Debugging large programs is a complex and time-consuming task, which has not been fully automated yet. Given a runtime error, the developer must first reproduce it. He then has to find the root cause of the error and create a proper bug fix. Automation can make this process significantly more efficient by reducing the amount of code the developer has to look through. This thesis introduces three methodologies of automatically reducing a given failing program into its minimal runnable subset. The techniques are based on existing findings in the field of debugging. The automatically minimized program must result in the same runtime error as the original program. The minimization focuses on optimal results in a domain of small and simple applications.

The goal of this thesis is to discuss techniques that are practical for program reduction. These techniques are implemented using Clang LibTooling, a library for standalone Clang tools. The inner workings of each implementation are explained, and their limitations are exposed. The implementations are benchmarked on a set of C and C++ source files. Performance is measured based on the size of the generated output and the running time of the algorithm.