Pedagogy of Programming Education for Higher Education Using Block Based Programming Environment

Daehoon Kim, Jaewoong Choi, In-Ho Jung, and Changbeom Choi(✉)

Handong Global University, Pohang-si, Kyunbuk 37554, Republic of Korea
{21200082, 21631005, 21631004, cbchoi}@handong.edu

Abstract. As the modern society utilizes various devices based on the Information, Communication Technology (ICT), the importance of the computer program has been increased. As the needs of the education of engineering increases, many researchers studied the pedagogy of the engineering education and the learning contents development. In general, the education of the programming language accompanies with the syntax learning and logic developments. The block-based programming language helps to build the logic of the students. Therefore, Block-based programming languages are used in the entry course to the engineering departments. However, the block-based learning languages are limited to develop conventional applications. The application of the block-based programming language requires the particular middleware to execute, and usually, the application cannot utilize the functionalities of the hardware. Therefore, students should learn the high-level programming language regardless of the block-based programming language to develop the ICT services. Unlike other pedagogy, this paper introduces the education contents and programming environment with high-level programming. Notably, this paper proposes the hybrid approaches to help students to build their programming logic and programming syntax.

Keywords: Human-computer-interactive learning · Web-based programming Programming education environment

1 Introduction

1.1 The Importance of Programming Education

As the era of the 4th industrial revolution emerged, the importance of the programming education increased. Many policymakers for K-12 curriculum have decided to adopt programming education into their curriculum. As the importance of the programming education increases, many educators considered making effective pedagogy to teach their students. There are several issues in teaching programming skills, and among them increasing motivation and understanding are essential. For those who are new to programming course, the visualized languages may give more motivation and interest rather than the grammar-based languages. Therefore, the educators have utilized to use visual programming environment to their classroom. The visualized programming environment, in general, utilize block image. By placing proper blocks and connecting them, a student may exercise to raise the computational thinking. Also, the educator may utilize visual programming environment to create various education contents by applying the environment to various domains, such as robotics, animation, application development.

Unfortunately, when the students are accustomed to the visual programming environment, and if they want to make complicated computer system, they must learn a grammar-based programming language, such as C++, Java, Python, and Scala. Therefore, an educator may teach visual programming environment to raise the computational thinking first, and after teaching visual programming environment, they have to teach grammar-based programming environment without taking advantage of the visual programming environment.

This research introduces a pedagogy for programming education using HIPE, the High-level Interactive Programming Environment to help students learn how to make a program using grammar-based programming language by taking advantage of the visual programming environment.

In this study, we introduce HIPE which provides ideal programming environment through interaction between human and computer and suggest how to interact with programming education. The rest of the paper is organized as follows. Section 2 introduces the background of the research. It introduces the characteristics of the visual programming environment and grammar-based programming environment. Also, HIPE will be introduced in this section. Section 3 shows programming education pedagogy using HIPE. Section 4 introduces the case study for HIPE, and finally, Sect. 5 concludes this paper.

2 Related Work

2.1 Text-Based Programming Environment

A well-known and popular programming environment is an environment in which a user can directly describe a program's code and complete a program. Such a programming environment can be defined as a text-based programming environment. Specifically, the text-based programming environment is an environment in which the programmer can directly insert and edit the program code in a place selected by the programmer using a keyboard and a mouse, and can convert the program code into a working program. A text-based programming environment consists of two tools. The first tool is a programming editing tool that allows the user to edit the program code using the tool. Another tool is an environment that provides a compile function and linking functions as a programming environment in which a program code described by a user is received as an executable program. A text-based programming environment can be divided into a console-based programming environment and an integrated development environment, depending on the editing tools and programming support tools. The console-based programming environment is a programming environment that utilizes the command line interface rather than the graphical user interface such as vi and gcc which are well-known tools for embedded programming environment or a

Linux programming environment. On the other hand, environments that can be edited and executed by using a graphical user interface without using a command line interface include a graphical code editing tool and a command line interface based programming support environment, or an integrated programming environment in which these tools are integrated have. Such a text-based programming environment can be utilized for learning programming while describing a user's intention as text and converting it into a form that can be executed, regardless of the graphical elements of the support tool.

In general, teachers and students use the text-based programming environment to gain programming knowledge in their classroom. The primary objectives of the teachers in programming course are two folds. First, the teachers should raise the computational thinking of the students. To help students to raise the computational thinking, the teachers prepare problem or contents, so that the students may build the logic of the program to solve the problem. On the other hand, the teachers teach the students to express their logic using programming language. Therefore, the teachers teach the syntax of the programming language to the students. As the primary objectives of the programming course are to build the computational thinking and improve the familiarity of the syntax usage of a programming language, the teachers use various tools and materials to help students. However, the text-based programming environments are not helpful to the teachers nor students. Since the text-based programming environment focuses on making a program rather than teaching students, the teachers and the students cannot take any advantages to improve the progress.

This research provides a visual educational environment for programming. The educational environment helps students to formulate logic and learn how to make a program by utilizing visual elements and generating an error-free code based on the visual elements.

2.2 Visual Programming Environment

In the programming education, a visual programming language is popular programming environment that lets users create a program by manipulating visual elements that represent the logic of the program. By visualizing logic of the program, the teachers and the students may utilize the programming environment to focus on the logic building. Also, the programming environment has an extension to control the animations, videos, music or robot to motivate students to improve the computational thinking. For instance, the Scratch, MIT App Inventor, and LEGO NXT are popular visual programming environment to teach programming beginners such as K-12 students in the STEM (Science, Technology, Engineering, and Mathematics) education.

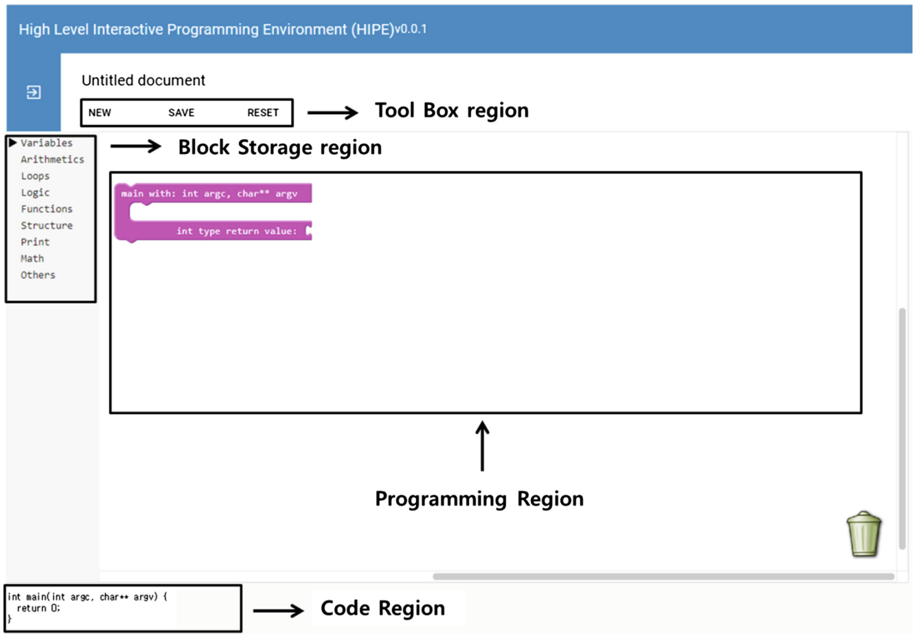
There are several characteristics of the visual programming environment. First, it can be easily programmed and shared via the Internet. For example, everyone can do programming and build a simple application that works on a smartphone in 30 min using the App Inventor. Therefore, the results of programming can be verified through a mobile device. Another feature is that the visual programming environment gives feedback to a user through a graphical user interface. For example, the user may combine logic blocks to complete the program. During the combining phase, the programming environment prohibits the incorrect combination of the logic blocks.

Accordingly, the user may build a program without knowing a complex programming language and other tools. Also, the user may drag and drop the logic or function blocks to complete program easily.

However, the visual programming environment is not suitable to develop a commercial application. In general, a commercial application has complex features and business logic, so that a developer may use hundreds and thousands of blocks to build the application. Also, the developer may not reuse the pre-developed library to implement an application. Moreover, the application of the visual programming environment may not utilize the full performance of a device. To use the full performance and the features of a device, the developer should understand the capabilities of a programming language and the target device. This research focuses on the fact. After a student develops the computational thinking using visual programming languages, the student should learn high-level programming languages to build a computer program based on the text-based programming environment. Since the students are familiar with the visual programming language, the teachers may need times and efforts to help students adopt the text-based programming environment.

2.3 High-Level Interactive Programming Environment

As the demand for programming language education grows, many services are proposed to help teach programming using visualization elements, the blocks. The services take advantage of the visualization features to express one's logic in real-time.



```
int main(int argc, char** argv) {
  return 0;
}
```

Fig. 1. High-level interactive programming environment web page screen

The feature may help a student who wants to improve the computational thinking; however, it may not help a student who wants to build a realistic program using high-level programming language. This research bridges the gap between two types of the students. The HIPE, High-level Interactive Programming Environment, is a web-based visualization programming editor that allows students to learn the structural and programming logic of the C/C++ language, intuitively.

The Fig. 1 shows the execution screen of the HIPE. The Toolbox region provides features to start building a new program or save the current status of the program. The Block storage region provides various blocks to a user based on the high-level programming language. The blocks are designed to reflect the functionalities of the high-level programming language. For example, the "print" block is designed to print out the contents to the standard output stream in C++ programming language. The Programming region is a region that the user may construct a program using block. Since the blocks are designed to prevent the syntax error, the user may not consider the syntax error of the code. Therefore, if a user tries to assemble the wrong combination of the block, the HIPE will give feedback to the user that the given block combination generates the syntax error.

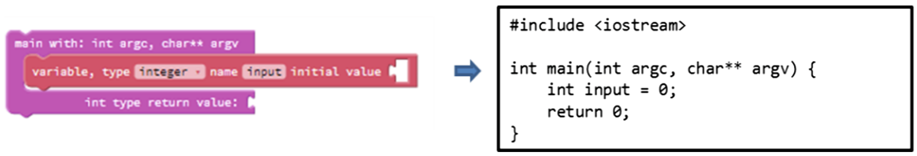
Finally, the Code region shows the code generation results concerning the assembled blocks in real-time.

3 Visual Programming Pedagogy Based on HIPE

Most programming education environments are a text-oriented environment. The tutor provides theoretical contents to the students and provides programming tasks to check the progress of a student. Such type of education can be a high entry barrier for beginners. On the other hand, a visual programming environment has low entry barrier to the beginners. The students may check their logic in real-time using feedback from the visual programming environment. Therefore, the visual programming environments are used to help beginners to form correct programming logic and design capability. However, almost all commercial applications are developed using a text-based programming language and environment. Consequently, when a student wants to build such application, the student should learn how to use the text-based programming language and the environment. The HIPE is a useful tool to fill in the gap between the visual programming language and the text-based programming language. The HIPE may be utilized to tutors who want to help students to improve the programming logic and high-level programming skills at the same time. The tutors and the students may use HIPE to interact each other. Following subsections show the educational phases and interactions among the tutors, the students, and the HIPE during the educational phases.

3.1 Phase 1: Interactive Learning

In Phase I, a tutor and a student interact with each other using HIPE. Since the HIPE shows the logic of the program, the tutor may use the tool to show the code generation results when the tutor puts a block to the HIPE. For example, when the tutor puts a block to the programming region of the HIPE, the HIPE generates the codes based on the assembled blocks. Since the HIPE prohibits the syntax error in real-time, students may practice forming correct logic. Also, the tutor may use the tools to make a simple pop-up quiz by using the code generation features. Figure 2 shows the logic block and the code generation results. As shown in the figure, a user may put blocks to the programming region of the HIPE. Then, the HIPE analyzes the assembled blocks and generates the corresponding C++ code.


main with: int argc, char** argv
variable, type integer name input initial value
int type return value:

```
#include <iostream>

int main(int argc, char** argv) {
    int input = 0;
    return 0;
}
```

Fig. 2. Blocks and codes for variables

Figure 3 shows the sequence diagram of interaction among the tutor, the student, and the HIPE. When a tutor teaches a student, the tutor may place the block incrementally. Then, a student may watch how the program completes. Also, during the teaching session, the tutor may use HIPE to answer the students' questions.

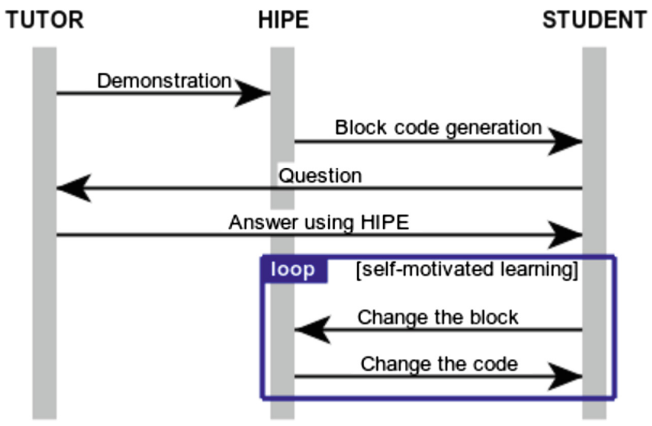
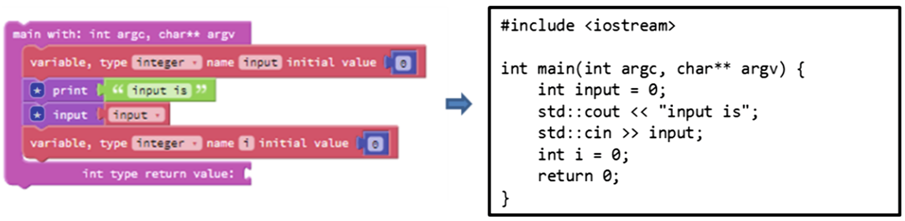


Fig. 3. Sequence diagram

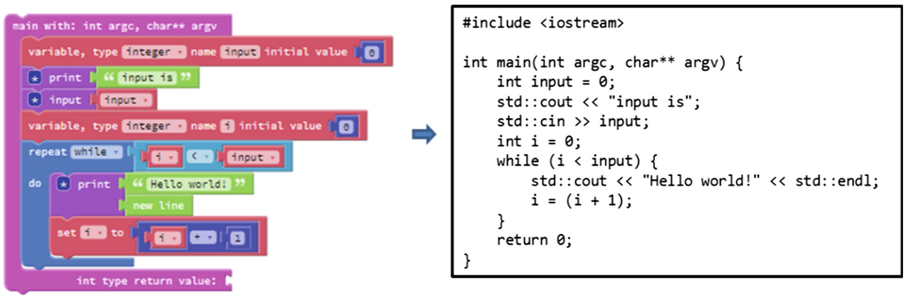
Figure 4 and 5 shows an example of the Phase I. Figure 4 shows the code generation results when a user adds print block and input block. When the user adds print block to the HIPE, the HIPE translate the block to code which uses the standard output stream. Similarly, when the user adds input block, the HIPE translate the blocks to code.


main with: int argc, char** argv
variable, type integer name input initial value 0
print "input is"
input input
variable, type integer name i initial value 0
int type return value:

```
#include <iostream>

int main(int argc, char** argv) {
    int input = 0;
    std::cout << "input is";
    std::cin >> input;
    int i = 0;
    return 0;
}
```

Fig. 4. Blocks and codes for input/output


main with: int argc, char** argv
variable, type integer name input initial value 0
print "input is"
input input
variable, type integer name i initial value 0
repeat while i < input
do print "Hello world!"
new line
set i to i + 1
int type return value:

```
#include <iostream>

int main(int argc, char** argv) {
    int input = 0;
    std::cout << "input is";
    std::cin >> input;
    int i = 0;
    while (i < input) {
        std::cout << "Hello world!" << std::endl;
        i = (i + 1);
    }
    return 0;
}
```

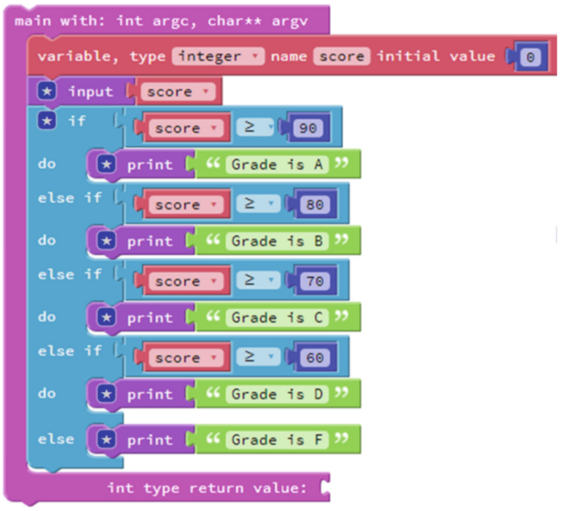
Fig. 5. Block and codes for while loop

Figure 5 shows the complex combination of the blocks. When the user adds the repeat blocks to the HIPE, the HIPE will generate the block into while block. During the session, a tutor may ask a question such as “Tell me what code is generated when you insert this block.”

3.2 Phase II: Learning from Difference

The next step of the pedagogy using HIPE is the “Learning from difference” phase. In Phase II, students are encouraged to study by themselves. First, a tutor gives a problem to the students. Notably, the problem is a diagram of code blocks which denotes the logic of the student a problem with the block created using HIPE and write the code for the problem. It is the process of actually mastering Programming Syntax learned through Phase 1. The student can identify and train the wrong part of himself by comparing his written code with the results of the HIPE.

Figure 6 shows the example of Phase II. Figure 6 is a block diagram which shows a program that receives a score from a user and outputs a grade corresponding to the score using the multiple conditional statements with HIPE. During the practice session, the tutor provides the diagram to the students and encourage them to develop the program using text-based programming language. Then, the students may solve the problem or may not solve the problem using text-based programming language. If the students are not yet familiar with the programming syntax, the students may go back to the Phase I, and use the HIPE to enhance their skills.



```
main with: int argc, char** argv
  variable, type integer name score initial value 0
  input score
  if score ≥ 90
  do print "Grade is A"
  else if score ≥ 80
  do print "Grade is B"
  else if score ≥ 70
  do print "Grade is C"
  else if score ≥ 60
  do print "Grade is D"
  else print "Grade is F"
int type return value:
```

Fig. 6. Complete code block and codes for conditional statements

3.3 Phase 3: Traditional Learning

Phase III is the final phase. In this phase, the students are familiar with the given programming syntax. Therefore, a tutor may use the traditional education method to teach students. During the phase, the tutor may present the programming problem to the student without using HIPE. If the students have a syntax error in their code, or if the students do not fully understand the syntax, they may solve the problems in the text-based programming environment.

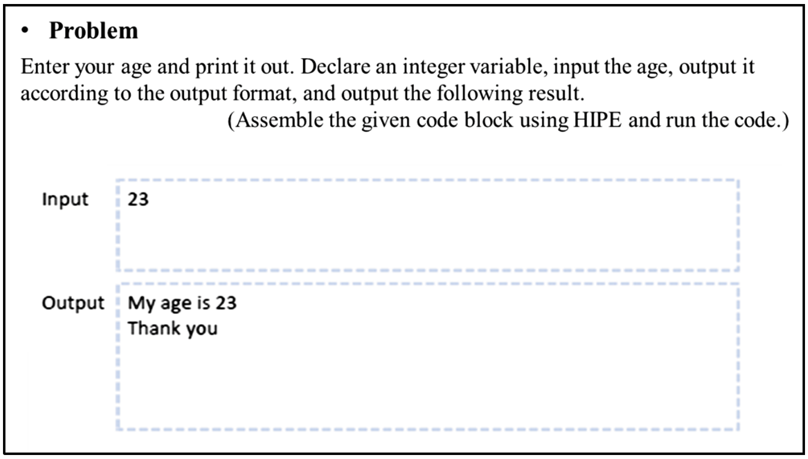
4 Case Study: Short-Term Programming Course

This section introduces the case study to show the effectiveness of the pedagogy at Handong Global University. Notably, the School of Global Entrepreneurship at Handong Global University opens short-term programming course twice a year for extracurricular activities. About 37 students have participated the programming course in 2017, and the course was specifically designed to apply the proposed pedagogy using HIPE. Approximately 70% of the participants have experienced the visual programming languages, and about 57% of the students have prior knowledge of C/C++ languages. After every session, a survey was taken to check the effectiveness of the pedagogy using HIPE.

4.1 Camp Operation

During the short-term programming course, the tutor explained the theoretical part of the C++ programming language, such as the basic syntax of C++ programming language, dynamic memory allocation, and other high-level features of the object-oriented programming language. After the lesson, there was a practice session to check the progress. The course had three parts as the proposed pedagogy. At the beginning of the course, HIPE was used for to help students to understand the difference between a visual programming language and a text-based programming language. After the programming lessons, the students were encouraged to use text-based programming environment during the practice session. Finally, at the end of the course, all students used the text-based programming language environment.

Figure 7 shows the problem used in the short-term programming course. The objective of the problem was to learn the syntax for input and output of the program. During the practice session, simple problems were provided to the students. Each problem has description part to introduce the objective of the program. Also, the problem has input/output boxes to provide a test case to check the correctness.



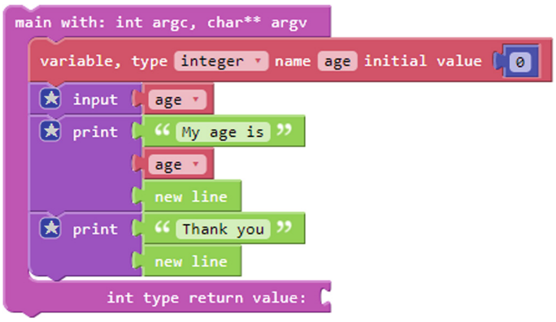
- **Problem**

Enter your age and print it out. Declare an integer variable, input the age, output it according to the output format, and output the following result.
(Assemble the given code block using HIPE and run the code.)

Input
```
23
```

Output
```
My age is 23
Thank you
```

Fig. 7. Problem of the practice session during Phase I



```
main with: int argc, char** argv
    variable, type integer name age initial value 0
    input age
    print "My age is "
          age
          new line
    print "Thank you"
          new line
    int type return value:
```

Fig. 8. Code blocks of the practice session during Phase I

During the practice session, students assembled a given block and checked the code generations. Figures 8 and 9 illustrate the code blocks and corresponding code of the practice session.

```
#include <iostream>

int main(int argc, char **argv){
    int age = 0;
    std::cin >> age;
    std::cout << "My age is" << age << std::endl;
    std::cout << "Thank you" << std::endl;
    retrn 0;
}
```

Fig. 9. Code of the practice session during Phase I

4.2 Questionnaire Results

Figure 10 shows the survey result. Almost 70% of the students are familiar with the visual programming language. Also, 56.8% of the students have prior knowledge of the C/C++ programming language. Based on the preliminary survey results, more than 40% of students begin with the visual programming languages to study programming.

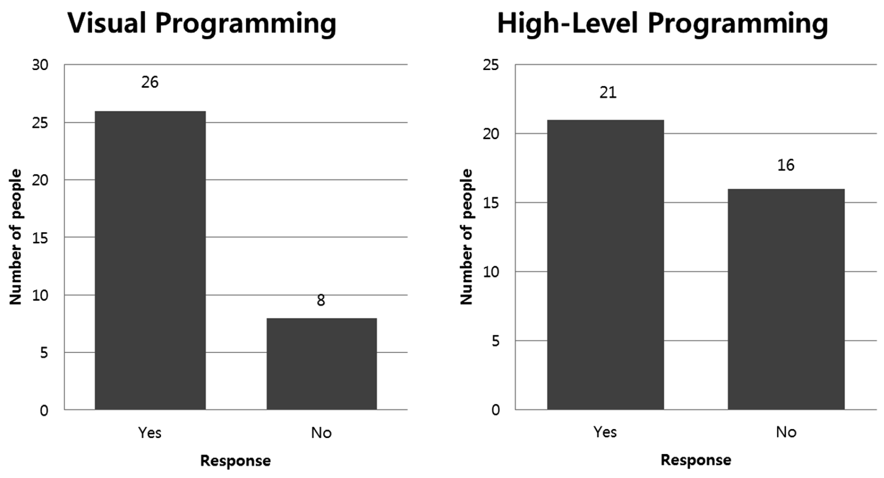


Fig. 10. Questionnaire result before short-term course

Figure 11 shows the survey results after the programming course. As a result, 81% of the students answered that the HIPE was helpful to understand the logic and the syntax of the programming language. This survey shows that the HIPE may be useful to the students who are familiar with the visual programming language, and the HIPE was useful to the students who have prior knowledge of the C/C++ programming language. Also, students have testified that the HIPE was useful to see the logic diagram and its code at the same time and it was helpful when the HIPE gives the feedback based on the combination of the blocks.

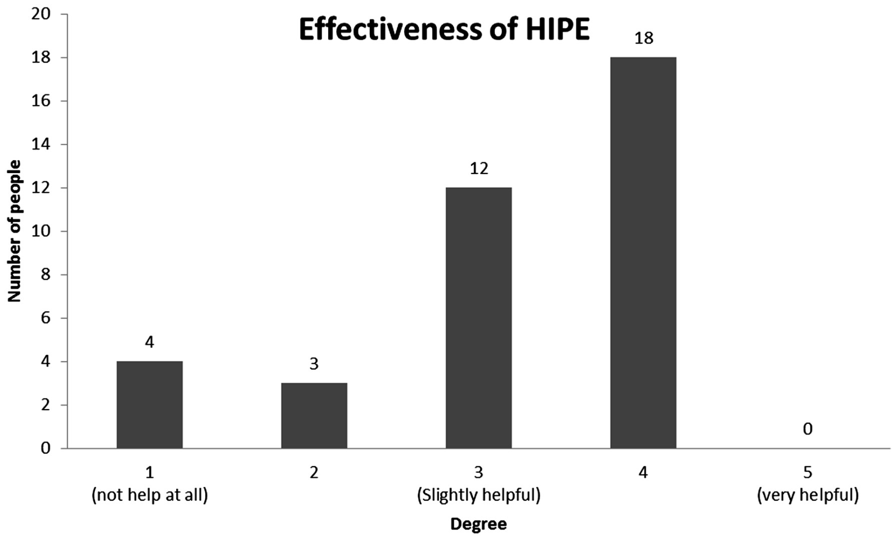


Fig. 11. Questionnaire result after short-term course

5 Conclusion

In the era of the fourth industrial revolution movement, the programming education is one of the hot issues in the education. To help beginners to form the computation thinking, many educators have utilized various teaching methods and tools. One of the programming education environment is block-based programming environment. The block-based programming environment is a visual programming environment using blocks to represent the logic of the program. Since the block-based programming language allows the correct block combination, a student may get feedbacks from the programming environment. After the students are familiar with the computational thinking, the students may want to build a complex application. Unfortunately, the students may not develop the complex or commercial application with the visual programming language, due to the limitation of the language. Therefore, the students should study text-based programming language. Since the students are familiar with the visual programming language, this research proposed a pedagogy using visual programming environment for a text-based programming language.

The programming environment, HIPE, fills the gap between building computational thinking and learning the syntax of the programming language. As proposed in the paper, a tutor may utilize the HIPE in various ways during the programming course.

For the future research, extending the programming pedagogy and programming environment to support specific topics, such as the Internet of Things and artificial intelligence.

References

1. https://www.vim.org/
2. https://gcc.gnu.org/
3. Malan, D.J., Leitner, H.H.: Scratch for budding computer scientists. ACM SIGCSE Bull. **39** (1), 223 (2007)
4. Wing, J.M.: Computational thinking and thinking about computing. Philos. Trans. Roy. Soc. A Math. Phys. Eng. Sci. **366**(1881), 3717 (2008)
5. Scratch official homepage. https://scratch.mit.edu/
6. App Inventor official homepage. http://appinventor.mit.edu/explore/
7. Lego NXT official homepage. http://www.lego.com/enus/mindstorms/?domainredir=mindstorms.lego.com
8. Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B.: Scratch: a sneak preview [education]. In: Second International Conference on Creating, Connecting, and Collaborating through Computing, pp. 104–109 (2004)
9. Shu, N.C.: Visual Programming. Van Nostrand Reinhold, New York (1988)