Technium Sustainability

Technium Sustainability
Vol. 1, Issue 1, pp.21-33 (2021)
ISSN: 2810-2991
www.techniumscience.com

Formation of a Programming Languages Stack and a methodology of teaching to students specialized in Computer Science at Technical Universities in the context of interdisciplinarity

Gregory Korotenko

Dnipro University of Technology, Ukraine
korotenko.g.m@nmu.one

Leonid Korotenko

Dnipro University of Technology, Ukraine
leonid_korotenko@ukr.net

Abstract. Rapid development of the computing hardware and software components has created many new technological directions for the use of computer tools. Thus, information, digital and end-to-end technologies for the processing of a variety of data exist and are developing, as well as many different types of platforms for their support, including digital platforms. At the same time, the active development of computing and its sub-disciplines directly impacts on the programming languages used in this area. This article attempts to assess the range of the most important and frequently used programming languages for the formation of a stack thereof and the methodology of their teaching in the context of the interdisciplinarity of modern computer specialties at technical Universities.

Keywords. computing, programming languages stack, methodology, teaching, programming, interdisciplinarity, technical university.

1. Introduction

One of the previous papers include an issue raised by their authors of the need to form a stack of programming languages (PL), which are the most effective in solving a wide range of computing problems studied at technical Universities [1]. The argument has been made that a rapid increase in the number of levels and the actual number of different platforms, as well as their ecosystems, imposes complex requirements on the technical Universities to include not only theoretical, but also practical materials directly related to programming into the content of new courses for the formation of students' relevant competencies. And here the question arises about the formation of a stack of the most promising programming languages, which are able to ensure, on the one hand, the best entry of future specialists into the subject area, and on the other hand, to meet the requirements of the modern labor market in the best possible way.

In the last CC-2020 document [2] a question remains open about recommendations for studying specific PLs. For example, see page 110 [2]: it is possible to draw attention at the following phrase regarding software development: “Evaluate and apply programming paradigms and languages to solve a wide variety of software design problems being mindful of trade-offs including maintainability, efficiency, and intellectual property constraints". However, unfortunately, this conclusion is not supported by any conceptual conclusions about the use of specific PLs in this process.

However, specialists have identified another stable tendency that exists in the programming community: specialists of this type of activity quite often change their places of work. A number of publications state that the duration of such time periods of employment can range from 2 to 4 years [3-6].

Overall, the impact of the rapidly developing computing area and its sub-disciplines on the formation of the corresponding profile curricula at technical Universities remains a priority issue (Fig. 1).

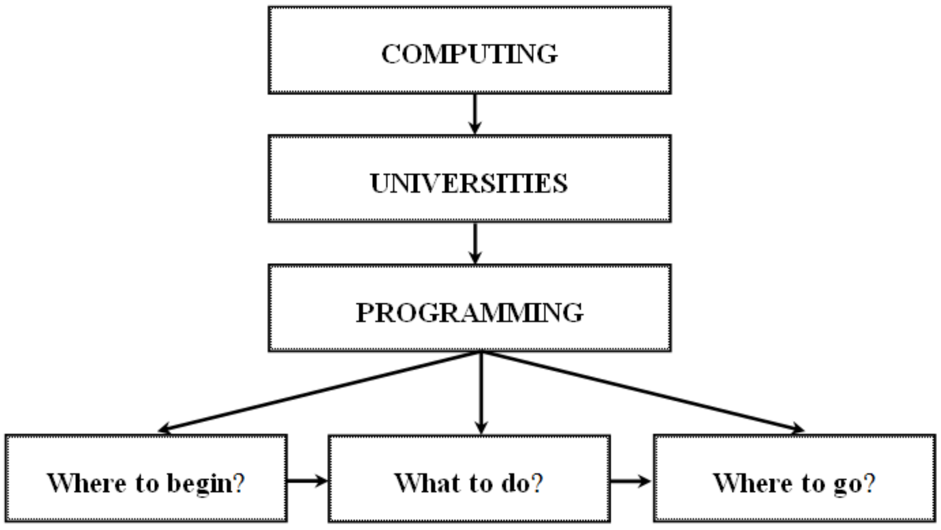

Fig. 1. Interaction between the Computing, Technical Universities and Programming

For example, the Higher Education Standards of Ukraine [7], which define the basic competencies related to the bachelors’ learning outcomes in the field of knowledge 12 "Information Technologies", have the latter to be formed in the course of studying disciplines (courses) that are part of the curriculum of educational and professional programs. Practically, the foundation for the future career of a programmer is laid in the curricula of all six specialties in this field of knowledge (121 Software Engineering, 122 Computer Science, 123 Computer Engineering, 124 Systems Analysis, 125 Cybersecurity, 126 Information Systems and Technologies) within three courses: "Algorithmization and Programming" (AP) , "Algorithms and Data Structures" (ASD) and "Object-Oriented Programming" (OOP) actually lay the foundation for the future career of a programmer.

The first of these courses (AP) usually involves the solution to educational and practical challenges by means of any of the well-known PLs (C ++, Java, C#, Python, etc.). The second (ASD), on the basis of the same PL, studies data structures and algorithms that constitute a

Technium Sustainability
Vol. 1, Issue 1, pp.21-33 (2021)
ISSN: 2810-2991
www.techniumscience.com

foundation for the modern program development methodology, as well as various implementations of abstract data types (ADT), ranging from standard lists, stacks, queues and ending with sets and mappings, which are used for the algorithms' informal description and implementation. In addition, methods to analyze the complexity of algorithms and their construction are studied. The third course (OOP) examines the concepts of object-oriented programming (encapsulation, inheritance and polymorphism) based on the creation and use of classes and objects, multiple inheritance, virtual functions, abstract classes, etc. This is usually taught in one PL. We hope.

2. Description of the method

In this Paper, the authors have made an effort to go along the path described below in order to determine a possible stack of languages. A website "50 Best Jobs in America for 2021" [8] contains a list of the most demanded areas for the application of the knowledge of specialists, including graduates of technical Universities. Some of the titles given on the website directly indicate knowledge of specific programming languages, for example, Java Developer. Other positions point to specific areas of activity where knowledge of PL is implied, for example, Data Scientist, Devops Engineer, Mobile Engineer, etc. Consequently, the jobs related to the fields of information technologies application have been selected for the research objectives, and some of the most popular of them have been added.

Each of the jobs under consideration, contains an analysis made in regard to the PLs [9-22] being mostly used in the structure of this type of activity. Consequently, Table 1 has been obtained, containing 14 types of jobs, for each of which the five most demanded PLs have been specified.

Table 1. Top programming languages should learn

№	**Profession**	**Programming Language Rating**				
		1	**2**	**3**	**4**	**5**
1.	Systems admins [9]	Python	Bash	PowerShell	Ruby	JavaScript
2.	Data Science and Machine Learning [10]	Python	R	SQL	Java	Scala
3.	Security admins [11]	Python	Golang	JavaScript	C	C++
4.	Mobile App Development [12]	JavaScript	Java	Swift	Kotlin	Dart
5.	Software Eng / Application solutions developers [13]	Java	JavaScript	C++	Python	CRM
6.	Front End [14]	React	Javascript	CSS	HTML	Angular
7.	Back End [15]	JavaScript	Python	Ruby	PHP	Java
8.	Cloud engineers [16]	Java	AngularJS	C++	Python	AWS
9.	CodinGame [17]	JavaScript	C#	C++	Java	HTML
10.	IoT Development [18]	C++	Java	JavaScript	PHP	Swift
11.	Network architects [19]	Python	Perl	C	TCL	Ansible
12.	Network admins [20]	Perl	Bash	Tcl	Golang	Python
13.	Database administrator [21]	SQL	PHP	Python	R	C#
14.	Blockchain Developer [22]	Solidity	JavaScript	C++	Python	Golang

Technium Sustainability
Vol. 1, Issue 1, pp.21-33 (2021)
ISSN: 2810-2991
www.techniumscience.com

An approach described below has been used to obtain a final estimate of the level of frequency of PLs use for all fourteen types of activities. For each of the jobs, 5 points have been granted to language, ranked number one, 4 points – to the PL placed second, 3 points – to the PL at the third place, etc. Further, these scores have been summed up for each PL and a new Table has been developed with languages arranged in descending order of the scores obtained for the use in all 14 jobs. And finally, a complex joint Table has been developed for all the languages under consideration, showing also the total score of each of the languages. The resulting Table 2 has been obtained after carrying out the corresponding calculations.

Table 2. Distribution of languages by areas of application

PL	Systems admins	Data Science and Machine Learning	Security admins	Mobile App Development	Software Engineer	Front End Developer	Back End Developer	Cloud engineers	CodinGame	IoT Development	Network architects	Network admins	Database administrator	Blockchain Developer	Sum
Python	5	5	5		2		4	2			5	1	3	2	34
JavaScript	1		3	5	4	4	5		5	3				4	34
Java		2		4	5		1	5	2	4					23
C++					3			3	3	5				3	17
Perl											4	5			9
SQL		3											5		8
Bash	4											4			8
PHP							2			2			4		8
Golang			4									2		1	7
R		4											2		6
React			1			5									6
C#									4				1		5
Solidity														5	5
AngularJS						1		4							5
Ruby	2						3								5
C			2								3				5
Tcl											2	3			5
Swift				3						1					4
PowerShell	3														3
CSS						3									3

Technium Sustainability
Vol. 1, Issue 1, pp.21-33 (2021)
ISSN: 2810-2991
www.techniumscience.com

PL	Systems admins	Data Science and Machine Learning	Security admins	Mobile App Development	Software Engineer	Front End Developer	Back End Developer	Cloud engineers	CodinGame	IoT Development	Network architects	Network admins	Database administrator	Blockchain Developer	Sum
HTML						2			1						3
Kotlin				2											2
Scala		1													1
Dart				1											1
CRM					1										1
AWS								1							1
Ansible											1				1

3. Case study

Based on the results of Table 2, a graph has been built, shown in Figure 2.

To clarify the results, the list of PLs studied has been limited to the most significant eight, which is displayed in the following graph (Fig. 3).

Figure 3 shows that the most popular PLs are the following five: Python, JavaScript, Java, C ++ and Perl. As for the Perl language, its appearance in this list is due to the importance of this language for performing work on system administration. This may indicate a possible need for optional acquaintance of students of a number of specialties with the possibilities and peculiarities of this language use.

4. Conclusions

Based on the results obtained above, it can be concluded that, apparently, it is necessary to carry out targeted actions to create a new methodology for the PL comprehensive study, taking into account the **increasing** relationship of the key disciplines of AP, ASD and OOP at all stages of their development.

At the same time, the fact should be taken into account that there is the broadest definition: if curly braces {} are used in the language to highlight functional blocks, then this is a C-style language. Many, but not all, C-style languages are strongly typed. In addition, some of the more popular C-style languages are object-oriented (e.g. C ++, Java, C#), but the C itself is not. The concepts of the formation of many other programming languages, such as Python, Perl, PHP and Ruby, the C language style effects are also present [23]. JavaScript has syntax similar to C, although the essence of the language itself is significantly different [24]. At the same time, the source [25] shows that Java, C++ and Perl are also C-like languages. Therefore, it has been proposed to start AP course studying with the C++ language, using its procedural paradigm. It is known that OOP greatly increases the amount of code, and linking of the object-oriented design with programming targets can significantly increase the time for developing and refining programs [26]. And while reusability is a big advantage of object-oriented languages, there is also the fact that OOP languages have extremely poor modularity in terms of extending and modifying classes. For example, it is easy to override a method that should not be overridden, or override a class in a way that causes problems in subclasses. Moreover, it should be noted that "OOP fails at the only task it was intended to address," argued Senior full-stack engineer Ilya Suzdalnitski, who published a 6,000-word essay calling object-oriented programming "a trillion dollar disaster." [27]. After writing a subsequent essay [28], Ilya Suzdalnitski found out that the two articles taken together have been read about half a million times in about a month!

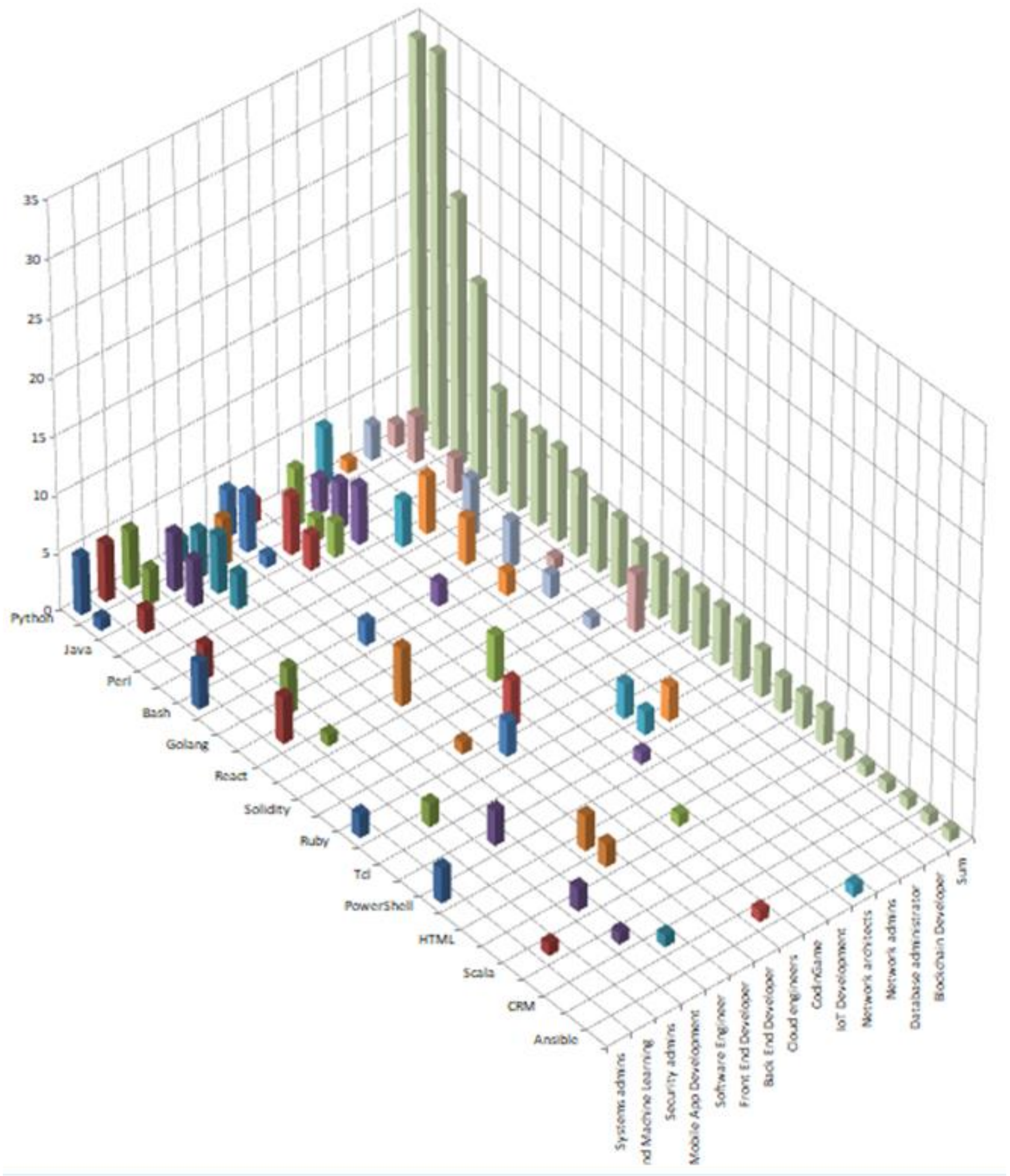

Fig. 2. Distribution of programming languages by type of work/jobs

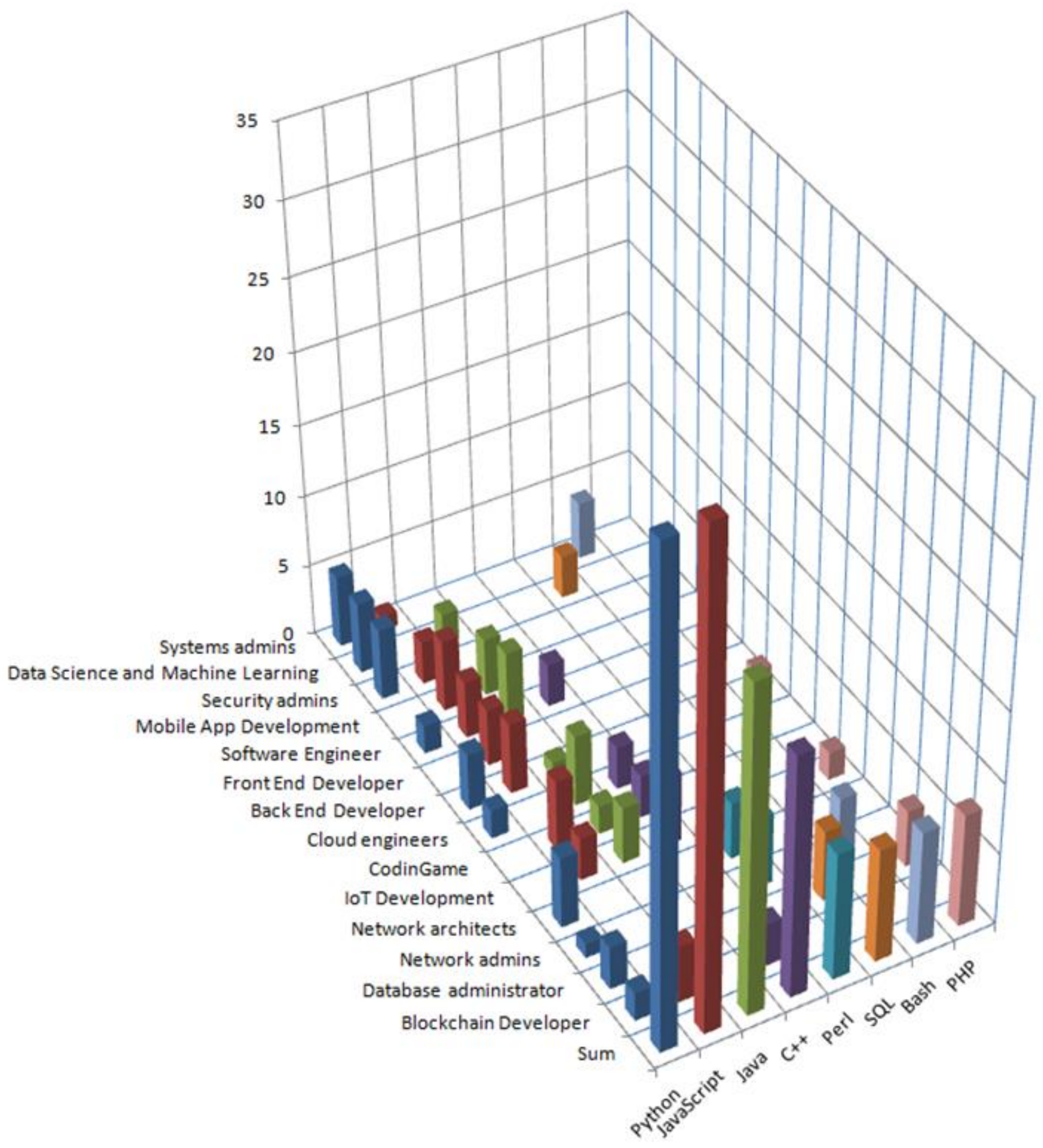

Fig. 3. Significance indicators for the eight PLs under study

Thus, focusing on the procedural paradigm in the AP course, it has been proposed to use a ***unified approach*** for different languages with an emphasis on ***conceptual notions.*** According to the authors, the main focus should be on the concept of built-in (base) and aggregate data types (for example: arrays, structures, unions, etc. in C++).

The next point in this approach is the emphasis on the structured programming based on the application of the Boehm-Jacopini Theorem [29]. According to Boehm and Jacopini, building the *logical structure* of a program requires three basic blocks, having one entrance and one exit:

– non-control *SEQUENCE* structure, including a linear sequence of instructions (including control operators) (Fig. 4)

Technium Sustainability
Vol. 1, Issue 1, pp.21-33 (2021)
ISSN: 2810-2991
www.techniumscience.com

– control conditional *SELECTION* structure (construction of making a binary or dichotomous decision*)* (Fig. 5);

– *ITERATION* control conditional structure with a precondition and a postcondition, etc. (Fig. 6).

Then, for any program composed of the indicated constructions, a proof of its correctness can always be obtained!

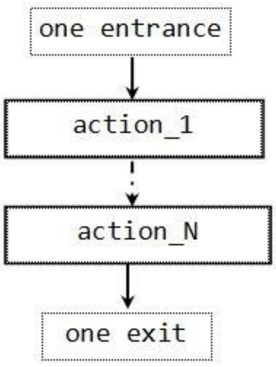

Fig. 4. Non-control *SEQUENCE* structure

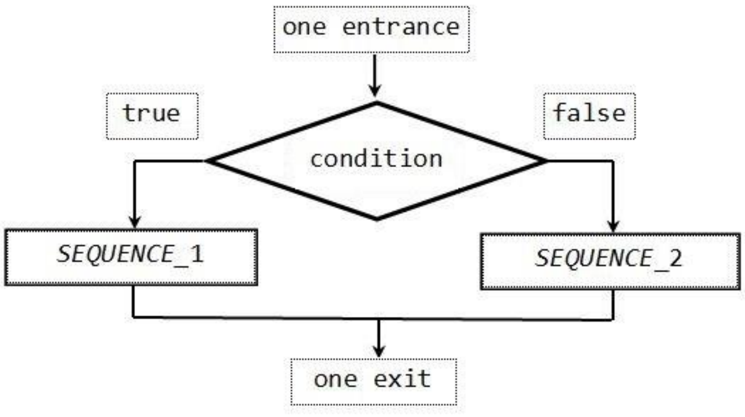

Fig. 5. Control conditional *SELECTION* structure

Moreover, all considered statements have one input and one output! In addition, each PL has conditional and looping statements implemented in their own way; but the approach to their study should be *the same*. It is proposed, at a later stage, to call them *control statements*, since they control the course of the program execution.

Thus, C++ includes three types of conditional control statements. The `if` statement performs (selects) an action (or a group of actions) if the condition is `true`, or skips it if the condition is `false`. The `if…else` statement performs an action (or a group of actions) if the condition is `true`, and performs another action (or a group of actions) if the condition is `false`. A `switch` statement performs one of many different actions (or a group of actions) depending on the value of the expression in it.

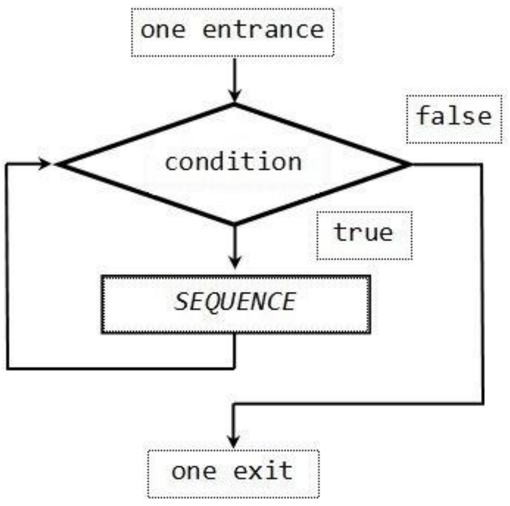

Fig. 5. *ITERATION* control conditional structure

C ++ also has three loop control statements (sometimes called repetition statements or iteration statements) that allow programs to repeat statements as long as a condition (called a loop continuation condition) is true. These include the control statements `while`, `do...while`, and `for`.

The `while` and `for` statements perform an action (or a group of actions) in their bodies zero or more times – if the loop-continuation condition is initially `false` or `true`. The `do…while` statement performs at least one action (or group of actions) in its body [30].

Thus, the student should clearly understand that the program is designed and constructed from the *SEQUENCE* and control structures that have one input and one output.

The principles of *top-down program development (design) by the method of stepwise refinement* should be used when studying modular programming (in C++, these are functions that either return or do not return a value).

The top-down program development (design) involves splitting a large problem into smaller subtasks, which can be considered separately and, in turn, broken up into smaller subtasks until all the features of the main problem being solved are clarified.

Niklaus Wirth was the first to introduce the principle of *Stepwise-Refinement* as a key to systematic program design; he published a detailed article on the use of this principle in the Communications of ACM Journal in 1971 [31].

The main idea of the top-down design by the method of Stepwise-Refinement has been explained clearly by Edward Yourdon [32]. Let's consider an example of any (arbitrary) program, conventionally named GLOP. First, let's try to imagine having a computer that can immediately implement this program. Then the only command has to be written:

```
GLOP
```

and it's done! Unfortunately, so far there are no such machines that would perform any actions at your request. Therefore, we'll have to break down the task, and therefore the program, into smaller elements. In most cases, the execution of the program implies the input of the necessary data, their processing and the output of the obtained results. And here

Edward Yourdon is missing an important point, since, before program designing, data structures have to be designed thereof! It is therefore necessary to rewrite the GLOP program as follows:

```
Design data structures!
Enter GLOP data
Perform GLOP calculations
Display GLOP results
```

In doing so, each line of our program can again be split into simpler operations, until they can be represented in the statements of a programming language at a disposal (in this case, it is C++). Moreover, when designing new modules, it is necessary to carefully transfer the corresponding fragments of data structures designed at the initial stage.

When using Structured Programming Technology:
– Application programs are easier to read and understand.
– Application programs are less likely to contain logic errors.
– Errors are more easily found.
– Higher productivity exists during application program development.
– Improved application program design is obtained.
– Application programs are more easily maintained. [33].

Since the teaching experience of the authors has not been sufficiently informative describing pseudocode in lecture materials, it is proposed to focus on programs with detailed comments. At the forefront, it is necessary to put students' mastering of approaches and skills in designing software units based on *reading*, *studying* and *understanding* someone else's code - the most important component of a team software development.

To do this, each section of the laboratory practice needs to be provided with programs containing detailed comments on each line. It is obligatory to mark each control statement with indents in front and behind, as well as to put a comment about its purpose immediately before it. The structure and the format of comments in computer programs offered for students teaching should be based on well-known standards, for example, "Google C++ Style Guide" (Google) [34]. In addition, the authors propose to use the so-called "CamelCase" from the Google standard. It is necessary to exclude some points related to the application of OOP due to the procedural paradigm used for the first-year students.

Based on the experience of teaching this course and taking into account the interdisciplinarity of its components, it has been proposed to consider the AP, ASD and OOP courses as a continuous chain. In the AP course, it is necessary to immediately focus on students' understanding of operations with embedded data of various types: integers, real, logical, symbolic, pointers, etc. It has been proposed that the students further should study aggregate data, including dynamic data. Also, the study of search and sorting methods, stacks and lists, as well as methods for their processing shall be transferred from the AP to the ASD course; and recursion (recursive function) shall be mastered when considering functions.

The concept of creating a graphical user interface should be developed on the use of the Qt framework - a set of widget tools for creating graphical user interfaces, as well as cross-platform applications that can run on various software and hardware platforms, such as Linux, Windows, macOS, Android. At the same time, an attention should be paid to embedded systems, which, with little or no changes to the code base, remain native applications with their own capabilities and execution speed.

It is also proposed to study HTML and CSS in addition to the specified PLs (Python, JavaScript, Java, C++ and Perl).

References

[1] The influence of Multi Platform Space on the formation of a programming languages stack in the competence-based approach to Computing training at Universities / G. Korotenko, L. Korotenko / International Journal of Innovative Science and Research Technology (IJISRT). Volume 3. – November. – Issue 11, 2018. – 6 p.
[2] Computing Curricula 2020. CC2020. Paradigms for Global Computing Education. Encompassing undergraduate programs in Computer Engineering, Computer Science, Cybersecurity, Information Systems, Information Technology, Software Engineering with data science. Association for Computing Machinery (ACM) and IEEE Computer Society (IEEE-CS). 2020 December 31. – 205 p.
[3] Why You Should Change Your Developer Job Every 4 Years, available at https://www.codemotion.com/magazine/dev-hub/cto/change-developer-job/ (accessed 11 October 2021).
[4] How Often Can a Software Engineer Switch Jobs?, available at http://codingsupply.com/how-often-can-a-software-engineer-switch-jobs/ (accessed 11 October 2021).
[5] Changing job - How often is too often?, available at https://dev.to/lankydandev/changing-job---how-often-is-too-often-4ph2 (accessed 11 October 2021).
[6] How often should you change jobs?, available at https://www.theladders.com/career-advice/often-changing-jobs (accessed 11 October 2021).
[7] Затверджені стандарти вищої освіти. МОН України. – Available at: https://mon.gov.ua/ua/osvita/visha-osvita/naukovo-metodichna-rada-ministerstva-osviti-i-nauki-ukrayini/zatverdzheni-standarti-vishoyi-osviti (accessed 11 October 2021).
[8] 50 Best Jobs in America for 2021 / https://www.glassdoor.com/List/Best-Jobs-in-America-LST_KQ0,20.htm (accessed 11 October 2021).
[9] Top 5 programming languages systems admins should learn, available at https://www.techrepublic.com/article/top-5-programming-languages-for-systems-admins-to-learn/ (accessed 11 October 2021).
[10] Top 5 Programming languages for Data Science and Machine Learning in 2021, available at https://medium.com/javarevisited/top-5-programming-language-for-data-science-and-machine-learning-badc2f8eff72 (accessed 11 October 2021).
[11] The Best Programming Languages for Cybersecurity in 2021, available at https://flatironschool.com/blog/best-programming-languages-cyber-security (accessed 11 October 2021).
[12] Top 5 Programming languages for Mobile App Development in 2021, available at https://medium.com/javarevisited/top-5-programming-languages-for-mobile-app-development-in-2021-19a1778195b8 (accessed 11 October 2021).
[13] 5 programming languages application solutions developers should learn, available at https://www.techrepublic.com/article/5-languages-for-application-solutions-developers-to-learn/ (accessed 11 October 2021).

[14] Top 10 Frontend Languages available at https://blog.back4app.com/front-end-languages/ (accessed 11 October 2021).
[15] The Best Ten Backend Languages available at https://blog.back4app.com/best-backend-language/ (accessed 11 October 2021).
[16] 5 programming languages cloud engineers should learn, available at https://www.techrepublic.com/article/5-best-languages-for-cloud-engineers-to-learn/ (accessed 11 October 2021).
[17] Best Programming Languages for Game Development available at https://gamedevacademy.org/best-game-development-languages/ (accessed 11 October 2021).
[18] 5 Best programming languages you should learn for IoT Development, available at https://content.techgig.com/5-best-programming-languages-you-should-learn-for-iot-development/articleshow/81533527.cms (accessed 11 October 2021).
[19] 5 programming languages network architects should learn, available at https://www.techrepublic.com/article/5-programming-languages-network-architects-should-learn/ (accessed 11 October 2021).
[20] 5 Programming languages that network admins need to learn, available at https://content.techgig.com/5-programming-languages-that-network-admins-need-to-learn/articleshow/77179923.cms (accessed 11 October 2021).
[21] Top 5 programming languages that every database administrator must learn, available at https://content.techgig.com/top-5-programming-languages-that-every-database-administrator-must-learn/articleshow/79627842.cms, (accessed 11 October 2021).
[22] Top Programming Languages For Blockchain Development, available at https://analyticsindiamag.com/top-programming-languages-for-blockchain-development/ (accessed 11 October 2021).
[23] The Influence of C in Python, available at https://realpython.com/lessons/influence-of-c/ (accessed 11 October 2021).
[24] Learn a C-style language. Improve your odds with the lingua franca of computing, available at https://www.oreilly.com/content/learn-a-c-style-language/ (accessed 11 October 2021).
[25] List of C-family programming languages, available at https://en.wikipedia.org/wiki/List_of_C-family_programming_languages (accessed 11 October 2021).
[26] Comparing Programming Paradigms: Procedural Programming vs Object-oriented Programming, available at https://www.codementor.io/learn-programming/comparing-programming-paradigms-procedural-programming-vs-object-oriented-programming (accessed 11 October 2021).
[27] Object-Oriented Programming – The Trillion Dollar Disaster. Why it's time to move on from OOP, available at https://betterprogramming.pub/object-oriented-programming-the-trillion-dollar-disaster-92a4b666c7c7 (accessed 11 October 2021).
[28] Object-Oriented Programming is The Biggest Mistake of Computer Science, available at https://suzdalnitski.medium.com/oop-will-make-you-suffer-846d072b4dce (accessed 11 October 2021).
[29] Böhm C., Jacopini G. Flow Diagrams, Turing Machines and Languages with Only Two Formulation Rules. Communications of the ACM, Volume 9, Issue 5, May 1966, pp 366–371, available at https://doi.org/10.1145/355592.365646 (accessed 11 October 2021).

Technium Sustainability
Vol. 1, Issue 1, pp.21-33 (2021)
ISSN: 2810-2991
www.techniumscience.com

[30] Deitel P., Deitel H. C++11 for Programmers. 2nd Edition. – Prentice Hall, 2013. – 1280 p.
[31] Niklaus Wirth. Program development by stepwise refinement / Communications of the ACM. Volume 14. Issue 4. April 1971. pp 221–227. https://doi.org/10.1145/362575.362577.
[32] Yourdon E. Techniques of Program Structure and Design. – Prentice Hall, 1975. – 409 p.
[33] Advantages of Structured Programming Macros, available at https://www.ibm.com/docs/en/tpfdf/1.1.3?topic=introduction-advantages-structured-programming-macros (accessed 11 October 2021).
[34] Google C++ Style Guide, available at https://google.github.io/styleguide/cppguide.html (accessed 11 October 2021).