**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана»**

**Факультет <<Информатика и системы управления>>**
**Кафедра ИУ5 <<Системы обработать информации и управления>>**

# Курс «Парадигмы и конструкции языков программирования»

Очете по Домашнему заданию

Выполнил:

Студент группы иу5-31Б

Кашима Ахмед

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Москва 2023Г.

# Реферат по языку программирования C#

## Введение

C# - это объектно-ориентированный язык программирования, разработанный Microsoft. Он является частью платформы .NET и чрезвычайно популярен для создания веб-приложений, настольных приложений, игр и других программных продуктов. C# объединяет в себе простоту языка C++, гибкость языка Java и некоторые инновационные возможности.

## 1. Основные концепции C#

```
using System;


class Program
{
   static void Main()
   {
      // Объявление переменной
      int number = 10;


      // Вывод значения переменной на экран
      Console.WriteLine("Значение переменной: " + number);
   }
}
```

## 2. Управляющие структуры

```
using System;

class Program
{
   static void Main()
   {
      int number = 5;

      // Условная конструкция
      if (number > 0)
      {
         Console.WriteLine("Число положительное");
      }
      else if (number < 0)
      {
         Console.WriteLine("Число отрицательное");
      }
      else
      {
         Console.WriteLine("Число равно нулю");
      }

      // Цикл
      for (int i = 0; i < 5; i++)
      {
         Console.WriteLine("Итерация цикла: " + i);
      }
   }
}
```

## 3. Объектно-ориентированное программирование

```
using System;

// Определение класса
class Car
{
    // Поля класса
    public string model;
    public int year;

    // Метод класса
    public void DisplayInfo()
    {
        Console.WriteLine("Модель: " + model);
        Console.WriteLine("Год выпуска: " + year);
    }
}

class Program
{
    static void Main()
    {
        // Создание объекта класса Car
        Car myCar = new Car();
        myCar.model = "Toyota";
        myCar.year = 2022;

        // Вызов метода объекта
        myCar.DisplayInfo();
    }
}
```

## Создание проекта: Простой калькулятор

Давайте создадим небольшой проект на C# - простой калькулятор. Этот калькулятор будет выполнять основные арифметические операции: сложение, вычитание, умножение и деление.

## Код проекта

```
using System;

class Calculator
{
   static void Main()
   {
      Console.WriteLine("Простой калькулятор");

      Console.Write("Введите первое число: ");
      double num1 = Convert.ToDouble(Console.ReadLine());

      Console.Write("Введите оператор (+, -, *, /): ");
      char op = Convert.ToChar(Console.ReadLine());

      Console.Write("Введите второе число: ");
      double num2 = Convert.ToDouble(Console.ReadLine());

      double result = 0;

      // Выполнение арифметических операций
      switch (op)
      {
         case '+':
```

```
            result = num1 + num2;
            break;
        case '-':
            result = num1 - num2;
            break;
        case '*':
            result = num1 * num2;
            break;
        case '/':
            if (num2 != 0)
            {
                result = num1 / num2;
            }
            else
            {
                Console.WriteLine("Ошибка: деление на ноль!");
                return;
            }
            break;
        default:
            Console.WriteLine("Ошибка: неверный оператор!");
            return;
    }

    Console.WriteLine("Результат: " + result);
  }
}
```

## Описание проекта

Этот проект представляет собой консольное приложение, которое запрашивает у пользователя два числа и оператор, затем выполняет соответствующую арифметическую операцию и выводит результат на экран. Если введен некорректный оператор или попытка деления на

ноль, программа выдаст сообщение об ошибке.
Это всего лишь пример того, как можно использовать C# для создания простого приложения. Язык C# обладает множеством возможностей и библиотек для более сложных проектов.

Давайте добавим возможность работы с разными типами данных (целые числа, дробные числа) и поддержку дополнительных операций (возведение в степень, извлечение корня и т.д.). Также мы можем добавить обработку ошибок ввода пользователя для улучшения надежности программы.

```
using System;

class Calculator
{
   static void Main()
   {
      Console.WriteLine("Калькулятор");

      Console.Write("Введите первое число: ");
      double num1;
      if (!double.TryParse(Console.ReadLine(), out num1))
      {
         Console.WriteLine("Ошибка: введено некорректное число!");
         return;
      }

      Console.Write("Введите оператор (+, -, *, /, ^, √): ");
      char op = Convert.ToChar(Console.ReadLine());

      double num2 = 0;
      if (op != '^' && op != '√')
```

```
{
    Console.Write("Введите второе число: ");
    if (!double.TryParse(Console.ReadLine(), out num2))
    {
        Console.WriteLine("Ошибка: введено некорректное число!");
        return;
    }
}

double result = 0;

// Выполнение арифметических операций
switch (op)
{
    case '+':
        result = num1 + num2;
        break;
    case '-':
        result = num1 - num2;
        break;
    case '*':
        result = num1 * num2;
        break;
    case '/':
        if (num2 != 0)
        {
            result = num1 / num2;
        }
        else
        {
            Console.WriteLine("Ошибка: деление на ноль!");
            return;
```

```
            }
            break;
        case '^':
            result = Math.Pow(num1, num2);
            break;
        case '√':
            if (num1 >= 0)
            {
                result = Math.Sqrt(num1);
            }
            else
            {
                Console.WriteLine("Ошибка: извлечение корня из отрицательного числа!");
                return;
            }
            break;
        default:
            Console.WriteLine("Ошибка: неверный оператор!");
            return;
    }

    Console.WriteLine("Результат: " + result);
  }
}
```

## Описание улучшенного проекта

В этой версии калькулятора мы добавили проверку ввода пользователя для обработки некорректных значений. Теперь пользователь может вводить как целые, так и дробные числа. Также добавлены новые операции: возведение в степень (^) и извлечение квадратного корня (√). Если пользователь вводит отрицательное число для извлечения корня, программа выдаст сообщение об ошибке.

Этот улучшенный калькулятор демонстрирует, как можно расширить функционал простого приложения, добавляя новые операции и учитывая возможные ошибки пользователя.
Надеюсь, это дополнение к проекту было полезным! Если у вас есть дополнительные вопросы или что-то еще, в чем я могу помочь, пожалуйста, сообщите мне!

---

**давайте рассмотрим основные различия между языком программирования C# и другими языками, в том числе с языком C.**

## 1. Объектно-Ориентированный Язык:

Одно из ключевых отличий C# от языка C заключается в том, что C# является объектно-ориентированным языком программирования (ООП). Это означает, что в C# все данные и операции над ними организованы в объекты, что облегчает структурирование и повторное использование кода.

## 2. Управляемый Код и .NET Фреймворк:

C# является частью .NET фреймворка, который обеспечивает управляемый код (managed code) с автоматическим управлением памятью (сборка мусора). Это означает, что разработчику не нужно вручную управлять выделением и освобождением памяти, что делает код более безопасным и предотвращает утечки памяти.

## 3. Синтаксис и Удобства:

Синтаксис C# более выразителен и удобочитаем, чем у языка C. C# предоставляет множество удобных функций, таких как автоматические свойства, лямбда-выражения, LINQ (Language Integrated Query), что значительно упрощает разработку программ. В C# также присутствует обширная стандартная библиотека классов (FCL - Framework Class Library), которая предоставляет множество готовых классов и методов для различных задач.

## 4. События и Делегаты:

C# включает в себя механизмы событий и делегатов, которые облегчают реализацию паттерна "наблюдатель" (observer pattern) и обработку асинхронных событий. Эти концепции позволяют создавать более гибкие и расширяемые приложения.

## 5. Поддержка Платформы Windows:

C# часто используется для разработки приложений под платформу Windows и тесно интегрирован с Windows API. Это делает его предпочтительным выбором для разработки приложений под Windows.

## 6. Разработка Веб-Приложений:

C# используется в технологии ASP.NET для создания веб-приложений. ASP.NET предоставляет мощные средства для разработки динамических и масштабируемых веб-приложений.

## 7. Интеграция с Другими Языками:

C# является языком, совместимым с другими языками платформы .NET. Это означает, что разработчики могут использовать компоненты, написанные на разных языках, в одном приложении.

В заключение, C# предоставляет удобный и безопасный опыт разработки благодаря своим объектно-ориентированным возможностям, управляемому коду и богатым библиотекам. Эти особенности делают его одним из популярных языков программирования для разработки различных типов приложений на платформе Windows.

**Вывод:**

C# - это мощный и гибкий язык программирования, который предоставляет разработчикам широкие возможности для создания разнообразных приложений, начиная от настольных и веб-приложений до игр и мобильных приложений. Он отличается от языка C и других языков программирования своей объектно-ориентированной природой, управляемым кодом и богатыми возможностями стандартной библиотеки классов.

Благодаря интеграции с .NET фреймворком и поддержке платформы Windows, C# позволяет разработчикам создавать высокопроизводительные и надежные приложения. Его синтаксис удобочитаем и выразителен, что делает процесс разработки более эффективным и приятным.

События, делегаты и другие продвинутые концепции языка позволяют создавать сложные приложения и обеспечивают легкость в разработке асинхронных и многопоточных приложений.

В целом, C# является важным инструментом для разработчиков, работающих на платформе Windows, и предоставляет им возможность создавать современные и эффективные программы.