**Lambda Expression Analysis**

Daiki Itoh  
 Computer Science  
 Arkansas State University  
 Jonesboro Arkansas U.S  
 daiki.itoh@smail.astate.edu

Lloyd Thomas  
 Computer Science  
 Arkansas State University  
 Jonesboro Arkansas U.S lloyd.thomas@smail.astate.edu

Zhiquan Tang  
 Computer Science  
 Arkansas State University  
 Jonesboro Arkansas U.S  
 zhiquan.tang@smail.astate.edu

**ABSTRACT**

As a senior student of computer science degree, one should have the capacity for some advanced level coding skills and knowledge. Lambda expressions are a kind of anonymous function which can shorten the lines of code in a program and are powerful to replace some regular functions.

In order to fully understand lambda expressions, our group will focus on the usage of lambda expressions in different aspects. The usage rate of lambda expressions, the different syntax and mechanisms of lambda expressions in different languages, and the best use cases for lambda expressions. We implemented a project using Python, Java and MySQL, which could scan a open-source project and count the usage of lambda expressions. Additionally, we read other research papers discussing lambda expressions.

According to the data we got from our project database, many people use lambdas in their code and usage appears to be increasing. It is better for us to use lambda expressions in simple and short functions. We investigated three different programming languages, C++, Python and Java. C++ would be the most complicated to use lambda expressions, but we believed most programmers can handle lambdas in C++ after completing the computer science course.

In conclusion, we considered that lambda expressions are useful, lambda expressions will be more and more popular in the programming community as time goes by. It will be a beneficial skill to learn and understand lambda expressions in undergraduate learning.

**KEYWORDS**

Lambda expression, Usage, Programming language

**1 INTRODUCTION**

Many programming languages provide support for defining local unnamed functions, as known as anonymous functions. Lambda expressions are a kind of anonymous function. In some programming languages, lambda expressions are the only anonymous function supported, and some languages cannot support lambda expressions directly, but through the use of a specific library to recreate the functionality of a lambda expression. Many sources and references use lambda functions to describe this kind of anonymous function, this term will be used interchangeably with the term “lambda expression”.

The first job that most undergraduate students can find are related to programming. None of the undergraduate level programming textbooks in Arkansas State University mentions lambda expressions, but they may be a useful and powerful language feature for programmers to know, and the syntax of a lambda expression may be difficult to understand if a programmer never learned lambda expressions prior to encountering them, so our group believed it would be necessary to know basic concepts and some advanced level knowledge about lambda expressions.

Because we needed to report statistics of the usage of language expressions from open-source projects,. Dr. Kim provided a group of Java and Python source code to us, helped us to understand how to scan the open-source project, count the language feature usage and extract that data to draw a completed graph. After we extracted the language feature usage from the jetty project as a test, we modified some code based on Dr. Kim’s source code to change which language feature is scanned and counted. We used a Java server, Python, MySQL database, Cygwin64 and Matlab in this project.

Group member Daiki and Vu from another group made key contributions in code implementation of the project. Every group member made contributions to their own research questions respectively.

**2 RELATED WORK**

In order to resolve the research questions that we mentioned before, we researched many related works, which includes studied the library in Java, we learned some basic knowledge of Python and MySQL database implementations.

One of the related works was done to understand the lambda expression as a whole was written by Mazinanian, Ketkar Tsantalis and Dig. The researchers surveyed 241 open-source projects, which included 100,540 lambda expressions. The researchers found the reason for lambda adoption by developers, pros and cons of the feature, it did not mention the rate of usage of lambda expressions. Therefore, further research has to be done to answer this question.

In addition, Jeremiah Willcock, J.Jarvi,G . Gregor, B.Stroustrup jointly published a research paper together to elaborate Lambda expression in C++ and its relationship with closures in 2006, which mentioned many library problems, implementation issues and design issues of lambda expressions. But the example they used was too profound to understand at an undergraduate level, and they did not discuss lambdas in other programming languages. Thus, a research question would be created to involve the usage of lambda expressions in different programming languages.

The effect lambda expressions have on a programmer’s experience greatly affects their ability to properly do their job. The question is how much? In order to answer this question, Philip Uesbeck, Andreas Stefik, and Stefan Hanenburg conducted an experiment to see the extent that lambda expressions affect a programmer’s experience. They created two groups (one using lambdas and the other using iterators) and had them solve four programming tasks each. The first task was a warmup and had nothing to do with lambdas or iterators. The programmers ranged in experience from college freshmen to professional. The results of the experiment demonstrated that lambda expression usage slowed down the speed of a programmer on average and had more of an effect on inexperienced programmers. Although, this paper is limited in several regards, including: language choice (C++ was utilized for all programming in this experiment), an individual’s propensity towards lambda expressions, and there is no concrete theory that explains why lambdas would slow a programmer down. Regardless of the limitations, this paper is an invaluable resource for determining how lambda expressions affect the industry as a whole.

As listed above, there are some deficiencies in the current research into lambdas. In order to understand and utilize this feature fully. Our research tries to fill those blanks so that the readers can apply the knowledge from our research to their research or projects. Moreover, they can convert this knowledge into other areas such lambda expression exploration and other features such as type inference.

**3 BACKGROUND AND OVERVIEW**

Lambda expressions are one of the primary features of a paradigm called functional programming. The general reasons for such usage in programming languages include improving the readability and conciseness of the existing code, demonstrating lazy evaluation of functions, avoiding duplicating code and so on.

In order to help programmers understand what lambda expressions are, our group members raised three research questions respectively. Group member Daiki raised question: How often are lambda expressions used in open source projects? To answer this, we attempted to report the rate of usage of lambda expression statistics from open source projects. Group member Lloyd raised question : What are the best cases to use lambda expressions? We attempted to analyze the most efficient way to use lambda expressions. Group member Zhiquan raised question: What are the differences between lambda expressions in different programming languages? A programmer may use different languages to program, so it would be necessary to know the similarities and differences of lambda expressions between different programming languages. In this research question, Zhiquan choose to analyze lambda expressions in C++, Java and Python. Those languages are popular both in the undergraduate education and the industry.

We used the following programming tools to conduct our research:

1. *Java*: Java is a befitting language in this case because we can find built-in a function to detect lambda expressions, which is not available in C++ or some other high-level language. We created a visitor Java file to detect lambdas. Which also meant the open source projects we picked to scan were implemented by Java.
2. *MySQL*: We needed to use a database to store data that we scan, MySQL is a good option because MySQL is free for public and the workbench of MySQL is powerful and easy to operate. So we created some lambda expression tables here and we imported the data to MySQL database server.
3. *Python*: In general, open-source projects can be very large, they can contain dozens of folders and every folder contains dozens of files, and the programmer has to expand the open-source project after they downloads it. So we used Python to expand the project completely and used python files to conduct the Java server to scan the projects from one Java file to another Java file till the end. Then another Python file can be used to conduct MySQL database to extract data to .tvs file. This kind of file (.tvs) can be imported to Matlab to draw a graph. Python is a connector between Java and MySQL database.
4. *Cygwin64*: Not all of us used Cygwin64 because some group members had Mac’s command line and some members had Linux Subsystem for Windows. If a programmer wants to use Cygwin64, they have to make sure that they download enough packages to control Python and MySQL. In Cygwin64, we typed on the command line to execute different files in the appropriate folder. If a programmer engages a command not found error and they did not make any syntax errors, that means they are missing a package for their Cygwin64. We did not choose to download all available packages for Cygwin64 because they were too large and not necessary for our project.
5. *Matlab*: This is a software to draw graphs by using imported .tvs files, we could only apply a 30 day free trial, or we would have had to buy it. There are some other similar software that are available for free. We learned how to use Matlab from the manual quickly.

**4 SOLUTION**

**4.1 Usage Rate of Lambda**

In order to detect lambda expressions, we added a LambdaExpressionVisitor file in Java server (Figure 1). This class is called when the server detects a lambda expression in an open-source project. It prints a message to let the programmer know its action. After visiting, the function inside the class is called, the line number the lambda expression was found on is saved and its enclosing container are written as outputs. We also added LambdaPattern in run.py (Figure 2). The class is responsible for the connections to MySQL database and sending the project data to the Java server.

All the data is exported from the lambda\_expressions and revisions tables to a .csv file and is parsed. Data time and lambda expression count are outputted to produce a graph. After exporting all the data to the .tsv file, necessary conversion of the file is done for the software to recognize the data format. Then, Matlab or Octave was used to produce a graph.

A screenshot of a social media post

Description automatically generated

Figure 1: **LambdaExpressionsVisitor Class**

**A screenshot of a cell phone

Description automatically generated**

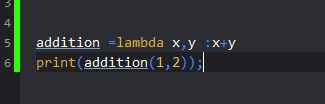
**Figure 2: LambdaPattern Class**

**4.2 Different Lambda Expressions**

*4.2.1 Lambda Expression in Python*

The syntax of lambda expression in Python is much easier than C++: lambda parameter:body

Here is an example of Lambda expression in Python in figure 3.



**Figure 3: Lambda Expression in Python**

The output is supposed to be 3 because this is an addition of 1+2. we can see that there is no capture section, no return type, only parameters list and body have been kept, there are do not even have any parentheses or brackets to separate each section. The biggest advantage of Python -styled lambda expressions is that the keyword must be added at the front of the statement, so everybody knows what is happening in the code, even a programmer who does not know anything related to lambdas, they can go to research it after seeing the keyword. The lambda syntax caters the ideal of Python: Make everything easy and clear.

*4.2.2 Lambda Expression in C++*

C++ doesn’t provide any support for lambda expressions before C++98. Even in C++98/03, a programmer cannot use lambda expressions directly, programmer has to include the library <Boost>, then use built-in the function Boost.Lambda to create an anonymous function. In C++11, programmer doesn’t have to use any library to implement a lambda expression. Here is the basic syntax of a lambda expression in C++:

[Capture](parameters)->return type{ body};

[Capture]: The capture section is the unique part of lambda expressions in C++, a regular function does not have a capture section because regular functions would only use a variable(s) inside of its body and the parameters list, if a programmer wants to use other variables inside a regular function, the programmer has to pass those variables by reference. Now the capture section provides another option for lambda expressions. This section allows a programmer to use the local variable(s) outside of lambda expressions. A programmer can write nothing inside the square bracket if they do not want to access any variable outside the lambda expression, but they cannot omit the square bracket or there would be a syntax error. If the programmer writes an assignment symbol inside the square bracket: [=], this means the programmer wants to make a copy of all local variables in the last stack.

If you write a lambda expression in the main function and add [=] in capture, the program would make a copy of all local variables in main (), you still cannot use the local variables of other functions in the lambda expression. If a programmer wants to access and make changes to all local variables from main directly, they can use [&]. The relationship between = and & is similar with pass-by-value and pass-by-reference in the parameter list. To copy or access all local variables would very difficult and may cost too much time, so the programmer can copy or access a specific number of variables to a lambda expression by using [=a] or [&a].There is no maximum variable limitation inside the capture section, programmer just needs a comma to separate each variable. In Java or Python, there is no capture section in lambda expression syntax, all variables outside of lambda expressions are defaulted to pass-by- reference.

Therefore the advantage of the structure of lambda expressions in C++ would be more privacy and security, the behavior can also make lambda expressions be more complicated to implement, a programmer may get an error from it if they are not familiar with the syntax.

(parameters): Parameter list in lambda expressions have almost no differences with regular functions except for three things:

1.There is no default value in a lambda parameter list

2. Every parameter in the list must have a name.

3.The parameter list is unchangeable in lambda

return type: Comparing to regular function, return type can be omitted in the two conditions below:

1.The return type for lambda expression is void.

2.There is only one sentence inside function body.

{body}: Function body of a lambda expression is same as regular function.

Here is a code segment of lambda expression (Figure 5). Most new programmers cannot figure out where the lambda expression is. Actually, the lambda expression is in the end of line 10 and line 11-13. This is the main benefit of lambda expression Lambda expressions significantly shorten the code. If we use a regular function to replace the lambda, we have to rewrite whole segment to begin with a function header, then write a for-loop inside the function body and make a function call in the main function, and possibly requires a prototype before the main function. Now we have lambda expression, we just need a single line after the for-loop statement. But this is also the main point of detriment, it is too difficult to figure out that this is a lambda expression even a programmer that knows basic knowledge of lambda expressions, most programmers would treat is as a part of for-loop.

In general, we don’t suggest programmers to use a lambda expression for complicated functions, if a programmer wants to write a complicated function, use a regular function and begin with a function header so that it can help other programmers and themselves to modify and understand this function precisely. The simplest version of a lambda expression in C++ can be written like this:

[] () {cout<<“hello world”};

empty capture section and parameters list, the arrow and the return type were omitted.

A screen shot of a computer

Description automatically generated

**Figure 4: Lambda Expression in C++**

*4.2.3 Lambda Expression in Java*

Java did not provide any support for lambda expressions until Java 8, it is one of the latest languages which support lambda expression, thus lambda package is a main package of Java 8, and Java 8 provides below functionalities.

1. Ability to treat functionality as a method argument, or code as data.
2. A function that can be created without belonging to any class.
3. A lambda expression can be passed around as if it was an object and executed on demand.

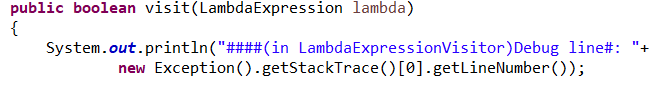
The syntax of lambda expressions in Java (figure 4):

parameters->expression

parameters->{ statements }

There are two types lambdas in Java. If there is only one expression after arrow symbol, programmer doesn’t have to add any brace, if there is a statement which was made up by more than one line of code, programmer needs braces for it. The difference will also present in return keyword. If lambda doesn’t have braces, no return keyword required, otherwise a return keyword is necessary when a programmer wants to return something.

There is a keyword to search for lambdas which was provided by the Eclipse Java library, it played a key role in our project to build the lambdaVisitor class.



**Figure 5: Lambda Expression in Java**

**4.3 Usage of Lambda Expression**

The main usage of lambda expressions is to shorten functions that would only be used a few times throughout the length of the program. For example in Java, before lambdas were introduced, there were two ways to define a functional interface (an interface that only defines one function): defining a separate class to create the function or use an anonymous inner class. See the following three images for examples of each (Figure 6, 7, & 8).

A screenshot of text

Description automatically generated

**Figure 6: Separate Class**

A screenshot of a cell phone

Description automatically generated

**Figure 7: Anonymous Function**

A screenshot of a cell phone

Description automatically generated

**Figure 8: Lambda Expression**

This example demonstrates the primary usage of a lambda expression: significantly reducing the lines of code required to identify a function. Lambda functions can also be used in parallel processing to speed up and further abstract code and make more generic projects. This is done by passing a lambda as a parameter to another function which uses the lambda as a check (called a predicate). The following is an example of predicate usage:

A screenshot of a cell phone

Description automatically generated

**Figure 9: Lambda Expression as a Predicate**

**5 EVALUATION**

After we finished the project, we discussed some points we did together. This was not an easy project to some undergraduate students to review problems and conclude implementation are necessary.

**5.1 Evaluation of Usage Rate of Lambda**

For this research question, we used two open-source projects, which are Guava and PMD (Figure 10). We can see that when lambda expressions were introduced in Java 8, which was in 2016, its usage was quite low. This is because developers were not used to it. However, as time went on, its usage began to gradually increase (Figure 11 and 12). We also see that at a certain time, the usage skyrocketed. We think that there is a refactoring of code and lambda expressions are heavily used in that situation. From this observation, we can conclude that the usage rate is increasing as lambda gets more and more popular due to its advantages.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Start** | **End** | **LOC** | **Lambda** |
| Guava | June, 2009 | November, 2019 | 517,054 | 157798 |
| PMD | June, 2002 | December, 2019 | 259,553 | 49444 |
| Jetty | August, 1998 | October 2019 | 1,334,412 | Unfinished |
| Flink | April, 2013 | December, 2019 | 1,305,760 | Unfinished |
| Apach Ant | January, 2000 | December, 2019 | 2,368,167 | Unfinished |
| XChange | February, 2002 | December, 2019 | 231,199 | Unfinished |

**Figure 10: Open-source Projects Used**

**A close up of a map

Description automatically generated**

**Figure 11: Graph of lambda usage in Guava**

**A screenshot of a map

Description automatically generated**

**Figure 12: Graph of lambda usage in PMD**

**5.2 Issues and errors**

*5.2.1 Command Not Found in Cygwin64*

There were some problems we encountered when we tried to scan the open-source project and extract the data from database. One big problem was the Command Not Found error. It always happens when someone did not download the correct Cygwin64 packages. Many people prefer to download parts of full packages because whole packages of Cygwin64 can be more than 20 gigabytes. We suggest to programmers to download these packages below, they were what we needed for this project (Figure 13). If any programmer still encounters the Command Not Found error. Check the syntax of command on the command line then download full packages of Cygwin64, it might be slow because it’s size , but it should help. If a programmer has some other problems related to Python or MySQL database, they should check to make sure they have the correct packages. .

A screenshot of a cell phone

Description automatically generated

**Figure 13: Cygwin64 Packages**

**5.3 Unknown Phenomena**

Group member Zhiquan Tang encountered an unknown phenomena after he scanned the open-source project. When he attempted to extract data from MySQL database to the .tvs file through the Python file, Cygwin64 shows the lines below (Figure 14):

A screenshot of a cell phone

Description automatically generated

**Figure 14: Error Zhiquan encountered**

Those lines did not show any distinct error in the Python file, which printed some lines of code inside the type-over-time-lambda.py file. Later, Zhiquan checked every line in the type-over-time-lambda.py file and used Dr. Kim’s source .py file as template and he found nothing wrong, everything complied with SQL syntax and the format of type-over-time-lambda was same as Dr. Kim’s template. This unknown problem stopped the formation of .tvs file, so Zhiquan could not draw the graph of lambda usage.

Group member Lloyd Thomas also encountered issues with displaying the graph of lambda usage. His issue revolved around getting the .tvs file’s data to make the graph. He ran the run.py file and the resulting .tvs file would be an empty table except for the headers of the table. He attempted to redownload the database and reinstall the environment programs, but the table would still be empty every time (Figure 15).

A screenshot of a cell phone

Description automatically generated

**Figure 15: Error Lloyd encountered**

**6 CONCLUSION**

In this paper. we discussed the basic features and importance of lambda expressions. To find how much lambdas are used in open-source projects, we utilized Java because it contains a built-in function to detect the syntax of a lambda expression, MySQL as a database to store the number of lambda expressions encountered in a given open-source project, Python to expand the open-source project that we were currently analyzing. Cygwin64 as a command-line, and Matlab to create a graph out of the data extracted from the open-source project. Once our environment was set up, we first tested the environment by extracting the lambda expression data from the jetty project. During this step two of our members has errors creating a graph which limits our research. Afterwards, the group member that did not have errors extracted the data from two additional projects: Guava and PMD. Utilizing the data and previously done research, we answered our three research questions: How often are lambda expressions used in open source projects?, What are the best cases to use lambda expressions?, and What are the differences between lambda expressions in different programming languages? First, the usage of lambda expressions in open-source projects has been increasing over time. Second, lambda expressions are best used to replace one-off functions (functions that will only be called once or a few times throughout the life of the program). Lastly, lambda expressions largely differ syntactically in various programming languages.

**REFERENCES**

[1] Davood Mazinanian, Ameya Ketkar, Nikolaos Tsantalis, and Danny Dig. 2017. Understanding the Use of Lambda Expressions in Java. Proc. ACM Program. Lang. 1, OOPSLA, Article 85 (October 2017), 31 pages. DOI: <https://doi.org/10.1145/3133909>

[2] Jakko jarvi. John freeman,2009. C++ lambda expressions and closures. Proc. ACM symposium on Applied computing 178-183(March 2008)

DOI: https://doi.org/10.1016/j.scico.2009.04.003

[3] Phillip Merlin Uesbeck, Andreas Stefik, and Stefan Hanenberg. 2016. An Empirical Study on the Impact of C++ Lambdas and Programmer Experience. Proc. Int’l Conf. Softw. Eng. (ICSE’16). (May 2016). 760–771. DOI: https://doi.org/10.1145/2884781.2884849