УДК 004.43

# ОСОБЕННОСТИ РЕАЛИЗАЦИИ ПЕРЕГРУЗКИ ОПЕРАТОРНЫХ ФУНКЦИЙ ДЛЯ АРИФМЕТИЧЕСКИХ, ЛОГИЧЕСКИХ ОПЕРАЦИЙ И СРАВНЕНИЙ: C# VS C++

*Голубь Надежда*

Национальный аэрокосмический университет имени Н.Е. Жуковского «ХАИ»

## Аннотация

*Статья посвящена особенностям реализации операторных функций в популярных языках объектно-ориентированного программирования C# и C++. Данная статья показывает на примере разницу в реализации перегрузки арифметических, логических операторов и операций сравнения с использованием этих двух популярных языков.*

## Abstract

*The article is devoted to peculiarities of the implementation of operator functions in popular languages of object-oriented programming C# and C++. This article shows by example the difference in the implementation of overloading arithmetic, logical operators and comparison operations using the two popular languages.*

## Введение

В языке C++ есть удобное средство объектно-ориентированного программирования – перегрузка операторов, реализующая специальный полиморфизм. Перегрузка большинства арифметических или логических операторов может быть сделана как с помощью member-, так и с помощью friend-функций. Однако отдельные операторы могут быть перегружены только через member-функции (например, ++ или --), а другие – с помощью только friend-функций (например, операции потокового ввода/вывода “<< или >>”) [1]. Язык C# тоже поддерживает перегрузку операторов, но немного по-другому.

Перегрузка процедур и функций на уровне общей идеи, как правило, не представляет сложности ни в реализации, ни в понимании. Однако даже в ней имеются некоторые «подводные камни», которые необходимо учитывать при реализации перегрузки операторов с использованием разных языков программирования [2].

Данная статья на примере показывает разницу в реализации перегрузки арифметических, логических операторов и операций сравнения с использованием этих двух популярных языков.

## Постановка задачи

В языке C# существуют только две формы операторных методов с явными операндами (operator): одна — для унарных операторов, другая для бинарных.

```
// общая форма перегрузки унарного оператора
public static возвращаемый тип operator op (тип_параметра операнд)
{ // операции }
// Общая форма перегрузки бинарного оператора
public static возвращаемый_тип operator op (тип_параметра1 операнд1,
                                             тип_параметра2 операнд2)
{ // операции }
```

Поэтому в качестве базового примера для исследования был взят и дополнен пример на языке C# [3], который затем был переделан в пример на языке C++. Это позволило увидеть разницу в реализации, а также некоторые трудности и «подводные камни».

## Изложение основного материала

Описание фрагмента класса с перегружаемыми операторами (язык C#).

```
class MyArr
   {
      // Координаты точки в трехмерном пространстве
      public double x, y, z;
      public MyArr(double x = 0, double y = 0, double z = 0)
      {
         this.x = x;
         this.y = y;
         this.z = z;
      }
      //==== Перегружаем бинарные операторы
      public static MyArr operator + (MyArr obj1, MyArr obj2)
      {
         MyArr arr = new MyArr();
         arr.x = obj1.x + obj2.x;
         arr.y = obj1.y + obj2.y;
         arr.z = obj1.z + obj2.z;
         return arr;
      }
     // по аналогии реализуются другие арифметические операции - * /
      //==== Перегружаем унарные операторы
      public static MyArr operator - (MyArr obj1)
      {
         MyArr arr = new MyArr();
         arr.x = -obj1.x;
         arr.y = -obj1.y;
         arr.z = -obj1.z;
         return arr;
      }
      public static MyArr operator ++ (MyArr obj1) // префиксный и постфиксный
      {
         obj1.x += 1;
         obj1.y += 1;
         obj1.z += 1;
         return obj1;
      }
      // по аналогии реализуется операция --
//==== перегружаем бинарный + для ДРУГИХ аргументов (в C++ аналога нет)
      public static string operator + (MyArr obj1, string s)
      {
         return s + " " + obj1.x + " " + obj1.y + " " + obj1.z;
      }
      //==== Перегружаем оператор сравнения ==
      public static bool operator == (MyArr obj1, MyArr obj2)
      {
         if ((obj1.x == obj2.x) && (obj1.y == obj2.y) && (obj1.z == obj2.z))
            return true;
         return false;
      }
      //==== Теперь обязательно нужно перегрузить его пару - оператор !=
      public static bool operator != (MyArr obj1, MyArr obj2)
```

```
        {
          if ((obj1.x != obj2.x) || (obj1.y != obj2.y) || (obj1.z != obj2.z))
            return true;
          return false;
        }
      }
```

Из анализа этого кода совершенно четко видно, что переопределение арифметических операций на языке C++ (как бинарные, так и частично унарные) можно без проблем реализовать, если объявить их как friend-функции. То же относится к логическим операциям и операциям сравнения. Причем последние не обязательно должны быть парными. При этом координаты точки могут стать закрытыми (private). А вот унарные операторы ++ и – в C++ могут быть только member-функциями. Причем исходные коды постфиксного и префиксного операторов будут разными [1] в отличие от C#.

```
// Перегружаем унарный ПРЕФИКСНЫЙ оператор ++
    MyArr& operator ++()
    {
        x++;
        y++;
        z++;
        return *this;
    }
```

Понадобится ещё и перегружаемая операция вывода координат точек

```
      friend ostream &operator<< (ostream &output, const MyArr &a)
      {
            output << a.x << "\t" << a.y << "\t" << a.z << "\n";
            return output;        // теперь можно выводить так:  cout  << a << b;
      }
```

## Выводы

- Операторные функции бинарных арифметических, логических операций и операций сравнения в языке C# реализуются в одной строго определенной форме. Операции сравнения при этом обязательно должны иметь описание своей пары.
- Операторные функции бинарных арифметических, логических операций и операций отношения в языке C++ могут быть реализованы в двух формах: member- или friend-функции. Операции сравнения при этом не обязательно должны иметь описание своей пары, если нет её явного вызова.
- Операторные функции унарных арифметических операций в языке C++ имеют свои особенности. Например, постфиксный и префиксный инкремент или декремент описываются по-разному.
- В C++ операции сдвига << или >> могут иметь двоякий смысл: побитовая операция сдвига или переопределение потокового ввода-вывода для объекта пользователя. В C# - это всегда бинарная побитовая операция сдвига.

## Список использованных источников:

1. Шилдт Г. Самоучитель C++. 3-е изд. - СПб.: BHV-Санкт-Петербург, 1998. – 688 с. – ISBN 5-7791-0086-1.

2. Перегрузка операторов [Электронный ресурс]. - Режим доступа : www. URL: https://ru.wikipedia.org/wiki/Перегрузка операторов - 18 августа 2016.

3. C#. Руководство по C#. Основы перегрузки операторов [Электронный ресурс]. - Режим доступа : www. URL: http://professorweb.ru/my/csharp/charp_theory/level6/6_4.php - 2016.