PFW as generator

László Menyhárt, Gáborné dr. Pap

Department of Media & Educational Informatics
Eötvös Loránd University, Budapest, Hungary
menyhart@inf.elte.hu
papne@inf.elte.hu

Abstract

Some months ago we presented on another conference the PFW: Programming Fundamentals Wizard, which is the improved version of helper tool for the subject Programming fundamentals. Our web-based application gives place to collect the information from the tasks and it contains an editor for Nassi-Shneiderman diagram. This guide helps the students to keep the expected order of problem solving namely at first thinking and secondly coding or even collecting the information, creating algorithm independently from program languages then creating source code. Unfortunately deep testing is not specific and documentation is required only as homework based on an example.

This time we present two new functions of PFW. One of the new functions is good to generate the first, raw version of the documentation which must be edited even. Second function generates C++-like source code. We would like to emphasize that the primary aim of this application is not the generation, it was not developed because of this but this beta version of C++-like source code helps intensively the beginner programmers in the implementation and understanding the code. Generating a compilable and runnable source code is possible with using program-language-specific expressions in the Nassi-Shneiderman diagram.

Keywords: teaching programming, code generation, documentation

MSC: 97B40, 97R99, 97Q60

1. Introduction

Computer science is a young and a diverse science. That is why education of computer science is very difficult. It is very important to clarify that who will learn it and on which level, because it defines the area and the depth. In this article we present some possibilities of a guide application (PFW) to the Programming Fundamentals course at University. There are more methods to teach programming
[1]. We use methodological and algorithm oriented programming on the foundation courses of students of informatics. This method takes in the whole process of implementation from creating specification to writing documentation. In an earlier paper we presented an own developed application, PFW, which helps the students in this systematic learning. We presented the PFW application and the method how it changes up the process of creating algorithm, helps the implementation and how it can be used in education. In this article we present two new functions of PFW. User is able to generate source code and documentation with these.

2. Short story of PFW

PFW is a guide on web platform. It helps to the students analyzing and thinking over their task in our programming courses. Data the tasks and their correlation can be collected on these pages so students are guided through the specification.

They can edit on a webpage the image representation of Nassi-Shneiderman diagram of the algorithm with mouse clicking. They can store the test cases and test results on this platform.

3. Implementation

This own developed, web-based application [14] was implemented in Java and it runs on our server. User must login for using it. Three possible ways are the Gmail identifier, user id in INF.ELTE domain or after a registration in the system with unique user identifier and password. Every user has an own workspace in the database where tasks are stored. The storage format is XML [6] from which the information on the screen and codes are generated.

3.1. Surface of the application

Figure 1 shows the initial screen of the application which contains introduction, link to the registration and login. The first menu has menu items for opening a task or creating a new task. Figure 2 shows the first page after the successful login.

The menu will be changed when the user open or create a task. The new menu supports the saving and closing of task and it’s solving steps. Figure 3 shows the main page.

Figure 4 shows the pages on which the user creates the specification. User fabricates the algorithm in NSD format using a graphical view of it. Figure 5 shows the floating menu of editing the NSD.

After these steps user can fill out the data on the testing and the possible future improvements pages.
Programming Fundamentals Wizard

What is this?
This assistant application helps the users to think over a problem from the Programming Fundamentals course in Eötvös Loránd University.

What does it do?
It generates a basic version of the required documentation and create sample codes of the solution.

What should I do?
You must follow the steps: create specification, create algorithm, developing the first version of the code, testing, update the basic version of documentation.

How can I use this assistance?
You must log in with your Gmail account or with your account in INF domain in ELTE or you should register at first.

Log in
• with your Gmail account
• with your account in INF.elte.hu domain
• with your PFW account

Registration
If you do not have or do not want to use your Gmail account or INF.elte.hu account, please register here.

3.2. Functions of generation
The user performed the design part of solving the task a la the previous paragraph. This is the most important step. After this the coding and creating the documen-
tation demand less thinking. Several parts of these can be mechanically produced. PFW’s new functions grab these parts and help the user with generation. They have not to double work so they save time. It is not our intention to offer correct source code and documentation. Students must correct and update both stuffs.

3.2.1. Generation of source code

Generation of C++ source code starts with a click on a button (See Figure 6) on the page General.

A compressed file is downloaded which contains a Code::Blocks project. In good case generation is required only one times. If the algorithm was modified, the new generation has a unique name as a version. Three files are in the project because the source code is taken apart to the main part and generated functions of subtasks. Main part contains the input and output handling, so the user must perform more work on this. NSD is very flexible and the boxes can contain anything because it is for human and not for the computer. But using the symbol system learned on the course is useful because the initiative C++ code is generated from this. The algorithm does not contain the reading input, writing output and other language specific items. For example operators must be rewrite in the source code. It is mechanical work, but sometimes it is difficult for the students. For instance

Figure 3: Example task
Figure 4: Editing the specification

Figure 5: Floating menu at NSD editing
the indexes of arrays need a bit thinking because C-like languages’ points to the beginning of an item, so brackets must contain one less value (0 instead of 1).

The generator produces a function to every subtask. The parameters of the function, with its name and type, are generated from the specification. That is why the specifications of subtasks are very important. The function body will be generated from the NSD diagram. User must translate this faulty source to the correct C++ specific expressions.

```c
void subtask_1(int N, int X[], int & S){
    S:=0;
    while (i:=1..N) {
        S:=S+X[i];
    }
}
```

Figure 7: Generated source code

Figure 7 shows the generated C++ source code. We marked with red color the places where user must modify it for getting a runnable code with correct syntax.

### 3.2.2. Generation of documentation

Generation of documentation starts with a click on a button (see. Figure 8) on the page General.

This document is based on an example with the same content and structure. The content is filled with every known data which are stored in the PFW. The expected but unknown places contain red colored “Update it!” expression where students must edit it. For instance users must paste the screenshots into the user manual and diagram of project structure must be copied into the development documentation.

### 3.3. Technology

The basic idea of the generation is XSLT. The XML Stylesheet Language defines the output of the transformation from the XML. Fortunately PFW stores the data in XML format. C++ source code is text which can be generated from XML.
4. Conclusion

We presented two new functions in our web-based application. These generations are good for students and teachers, too. They can save time with these and the students can study the connection between the specification and data structure of the application and between the algorithm and source code.

References


