Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

A Comparative Study of Computer Programming Challenges of Computing and Non-Computing First-Year Students

Alain Kabo Mbiada[1], **Bassey Isong**[2], **Francis Lugayizi**[3]
mbidaalain@gmail.com, bassey.isong@nwu.ac.za. francis.lugayizi@nwu.ac.za.
Department of Computer Science, North-West University, Mafikeng, South Africa

Article Information

Submitted : 3 Aug 2023
Reviewed: 11 Aug 2023
Accepted : 27 Aug 2023

Keywords

IP, Programming concepts, Programming challenges, Computing/Non-computing students, Learning.

Abstract

The learning of computer programming comes with unique difficulties that vary among students depending on their backgrounds, learning methods, and objectives. This paper investigates the programming challenges first-year students from non-computing at the North-West University, South Africa, and computing backgrounds at the University of Dschang, Cameroon face. A questionnaire-based data collection method is utilized and categorises participants based on their gender, age, fields of study, prior experiences in mathematics, statistics, English, and programming languages, lab use/access, learning strategies, and material preferences. The aim is to identify and analyse the student's understanding of the basic programming concepts and the specific challenges met during introductory programming modules. Analysis of the collected data shows that while a considerable percentage of non-computing students have prior experience in mathematics and English, they lack familiarity with programming. Equally, while most computing students are proficient in spoken English, they face significant challenges in programming, mathematics, and written English. Notable difficulties are experienced in grasping concepts like recursion, arrays, error handling, and function/procedure methods. Moreover, a comparative study reveals that both groups of students encounter similar challenges, however, non-computing students' difficulties are more than their computing counterparts. This paper, therefore, suggests designing teaching methods and learning materials to specifically meet the needs of non-computer science students, and enhance their understanding and proficiency in computer programming.

Attribution-ShareAlike 4.0 International License

A. Introduction

Computing programming is a core element of any introductory Computer Science (CS) module due to its effect on the precision of thought, thorough presentation, and concern for details. It promotes the development of computational thinking and problem-solving, encouraging students to explore, learn and reflect. However, a vast body of literature in the field of CS education emphasizes that novices find programming particularly difficult, leading to high percentages of failure and dropout among them [1]–[3]. This challenge is due to several factors such as the individual's background, learning style, goals, instruction complexity, diversity of languages, lack of resources, logical understanding difficulty, outcomes uncertainty, etc. To address this challenge, several researchers and academics have carried out numerous studies to identify the difficulties novice students face in introductory programming (IP) courses ranging from efficient teaching methods to learning material tools that could help novices learn better [1], [4]–[7]. However, most of these studies have focused on undergraduate CS students and unfortunately, only a fewer of them have paid attention to non-computing students from other study fields who enrolled for IP as an elective module. These sets of students are the worst affected by the challenge posed by programming difficulties. Thus, there is an urgent need to also research the various challenges they face while offering programming modules as electives. Moreover, it is also of paramount importance that these studies are also performed on different institutions or nations to identify if this challenge is a general one or unique to a particular institution or nation.

Therefore, this paper comprehensively surveyed to identify the challenges that still exist, and students faced despite the several interventions put in place to enhance the learning of programming in IP modules. This was achieved by designing a quantitative survey via a questionnaire that targeted first-year students in the North-West University (NWU)'s Faculty of Natural and Agricultural Sciences (FNAS), South Africa, and the first-year computing students in the Department of CS, University of Dschang (UDS), Cameroon. We targeted the NWU students who are not pure computing students but double major students offering CS modules alongside other disciplines such as Agricultural, Geography, Physics, Mathematics, Chemistry, Biology, etc. The objective of this study was to identify the students' background profiles, teaching and learning conditions, learning material preferences, and the challenges faced in their IP modules. in addition, it was designed to identify if the programming challenge is institution-based, or a general challenge faced by computing and non-computing students. The questionnaires were distributed to participants, and data was collected and analyzed. The analysis of the results and comparative study of these two groups on programming challenges led to several recommendations for the design of an effective teaching method and learning environment for non-computing students.

The paper is structured as follows: Section 1 presents the background information, Section 2 presents some related work, Section 3 describes the survey methodology, Section 4 presents the survey results and the analysis, Section 5 presents the comparative analysis, Section 6 discusses the results and provides

some recommendations, Section 7 discusses the validity threats, and Section 8 concludes the paper.

B. Lterature Review

1. Computer programming

Computer programming is a process of creating computer-readable and executable instructions to solve a problem. It incorporates a set of skillful, logical, and critical thinking activities that range from analysis, and design, to maintaining the program [8]. Creating the instructions involves using languages such as Java, C, C++, Python, PHP, JavaScript, etc. on tools such as text editors, integrated development environment (IDE), and so on. Computing programming which plays a critical role in the advancement of technology, has a wide range of applications, and its skill has become a core competence of any computing and related computing students as it allows them to develop software solutions and create innovative technologies. With the increasing demand for technology in various industries, the ability to code has become an essential skill. However, its activities are complex, requiring logical thinking, problem-solving skills, and attention to detail. Thus, understanding programming logic has been considered difficult and often poses a source of frustration to first-year students, sometimes leading to a lack of motivation for learning and failure and drop-out. The challenges in grasping IP modules may have various origins [9]–[12] such as unsuitable teaching strategies, lack of attention to novices' initial backgrounds, use of expert-level content explanation, instructor skill in the subject, learners being busy with other courses, students are used to memorizing instead of understanding, large class size, difficulty in combining programming concepts to solve programming problems, mix-up of native languages and programming syntaxes, use of line-by-line methods for solutions, short module duration, absence of computer mental model for novices, etc. Though programming is hard, it is not impossible to learn, and being aware of the sources of the challenge is a *sine qua non* to designing and implementing a good programming teaching and learning strategies.

2. Related works

This section presents some of the studies conducted over the years to identify the difficulties encountered by beginners in IP modules. Lahtinen et al. [1] surveyed to identify the difficulties encountered and perceived by students and teachers during object-oriented (OO) or imperative programming courses. They found that students needed help with program construction and error correction. Recursion, arrays, pointers, error handling, and the use of libraries were found to be the most difficult basic programming concepts experienced by students. Mainly, working with C++, an OO programming language, Milne and Rowe [4] surveyed second-year students and instructors on individual language concepts they struggle to learn and, consequently, to teach, and found that students have poor mental model representation of what happens during program execution. In the same vein, Thomasson et al. [5] studied the difficulties beginners faced in designing an OO class. They identified two common errors, namely unreferenced classes and problems linked to class attributes and cohesion. In addition, Tan et al.

[6] surveyed undergraduate students enrolled in a computing programming course at a Malaysian university, to identify the challenges they faced in learning the programming language and their perceptions of how the learning should proceed naturally. Furthermore, Piteira et Costa [7] identified the basic concepts of IP modules and organized them from a low to a high level of students' understanding. The concepts with a low comprehension level are arrays, structured data types, recursion, pointers and references, while concepts with a high level of comprehension are selection structures, variables, loop structures, operators and precedence.

The above highlighted some of the studies conducted to elicit programming difficulties in students. While these studies in [1],[4-7] focus on computing students and specific institutions, our study is designed to identify the challenges both computing and computing students face and across institutions.

C. Methodology

This section presents the methdodlogy and research design of this study.

Research design and participants: In this paper, we conducted a comprehensive survey based on a quantitative research design methodology using a structured questionnaire. The study was intended to answer the following research questions (RQs) which is to identify the challenges students face in computing programming and how unique the challenges are across institutions. The RQs include:

> *RQ1: What are the attributes possessed by computing and non-computing students offering IP modules and what challenges do they face?*
>
> *RQ2: How are these challenges affect computing and non-computing students?*

The target population included first-year students of FNAS from NWU, South Africa, as non-computer science students (Group 1) which comprise two campuses: Mafikeng and Potchefstroom, and the first-year CS students from UDS, Cameroon, (Group2). The study adopted a cross-sectional design method which led to a sample size of 260 respondents where 81% are from NWU and 19% from UDS.

Data collection and analysis: For logistical reasons, we conducted both online and face-to-face surveys, and the questionnaires used for data collection were distributed to the participants: NWU Mafikeng campus (face-to-face), NWU Potchefstroom campus (online), and UDS (online). The questionnaire was designed with six sections, each having a set of structured questions which are short and precise to reduce the risk of misunderstanding. Section A identifies important information about students' backgrounds, Section B assesses students' learning situation preferences in programming modules, Section C discusses students' preferences for learning materials, Section D investigates students' knowledge of core programming concepts, Section E examines students' general perception of programming issues, and Section F looks at the different programming concepts taught at the beginning of learning programming. The study used an exhaustive approach for building responses to questions and the questions required participants to give answers based on the following ordinal scale values: {Poor (1), Insufficient (2), Average (3), Good (4), Very good (5)}.

Moreover, the data collected were analyzed using descriptive statistics on the SPSS software tool, and the results were presented. The results of sections A, B, and C of the questionnaire will answer RQ1, while the analysis of the data obtained in sections D, E, and F will answer RQ2. The IP modules of interest were C and Python programming languages modules.

Ethical considerations: Before data collection from NWU students, the ethics application was submitted and approved, and a certificate of ethics was issued by the FNAS ethics committee. This signifies the study's commencement and contrast; no requirements were made before data collection from NWU students.

D. Result and Analysis

This section presents the results of research and their analysis.

1. Participant characteristics

Background profile: This paper studies the general background of both groups' participants relative to gender, prior programming, maths and English experiences, lab accessibility and usage, and course duration, which are important success factors in IP courses [13]–[22]. The knowledge of these features can be helpful in the design of a novel teaching method and learning tools that meet the needs of these students. The surveyed population includes both males (72.3%) and females (27.3%), largely aged between 18 and 20 years at 57.7%. Over 52% of participants felt that the time allocated to the IP modules was insufficient, and 67% have no prior programming experience. For Group 1 specifically, Fig. 1. depicts their distribution into respective fields. The analysis reveals that most Group 1 respondents (52.2%) expressed at least an average level in mathematics, while over 50% from Group 2 had at most an insufficient level in mathematics as shown in Fig. 2. The information demonstrates that students offering these IP modules were young and tend to have very diverse experience levels, making it cumbersome to plan instruction in a way that is engaging and useful for all learners.

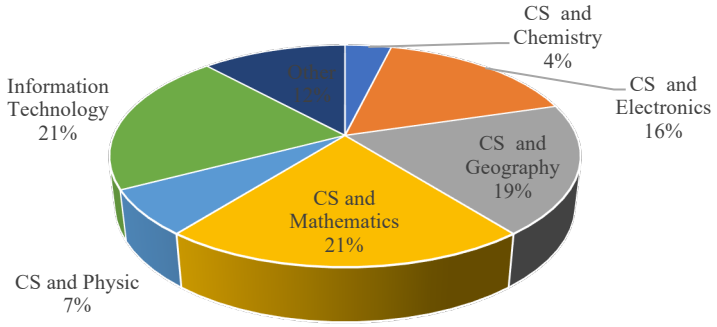


Figure1. Distribution of NWU participants by field of study

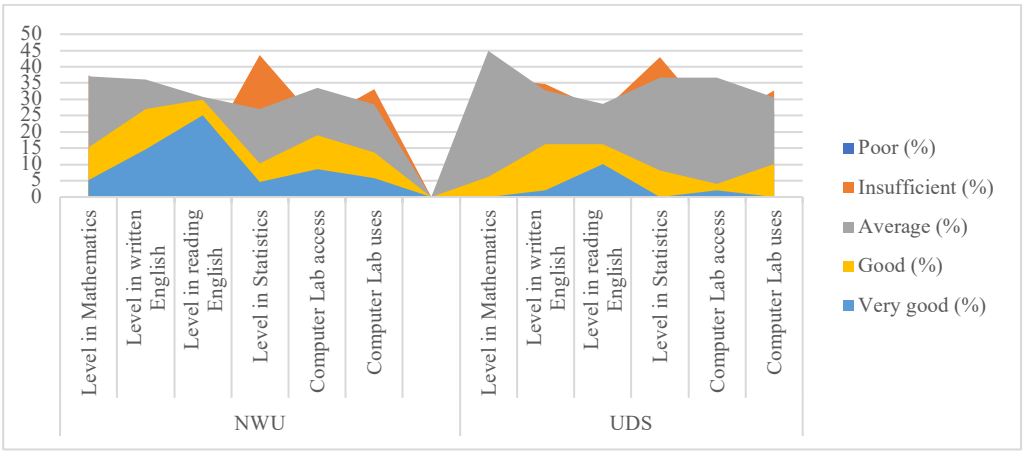


Figure 2. Participants backgound information

Learning situations preferences: The analysis of results in Fig. 3 shows that most of the Group 1 respondents learn less during lecture sessions and are not self-confident when it comes to studying alone and working alone on programming assignments. They enjoyed the other learning situations. Similarly, Group 2 respondents learned also less during lecture sessions, however, they are self-confident to learn alone and enjoy other learning situations. Some studies [1], [6] also identified lecture sessions as activities that decrease students' commitment. This highlights the issue of teaching methods that interest and engage Group 1 respondents in their learning activities and develop self-confidence.

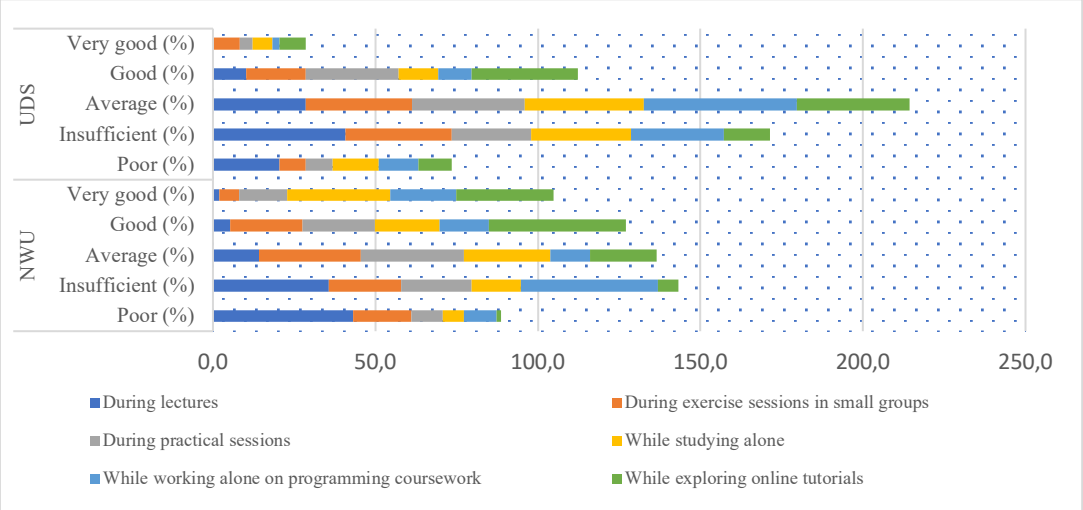


Figure 3. Students' situational learning preferences

Learning materials preferences: Analysis of results in Fig. 4 reveals that the majority of respondents find programming textbooks, lecture notes, and PowerPoint slides less useful than other learning materials. Programming textbooks have also been highlighted in other studies such as [1], [23] as being less useful to students. Thus, teaching methods and learning tools should then be more innovative in terms of visualization and interaction, enabling students to easily understand programming texts and manuals used in or out of the classroom.

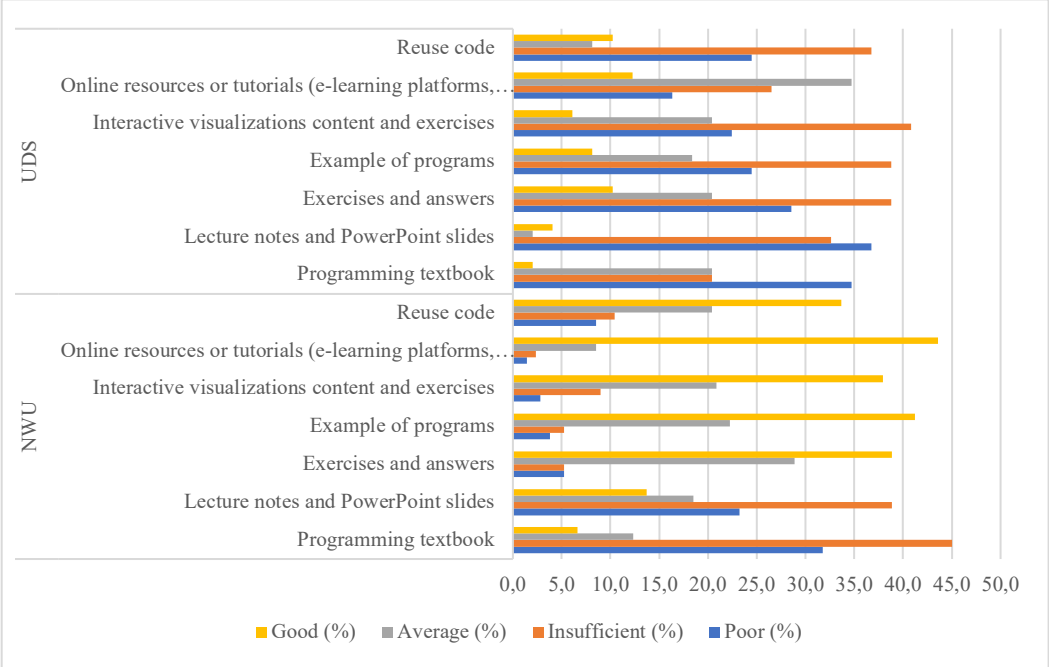


Figure 4. Students' learning materials preferences

2. Difficulties faced in learning IP modules

Based on the IP modules considered in this study, C and Python programming languages, the analysis of the results in Fig. 5 illustrates that most issues of this part were rated at most "*Insufficient*" by the respondents. They, therefore, faced the same difficulties as most programming beginners. Other studies came out also with similar results [1], [6], [23]. A well-designed teaching strategy and learning tools are then needed to help these students grasp most of these programming issues.

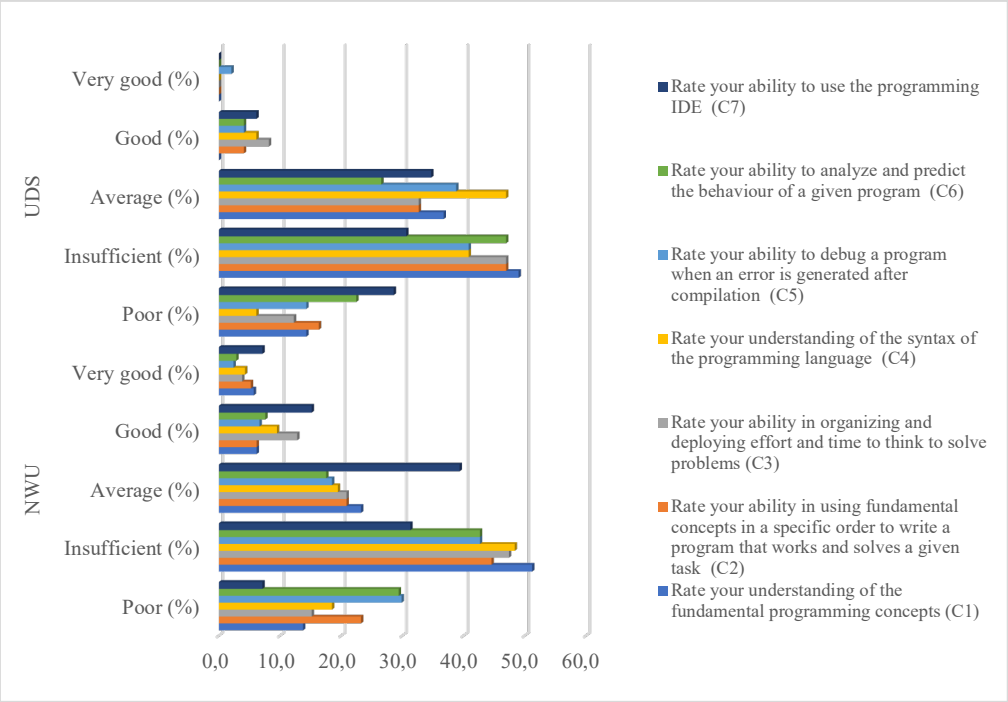
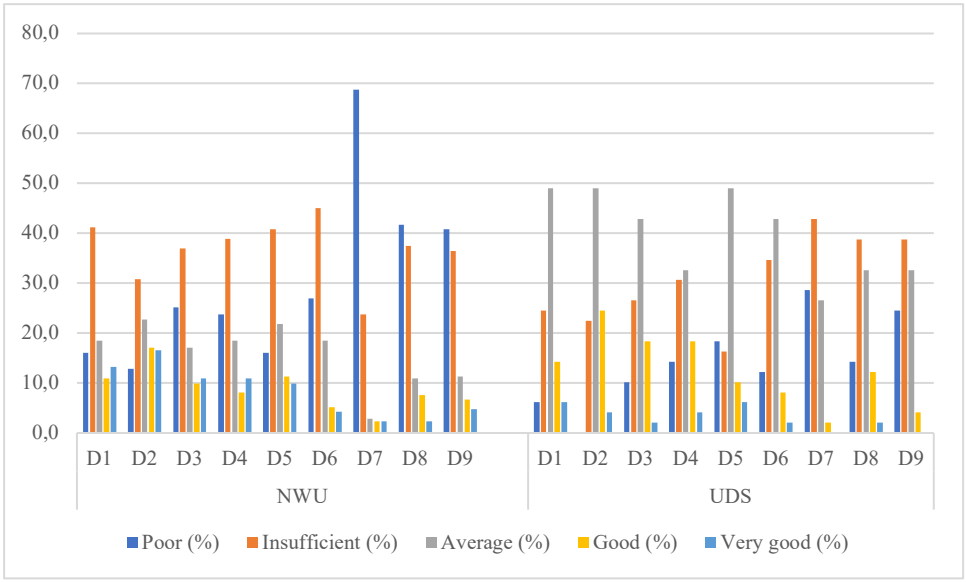


Figure 5. Students' geeneral perception of IP modules

On the other hand, the analysis of results in Fig. 6 in F depicts that the understanding of core programming concepts in most cases is at most rated "*Insufficient*" for most Group 1 respondents. The comparison operator concept was the only one rated as "*Average*", with 66% for this group. Respondents of Group 2 rated most of the concepts at least "average". The only concepts rated as "insufficient" at most by this group of respondents were recursion and error handling. The level of appreciation of the basic programming topics is relatively weak for Group 1 respondents and the difficulties observed are similar to those encountered by most beginners. When going into detail, the following programming concepts appear to be more difficult for both groups: recursion, arrays, and error handling. Some studies also presented recursion and error handling as challenging concepts in programming [1], [6], [7], [23]. These results detail most of the topics that make learning programming difficult for these students who need appropriate strategies during class by the instructor to improve their learning.

Furthermore, participants were also asked to select the concepts that were discussed in class when they were first learning to program. The analysis of the corresponding results in Fig. 6 reveals that some respondents could not remember certain core programming concepts which might highlight their lack of interest or commitment to the module since there is no way for example, that a programming course could be taught without variables being introduced.



D1: Variables; D2: Comparison operator; D3: Loop structures; D4: Sequential and conditional structure; D5: Input/output instructions; D6: Functions and procedures or Methods; D7: Recursion; D8: Arrays; D9: Error handling

Figure 6. Students' understanding of some core programming concepts

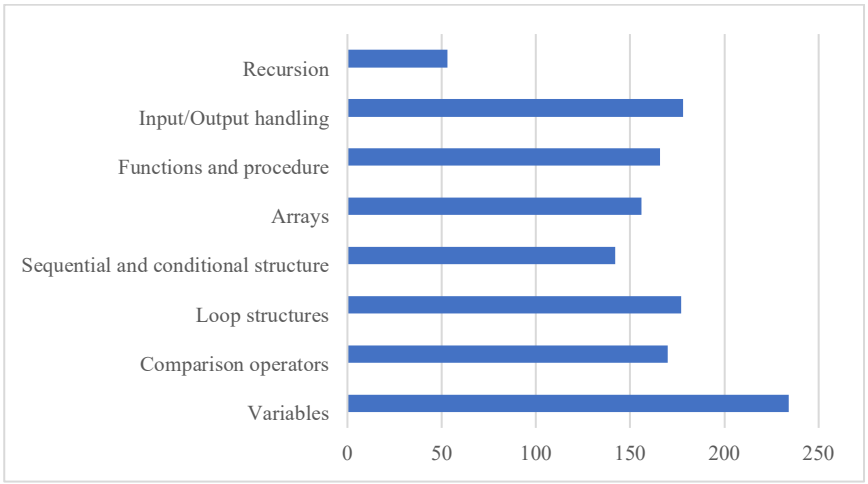


Figure 7. Frequency of selection of some core programming concepts

3. Comparative analysis

To determine the level of effect the challenges have on both non-computing and computing students across NWU and UDS and answer RQ2, this paper performed a comparative study of the data collected on the two groups and analysed the results. The data consists of the difficulty levels of respondents in both groups on the variables D1, D2, D3, D4, D5, D6, D7, D8, and D9 presented in Fig. 6.

To determine the level of significance of the challenges, we formulated and tested hypotheses based on the understanding of each programming concept as shown in Fig. 6. The null hypotheses used in assessing the significance of the equality of respondents' difficulties levels in the two groups (i.e. the difficulty level

on any concept is the same for both non-computing and computing students) are as follows respectively:

H_{01}: $\mu_{D1} = 0$, H_{02}: $\mu_{D2} = 0$, H_{03}: $\mu_{D3} = 0$, H_{04}: $\mu_{D4} = 0$, H_{05}: $\mu_{D5} = 0$, H_{06}: $\mu_{D6} = 0$, H_{07}: $\mu_{D7} = 0$, H_{08}: $\mu_{D8} = 0$, *and* H_{09}: $\mu_{D9} = 0$.

Furthermore, the formulated hypotheses were tested on the SPSS software tool using the Mann-Whitney U-test technique for the two independent groups. Due to the percentages involved in the groups, we randomly selected 49 respondents from Group 1 to match the number in Group 2. The variables considered showed a non-normal distribution, which was confirmed by the Shapiro-Wilk Test and applied $\alpha = 0.05$ significance level where p-value, $\rho > 0.05$ will be accepted.

Table 1. Hypotheses testing results

	D1	D2	D3	D4	D5	D6	D7	D8	D9
Mann-Whitney U	772.500	965.000	661.500	846.500	879.000	804.000	611.000	543.000	852.000
Z-index	-3.198	-1.747	-4.024	-2.640	-2.398	-3.017	-4.599	-4.924	-2.639
P-value Asymptotic. Sig.	0.001	0.081	0.000	0.008	0.016	0.003	0.000	0.000	0.008

Table 1 presents the results of the Mann-Whitney U-test and the analysis of these results shows that for the variable D1, the H_{01} is rejected, as $\rho=0.001 < 0.05$. This means that there is a significant difference between both groups. The CS students at UDS (Group 2) expressed fewer difficulties than non-CS students at NWU (Group 1), as Group 1 has a Mean Rank of $T_1=44.77$ and Group 2 has a Mean Rank of $T_2=58.23$. For the comparison operators variable (D2), H_{02} is accepted since $\rho=0.082 > 0.05$, indicating that the level of difficulties is approximately the same for both groups. For the rest of the variables, we achieved $\rho < 0.05$ in each which signifies the rejection of the following H_{03}, H_{04}, H_{05}, H_{06}, H_{07}, H_{08}, and H_{09}. Thus, the level of difficulty of Group 1 respondents is higher than that of Group 2 respondents in terms of the following basic programming concepts: loop structures, sequential and conditional structures, input/output instructions, functions and procedures or methods, recursion, arrays and error handling, respectively. These results are perfectly in line with those presented above and further confirm that computing students encounter fewer difficulties than non-computer science students in IP modules.

E. Discussion

The survey results from the background profile confirmed that NWU respondents are from different fields of study and correspond to non-computing students. On the other hand, the mathematics and English experiences they expressed did not guarantee their performance in the module, given the difficulties expressed and identified, which are in contradiction with some studies detailed here [13], [15], [24] that have confirmed these elements as success factors in programming courses. We suspect they overestimated their level in mathematics. They may also, like most beginners, be confused between the English language and the syntax of the programming language [11]. By contrast, the lack of prior

programming experience expressed in the majority of cases is consistent with the fact that they are struggling and confirms some studies detailed here [16], [17]. The core programming concepts are a challenge for most of these respondents as for most other beginners as well. Most challenges reported in this study are recursion, arrays, and error handling and were also confirmed in the previous studies [1], [6], [23]. These difficulties were also observed among Group 2 respondents, most of whom had no prior programming experience, and the majority of whom expressed an "*Insufficient*" level of mathematics, statistics, and written English, as well as insufficient access to and use of laboratories. However, using the Mann-Whitney U-test technique to compare these two groups, we observed that Group 2 respondents expressed fewer difficulties with basic programming concepts than the non-computing students. Note that the most difficult basic programming concepts remain the same as those expressed by Group 1 participants. This means that these concepts require special attention from the instructor. Therefore, non-computing students cannot be assimilated into CS students in terms of teaching approach and learning materials tools.

Thus, this study assumes that they need specific teaching approaches and learning materials to improve their motivation, engagement, and performance in IP courses. Elsewhere, these challenges observed can be explained by the insufficient time devoted to IP modules, as pointed out, and by the respondents' lack of motivation and engagement. Indeed, most respondents reported being unable to well organize and deploy effort and time to think to solve programming problems (see Fig. 6), and some of them did not be able to recall certain core programming concepts. Nevertheless, all students of IP courses must be aware of them and should master these concepts. They should always be taught, regardless of the programming paradigm engaged, whether it is imperative or object-oriented [7]. Moreover, the results of the survey on learning situations and learning material preferences revealed participants' disregard for lecture sessions on the one hand, and for programming manuals and PowerPoint slides on the other, as also noted in other studies detailed here [1], [23]. The teaching strategies could be an issue as well as students' motivation and engagement. A novel teaching strategy must therefore, take into consideration respondents' interest in interaction expressed through practical sessions and collaborative programming which can have a good impact on students' performance in programming as confirmed in some studies [25]–[28].

Again, NWU respondents are not self-confident when it comes to studying alone compared to UDS participants. Still, the teachers will provide them with appropriate learning material tools such as online tutorials with interactive content and assignments that correspond to their background profile as well as exercises and answers, reuse code, and examples of programs. The new teaching method will be designed to give students time to work on their own during the course and to develop their self-efficacy and self-confidence. The learning material tools must also be easily integrated into the teaching practices for direct instruction and to enable students to discover concepts, uncover knowledge, and put skills into practice. Moreover, for these students, we recommend a cognitive teaching approach, rather than the constructivism-based approach that has inspired many IP teaching methods [29], as most of them have no prior

programming experience, need quality guidance from their instructors, and lack in-depth, well-organized necessary knowledge to implement effective problem-solving processes. The teaching method must then prepare students by defining a quality strategy for each basic programming concept, whether the instructor wishes to practice collaborative or individual learning. The lecture time must be capitalized and captured as much as possible student's attention, as non-computing students are busy with other subjects that have nothing to do with computing and find it hard to manage their time and effort to solve programming problems. Also, to reduce non-computing students' frustration and capture more of their attention, and thus, improve their debugging skills, the teaching method and learning material must include students' misconceptions about programming and how experts' reason to solve programming problems. These measures will then prevent and avoid them to make the same mistakes as other beginners or experts.

F. Validity Threats

In the survey conducted, it was not possible to collect more data from all students, especially from foundation students at the NWU. Both the foundation and the first-year students attend the same module given at different time frames and with different instructors. Moreover, students at the NWU's Vaal campus were not considered and the design of the questions could negatively impact the findings and their generalization. However, some measures were taken to minimize the threats. We conducted comparative studies to sustain the generalization of the results. Moreover, to guarantee the reliability of data and avoid response bias, the questions were made simple for ease of comprehension. Furthermore, to eliminate misunderstanding, module instructors provided some explanations during the distribution, and the participants were informed to answer with the most sincerity without fear. They were not forced to respond, and to prevent them from answering more than once, only one questionnaire paper was given to each for the face-to-face survey. For the online survey, students were not allowed to answer the survey more than once using the same computer, and the link was sent to their WhatsApp group by the teachers with the necessary instructions. The analysis of results showed that all the questions were answered by all participants and included all the group ages representative of the population. We are confident the exclusion of Foundation students will not adversely affect the findings presented.

G. Conclusion

This paper described key characteristics of the first-year non-CS students of the Faculty of Natural and Agricultural Sciences, of the North-West University in South Africa, and first-year CS students of the Department of CS, University of Dschang, Cameroon, who attend IP courses. This population includes males and females, aged between 17 and 20 years. NWU respondents are from various fields such as computer biology, computer geography, etc., with good experiences in English and Mathematics, but weak experiences in programming. The study found that NWU respondents are not self-confident when it comes to studying alone. Respondents of the two groups learn less programming during lectures session and while working on assignments and they learn better in the other learning situations presented, such as group work, practical sessions, etc. Similarly,

programming books and PowerPoint slides are different from their preferred learning materials, unlike others such as online tutorials, reuse code, the example of programs, etc. Their first programming languages were C and Python, and they faced programming problems like those of most beginners, as described in the literature, but with much more difficulties than some CS beginners. Those difficulties include debugging, deployment effort and time, understanding basic programming concepts, etc. Furthermore, most programming core concepts are also a challenge for most beginners. The following are the most difficult recursion, arrays, and error handling, and reveal that the results of this study match up well with the main findings of other researchers on the subject. Although these populations expressed a certain similarity in terms of difficulties in the IP modules, the comparative study carried out using Mann-Whitney U-test showed that NWU respondents have more difficulties with programming than those from UDS. This information was used to formulate some recommendations for the design of a new and efficient pedagogical approach and learning tools for non-computing students. For example, unlike other studies that have mainly used the constructivism-based teaching approach, we recommend that the new method of teaching IP modules to non-computer science students be based on the cognitivist approach to teaching. That novel pedagogical approach should encourage individual and pair programming, as well as the teaching of strategies that detail the steps needed to grasp a programming concept, as these students are young and need quality follow-up support. Moreover, both new pedagogical methods and learning tools should include the use of results of students' misconceptions studies to reduce non-computing students' frustration and improve their motivation and engagement in the course.

H. Acknowledgment

This research was supported by the UDSC, the Department of Computer Science at the North-West University Mafikeng campus and the Council for Scientific and Industrial Research (CSIR).

I. References

[1] R. Ramos, "Thesis Title," Ph.D Thesis, College van Dekanen, University of Twente, The Netherland, 1992.

[2] J. Bennedsen and M. E. Caspersen, 'Failure rates in introductory programming: 12 years later', *ACM Inroads*, vol. 10, no. 2, pp. 30–36, 2019.

[3] P. A. Kirschner, J. Sweller, and R. E. Clark, 'Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching', *Educ. Psychol.*, vol. 41, no. 2, pp. 75–86, 2006.

[4] I. Milne and G. Rowe, 'Difficulties in learning and teaching programming—views of students and tutors', *Educ. Inf. Technol.*, vol. 7, pp. 55–66, 2002.

[5] B. Thomasson, M. Ratcliffe, and L. Thomas, 'Identifying novice difficulties in object-oriented design', *ACM SIGCSE Bull.*, vol. 38, no. 3, pp. 28–32, 2006.

[6] P.-H. Tan, C.-Y. Ting, and S.-W. Ling, 'Learning difficulties in programming courses: undergraduates' perspective and perception', in *2009 International Conference on Computer Technology and Development*, IEEE, 2009, pp. 42–46.

[7] M. Piteira and C. Costa, 'Computer programming and novice programmers', in *Proceedings of the Workshop on Information Systems and Design of Communication*, 2012, pp. 51–53.
[8] A. Robins, J. Rountree, and N. Rountree, 'Learning and teaching programming: A review and discussion', *Comput. Sci. Educ.*, vol. 13, no. 2, pp. 137–172, 2003.
[9] R. Horváth and S. Javorskỳ, 'New teaching model for Java programming subjects', *Procedia-Soc. Behav. Sci.*, vol. 116, pp. 5188–5193, 2014.
[10] N. Salleh, M. S. I. Abdullahi, A. Nordin, and A. A. Alwan, 'Cloud-based learning system for improving students' programming skills and self-efficacy', *J. Inf. Commun. Technol.*, vol. 17, no. 4, pp. 629–651, 2018.
[11] N. Shi, Z. Min, and P. Zhang, 'Effects of visualizing roles of variables with animation and IDE in novice program construction', *Telemat. Inform.*, vol. 34, no. 5, pp. 743–754, 2017.
[12] S. I. Malik and J. Coldwell-Neilson, 'A model for teaching an introductory programming course using ADRI', *Educ. Inf. Technol.*, vol. 22, no. 3, pp. 1089–1120, 2017.
[13] Y. Ayalew, E. Tshukudu, and M. Lefoanea, 'Factors affecting programming performance of first year students at a University in Botswana', *Afr. J. Res. Math. Sci. Technol. Educ.*, vol. 22, no. 3, pp. 363–373, 2018.
[14] M. Ayub and O. Karnalim, 'Predicting outcomes in introductory programming using J48 classification', *World Trans. Eng. Technol. Educ. WTETE*, vol. 15, no. 2, pp. 132–136, 2017.
[15] Y. Qian and J. D. Lehman, 'Correlates of success in introductory programming: A study with middle school students.', *J. Educ. Learn.*, vol. 5, no. 2, pp. 73–83, 2016.
[16] A. K. Veerasamy, D. D'Souza, and M.-J. Laakso, 'Identifying novice student programming misconceptions and errors from summative assessments', *J. Educ. Technol. Syst.*, vol. 45, no. 1, pp. 50–73, 2016.
[17] X. Zhang, C. Zhang, T. F. Stafford, and P. Zhang, 'Teaching introductory programming to IS students: The impact of teaching approaches on learning performance', *J. Inf. Syst. Educ.*, vol. 24, no. 2, pp. 147–155, 2013.
[18] K. Longi and others, 'Exploring factors that affect performance on introductory programming courses', 2016.
[19] R. A. Alturki and others, 'Measuring and improving student performance in an introductory programming course', *Inform. Educ.- Int. J.*, vol. 15, no. 2, pp. 183–204, 2016.
[20] S. Wiedenbeck, D. Labelle, and V. N. Kain, 'Factors affecting course outcomes in introductory programming.', in *PPIG*, 2004, p. 11.
[21] C. O'Malley and A. Aggarwal, 'Evaluating the Use and Effectiveness of Ungraded Practice Problems in an Introductory Programming Course', in *Proceedings of the Twenty-Second Australasian Computing Education Conference*, 2020, pp. 177–184.
[22] G. Sharma, 'Computer Programming Comp103: Who Does Better?', 2019.
[23] M. Othman and M. Othman, 'The proposed model of collaborative virtual learning environment for introductory programming course', *Turk. Online J. Distance Educ.*, vol. 13, no. 1, pp. 100–111, 2012.
[24] J. Bennedsen and M. E. Caspersen, 'An investigation of potential success factors for an introductory model-driven programming course', in *Proceedings of the*

first international workshop on Computing education research, 2005, pp. 155–163.

[25] R. Gonzalez and A. Biørn-Hansen, 'Web-Based Collaborative Learning in CS1: A Study on Outcomes of Peer Code Review', in *Norsk IKT-konferanse for forskning og utdanning*, 2020.

[26] B. Isong, T. Moemi, N. Dladlu, N. Motlhabane, O. Ifeoma, and N. Gasela, 'Empirical Confirmation of Pair Programming Effectiveness in the Teaching of Computer Programming', in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2016, pp. 276–281. doi: 10.1109/CSCI.2016.0060.

[27] L. Williams and R. R. Kessler, *Pair programming illuminated*. Addison-Wesley Professional, 2003.

[28] M. Laal and M. Laal, 'Collaborative learning: what is it?', *Procedia-Soc. Behav. Sci.*, vol. 31, pp. 491–495, 2012.

[29] A. K. Mbiada, B. Isong, F. Lugayizi, and A. Abu-Mahfouz, 'Introductory Computer Programming Teaching and Learning Approaches: Review', in *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, IEEE, Jul. 2022. doi: 10.1109/icecet55527.2022.9873427.