Disynergy

Страница материала ↗

EXQT
Готовые ответы — на EXQT
Доступ к ответам приобретается отдельно.
Найти ответы ↗

Методические рекомендации

Базовые понятия C#, структурное и ОО-программирование, UML и отладка

C# .NET Visual Studio VS Code UML

1. Теория

В этом документе собраны краткие методические рекомендации для начального курса по программированию на C#, охватывающие ключевые этапы жизненного цикла программного обеспечения, принципы структурного и объектно-ориентированного подходов, базовые средства языка и приёмы проектирования.

1.1. Этапы жизненного цикла программного обеспечения

Жизненный цикл ПО — это последовательность этапов, через которые проходит программный продукт от идеи до завершения эксплуатации.

Основные этапы

— **Анализ требований.** Выявление задач, потребностей пользователей, функциональных и нефункциональных требований. *Пример:* подготовка ТЗ для модуля «Учёт студентов» (ввод, поиск, сортировка).

— **Проектирование.** Определение архитектуры, модулей, интерфейсов, классов, структур данных. Применение UML-диаграмм (классов, прецедентов, последовательностей).

— **Реализация (кодирование).** Написание исходного кода на выбранном языке (C#), модульное тестирование.

— **Тестирование.** Проверка корректности и полноты реализации требований (юнит, интеграционные, системные тесты).

— **Внедрение и эксплуатация.** Передача пользователям, обучение, техподдержка.

> Совет: для учебных проектов фиксируйте артефакты каждого этапа — *ТЗ, диаграммы, README, тест-план, отчёт о внедрении*. Это облегчит проверку и дальнейшую поддержку.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер

disynergy.ru

— **Сопровождение и модернизация.** Исправления, добавление функций, оптимизация.

2. Принципы структурного программирования

Структурное программирование опирается на три базовые управляющие структуры: следование, ветвление и цикл. Код разбивается на логически завершённые функции и процедуры.

```
using System;

class Program
{
    static void Main()
    {
        int a = 10, b = 5;
        if (a > b)
        {
            Console.WriteLine("a больше b");
        }
        else
        {
            Console.WriteLine("a меньше или равно b");
        }

        for (int i = 1; i <= 5; i++)
        {
            Console.WriteLine($"Итерация {i}");
        }
    }
}
```

— используйте управляющие структуры вместо `goto`;

— соблюдайте модульность;

— пишите читаемо и предсказуемо.

3. Основы объектно-ориентированного программирования (ООП)

ООП рассматривает программу как совокупность взаимодействующих объектов.

Инкапсуляция

```
class Student
{
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
    private string name;
    public void SetName(string n) => name = n;
    public string GetName() => name;
}
```

Наследование

```
class Person
{
    public string Name { get; set; }
}

class Student : Person
{
    public string Group { get; set; }
}
```

Полиморфизм

```
class Shape
{
    public virtual void Draw() => Console.WriteLine("Фигура");
}

class Circle : Shape
{
    public override void Draw() => Console.WriteLine("Круг");
}

class Square : Shape
{
    public override void Draw() => Console.WriteLine("Квадрат");
}

class Program
{
    static void Main()
    {
        Shape s = new Circle();
        s.Draw();  // Выведет: Круг
    }
}
```

4. Типы данных, переменные и выражения

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

В C# каждый объект имеет тип, определяющий диапазон и допустимые операции.

Категории типов

Категория	Примеры	Диапазон/ значения
Целые числа	`int`, `long`, `byte`	от −2 147 483 648 до 2 147 483 647 (для `int`)
Вещественные	`float`, `double`, `decimal`	до 15–28 значащих цифр
Логические	`bool`	`true`, `false`
Символьные	`char`	один символ Unicode
Строковые	`string`	последовательность символов

```
int age = 20;
double price = 19.99;
bool isActive = true;
string name = "Иван";

Console.WriteLine($"{name}, возраст: {age}");

// Константы
const double Pi = 3.14159;
```

5. Управляющие конструкции: ветвления и циклы

Ветвления

```
if (x > 0)
    Console.WriteLine("Положительное");
else if (x < 0)
    Console.WriteLine("Отрицательное");
else
    Console.WriteLine("Ноль");
```

Switch

```
switch (day)
{
    case 1: Console.WriteLine("Понедельник"); break;
    case 2: Console.WriteLine("Вторник"); break;
    default: Console.WriteLine("День не найден"); break;
}
```

Циклы

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
for (int i = 0; i < 5; i++) Console.WriteLine(i);
while (count < 10) count++;
do { Console.WriteLine("Минимум одна итерация"); } while (false);
```

6. Функции и методы

Методы позволяют структурировать код и избегать дублирования.

```
static int Sum(int a, int b)
{
    return a + b;
}

static void Main()
{
    Console.WriteLine(Sum(3, 4)); // 7
}

// Передача по ссылке
static void AddOne(ref int x) { x++; }
```

7. Массивы и строки

```
int[] numbers = { 1, 2, 3, 4, 5 };
foreach (int n in numbers)
    Console.WriteLine(n);

string text = "Hello";
Console.WriteLine(text.Length);

// Многомерный массив
int[,] matrix = new int[2, 3] { { 1, 2, 3 }, { 4, 5, 6 } };
```

8. UML-диаграммы и проектирование программных модулей

UML помогает визуализировать структуру и поведение программных систем.

Диаграмма классов (упрощённый пример)

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
+------------------+
|     Student      |
+------------------+
| - name: string   |
| - group: string  |
+------------------+
| +GetInfo(): void |
+------------------+
```

Диаграмма последовательности (упрощённый пример)

```
Пользователь → Программа: Ввод данных
Программа → Модуль проверки: Проверка корректности
Модуль проверки → Программа: Результат
Программа → Пользователь: Вывод отчёта
```

9. Отладка и тестирование

В Visual Studio и VS Code используйте точки останова (breakpoints), окно *Watch* и шаги *Step Into/Over* для анализа выполнения.

```
using System.Diagnostics;
Debug.Assert(Sum(2, 3) == 5);
```

Типичные ошибки

— деление на ноль (`DivideByZeroException`);

— выход за границы массива;

— обращение к `null`.

10. Структуризация и повторное использование кода

— используйте пространства имён (`namespace`);

— делите проект на классы и файлы;

— подключайте библиотеки (`using System.IO;`).

```
namespace MyApp.Utils
{
    public static class MathHelper
    {
        public static double Average(int[] nums)
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
            => nums.Sum() / (double)nums.Length;
    }
}
```

2. Выполнение задания

Описание задания

1. Выбрать предметную область (из списка в документе).
2. Проанализировать объект моделирования: определить основные сущности и их характеристики (например: книга, студент, товар, клиент, заказ).
3. Разработать UML-диаграмму классов, отражающую структуру данных и связи между объектами.
4. Создать консольное приложение, реализующее базовые функции:
    - добавление нового объекта в коллекцию;
    - просмотр всех записей;
    - поиск по ключевому полю (например, ID, фамилия, название);
    - редактирование и удаление данных;
    - сохранение и загрузка данных из файла (текстового или JSON).
5. Реализовать в коде:
    - использование условных операторов (`if`, `switch`);
    - применение циклов (`for`, `while`, `foreach`);
    - написание функций и методов с параметрами и возвращаемыми значениями;
    - применение массивов, списков или коллекций;
    - создание классов и объектов, использование конструкторов, свойств, перегрузки методов.
6. Провести отладку и тестирование программы: проверить корректность ввода данных, устойчивость к ошибкам, корректность расчётов.
7. Подготовить отчёт с описанием этапов работы:
    - анализ предметной области;
    - UML-диаграмма классов;
    - фрагменты исходного кода;
    - результаты тестирования (скриншоты работы программы);
    - выводы.

Шаг 1. Анализ предметной области

Цель: определить, какие данные нужно хранить и как они связаны.

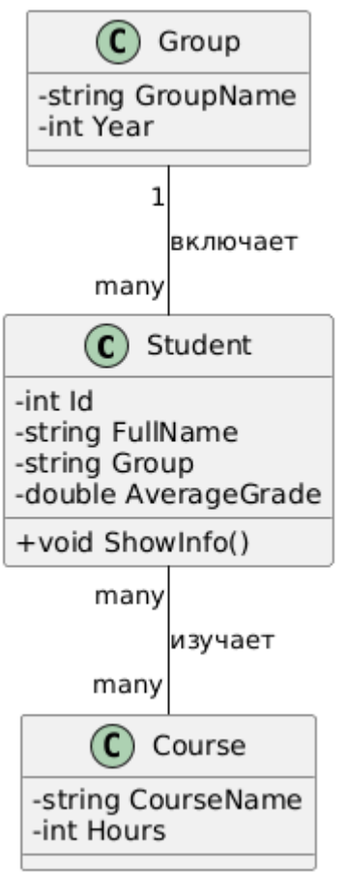
Объект моделирования: система учёта студентов.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Сущность	Атрибут	Тип данных	Пример
Student	ID	int	1
	FullName	string	Иванов Иван
	Group	string	ИС-23
	AverageGrade	double	4.5
Group	GroupName	string	ИС-23
	Year	int	2025
Course	CourseName	string	Программирование
	Hours	int	72

Шаг 2. UML-диаграмма классов

Диаграмма ниже рендерится автоматически из текста PlantUML (требуется интернет при просмотре документа).



Шаг 3. Создание проекта в Visual Studio

1. Откройте Visual Studio.
2. Выберите «Создать проект» → «Консольное приложение» (.NET 8 / .NET 6).
3. Назовите проект `StudentManager`.
4. Сохраните проект в папку `Учебная_практика_ПМ01`.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Шаг 4. Создание классов

Student.cs

```
public class Student
{
    public int Id { get; set; }
    public string FullName { get; set; }
    public string Group { get; set; }
    public double AverageGrade { get; set; }

    public void ShowInfo()
    {
        Console.WriteLine($"{Id}: {FullName}, группа {Group}, средний балл {Aver
    }
}
```

Group.cs

```
public class Group
{
    public string GroupName { get; set; }
    public int Year { get; set; }
}
```

Course.cs

```
public class Course
{
    public string CourseName { get; set; }
    public int Hours { get; set; }
}
```

Шаг 5. Основной модуль программы (Program.cs)

```
using System;
using System.Collections.Generic;
using System.Text.Json;
using System.IO;

class Program
{
    static List<Student> students = new List<Student>();
    static string filePath = "students.json";

    static void Main()
    {
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
        LoadData();

        bool exit = false;
        while (!exit)
        {
            Console.WriteLine("\n--- Меню управления студентами ---");
            Console.WriteLine("1. Добавить студента");
            Console.WriteLine("2. Показать всех студентов");
            Console.WriteLine("3. Найти по ID");
            Console.WriteLine("4. Редактировать данные");
            Console.WriteLine("5. Удалить студента");
            Console.WriteLine("6. Сохранить и выйти");
            Console.Write("Выбор: ");
            string choice = Console.ReadLine();

            switch (choice)
            {
                case "1": AddStudent(); break;
                case "2": ShowAll(); break;
                case "3": FindStudent(); break;
                case "4": EditStudent(); break;
                case "5": DeleteStudent(); break;
                case "6": SaveData(); exit = true; break;
                default: Console.WriteLine("Неверный выбор."); break;
            }
        }
    }
```

Шаг 6. Реализация функций

Добавление студента

```
static void AddStudent()
{
    Console.Write("Введите ID: ");
    int id = int.Parse(Console.ReadLine());
    Console.Write("Введите ФИО: ");
    string name = Console.ReadLine();
    Console.Write("Введите группу: ");
    string group = Console.ReadLine();
    Console.Write("Введите средний балл: ");
    double grade = double.Parse(Console.ReadLine());

    students.Add(new Student { Id = id, FullName = name, Group = group, AverageG
    Console.WriteLine("Студент добавлен.");
}
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Просмотр всех

```
static void ShowAll()
{
    Console.WriteLine("\n--- Список студентов ---");
    foreach (var s in students)
        s.ShowInfo();
}
```

Поиск

```
static void FindStudent()
{
    Console.Write("Введите ID студента: ");
    int id = int.Parse(Console.ReadLine());
    var s = students.Find(x => x.Id == id);
    if (s != null) s.ShowInfo();
    else Console.WriteLine("Студент не найден.");
}
```

Редактирование

```
static void EditStudent()
{
    Console.Write("Введите ID для редактирования: ");
    int id = int.Parse(Console.ReadLine());
    var s = students.Find(x => x.Id == id);
    if (s != null)
    {
        Console.Write("Новое имя: ");
        s.FullName = Console.ReadLine();
        Console.Write("Новая группа: ");
        s.Group = Console.ReadLine();
        Console.Write("Новый средний балл: ");
        s.AverageGrade = double.Parse(Console.ReadLine());
        Console.WriteLine("Данные обновлены.");
    }
    else Console.WriteLine("Студент не найден.");
}
```

Удаление

```
static void DeleteStudent()
{
    Console.Write("Введите ID для удаления: ");
    int id = int.Parse(Console.ReadLine());
    students.RemoveAll(s => s.Id == id);
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
    Console.WriteLine("Студент удалён.");
}
```

Шаг 7. Работа с JSON-файлом

JSON — лёгкий текстовый формат хранения данных, используется для обмена между программами. Данные хранятся как пары «ключ–значение».

Пример JSON-файла:

```
[
  {
    "Id": 1,
    "FullName": "Иванов Иван",
    "Group": "ИС-23",
    "AverageGrade": 4.5
  },
  {
    "Id": 2,
    "FullName": "Петров Пётр",
    "Group": "ИС-23",
    "AverageGrade": 3.9
  }
]
```

Создание JSON программно

```
static void SaveData()
{
    string json = JsonSerializer.Serialize(students, new JsonSerializerOptions {
    File.WriteAllText(filePath, json);
    Console.WriteLine("Данные сохранены в students.json");
}

static void LoadData()
{
    if (File.Exists(filePath))
    {
        string json = File.ReadAllText(filePath);
        students = JsonSerializer.Deserialize<List<Student>>(json);
        Console.WriteLine("Данные загружены из файла.");
    }
}
```

Справка:

- `Serialize` — превращает коллекцию объектов в текст JSON;
- `Deserialize` — восстанавливает объекты из файла;

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

- `WriteIndented = true` — форматирует JSON для удобства чтения.

После запуска программы файл `students.json` появится в папке проекта (`bin/Debug/net8.0/`).

Шаг 8. Тестирование и отладка

1. Запустите программу клавишами **Ctrl + F5**.
2. Добавьте 2–3 студентов.
3. Перезапустите программу — убедитесь, что данные загружаются автоматически.
4. Проверьте функции поиска, редактирования и удаления.
5. Откройте файл `students.json` и убедитесь, что он содержит актуальные данные.

Ссылка на тестовый проект: GitHub — YP.PM.01-testProject - https://github.com/OlgaKraven/YP.PM.01-testProject.git

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru