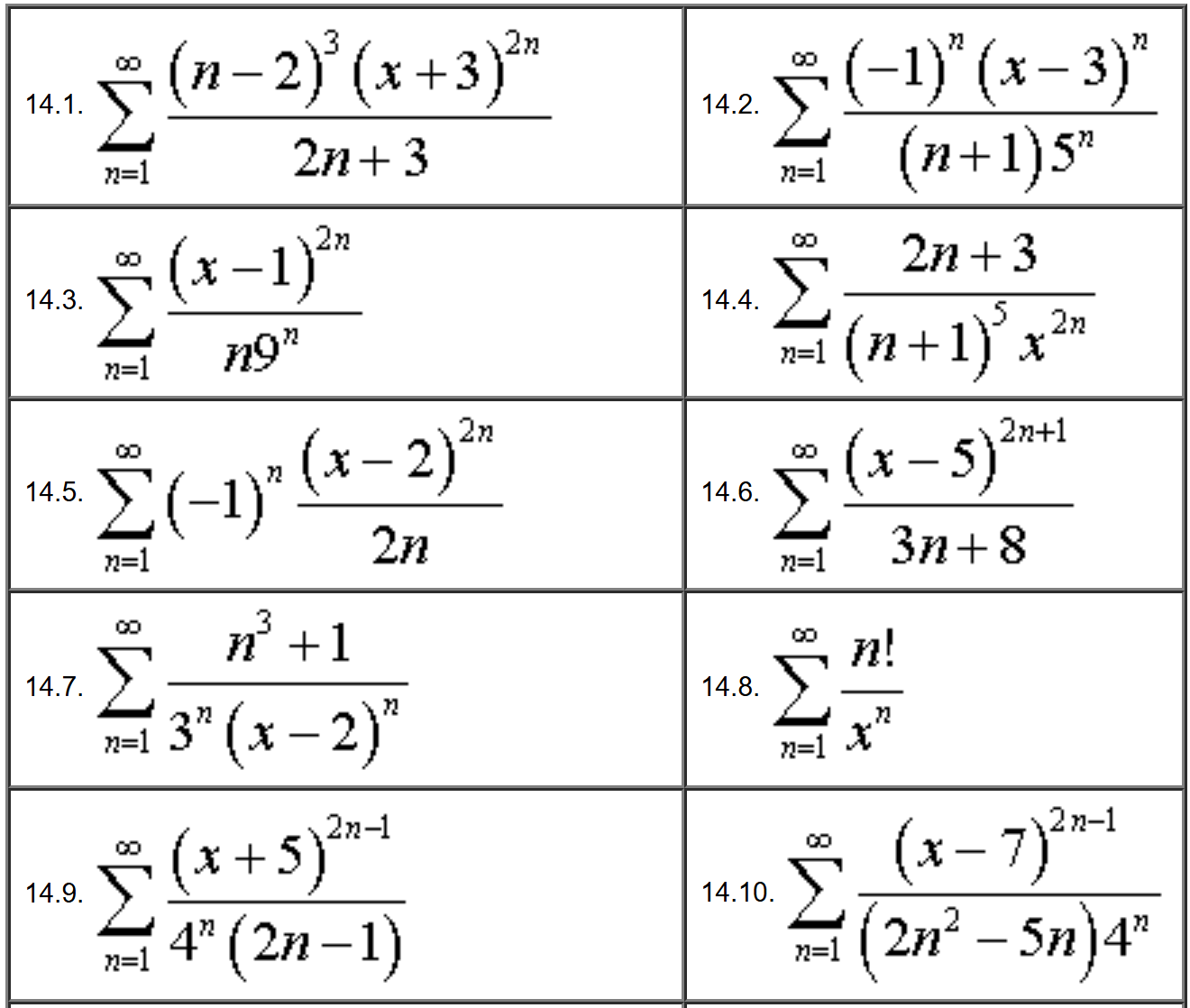
2.3. **Для оценки отлично:** выбор метода интегрирования, реализованный с помощью паттерна стратегия.

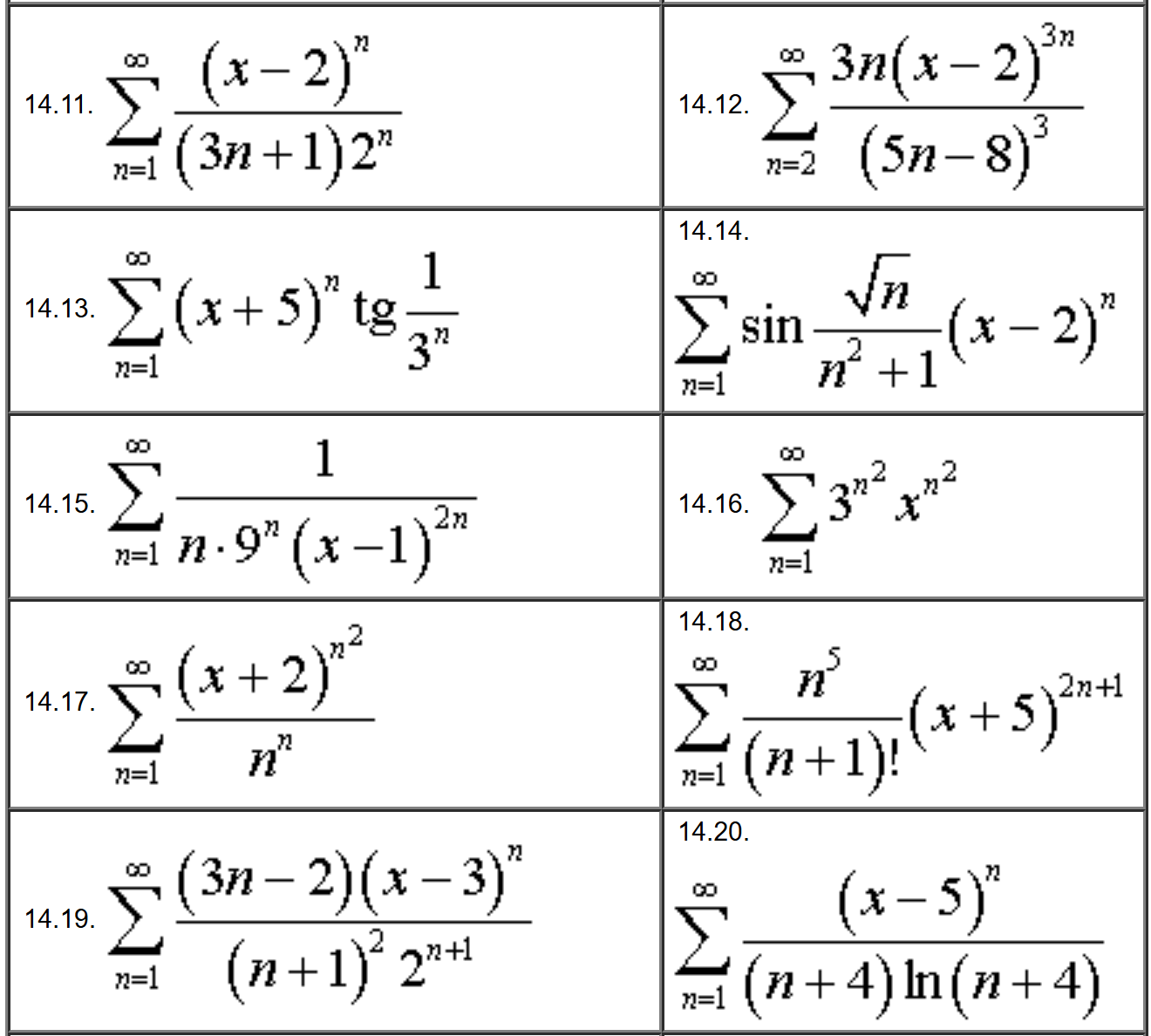
1. Для успешной защиты:

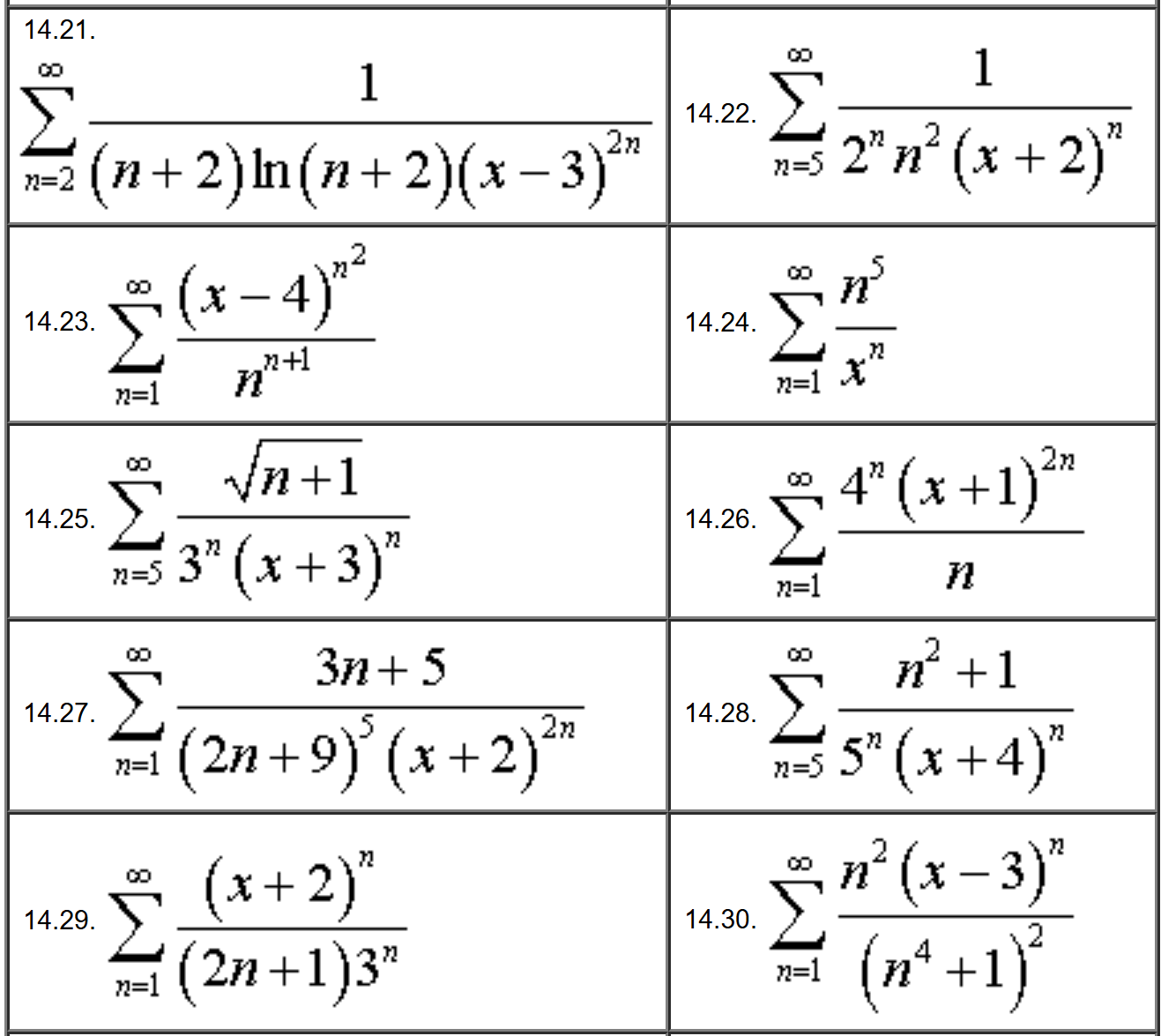
* программа должна быть задокументирована с помощью комментариев,
* код программы не должен содержать не используемые блоки, должен быть «чистым» и стилистически верно оформлен,
* интерфейс программы должен быть гибким и удобным.

**Варианты заданий на лабораторную работу**

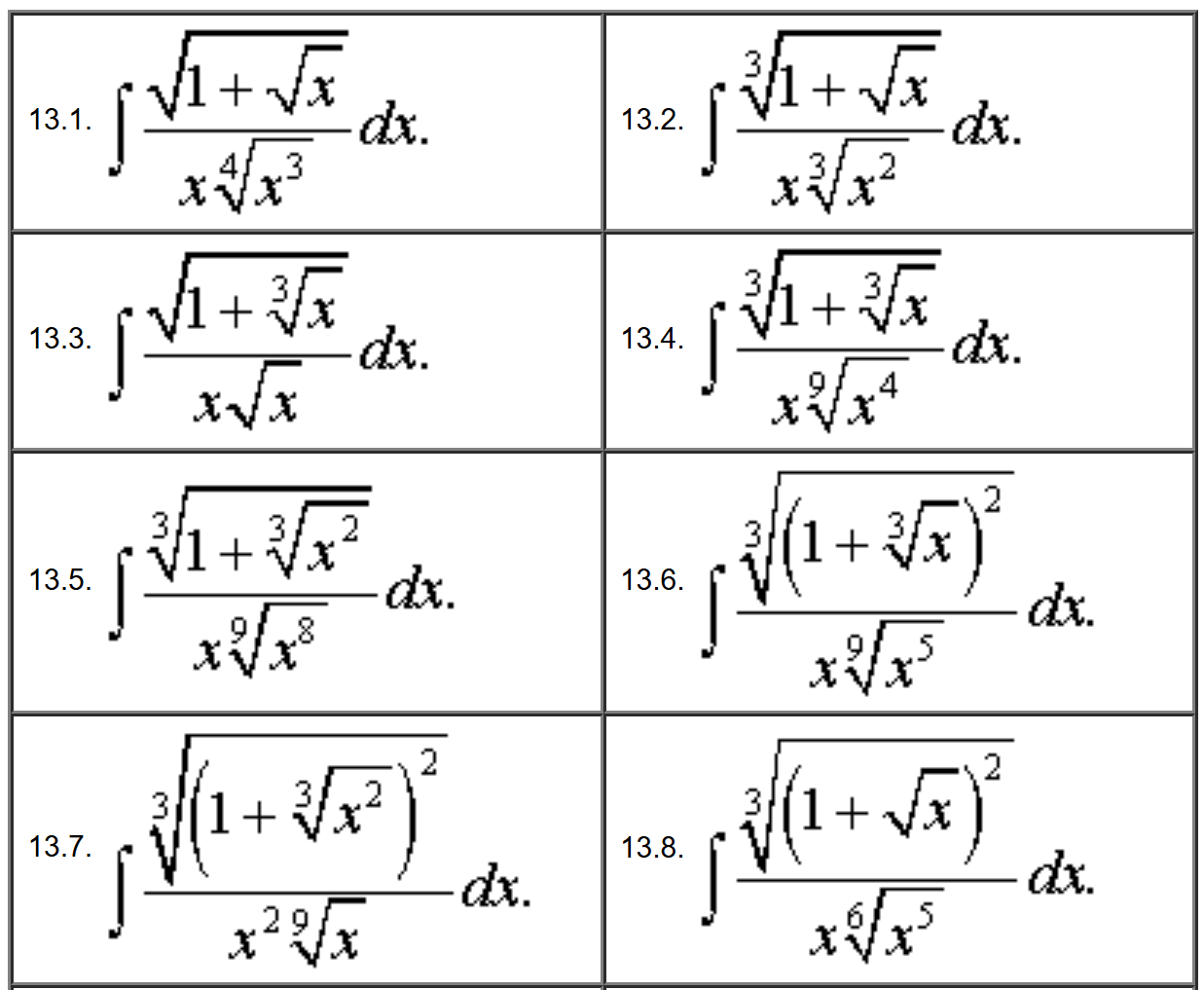
Ряды (*x* задаётся пользователем).

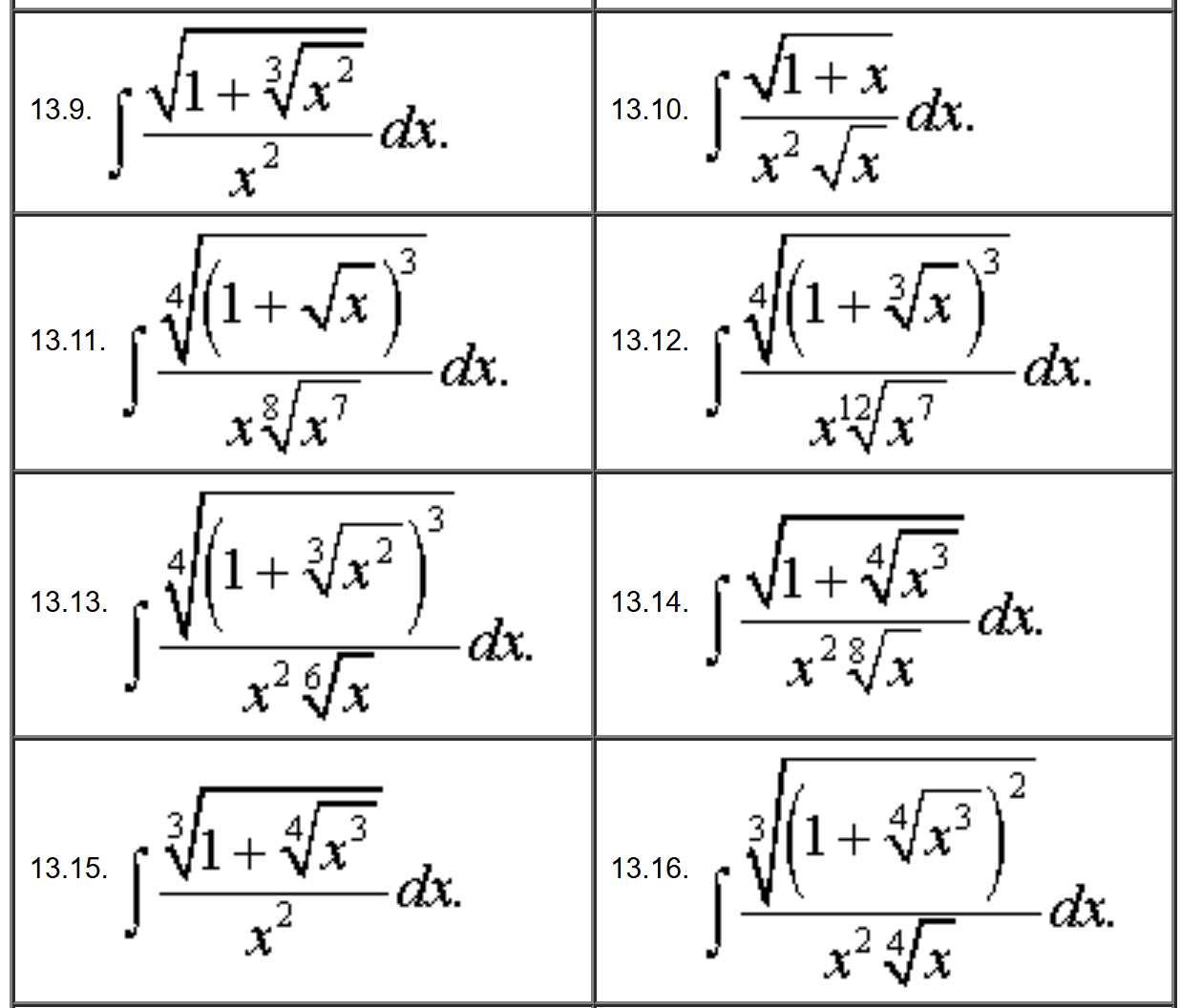


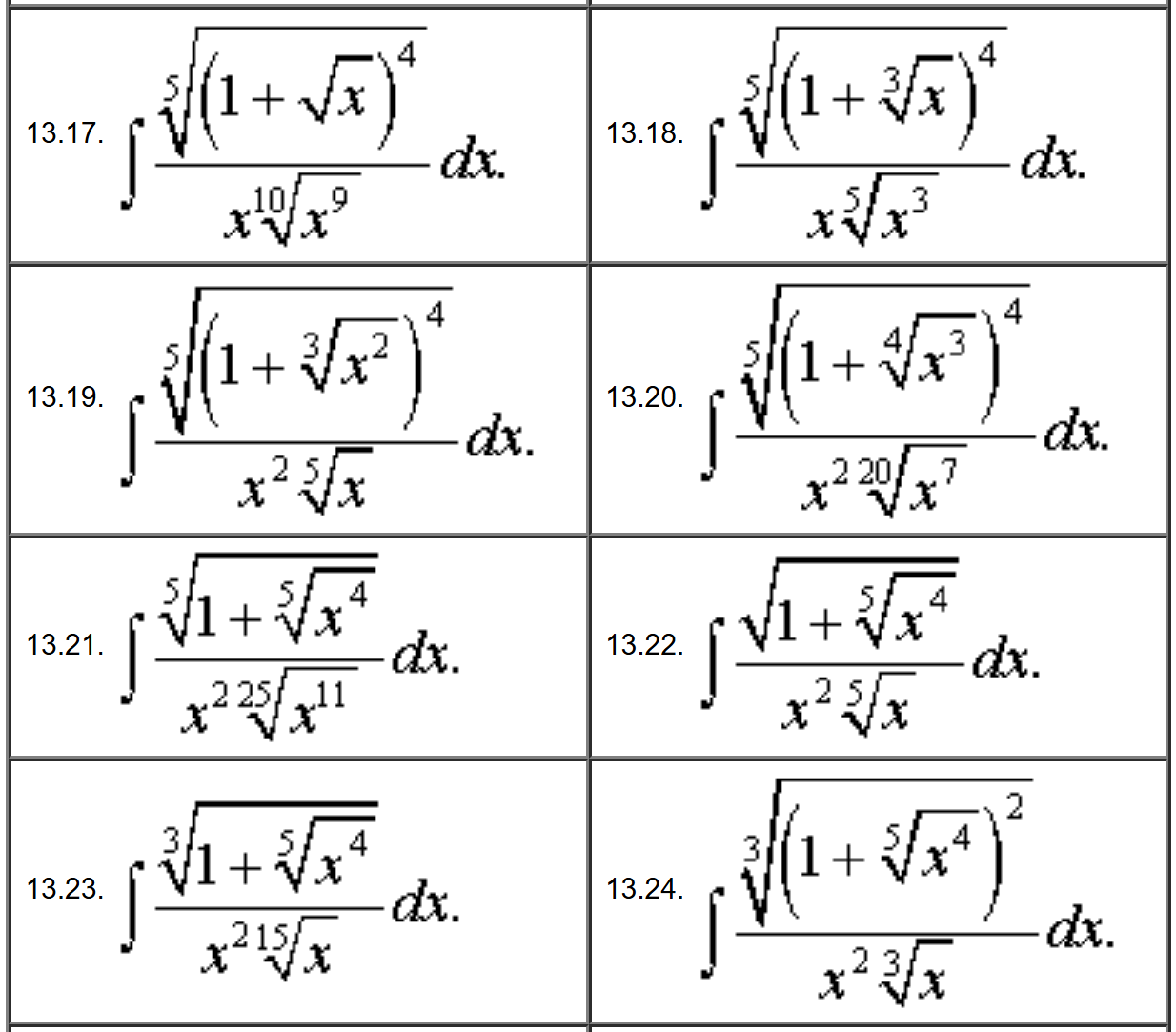


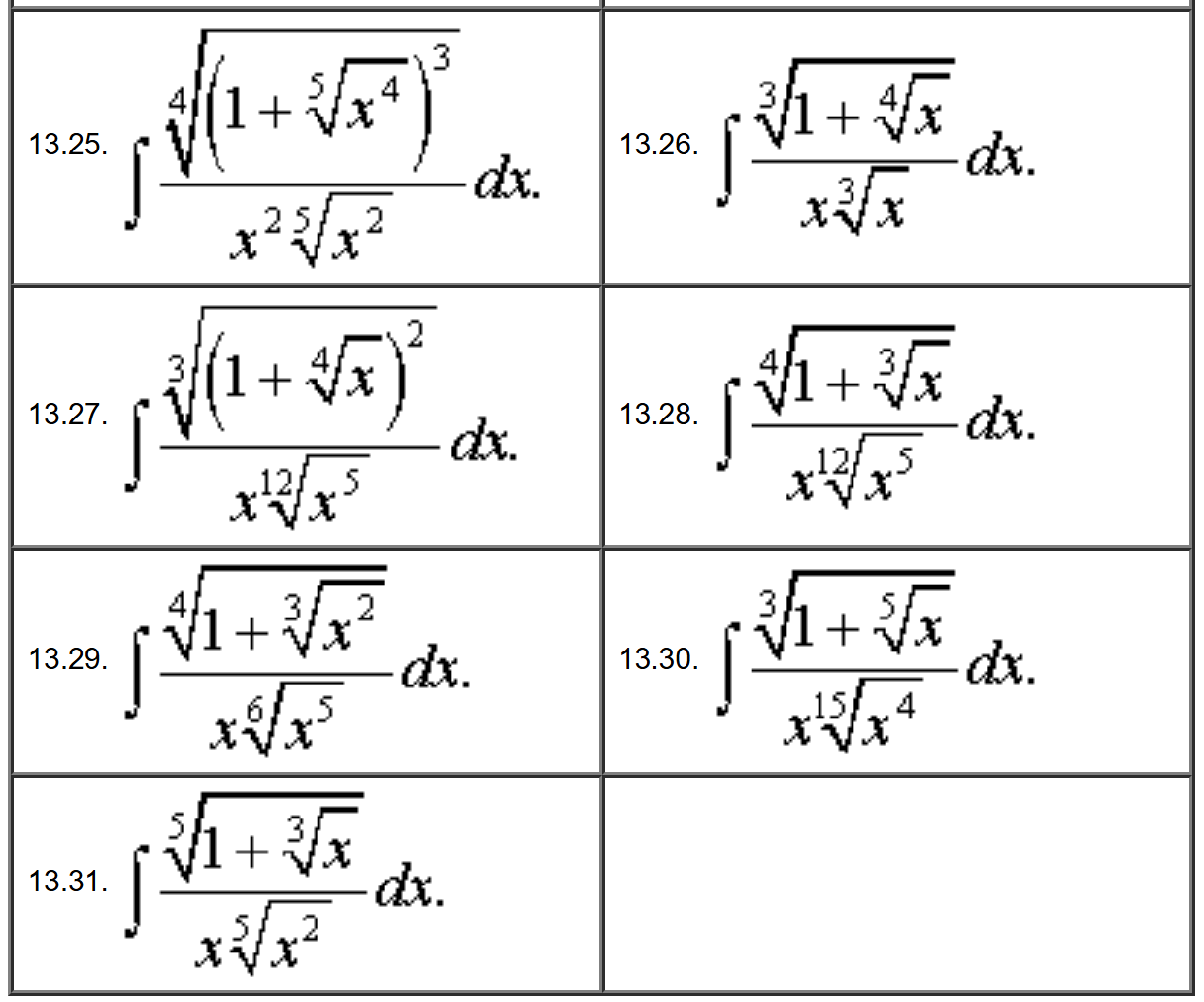


Интегралы









Методы интегрирования

<https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D1%83%D1%80%D0%BD%D1%8B%D1%85_%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB>

1. Прямоугольников, Гаусса-2, Симпсона
2. Трапеция, Гаусса-4, Прямоугольников
3. Гаусса-3, Симпсона, Гаусса-5
4. Гаусса-2, Трапеция, Прямоугольников
5. Симпсона, Гаусса-4, Гаусса-3
6. Прямоугольников, Гаусса-5, Трапеция
7. Гаусса-4, Симпсона, Гаусса-2
8. Трапеция, Прямоугольников, Гаусса-3
9. Гаусса-5, Гаусса-2, Симпсона
10. Прямоугольников, Гаусса-3, Трапеция
11. Симпсона, Гаусса-4, Прямоугольников
12. Гаусса-2, Гаусса-5, Трапеция
13. Гаусса-3, Прямоугольников, Симпсона
14. Трапеция, Гаусса-2, Гаусса-4
15. Гаусса-5, Симпсона, Прямоугольников
16. Гаусса-4, Трапеция, Гаусса-3
17. Симпсона, Прямоугольников, Гаусса-5
18. Гаусса-2, Гаусса-3, Трапеция
19. Прямоугольников, Гаусса-4, Симпсона
20. Гаусса-5, Трапеция, Гаусса-2
21. Гаусса-3, Симпсона, Прямоугольников
22. Трапеция, Гаусса-5, Гаусса-4
23. Симпсона, Гаусса-2, Гаусса-3
24. Прямоугольников, Гаусса-2, Гаусса-5
25. Гаусса-4, Симпсона, Трапеция
26. Гаусса-3, Прямоугольников, Гаусса-5
27. Трапеция, Симпсона, Гаусса-2
28. Гаусса-5, Гаусса-4, Прямоугольников
29. Гаусса-2, Трапеция, Симпсона
30. Прямоугольников, Гаусса-3, Гаусса-4

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»