Класс “Полиномы”

Класс "Полиномы" предоставляет методы для выполнения основных операций над полиномами, таких как сложение, вычитание, умножение и деление. Он также позволяет вычислять значения полинома для заданных значений переменной.

Операция сложения полиномов

Операция сложения полиномов осуществляется поэлементно. Для сложения полиномов, необходимо сложить коэффициенты соответствующих членов полиномов. Если один полином имеет более высокую степень, то оставшиеся члены с более высокими степенями просто дописываются к результату.

В приведенном коде на C# представлен оператор сложения двух полиномов. Он создает новый полином, определяет его степень как максимальную степень среди полиномов a и b. Затем он проходит по всем коэффициентам от 0 до минимальной степени среди полиномов и складывает соответствующие коэффициенты. После этого, если степень одного полинома больше другого, он дописывает оставшиеся коэффициенты в новый полином.

Таким образом, операция сложения полиномов позволяет объединять их члены и получать новый полином, который представляет собой сумму исходных полиномов.

```
public static Polinom operator +(Polinom a, Polinom b)
{
```

```
      Polinom p;
      int n, m;
      n= Math.Max(a.n, b.n);
      m = Math.Min(a.n, b.n);
      p = new Polinom(n);

      for (int i = 0; i <= m; i++) //сложение коэффициентов
      {
           p.koef[i] = a.koef[i] + b.koef[i];
      }
      for (int i = m+1;i <= n; i++) //дописывание старших коэффициентов
      {
           p.koef[i]=(a.n>=b.n) ? a.koef[i] : b.koef[i];
      }

      return p;
}
```

Операция вычитания полиномов

Операция вычитания полиномов осуществляется поэлементно. Для вычитания полиномов, необходимо вычесть коэффициенты соответствующих членов полиномов. Если один полином имеет более высокую степень, то оставшиеся члены с более высокими степенями просто копируются с противоположным знаком в результат.

В приведенном коде на C# представлен оператор вычитания двух полиномов. Он создает новый полином, определяет его степень как максимальную степень среди полиномов a и b. Затем он проходит по всем коэффициентам от 0 до минимальной степени среди полиномов и вычитает соответствующие коэффициенты. После этого, если степень одного полинома больше другого, он копирует оставшиеся коэффициенты с противоположным знаком в новый полином.

Таким образом, операция вычитания полиномов позволяет вычитать их члены и получать новый полином, который представляет собой разность исходных полиномов.

```
public static Polinom operator +(Polinom a, Polinom b)
{
      Polinom p;
      int n, m;
      n= Math.Max(a.n, b.n);
      m = Math.Min(a.n, b.n);
      p = new Polinom(n);
```

```
      for (int i = 0; i <= m; i++) //сложение коэффициентов
      {
           p.koef[i] = a.koef[i] - b.koef[i];
      }
      for (int i = m+1;i <= n; i++) //дописывание старших коэффициентов
      {
           p.koef[i]=(a.n>=b.n) ? a.koef[i] : -b.koef[i];
      }

      return p;
}
```

Операция умножения полиномов

Операция умножения полиномов выполняется путем перемножения коэффициентов соответствующих членов полиномов и суммирования их результатов. Для умножения полиномов a и b, создается новый полином p, у которого степень равна сумме степеней полиномов a и b.

В приведенном коде на C# представлен оператор умножения двух полиномов. Он создает новый полином p с размерностью, равной сумме степеней a и b. Затем происходит двойной цикл для перемножения всех членов полиномов a и b. Внутренний цикл перемножает коэффициенты соответствующих членов полиномов и суммирует результаты в соответствующие коэффициенты полинома p.

Таким образом, операция умножения полиномов позволяет получить новый полином, который представляет собой результат перемножения исходных полиномов.

```
public static Polinom operator *(Polinom a, Polinom b)
{
    Polinom p;
    int n =a.n;
    int m = b.n;
    int k = 0;
    p = new Polinom(n+m);

    for (int i = 0; i <= n + m; i++)
    {
        for (int j = 0; j <= Math.Min(i,n); j++)
        {
            k = i - j;
            if (k <= m)
            {
                p.koef[i] += a.koef[j] * b.koef[k];
            }
```

```
        }
    }

    return p;
}
```

Операция деления полиномов

Операция деления полиномов позволяет получить результат деления одного полинома на другой. Результатом деления является новый полином, который представляет собой частное от деления исходных полиномов.

В приведенном коде на C# представлен оператор деления двух полиномов. Он создает новый полином p и временный полином t, копирует коэффициенты полинома a в полином t. Затем происходит цикл деления, где каждый раз из полинома t вычитается произведение коэффициента полинома b и частного от деления коэффициента полинома a на коэффициент полинома b. Результатом деления является полином p.

Таким образом, операция деления полиномов позволяет получить новый полином, который представляет собой результат деления исходных полиномов.

```
public static Polinom operator /(Polinom a, Polinom b)
{
    Polinom p,t;
    int n = a.n;
    int m = b.n;
    if (n<m)
    {
        return new Polinom(0);
    }
    double d = 0;
    p = new Polinom(n-m);
    t = new Polinom(a.koef);
    for (int i = 0; i <= n - m; i++)
    {
        d = t.koef[n-i]/b.koef[m];
        p.koef[n-m-i] = d;
        t.koef[n - i] = 0;
        for (int j = 0; j <= m; j++)
        {
            t.koef[n-i-j] = t.koef[n-i-j] - d*b.koef[m-j];
        }
    }
    return p;
}
```

Операция нахождения остатка от деления

Тут пол кода — деление, а другая половина представляет собой алгоритм для поиска остатка

```
public static Polinom operator %(Polinom a, Polinom b)
{
    Polinom p, t;
    int n = a.n;
    int m = b.n;
    if (n < m)
    {
        return new Polinom(0);
    }
    double d = 0;
    p = new Polinom(n - m);
    t = new Polinom(a.koef);
    for (int i = 0; i <= n - m; i++)
    {
        d = t.koef[n - i] / b.koef[m];
        p.koef[n - m - i] = d;
        t.koef[n - i] = 0;
        for (int j = 0; j <= m; j++)
        {
            t.koef[n - i - j] = t.koef[n - i - j] - d * b.koef[m - j];
        }
    }
    int k = 0;
    double[] af;
    while (k <= n && t.koef[n - k] == 0)
    {
        k++;
    }
    if (k <= n)
    {
        af = new double[n - k + 1];
        for (int i = 0; i <= n; i++)
        {
            af[i] = t.koef[i];
        }
    }
    else
    {
        af = new double[1];
    }

    return new Polinom(af);
}
```