Programming Language

[1]Rakesh Sinha,[2]Oly Rajak,[3] Prof Subhashis Kumar Chandra
[1] Student,[2] Student,[3] Professor
Computer Science,
[1] T.D.B College , Raniganj, India

Abstract: A programming language is one kind of written instruction that tells computers what to do. It is a grammatical formula for a computer system. Any kind of computer program or computer software is built by a programming language. It can be said that programming language is a set of instructions that computer follows to do some specific task.

All programming languages have a unique set of keywords and it has a special syntax for constructing programming knowledge. Since programming languages are primarily performed in sequence using only line code, and it includes a function among the first layers of abstraction, and then the class and the object are created abstractly.

The algorithm is the backbone of programming language. In our everyday life whatever type of work we want to do or we have to do that case we must follow an algorithm to proceed it. The algorithm is plotted in a programming language. For example, we can say if we want to search anything we use either binary search or linear search or sequential search technique.

But the real fact is that we don't think about it.But there are too many situations where algorithm doesn't hold that's why there is a recent addition which is named as Big Data or Machine learning**.**

Index Terms– **Introduction, Type of programming Language , Operator of programming language , Principle of programming language, Elements of programming language , Generation of programming language , Some programming language.**

I. Introduction

A programming language is one of the most important parts of software technologies. It is very important for describing hardware and software with an intention. It is a formal language that is used to develop many software programs. It is a main key factor for every software. It intermediate between Human and computer. This is one type of formal language that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of instructions for a computer. A programming language can be also used to create programs that implement a specific type of algorithms.

II. Type of Programming Language

i) High-level language
ii) Low-level language

Low-level language level language are two types-

i) Machine language-

The machine language is that custom programming is then possible. The computer comes with a disk operating system (DOS) and basic (or other "high level" language). After a while, you'll probably be limited by the rules or commands available in these languages. You want to add them to customize them. Need to understand machine language, if you want to add new words to BASI, modify a word processor or personalize your computer - to behave properly according to your preferences.

Machine language defines as 0&1.

ii) Assembly language -

Assembly language is a utility program converted into an executable machine code which is considered to be a connector.

The conversion process is referred to as a rally, or the source code is merged.

We use ADD, SUB, LOAD, and MUL in Assembly Language.

Programming Language are widely used in C,C++,JAVA,PYTHON,PHP,C#,JAVA SCRIPT,SWIFT ,Parrot ,Perl etc.

III. Operator of Programming Language

Relational Operator:

- "==" is used for check two operant is equal or not, if they are equal then the condition true.
- "! =" is used for check two operant is equal or not, if they are not equal then the condition is true.
- ">" is used for check the value of left operand is greater than the value of right operand then the condition is true otherwise it is false.
- "<" is used for check the value of the right operand is greater than the left operand then the condition is true otherwise it is false.
- ">=" is used for the value of left operand is greater than or equal to the right operand then the condition is true.
- "<=" is used for the value of the right operand is greater than or equal to the left operand then the condition is true.

Logical Operator:

- "&&" is a logical AND operator.
- "||" is a logical OR operator.
- "!" is a logical NOR operator.

Binary Operator:

- "&" is a binary AND operator.
- "|" is a binary OR operator.
- "^" is a binary XOR operator.
- "~" is a binary one's complement operator.
- "<<" is a binary left shift operator.
- ">>" is a binary right shift operator.

Assignment Operator:

- "=" it a simple assignment operator it is used for assign the value to the right side of the operator.
- "+=" it is used for add the right operand and the left operand.
- "-=" it is used for subtracting the right operand from the left operand.
- "*" it is used to multiply the right operand and the left operand.
- "/=" it is used to divide the right operand by the left operand.
- "% =" it is a modular AND operator.
- ">>=" it is a right shift operator.
- "<<=" it is a left shift operator.
- "&=" it is an operator that is true for bitwise AND operator.
- "^=" it is an exclusive OR operator.
- "|=" it is an inclusive OR operator.

IV. Principle of Programming Language

Language is the notation that we use to express ideas and to a large extent, that notation is a product of a kind of ideas that we look to express. We will see that in many ways many of the factors that shape human languages will also shape programming languages given that programming languages are a determining factor in how we write programs and even how we design Hardware is becoming very important to understand how they come to take the forms that they have.

A programming language is a notation that we use to describe an algorithm or some other computation that we require the computer to perform. There are two imperatives that conflict with each other making languages close enough to natural or human language that they can replicate how we express ideas, but also close enough to the instructions in the computer's own instruction set. High-level languages are closer to the first goal while lower level languages are closer to the second goal.

A language is a primary tool in shaping the ideas that we form. It's difficult to express and even concede the ideas without the words to express them and a way to put these words together and that is exactly what language is. Ideas need to be transformed into algorithms they can't become programs without language to write them in. A language leads to better ideas which lead to better programming. It's important to understand how a language is implemented to be able to use it correctly. This is especially important when trying to fix bugs in the programs that you have written using that language.

Understanding a language helps you learn how to make better use of its features.

A hammer is great for banging in nails and screwdriver is better for inserting and removing screws. The same is true for a programming language. If you are doing large-scale calculations FORTRAN may be better, for business reports that might COBOL or Sequel, but this requires knowing something about the languages themselves.

V. Element of Programming Language

1) *Array*

An array is a collection of data. In a programming language, there is a way to save many items of an array. These items should have the same type of reason because an array cannot store various items. The same type of an array is used for data gathering in one variable together. An array is used for collecting data of the same type in one variable. In this section, we will discuss some of the cases where an array is a suitable data structure to use.

Declaration of an array-Data _type array_name[array_size];

2) *Pointer*

A pointer is a variable that holds the address of the memory of another variable. We can have a pointer to any variable type.It is a variable which contains the address in memory of another variable.We can have a pointer to any variable type.

Declaration of a pointer-Type *var_name; here var name means the variable name of pointer type.

3) *Stack*

A Stack is the abstract type of data that is mostly used in a programminglanguage. In stack two operationsis used –

i) For insertion we can use 'PUSH' function and

ii) For deletion we can use 'POP'. It works through the concept of Last in first out.

The real-life example is a stack of books in the cupboard, Bangles wear or removing.

4) *Variable*

In computer programming, a variable is a storage location attached to the associated chamber name, which includes some known or unknown information that is known as a value. A variable can contain any type of information by a particular data type, for example, a string, a digit or a letter.

5) *Function*

A function is a unit of code which is often defined by its role inside a larger code structure. In particular, a function contains a unit of code that works in different inputs, which are many variables, and concrete results based on the change in the value of the variable or the actual operations based on the input.

Some example of function are:-

```
i)  Input "A"
        If A>=5 then
Print A
        End
ii) Input "A"
       If A%5==2 then
       Print A
       End
```

6) *String*

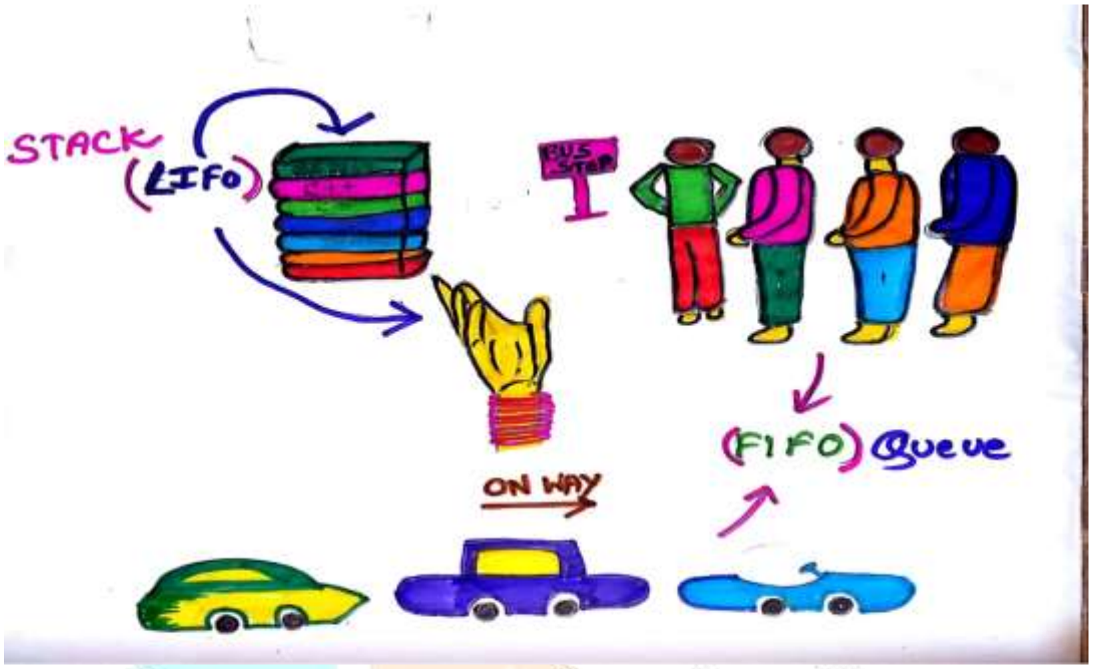
A string is traditionally a sequence of characters, either as a literal constant or as a variable of some kind. The latter may change its components and the length may vary, or it can be corrected. A string is usually understood as a data type and is often applied as an array data froma byte, which uses some character encoding to generally store a character's content. A string may also refer to more common arrays or other sequence information types and structure.

Declaration of string- Char string_variable_name [size];

7) *Queue*

It works through the concept of FIFO means first in first out. The queue has two function front and rear. By using front we can insert element and by using rear we can delete the element.

Real life example. Are Ticket counter,single lane in one way road?



VI. Generation of programming language

The generation of a programming language are five types-

1) First Generation Language.
2) Second Generation Language.
3) Third Generation Language.
4) Fourth Generation Language.
5) Fifth generation language.

1) First Generation Programming Language:

A first-generation programming language is a machine-level programming language. Originally, no translator was used to compile or assemble the first-generation language. The first generation programming instructions were entered through the front panel switches of the computer system. A first generation language is a grouping of programming languages that are machine level languages used to program first-generation computers. The instructions were given through the front panel switches of these computers, directly to the CPU. There was originally no compiler or assembler to process the instructions in 1GL. The instructions in 1GL are made of binary numbers, represented by 1s and 0s. This makes the language suitable for the understanding of the machine but very much more difficult to interpret and learn by the human programmers. The main advantage of programming in 1GL is that the code can run very fast and very efficiently, precisely because the instructions are executed directly by the CPU. One of the main disadvantages of programming in a low-level language is that when an error occurs, the code is not as easy to fix. First generation languages are very much adapted to a specific computer and CPU, and code portability is therefore significantly reduced in comparison to higher level languages. Modern day programmers still occasionally use machine level code, especially when programming lower level functions of the system, such as drivers, interfaces with firmware and hardware devices. Modern tools, such as native-code compilers are used to produce machine level from a higher-level language.

2) Second Generation Language

Second generation programming language is a generational way to categorize assembly languages. The term was coined to provide a distinction between higher level third-generation programming languages such as COBOL and earlier machine code languages. Second generation programming languages have the following properties:

The code can be read and written by a programmer. To run on a computer it must be converted into a machine-readable form, a process called an assembly. The language is specific to a particular processor family and environment. Second generation languages are sometimes used in kernels and device drivers, but more often find use in extremely intensive processing such as games, video editing, and graphics manipulation/rendering. One method for creating such code is by allowing a compiler to generate a machine-optimized assembly language version of a particular function. This code is then hand-tuned, gaining both the brute-force insight of the machine optimizing algorithm and the intuitive abilities of the human optimizer.

3) Third Generation Language

A third-generation programming language is a generational way to categorize high-level computer programming languages. Where assembly languages, categorized as second generation programming languages, are machine-dependent, 3GLs are much more machine independent and more programmer friendly. This includes features like improved support for aggregate data types and expressing concepts in a way that favors the programmer, not the computer. A third-generation language improves over a second generation language by having the computer take care of non-essential details. 2GLs feature more abstraction that previous generations of languages, and thus can be considered higher level languages than their first and second generation counterparts. First introduced in the late 1950s, FORTRAN, ALGOL, and COBOL are early examples of this sort of language. Most popular general-purpose languages today, such as C, C++, C#, Java, BASIC and Pascal, are also third generation languages, although C++, Java, and C# follow a completely different path as they are object-oriented in nature. The third generation focused on structure and had no meaning for the object-oriented concepts. Most 3GLs support structured programming. A programming language such as C, FORTRAN, or PASCAL enables a programmer to write programs that are more or less independent of a particular type of computer. Such languages are considered high-level because they are closer to human languages and further from machine languages. In contrast, assembly languages are considered low-level because they are very close to machine languages. The main advantage of high-level languages over low-level languages is that they are easier to read, write and maintain. Ultimately programs written in a high-level language must be translated into machine language by a compiler or interpreter. The first high-level programming languages were designed in the 1950s. Examples of early high-level languages are ALGOL, COBOL, and FORTRAN. These programs could run on different machines so they were machine-independent. As new, more abstract languages have been developed, however, the concept of high and low-level languages has become rather relative. Many of the early "High Level" languages are now considered relatively low level in comparison to languages such as Python, Ruby, and Common Lisp.

4) Fourth Generation Language

A fourth-generation programming language (1970s-1990) is a programming language or programming environment designed with a specific purpose in mind, such as the development of commercial business software. In the history of computer science, the 4GL followed the 3GL in an upward trend toward higher abstraction and statement power. The 4GL was followed by efforts to define and use a 5GL. The natural-language, block-structured mode of the third-generation programming languages improved the process of software development. However, 3GL development methods can be slow and error-prone. It became clear that some applications could be developed more rapidly by adding a higher-level programming language and methodology which would generate the equivalent of very complicated 3GL instructions with fewer errors. In some senses, software engineering arose to handle 3GL development. 4GL and 5GL projects are more oriented toward problem-solving and systems engineering.

All 4GLs are designed to reduce programming effort, the time it takes to develop software and the cost of software development. They are not always successful in this task, sometimes resulting in inelegant and unmaintainable code. However, given the right problem, the use of an appropriate 4GL can be spectacularly successful as was seen with MARK-IV and MAPPER. The usability improvements obtained by some 4GLs (and their environment) allowed better exploration for heuristic solutions than did the 3GL.

5) Fifth generation language.

A fifth-generation programming language is a programming language based on solving problems using constraints given to the problem, rather than using an algorithm written by a programmer. Most constraint-based and logic programming languages and some declarative languages are fifth-generation languages. While fourth generation programming languages are designed to build specific programs, fifth-generation languages are designed to make the computer solve a given problem without the programmer. This way, the programmer only needs to worry about what problems need to be solved and what conditions need to be met, without worrying about how to implement a routine or algorithm to solve them. Fifth-generation languages are used mainly in Artificial Intelligence research. Prolog, OPS5, and Mercury are examples of fifth-generation languages. These types of languages were also built upon Lisp, many originating on the Lisp machine, such as ICAD. Then, there are many frame languages, such as KL-ONE. In the 1980s, fifth-generation languages were considered to be the wave of the future, and some predicted that they would replace all other languages for system development, with the exception of low-level languages. Most notably, from 1982 to 1993 Japan put much research and money into their fifth generation computer system project, hoping to design a massive computer network of machines using these tools. However, as larger programs were built, the flaws of the approach became more apparent. It turns out that, given a set of constraints defining a particular problem, deriving an efficient algorithm to solve it is a very difficult problem in itself. This crucial step cannot yet be automated and still requires the insight of a human programmer. A common misconception, Vendors have been known on occasion to advertise their languages as 5GL. Most of the time they actually sell 4GLs with a higher level of automation and knowledge base. Since the 5GL awareness has dropped because the hype of the 1980s quickly faded away and the projects were eventually all dropped; this has opened doors to the vendors to re-use the term in marketing their new tools, without causing much controversy among the current generations of programmers.

VII. Some Programming Language

- ***C:***

C is a programming language that has been mostly used in operating system and application and that had widely following by the academic community. C was designed to be compiled using a relatively straightforward compiler.

- ***C++:***

C ++ is a compiled as general purpose programming language. Both combine the topics of both upper and lower level languages, the real name of C ++ is "Class with.

Difference between C and C++

Sl No	C Language	C++ Language
1	In c we use scanf() function for input.	In c++ we use cin>> function for input
2	In c we use print() function for output.	In c++ we use cout<< for output
3	C was developed by C-Danish Richie in 1969 and 1973 by It & T Bell Labs.	C++ was developed by Bjarne Stroup in 1979.
4	C is a subset of C++.	C++ is total set.
5	All c programs are save in C extension.	All C++ programs are save in Cpp extension.
6	In c the feature namespace is absent.	In C++ the feature namespace is present.

- ***JAVA:***

Java is a programming language that was designed by James Gosling from sun microsystems in 1991.It is a type of programming language that enters one time into the program and then runs this program on multiple operating systems. Java runs different platforms such as Windows, Mac OS, and UNIX versions. This tutorial gives you a full understanding of Java. While learning the Java programming language, this reference will take you through a simple and realistic approach.

- ***Python:***

Python explains a general purpose, interactive, object-oriented and high-level programming language. It was designed by Goido Van Rossum in between 1985-1990. It is very simple language but when 1st time one seen it is difficult but it has easy structure and simple syntax that one can easily understand. This language automatically supports garbage collection.

- ***PHP:***

Php is a broadcasting language that was designed by Rasmus Lerdorf in 1994.It is also a powerful tool for creating web pages. Php platform web server it interprets a high level of scripting, but it does not have access to the web server at the level of language.
A PHP always starts with <?php and ends with ?>:

- ***Script:***

Scripting languages can be very fast to learn and write, such as interactively via short source code files or in a read-print loop. It usually refers to relatively simple syntax and acoustics; usually, a script comes into effect from the beginning to the end, as a script, without a clear entry point. Some example of script language- JavaScript, HTML, XML, DHTML, shtml, XHTML, Unix Shell, VBScript, Numpy.

- ***Swift:***

Swift is a new programming language that was created by Apple for iOS and OS x development. In this language, Multi-line comments start with /* and terminate with the characters */.

- ***RUBY:***

Ruby is one kind of scripting programming language. It was designed by Yukihiro Matsumoto. This language can be used on various platforms such as windows MacOs and various version of UNIX. This language is highly scalable and the programs which are written in this language is very easy to use. This language is also used for development in an internet application.

- ***Perl:***

This programming language is developed by Larry Wall. The full form of Perl is Practical Extraction and Report Language. This is an open source software language.

- ***Parrot:***

Parrot language is written in most popular programming language C. Parrot is a virtual machine designed to execute compile and bytecode efficiently for spoken languages. Parrot finally target the Perl 6 Compiler.

VIII. CONCLUSION

A related to the idea that computer code preferred method computer our guidelines contact. This approach to our specified and expressive be able to be, it is our actions a complete record available, and others can our work copy. The process, stores computer write code or display information a task that enough order with will take place. It careful the small piece and code need to develop, and it tidy is generate codes important and reality formed. This order that the future of the results of both now and will produce writing code essential. Another important concept of dry policy. Information, whether the information or computer code, the only information each important unit one copy is that in such a way to organize will. We information storage terms and conditions in the idea of the most definitely have seen, XML documents design and relational database design in. But ideas organizational information skills affects how can we are a lot of ways to work. The dry policy also how can be applied we computer code to write, especially in the loop and function of use in. This is how can be applied we file in our code collection and how we directory or folder in a file organized.

IX. REFERENCE

[1] "Programming Language Popularity". 2009. Archived from the original on 13 December 2007. Retrieved 16 January 2009

[2] "Using C for CGI Programming". Linuxjournal.com. 1 March 2005. Retrieved 4 January 2010

[3] Guttag, John V. (2016-08-12). Introduction to Computation and Programming Using Python: With Application to Understanding Data.

[4] Stroustrup, Bjarne. "The C++ Programming Language" (First ed.). Retrieved 16 September 2010.

[5] Stroustrup, Bjarne. "The C++ Programming Language" (Second ed.). Retrieved 16 September 2010.

[6] Wall, Larry, Tom Christiansen and Jon Orwant (July 2000). Programming Perl, Third Edition. O'Reilly Media.ISBN 0-596-00027-8.

[7] Preface (Modern Perl 2011-2012). Modernperlbooks.com. Retrieved on 2013-07-17

[8] Abelson, Sussman, and Sussman. "Structure and Interpretation of Computer Programs". Archived from the original on 26 February 2009. Retrieved 3 March 2009.

[9] Brown Vicki (1999). "Scripting Languages". mactech.com. Archived from the original on 7 December 2017. Retrieved 17 November 2014.

[10] Georgina Swan (21 September 2009). "COBOL turns 50". computerworld.com.au. Archived from the original on 19 October 2013. Retrieved 19 October 2013.

[11] Ed Airey (3 May 2012). "7 Myths of COBOL Debunked". developer.com. Archived from the original on 19 October 2013. Retrieved 19 October 2013.

[12] Nicholas Enticknap. "SSL/Computer Weekly IT salary survey: finance boom drives IT job growth". Computerweekly.com. Archived from the original on 26 October 2011. Retrieved 14 June 2013.

[13] "Counting programming languages by book sales". Radar.oreilly.com. 2 August 2006. Archived from the original on 17 May 2008. Retrieved 3 December 2010.

[14] Bieman, J.M.; Murdock, V., Finding code on the World Wide Web: a preliminary investigation, Proceedings First IEEE International Workshop on Source Code Analysis and Manipulation, 2001

[15] "Programming Language Popularity". langpop.com. 25 October 2013. Archived from the original on 16 January 2009. Retrieved 2 January 2014.

[16] Carl A. Gunter, Semantics of Programming Languages: Structures and Techniques, MIT Press, 1992, ISBN 0-262-57095-5, p. 1

[17] "TUNES: Programming Languages". Archived from the original on 20 October 2007.

[18] Wirth, Niklaus (1993). "Recollections about the development of Pascal". Proc. 2nd ACM SIGPLAN conference on the history of programming languages: 333–342. doi:10.1145/154766.155378. ISBN 0-89791-570-4. Retrieved 30 June 2006.

[19] New supported release 8.1.0 "Andean Parakeet". Parrot Foundation. 2016-02-16. Retrieved 2016-09-26.

[20] "Parrot Contributor License Agreement 1.0" (PDF). Parrot Foundation. Retrieved 2009-03-18.

[21] "Parrot Roadmap". Parrot Foundation. 2008-11-20. Retrieved 2008-11-20.

[22] "The Story Behind the Parrot Prank - O'Reilly Media". Oreilly.com. 2001-04-06. Retrieved 2014-02-25.

[23] "Programming Parrot". Perl.com. Archived from the original on 2010-07-18. Retrieved 2014-02-25.

[24] Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; Buckley, Alex (2014). The Java® Language Specification (PDF) (Java SE 8 ed.).

[25] Gosling, James; Joy, Bill; Steele, Guy L., Jr.; Bracha, Gilad (2005). The Java Language Specification (3rd ed.). Addison-Wesley. ISBN 0-321-24678-0.

[26] Lindholm, Tim; Yellin, Frank (1999). The Java Virtual Machine Specification (2nd ed.). Addison-Wesley. ISBN 0-201-43294-3.

[27] Lerdorf, Rasmus (2007-04-26). "PHP on Hormones" (mp3). The Conversations Network. Retrieved 2009-06-22.

[28] Lerdorf, Rasmus (2007). "Slide 3". slides for 'PHP on Hormones' talk. The PHP Group. Retrieved 2009-06-22.

[29] Lerdorf, Rasmus (June 8, 1995). "Announce Personal Home Page Tools (PHP Tools)". Retrieved 7 June 2011.

[30] Swift Has Reached 1.0". Apple. September 9, 2014. Retrieved March 8, 2015.

[31] https://swift.org/download/#releases/

[32] David A. Schmidt, The structure of typed programming languages, MIT Press, 1994, ISBN 0-262-19349-3, p. 32

[33] Pierce, Benjamin (2002). Types and Programming Languages. MIT Press. p. 339. ISBN 0-262-16209-1 David A. Schmidt, The structure of a typed programming language

[34] David A. Schmidt, The structure of typed programming languages,

[35] Robert W. Sebesta: Concepts of Programming Languages, 9th ed., Addison Wesley 2009.[

[36] Franklyn Turbak and David Gifford with Mark Sheldon: Design Concepts in Programming Languages, The MIT Press 2009.

[37] Peter H. Salus. Handbook of Programming Languages (4 vols.). Macmillan 1998.

[38] Ravi Sethi: Programming Languages: Concepts and Constructs, 2nd ed., Addison-Wesley 1996.

[39] Michael L. Scott: Programming Language Pragmatics, Morgan Kaufmann Publishers2005

[40] Abelson, Harold; Sussman, Gerald Jay(1996). Structure and Interpretation of Computer Programs (2nd Ed.). MIT Press.

[41] Raphael Finkel: Advanced Programming Language Design, Addison Wesley 1995.

[42] Daniel P. Friedman, Mitchell Wand, Christopher T. Haynes: Essentials of Programming Languages, The MIT Press 2001.

[43] Maurizio Gabbrielli and Simone Martini: "Programming Languages: Principles and Paradigms", Springer, 2010.

[44] David Gelernter, Suresh Jagannathan: Programming Linguistics, The MIT Press 1990.

[45] Ellis Horowitz (ed.): Programming Languages, a Grand Tour (3rd ed.), 1987.

[46] Ellis Horowitz: Fundamentals of Programming Languages, 1989.

[47] Shriram Krishnamurthi: Programming Languages: Application and Interpretation, online publication.

[48] Bruce J. MacLennan: Principles of Programming Languages: Design, Evaluation, and Implementation, Oxford University Press1999.

[49] John C. Mitchell: Concepts in Programming Languages, Cambridge University Press2002.

[50] Benjamin C. Pierce: Types and Programming Languages, The MIT Press 2002