https://doi.org/10.26160/2474-5901-2024-41-83-87

ПРОИЗВОДИТЕЛЬНОСТЬ ЛЯМБДА-ВЫРАЖЕНИЙ В ЯЗЫКАХ ПРОГРАММИРОВАНИЯ ВЫСОКОГО УРОВНЯ

Корниенко О.В.
Донской государственный технический университет, Ростов-на-Дону, Россия

Ключевые слова: лямбда-выражения, C#, Java, Python, фильтрация, анонимные функции, сравнение производительности.
Аннотация. Функциональное программирование — это парадигма программирования, которая становится все более популярной среди разработчиков программного обеспечения. Отчасти это связано с развитием распределенных систем и потребностью в более надежном и масштабируемом коде. В статье представлен обзор синтаксиса и возможностей лямбда-выражений в трех языках программирования – C#, Java и Python.

PERFORMANCE OF LAMBDA EXPRESSIONS IN HIGH LEVEL PROGRAMMING LANGUAGES

Kornienko O.V.
Don State Technical University, Rostov-on-Don, Russia

Keywords: lambda expressions, C#, Java, Python, filtering, anonymous functions, performance comparison.
Abstract. Functional programming is a programming paradigm that is becoming increasingly popular among software developers. This is partly due to the development of distributed systems and the need for more reliable and scalable code. This article presents an overview of the syntax and features of lambda expressions in three programming languages – C#, Java and Python.

Функциональное программирование является ключевой частью современных языков программирования. Оно основано на математических функциях, условных выражениях и рекурсии. Одной из важных особенностей функционального программирования является использование лямбда-выражений, которые представляют собой краткую и гибкую альтернативу функциям [1].

Лямбда-выражения сравнительно недавно появились в распространенных языках программирования и способны упростить и оптимизировать написание и использование функций. Они представляют собой анонимные функции, которые используются в качестве аргументов других функций или значений переменных.

В Java 8 появилась поддержка лямбда-выражений, которые позволяют писать функции в виде кратких анонимных функций, которые можно передавать в качестве аргументов другим функциям или хранить в переменных [2]. Лямбда-выражения в Java записываются с помощью синтаксиса, показанного на рисунке 1, где **argumets** – это список параметров, разделенных запятыми, а **expression** – это тело функции.

```
(arguments) -> expression
```

Рис. 1. Синтаксис лямбда-выражений на Java

Лямбда-выражения в Java можно, например, использовать в качестве краткой альтернативы классу, который определяется и инстанцируется в одной строке кода, без имени. Рассмотрим анонимный внутренний класс на рисунке 2.

```
new Thread(new Runnable() {
public void run() {
System.out.println("Hello from the thread");
}
}).start();
```

Рис. 2. Анонимный внутренний класс на Java

Этот код можно переписать с помощью лямбда-выражения, как показано на рисунке 3.

```
new Thread(() -> System.out.println("Hello from the
thread")).start
```

Рис. 3. Переписанный при помощи лямбда-выражения код на Java

В C# 3.0 появилась поддержка лямбда-выражений, которые по синтаксису и функциональности похожи на выражения в Java [3]. Лямбда-выражения в C# имеют синтаксис, указанный на рисунке 4.

```
(arguments) => expression
```

Рис. 4. Синтаксис лямбда-выражений на C#

Делегаты – это безопасный для типов объектно-ориентированный механизм вызова методов, который широко используется в C#. Лямбда-выражения могут использоваться в качестве аргументов делегатов, обеспечивая гибкий способ определения и использования функций. На рисунке 5 показан делегат типа **Func**, который принимает два целочисленных аргумента и возвращает целое число.

```
delegate int Func(int a, int b);

Func sum = (a, b) => a + b;
```

Рис. 5. Использование лямбда-выражения в качестве аргумента делегата

С самых ранних версий Python поддерживает лямбда-выражения, которые записываются с помощью синтаксиса, указанного на рисунке 6 [4].

```
lambda arguments: expression
```

Рис. 6. Синтаксис лямбда-выражений на Python

Один из основных вариантов использования лямбда-выражений в Python – это компактный способ задать функцию для функции более высокого порядка. Одной из таких функций является функция **map** в Python. Она

преобразует коллекцию в другую коллекцию путем применения функции ко всем элементам, как показано на рисунке 7.

```
numbers = [1, 2, 3, 4, 5]

squaredNumbers = list(map(lambda x: x**2, numbers))
```

Рис. 7. Пример использования функции **map** на Python

Функция **map** применяется к списку чисел, а лямбда-выражение **lambda x: x**2** используется для указания функции, которая применяется к каждому числу. В результате получается список квадратов чисел.

Проведенные тесты были направлены на расчет времени выполнения некоторых заданных функций с использованием и без использования лямбда-функций на трех рассматриваемых языках программирования. Тестовая среда состоит из процессора Intel Core CPU i7-11700, 8 ГБ DDR4 и операционной системы Windows 11 Pro. Используются следующие версии языков программирования: C# 11.0 с .NET 7.0, Java SE 19, Python 3.11.2.

Все тесты применяются к списку объектов класса Employee, который содержит три поля: *номер*, *имя*, *зарплата*. Перед тем как представить тесты, необходимо сделать следующее замечание. Все тесты выполняются на списке из 200000 объектов со случайно сгенерированными значениями для поля зарплата. Для измерения времени выполнения используются методы класса Stopwatch на C# и его аналоги на Java и Python. – Все тесты выполняются по сто раз и для более точных выводов берется среднее время всех выполнений.

На рисунке 8 продемонстрирован первый тест, включающий сортировку списка объектов Employee по зарплате с помощью лямбда-выражений.

```
C#
employees.Sort((o1, o2) =>
o1.getSalary().CompareTo(o2.getSalary()));

Java
employees.sort((o1, o2) -> Double.compare(o1.getSalary(),
o2.getSalary()));

Python
emp_list = sorted(emp_list, key=lambda t: t.getSalary())
```

Рис. 8. Первый тест

На рисунке 9 продемонстрирован второй тест, в котором применяется стандартная функциональность сортировки с помощью метода CompareTo или его альтернатив.

```
C#
employees.Sort();

Java
Collections.sort(employees);

Python
emp_list = sorted(emp_list)
```

Рис. 9. Второй тест

На рисунке 10 продемонстрирован третий тест, в котором проводится исследование возможностей фильтрации функций лямбда. Он измеряет время фильтрации списка и извлечения только тех объектов, у которых значение поля *зарплата* больше 50000.

```
C#
empFiltLst = employees.Where(employee => employee.getSalary() > 50000);

Java
Stream <Employee> empStm = employees.stream().filter(employee -> employee.getSalary() > 50000);

Python
filteredResults = filter(lambda x: x.getSalary() > 50000, emp_list)
```

Рис. 10. Третий тест

На рисунке 11 продемонстрирован финальный тест фильтрации, но расширенный за счет трудоемкой операции преобразования отфильтрованного результата в список и получения количества отфильтрованных элементов. Результаты всех испытаний представлены в таблице 1.

```
C#
filteredResultsCnt = empFiltLst.Count();

Java
filteredResultsCnt = empStm.collect(Collectors.toList()).size();

Python
filteredResultsCnt = len(list(filteredResults))
```

Рис. 11. Финальный тест

Табл. 1. Результаты всех испытаний

Тест	C#	JAVA	Python
Сортировка с помощью лямбды-выражений	43,32 мс	54,36 мс	68,04 мс
Сортировка с помощью CompareTo	41,52 мс	52,84 мс	67,23 мс
Фильтр без подсчета	0,01 мс	0,01 мс	0,01 мс
Фильтр со счетчиком	1,42 мс	1,98 мс	31,14 мс

Из полученных результатов можно сделать следующие выводы. Использование лямбда-выражений для сортировки быстрее всего в C# и медленнее всего в Python. Лямбда-выражения имеют такую же или даже меньшую производительность при сортировке, чем в случае использования CompareTo.

Список литературы

1. Слоннегер К., Курц Б. Формальный синтаксис и семантика языков программирования: лабораторный подход. – Аддисон-Уэсли Лонгман, 1995. – 139 с.
2. Лямбда-выражения в Java 8 [Электронный ресурс]. – Режим доступа: https://clck.ru/3AJVca.
3. Узаир С. Освоение C#: Первые шаги. – CRC Press, 2022. – 152 с.
4. Лямбда-функция в Python простыми словами [Электронный ресурс]. – Режим доступа: https://clck.ru/3AJW36.

References

1. Slonneger K., Kurtz B. Formal syntax and semantics of programming languages: a laboratory approach. – Addison-Wesley Longman, 1995. – 139 p.
2. Lambda-expressions in Java 8 [Electronic resource]. – Access mode: https://clck.ru/3AJVca.
3. Uzair S. Mastering C#: First Steps. – CRC Press, 2022. – 152 p.
4. Lambda function in Python in simple words [Electronic resource]. – Access mode: https://clck.ru/3AJW36.

Корниенко Олег Вячеславович – студент	**Kornienko Oleg Vyacheslavovich** – student
oleg.korn1enko@yandex.com	

Received 24.04.2024