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.

The goal of this thesis is to propose and discuss automated techniques for reducing a given failing program into its minimal runnable subset. We introduce three methodologies that are practical for program reduction. The automatically minimized program must result in the same runtime error as the original program. The process of minimization focuses on producing optimal results for the domain of small and simple applications.

All three techniques are implemented using Clang LibTooling, a library for standalone Clang tools. In the thesis, we explain the inner workings of each implementation and discuss their limitations. Implementations are benchmarked on a set of C and C++ source files. Performance is evaluated with respect to the size of the generated output and the algorithm's running time.