# Лабораторная работа №8

## Перегрузка операций

### Вариант 1

Цель работы: приобретение практических навыков реализации операций в классах на языке C#.

### Задание к работе

В класс Point из лабораторной работы №6 добавить перегрузку:

* операции ++(−−): одновременно увеличивает (уменьшает) значение полей X и Y на 1;
* констант true и false: обращение к экземпляру класса дает значение true, если значение полей X и Y совпадает, иначе false;
* операции бинарный +: одновременно добавляет к полям X и Y значение скаляра;
* преобразования типа Point в string (и наоборот).

# Решение

## Описание класса

Для хранения точки в классе Point используется два поля: x и y.

Для инициализации экземпляра класса используются два конструктора: первый, когда нужно инициализировать точку нулевыми координатами, второй — заданными.

### Краткое описание методов, свойств и перегрузок класса

1. Print() — общедоступный метод для вывода координат точки.
2. Vector() — общедоступный метод, возвращающий расстояние от начала координат до точки.
3. Transition(int a, int b) — общедоступный метод, перемещающий точку на плоскости на вектор (a, b).
4. X — общедоступное свойство, устанавливающее или отдающее координату X.
5. Y — общедоступное свойство, устанавливающее или отдающее координату Y.
6. Multiplication — общедоступное свойство, доступное только для записи, умножающее координаты на скаляр.
7. **IsEqual(Point value)** — приватный метод, сравнивающий значение полей X и Y.
8. **Add(Point value)** — приватная метод, возвращающий точку с координатами, увеличенными на заданный скаляр.
9. **++** — перегруженный оператор, при вызове инкрементирующий одновременно поля X и Y.
10. **−−** — перегруженный оператор, при вызове декрементирующий одновременно поля X и Y.
11. **true** — перегрузка; при равенстве полей X и Y.
12. **false** — перегрузка; при неравенстве полей X и Y.
13. **+** — перегруженный бинарный оператор (в двух вариациях, для операций в разном порядке), возвращает точку с координатами увеличенными, на заданный скаляр.
14. **string** — перегрузка; приводит экземпляр класса Point к строковому представлению (используя формат «(x; y)»)
15. **Point** — перегрузка; переводит строку в представление в виде экземпляра класса Point.

## Результат работы программы

|  |
| --- |
| Сообщение на экране |
| Новая точка: (3; 2)  После инкремента: (4; 3)  После декремента: (3; 2)  Координаты + 2: (5; 4)  3 + координаты: (8; 7)  X = Y? Нет.  Неявное преобразование:  Точка: (8; 7)  Явное преобразование ("1 2" в Point):  Точка: (1; 2) |

## Текст программы

|  |  |
| --- | --- |
| 001 | **using** System; |
| 002 |  |
| 003 | **namespace** OOP\_08 |
| 004 | { |
| 005 | **class** Program |
| 006 | { |
| 007 | **static void** Main(**string**[] args) |
| 008 | { |
| 009 | Point point = **new** Point(3, 2); |
| 010 |  |
| 011 | Console.Write(*"Новая точка: "*); |
| 012 | point.Print(); |
| 013 |  |
| 014 | point++; |
| 015 | Console.Write(*"После инкремента: "*); |
| 016 | point.Print(); |
| 017 |  |
| 018 | point--; |
| 019 | Console.Write(*"После декремента: "*); |
| 020 | point.Print(); |
| 021 |  |
| 022 | point = point + 2; |
| 023 | Console.Write(*"Координаты + 2: "*); |
| 024 | point.Print(); |
| 025 |  |
| 026 | point = 3 + point; |
| 027 | Console.Write(*"3 + координаты: "*); |
| 028 | point.Print(); |
| 029 |  |
| 030 | Console.Write(*"X = Y? "*); |
| 031 | **if** (point) |
| 032 | { |
| 033 | Console.WriteLine(*"Да."*); |
| 034 | } |
| 035 | **else** |
| 036 | { |
| 037 | Console.WriteLine(*"Нет."*); |
| 038 | } |
| 039 |  |
| 040 | Console.WriteLine(*"Неявное преобразование: "*); |
| 041 | **string** s = *" Точка: "* + point; |
| 042 | Console.WriteLine(s); |
| 043 |  |
| 044 | Console.WriteLine(*"Явное преобразование (\"1 2\" в Point): "*); |
| 045 | Console.Write(*" Точка: "*); |
| 046 | s = *"1 2"*; |
| 047 | point = (Point)s; |
| 048 | point.Print(); |
| 049 |  |
| 050 | Console.ReadKey(); |
| 051 | } |
| 052 | } |
| 053 |  |
| 054 | **public** **class** Point |
| 055 | { |
| 056 | **private** **int** x; |
| 057 | **private** **int** y; |
| 058 |  |
| 059 | // Конструктор класса, задающий нулевые значения |
| 060 | **public** Point() |
| 061 | { |
| 062 | x = 0; |
| 063 | y = 0; |
| 064 | } |
| 065 |  |
| 066 | // Конструктор класса, с заданными координатами |
| 067 | **public** Point(**int** a, **int** b) |
| 068 | { |
| 069 | x = a; |
| 070 | y = b; |
| 071 | } |
| 072 |  |
| 073 | // Выводит координаты точки на экран |
| 074 | **public** **void** Print() |
| 075 | { |
| 076 | Console.WriteLine(*"({0}; {1})"*, x, y); |
| 077 | } |
| 078 |  |
| 079 | // Находит расстояние от начала координат до точки |
| 080 | **public** **double** Vector() |
| 081 | { |
| 082 | **return** Math.Sqrt(x \* x + y \* y); |
| 083 | } |
| 084 |  |
| 085 | // Перемещает точку на плоскости на вектор (a, b) |
| 086 | **public** **void** Transition(**int** a, **int** b) |
| 087 | { |
| 088 | x += a; |
| 089 | y += b; |
| 090 | } |
| 091 |  |
| 092 | // Позволяет задать или получить координату X |
| 093 | **public** **int** X |
| 094 | { |
| 095 | get { **return** x; } |
| 096 | set { x = **value**; } |
| 097 | } |
| 098 |  |
| 099 | // Позволяет задать или получить координату Y |
| 100 | **public** **int** Y |
| 101 | { |
| 102 | **get** { **return** y; } |
| 103 | **set** { y = **value**; } |
| 104 | } |
| 105 |  |
| 106 | // Позволяет умножить координаты на скаляр (только для записи) |
| 107 | **public** **int** Multiplication |
| 108 | { |
| 109 | **set** |
| 110 | { |
| 111 | x \*= **value**; |
| 112 | y \*= **value**; |
| 113 | } |
| 114 | } |
| 115 |  |
| 116 | // Сравнивает поля X и Y |
| 117 | **private** **static** **bool** IsEqual(Point value) |
| 118 | { |
| 119 | **return** value.x == value.y; |
| 120 | } |
| 121 |  |
| 122 | // Возвращает точку сдвинутую на вектор rhs |
| 123 | **private** **static** Point Add(Point lhs, **int** rhs) |
| 124 | { |
| 125 | **return** **new** Point(lhs.x + rhs, lhs.y + rhs); |
| 126 | } |
| 127 |  |
| 128 | **public** **static** Point **operator** ++(Point value) |
| 129 | { |
| 130 | value.Transition(1, 1); |
| 131 | **return** value; |
| 132 | } |
| 133 |  |
| 134 | **public** **static** Point **operator** --(Point value) |
| 135 | { |
| 136 | value.Transition(-1, -1); |
| 137 | **return** value; |
| 138 | } |
| 139 |  |
| 140 | **public** **static** **bool** **operator** **true**(Point value) |
| 141 | { |
| 142 | **return** IsEqual(value); |
| 143 | } |
| 144 |  |
| 145 | **public** **static** **bool** **operator** **false**(Point value) |
| 146 | { |
| 147 | **return** !IsEqual(value); |
| 148 | } |
| 149 |  |
| 150 | **public** **static** Point **operator** +(Point lhs, **int** rhs) |
| 151 | { |
| 152 | **return** Add(lhs, rhs); |
| 153 | } |
| 154 |  |
| 155 | **public** **static** Point **operator** +(**int** lhs, Point rhs) |
| 156 | { |
| 157 | **return** Add(rhs, lhs); |
| 158 | } |
| 159 |  |
| 160 | **public** **static** **implicit** **operator** **string**(Point Point) |
| 161 | { |
| 162 | **return** String.Format(*"({0}; {1})"*, Point.x.ToString(),   Point.y.ToString()); |
| 163 | } |
| 164 |  |
| 165 | **public** **static** **explicit** **operator** Point(**string** s) |
| 166 | { |
| 167 | **string**[] splitted = s.Split(' '); |
| 168 | **return** **new** Point(Convert.ToInt32(splitted[0]),   Convert.ToInt32(splitted[1])); |
| 169 | } |
| 170 | } |
| 171 | } |