MINISTRY OF NATIONAL EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

SueC - An Editor and Interpreter for Pseudocode

LICENSE THESIS

Graduate: Mihai PÎȚU
Supervisor: dr. eng. Emil Ștefan CHIFU

2019

MINISTRY OF NATIONAL EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

DEAN,
Prof. dr. eng. Liviu MICLEA

HEAD OF DEPARTMENT,
Prof. dr. eng. Rodica POTOLEA

Graduate: **Mihai PÎȚU**

SueC - An Editor and Interpreter for Pseudocode

1. **Project proposal:** *Short description of the license thesis and initial data*
2. **Project contents:** *(enumerate the main component parts) Presentation page, advisor's evaluation, title of chapter 1, title of chapter 2, ..., title of chapter n, bibliography, appendices.*
3. **Place of documentation:** *Example*: Technical University of Cluj-Napoca, Computer Science Department
4. **Consultants:**
5. **Date of issue of the proposal:** November 1, 2017
6. **Date of delivery:** February 18, 2019 *(the date when the document is submitted)*

Graduate: _____

Supervisor: _____

MINISTRY OF NATIONAL EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnatul(a) _____, legitimat(ă) cu _____ seria _____ nr. _____
CNP _____, autorul lucrării _____
_____
_____
elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la Facultatea de Automatică şi Calculatoare, Specializarea _____
din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea _____ a anului universitar _____, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

În cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data _____

Nume, Prenume _____

Semnătura

De citit înainte (această pagină se va elimina din versiunea finală):

1. Cele trei pagini anterioare (foaie de capăt, foaie sumar, declaraţie) se vor lista pe foi separate (nu faţă-verso), fiind incluse în lucrarea listată. Foaia de sumar (a doua) necesită semnătura absolventului, respectiv a coordonatorului. Pe declaraţie se trece data când se predă lucrarea la secretarii de comisie.

2. Pe foaia de capăt, se va trece corect titulatura cadrului didactic îndrumător, în engleză (consultaţi pagina de unde aţi descărcat acest document pentru lista cadrelor didactice cu titulaturile lor).

3. Documentul curent **nu** a fost creat în MS Office. E posibil sa fie mici diferenţe de formatare.

4. Cuprinsul începe pe pagina nouă, impară (dacă se face listare faţă-verso), prima pagină din capitolul *Introducere* tot aşa, fiind numerotată cu 1.

5. E recomandat să vizualizaţi acest document şi în timpul editării lucrării.

6. Fiecare capitol începe pe pagină nouă.

7. Folosiţi stilurile predefinite (Headings, Figure, Table, Normal, etc.)

8. Marginile la pagini nu se modifică.

9. Respectaţi restul instrucţiunilor din fiecare capitol.

Contents

Chapter 1

Introduction - Project Context

1.1 Project Context

Computer science and programming is taught in schools around Romania for at least 30 years, especially in high schools, but also in secondary schools, starting with the 5^{th} grade. Before introducing directly to a programming language, many teachers use a pseudocode language which serves as a mean of understanding programming concepts in a more universal manner, bringing it closer to the natural spoken language. I have decided to make an implementation of this pseudocode by creating an editor and interpreter for it.

The purpose of this project is to create an easier way of learning programming concepts for students who are new into this domain. This will serve as a fresh renewal of software used in schools today, as older tools such as Code::Blocks and/or Free Pascal are still used in schools and programming contests.

SueC is the name of the editor which creates, edits and compiles files which represent pseudocode files. This editor will work also like any other editors, providing some error-checking mechanisms and returning the result after compiling a pseudocode file.

1.2 Motivation

During high school, many of my colleagues have struggled learning programming and computer science as they had issues in understanding the simple paradigms because of C programming language. They have improved throughout the high school due to the teacher using pseudocode as a mean of explaining simple algorithms and paradigms, but there were some struggle shown for some when changing the pseudocode into implementations in C.

Nowadays, this issue is still persistent in schools in Romania as C and Pascal are used as main programming languages for teaching, exams and computer science contests. There are some more interactive programming languages such as Scratch which uses a graphical interface for implementing simple programs, but since the target audience is for primary school students, there is a need for an attractive way of making secondary and

high school students for understanding programming at their age group.

At the moment, there are platforms for learning code such as CodeCademy and Udemy which contain basic courses for people at every age, but the main focus is for people who have a little background in programming and computer science.

Chapter 2

Project Objectives and Specifications

As the title of the project suggests - "An Editor and Interpreter for Pseudocode" - this is an application which will serve as an educational tool for using the pseudocode as a programming language.

For the users of this application(students and/or teachers), the main functionalities of this application are:

- Creating/opened a file in which pseudocode can be implemented.
- Writing pseudocode in the file created/opened.
- Compiling the file and obtaining the desired result or error(s) if there are occured.
- Running some step-by-step basic tutorials which are aimed for learning the language.

The main objectives of this project are:

- Developing an user-friendly application which handles the main file handling operations and communicating with the compiler of the pseudocode source files.
- Creating an understandable programming language that resembles the pseudocode used by teachers in schools and/or universities. For a technical point of view, the pseudocode will be created like any other programming languages, having similar elements to existing ones that are used nowadays, but also with specific structural elements bringing it closer to the natural language.
- Developing a compiler for this programming language by defining a lexical and syntactic analyzer respectively. These analyzers contain the set of rules that apply to the programming language.

Chapter 3

Bibliographic research

For this project, my research done was focused on the main components and technologies included in the project:

1. **C Programming Language**
2. **Python Programming Language**
3. **Lex**
4. **Yacc**

3.1 C Programming Language

C is a general-purpose, procedural computer programming language supporting structured programming, lexical variable scope, and recursion, while a static type system prevents unintended operations. This programming language was created between 1972 and 1973 as a way of making utilities work in Unix operating system, later being used for reimplementing the kernel of this OS. Since 1980s, C has gained enough popularity becoming one of the most widely used programming languages in the world. During this time, there were several C compilers created by several vendors for being available for the majority of existing computer architectures and operating systems. Since 1989, C has been standardized by ANSI (American National Standards Institute) and by the International Organization for Standardization (ISO).

Being an imperative procedural language, C was designed to be compiled using a relatively straightforward compiler to provide low-level access to memory and language constructs that map efficiently to machine code instructios all with minimal runtime support. This language supports cross-platform programming, making it available in numerous platforms, from embedded microcontrollers and supercomputers. It also stood as a big influence in the creation of other programming languages, such as:

- C++

- C#
- Java
- Python
- Go

The C programming language syntax is defined by a formal grammar, having specific keywords and rules based on statements to specify different actions. The most common statement is an expression statement, consisting of an expression to be evaluated followed by a semicolon. The main structure of a C program consists of declarations and function definitions, which in turn contain declarations and statements.

Besides exxpressions, the main sequence execution of statements can contain several control-flow statements defined by reserved keywords:

- Conditional execution

  This is defined by *if* and *else* statements. These statements contain an expression that the *if* checks if it is true or not and execute statements based on a condition.

  Alongside those statements, there exists the *switch* statement in which it displays a *case* based on the expression given.

- Iterative execution (Looping)

  This is defined by *while* , *do-while* and *for* statements which can loop through a certain set. The *for* statement contains separate expressions for initialization, testing and reinitialization, any of which can be omitted.

3.2 Python Programming Language

Python is an interpreted, high-level, general-purpose programming language with the aim to help programmers with clear, logical code for small and large-scale projects. It was conceived in the late 1980s as a successor to ABC language, but it was released in 1991. Python is dynamically typed and garbage-collected, supporting multiple programming paradigms, such as: procedural, object-oriented and functional. Due to this and its comprehensive standard library, Python is often described as a "batteries included" language.

Due to supporting multiple programming paradigms, Python is used in a lot of domains. Object-oriented programming and structured programming are fully supported, but it also includes features from functional programming and aspect-oriented programming. Other paradigms can be supported by Python via extensions, including even logic programming and design by contract.

Python is meant to be an easily readable language due to its syntax and semantics. The format is visually uncluttered, using English keywords more often than punctuation. Curly brackets are not used to delimit blocks and semicolons are optional. For block delimitation, whitespace indentation is used. A decrease in indentation shows that the current block of code is finished. With this method, it is shown that the program's visual structure accurately represents the program's semantic structure.

Chapter 4

Analysis and Theoretical Foundation

Together with the next chapter takes about 60% of the whole paper

The purpose of this chapter is to explain the operating principles of the implemented application. Here you write about your solution from a theory standpoint - i.e. you explain it and you demonstrate its theoretical properties/value, e.g.:

- used or proposed algorithms
- used protocols
- abstract models
- logic explanations/arguments concerning the chosen solution
- logic and functional structure of the application, etc.

YOU DO NOT write about implementation.

YOU DO NOT copy/paste info on technologies from various sources and others alike, which do not pertain to your project.

4.1 Title

4.2 Other title

Chapter 5

Detailed Design and Implementation

Together with the previous chapter takes about 60% of the paper.

The purpose of this chapter is to document the developed application such a way that it can be maintained and developed later. A reader should be able (from what you have written here) to identify the main functions of the application.

The chapter should contain (but not limited to):

- a general application sketch/scheme,
- a description of every component implemented, at module level,
- class diagrams, important classes and methods from key classes.

Chapter 6

Testing and Validation

About 5% of the paper

6.1 Title

6.2 Other title

Chapter 7

User's manual

In the installation description section your should detail the hardware and software resources needed for installing and running the application, and a step by step description of how your application can be deployed/installed. An administrator should be able to perform the installation/deployment based on your instructions.

In the user manual section you describe how to use the application from the point of view of a user with no inside technical information; this should be done with screen shots and a stepwize explanation of the interaction. Based on user's manual, a person should be able to use your product.

7.1 Title

7.2 Other title

Chapter 8

Conclusions

About. 5% of the whole
Here your write:

- a summary of your contributions/achievements,
- a critical analysis of the achieved results,
- a description of the possibilities of improving/further development.

8.1 Title

8.2 Other title

Bibliography

[1] E. Bellucci, A. Lodder, and J. Zeleznikow, "Integrating artificial intelligence, argumentation and game theory to develop an online dispute resolution environment." in *16th International Conference on Tools with Artificial Intelligence*, 2004, pp. 749–754.

[2] G. Antoniou, T. Skylogiannis, A. Bikakis, M. Doerr, and N. Bassiliades, "Dr-brokering: A semantic brokering system." *Knowledge-Based Systems*, vol. 20, no. 1, pp. 61–72, 2007.

[3] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, 1995, vol. 2.

[4] W. Strunk, Jr. and E. B. White, *The Elements of Style*, 3rd ed. Macmillan, 1979.

Appendix A

Relevant code

```
 /** Maps are easy to use in Scala. */
object Maps {
  val colors = Map("red" -> 0xFF0000,
                   "turquoise" -> 0x00FFFF,
                   "black" -> 0x000000,
                   "orange" -> 0xFF8040,
                   "brown" -> 0x804000)
  def main(args: Array[String]) {
    for (name <- args) println(
      colors.get(name) match {
        case Some(code) =>
          name + " has code: " + code
        case None =>
          "Unknown color: " + name
      }
    )
  }
}
```

Appendix B

Other relevant information (demonstrations, etc.)

Appendix C

Published papers