Turing project: ALGOL 60 Tutorial

Viktor Gsteiger
University of Basel
Matriculation Number: 18-054-700

November 21, 2020
Seminar: 58826-01 - Turing Award Winners and Their Contributions

Abstract

The difficulty of learning a new programming language is inherently great. One may have no previous experience all together, one may have some experience but with another language or one may have some knowledge about the language at hand but may have forgotten large parts of the learn things again. The difficulty of learning a programming language that is not used any more and never had great commercial success is even greater, however, in the case of ALGOL 60, I am convinced, that the effort is not without benefits. ALGOL 60 is one of the grandparents of most modern programming languages and thus a direct predecessor of the tools we use every day. It is thus important to study the roots of our tools, to learn from past experiences and correct past mistakes.

Contents

1 Introduction

This tutorial aims to give the reader an introduction into the ALGOL 60 programming language. The reader should be able to program small to mid size procedures after reading this tutorial and should be able to translate and execute the ALGOL 60 program with the help of the marst translator. This tutorial does not aim to be complete and due to the inherent difficulty of learning a programming language it does not aim to lead to success.

2 Background

ALGOL 60 was the direct successor of the International Algebraic Language (IAL or later called ALGOL 58) and was a joint effort of European as well as American computer scientists in the years 1958 to 1960. With the help of the ALGOL Bulletin, a publication edited by Peter Naur, and several conferences the ALGOL 60 report could be published in 1960. ALGOL 60 did not have great commercial success on its own, however, the concepts introduced by the language can be witnessed in programming languages until nowadays.

The GNU marst translator translates programs written in ALGOL 60 into the ANSI C 89 programming language. It is part of the GNU project and currently maintained by Andrew Makhorin and the last release dates back to 2013.

3 ALGOL 60 Environment Setup

Before we can start programming in ALGOL 6, we will need to install some prerequisites to edit, translate and execute ALGOL 60 programs.

3.1 Text Editor

To edit any kind of text document, one will need a text editor. Examples include Windows Notepad, vim, EMACS, Atom or similar text editors.

The files created with the text editor are source files with ALGOL 60 programs usually having the extension ".alg".

3.2 The C Compiler

The C Compiler translates the human readable source code into executable machine language. In the case of writing ALGOL 60 programs the C Compiler is not directly accessed by the user but rather compiles the translated ALGOL 60 program into machine code.

The C Compiler usually used is the GNU C/C++ compiler. In the following subsection I will discuss on how to install the C compiler on the UNIX based Operating Systems. It is sadly not possible for me to install it on Microsoft Windows and thus I will focus on the UNIX based OS.

3.3 C Compiler Installation

3.3.1 Installation on UNIX

The GNU C/C++ compiler is mostly already installed on UNIX systems. To check whether the compiler is already install type the following into the command line:

```
$ gcc -v
```

If the GNU compiler is already installed then something like the following will be printed out to the command line:

```
Using built-in specs.
Target: i386-redhat-linux
Configured with: ../configure --prefix=/usr .......
Thread model: posix
gcc version 4.1.2 20080704 (Red Hat 4.1.2-46)
```

If the GNU compiler is not installed on your UNIX system you will need to install it from an official GNU distribution. See the documentation on the download page for reference.

3.3.2 Installation on Mac OS

The easiest way to install the GNU compiler on a Mac OS X is to install the Xcode development environment provided by Apple. See the documentation on the download page for reference.

This tutorial has been written based on Mac OS and the examples have been translated and compiled on Catalina.

3.4 ALGOL 60 Translator Installation

The marst ALGOL 60 translator can be downloaded from any gnu mirror under `/gnu/marst/`. We will be using version 2.7 or marst released in 2013. Download the tar directory and uncompress it.

To install marst on your OS type the following into the command line at the location of the `marst-2.7` directory:

```
$ ./configure; make; make install
```

This should configure, build, and install the marst package. For more information see the `README` or the `INSTALL` file.

4 ALGOL 60 Program Structure

Before we introduce the building blocks involved in developing an ALGOL 60 program we will introduce an example ALGOL 60 program and its structure so that we may use it again for reference in the following sections.

4.1 ALGOL 60 Hello World Example

An ALGOL 60 program can be constructed with the following parts:

- Procedures
- Variables
- Statements
- Comments

A simple example to display various parts of an ALGOL 60 program would be the following:

```
procedure main();
        comment a first ALGOL 60 program
        begin
                outstring(1, "Hello, world!\n")
        end
end main;

main();
```

The parts of the above program are the following:

1. The first line declares the procedure which we called the main procedure. The name of the procedure can be changed.
2. The next line is a comment which will be ignored by the translator and is used to comment on the code at hand to make it easier for fellow programmers to understand the intention of the program.
3. The `begin` keyword signifies that a block of the procedure starts here.
4. Following comes an output keyword `outstring` which displays the string given to the first output channel.
5. The last line calls the main procedure and executes it with it.

4.2 Compile and Execute an ALGOL 60 Program

We will now save the ALGOL 60 program, translate it, compile it, and run it. The steps to do this are the following:

1. Open your text editor and type in the above program.
2. Save the file as `hello.alg`.
3. Open a command line and navigate to the directory where the above program has been saved.

4. Type `marst hello.alg -o hello.c`.
5. If there are no errors the translator creates the C file `hello.c`.
6. Compile and link the file with the following command `gcc hello.c -lalgol -lm -o hello`.
7. Run your executable `./hello`.
8. If everything worked fine you should see `"Hello, world!"` printed on the command line.