Generated code

We have decided to generate C++ code, for several reasons. First, C++ has been around for many year and has been worked on by many people. Compared to more recent programming languages, there has been a lot of work towards the optimization of C++ compilers. Second, we chose C++ over low-level code such as assembly code because we lack the knowledge in those language to produce well designed code in a reasonable amount of time. We chose C++ over C because of GoLite’s support for unbound slice types. C++ vector library allows us to implement the slice type in a clean fashion and is probably more efficient than any solution we can come up with. We chose C++ over Java, Python or other higher level languages simply because C++ is lower level and should offer better performance.

Code generation pattern

For most of the program’s structure and control flow, such as for loops and if statements, we simply use the constructs provided by C++.

Design decisions

1. There are 2 ways to generate code about alias types. One way is to simply generate the code for the underlying type. The other way uses C++ built-in aliasing feature in **typedef** statements. We have decided to go with the first alternative. Although the second option is a more accurate representation of the source code, generating the basic types directly can improve the readability of the generated code.
2. To match Go’s design, we have decided to make slice into pointers, which means that assignment will only perform a shallow copy and slices will be passed as reference in function calls. In contrast, arrays assignments will be deep copies and will be passed by value in function calls.