CS323 Project 2 Report

Name: 陈宇恒 (Chen Yuheng)

Student ID: 11711111

The bonus features that I have realized:

- Revoke Assumption 6, support variables share the same identifier in different scope.
- Revoke Assumption 7, adopt structural equivalence for struct variables. (with order)

Main Program

The main program is inherited from the program of project 1.

Based on the program of project 1, I did the following augmentations:

- Build 3 **symbol tables** during parsing: global variable symbol table, function symbol table and structure symbol table. These symbol tables are maps from ID to type.
- Add a parsing system for parsing **type** infomation.
- Did some **duplicate checkings** during parsing.
- Did most **type checkings** after the syntax tree building.

Bonus 1

Bonus 1: Revoke Assumption 6, support variables share the same identifier in different scope.

This is achieved by implementing a **symbol table stack** for variables.

The global variable symbol table is in the bottom of this stack. When entering a new scope, a new symbol table will be pushed into the stack, and when exiting a scope, the symbol table on the top will be popped out of the stack.

The search of the symbol table stack is from top to bottom, until search successful. If search failed in the stack bottom (the global variable symbol table), then the whole search failed.

The insertion, however, only checks the symbol table in the stack top, if no duplicate key in the top symbol table, the insertion will success.

Bonus 2

Bonus 2: Revoke Assumption 7, adopt structural equivalence for struct variables. (with order)

This is achieved by using the **Type structure** provided by the document of project 2.

This structure naturally deals with order of structure fields by an **ordered linked list of types**. So we can simply apply a recursive function to compare two structure types.

One point to mention is that if we directly recursively compare two types, it may stuck in an endless loop, because a structure may have a field with the type of itself (as below code shows). To avoid such endless loop, I only apply name equivalence to the structure in a field.

```
struct Node {
    int value;
    struct Node next;
}
```