TDDD55 Programming Exercise 4: Intermediate Code Generation

1 Introduction

The purpose of this exercise is to learn a little about how parse trees can be translated into intermediary code. Although there are powerful tools that can be used to generate code generators, it is still often done by hand.

2 Requirements

The file `codegen.cc` contains methods for generating code from most types of abstract syntax tree nodes, but you need to write the methods for if statements (including the elseif and else branches), for array references and assignments to array elements and for all binary operators and relations by implementing the function `BinaryGenerateCode` which is used for all binary operators and relations. Write the missing methods and add calls to GenerateCode in the parser specification. When completed, you should have a program that is capable of generating intermediate code for the small programming language used in exercises two, three and four.

Hand in the following:

- A listing of `codegen.cc` with your changes clearly marked.
- Listings of any other files you modified.
- Answers to the questions in the next section.

Demonstrate your solution to your lab assistant during a laboratory session. Send an e-mail (one e-mail per group) with your modified code and answers to the questions to the same assistant, put TDDD55, assignment number and your LiU logins in the e-mail subject line.

3 Questions

Question 1 The code generator generates terribly inefficient code. For example, assigning a constant to a variable causes two quads to be generated, where one would have been enough. There are a number of other situations where equally bad code is generated. Suggest at least one way of eliminating most of the bad code that is generated.