**README**

输入文件：[txt\input.txt](txt/input.txt)

抽象语法树格式化输出文件：[txt\astOutput.txt](txt/astOutput.txt)

终端语义分析输出文件：[txt\logOutput.txt](txt/logOutput.txt)

hxx输出目录（文件名与用例序号一致）：txt

所有测试用例报表:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 检查的功能 | 输入 | 预期结果 | 实际结果 |
| 1 | 命名冲突 | struct A{  short num;  long num;  }; | 报错，同一个struct空间下，不能有同名变量 | 与预期一致 |
| 2 | 命名冲突 | module space{  struct A{  short a;  };  struct B{  short a;  };  }; | 通过，同一个module下，不同的struct可以有同名变量 | 与预期一致 |
| 3 | 命名冲突 | module space{  struct A{  short a;  };  struct A{  short a;  };  }; | 报错，同一个module下，不能出现同名的struct | 与预期一致 |
| 4 | 命名冲突 | struct a;  struct b;  struct b; | 报错，检查全局下的重名struct | 与预期一致 |
| 5 | 命名冲突 | module a{  struct b{};  };  module a{  struct b{};  }; | 报错，检查全局下的重名module | 与预期一致 |
| 6 | 命名冲突 | module a{  module b{  struct c{};  };  module b{  struct d{};  };  }; | 报错，检查同一module下的重名module | 与预期一致 |
| 7 | 命名冲突 | module a{  struct b{  uint8 c, c;  };  }; | 报错，检查同一member\_list中的对象重名报错 | 与预期一致 |
| 8 | 命名冲突 | module a{  struct b{  uint8 e;  string e;  };  }; | 报错，同一struct内不同类型对象重名报错 | 与预期一致 |
| 9 | 命名冲突 | module a{  struct A{};  struct b{  uint8 e;  A e;  };  }; | 报错，同一struct内与struct类型对象重名报错 | 与预期一致 |
| 10 | 未定义即使用 | struct A{  short a;  B b;  }; | 报错，B结构应该先定义才能引用类型 | 与预期一致 |
| 11 | 未定义即使用 | module space1{  struct B{};  };  module space2{  struct A{  short a;  B b;  };  }; | 报错，虽然B结构定义了，但是命名空间的引用不对，应该是 space1::B | 与预期一致 |
| 12 | 未定义即使用 | module a{  struct b{  int d;  };  }; | 报错，int不在普通类型中，这里视为type\_spec，但int未定义 | 与预期一致 |
| 13 | 未定义即使用 | module a{  struct B{};  };  module a1{  struct C{  a::B b;  ::a::B c;  };  }; | 通过，检查对另一个module中声明的type\_spec对象的声明 | 与预期一致 |
| 14 | 未定义即使用 | module A{  module B{  struct C{  struct D{}i1;  };  };  };  module a1{  struct C{  A::B::C::D b;  ::A::B::C::D c;  };  }; | 通过，检查对另一个多层module多层struct中声明的type\_spec对象的声明 | 与预期一致 |
| 15 | 字面量类型检查 | struct A{  short a='a';  }; | 报错，a是整型变量，字面量却是字符类型 | 与预期一致 |
| 16 | 字面量类型检查 | struct A{  short a=100000;  }; | 报错，short为有符号短整型，最大值不超过 2^15-1 | 与预期一致 |
| 17 | 字面量类型检查 | struct A{  short a=15.24;  }; | 报错，a是整型变量，字面量却是浮点类型 | 与预期一致 |
| 18 | 字面量类型检查 | struct A{  short a[4]=[10,12,45.34,'a'];  }; | 报错，a是整型变量 a是整型数组，数组字面量里必须保证数据类型的统一 | 与预期一致 |
| 19 | 字面量类型检查 | struct A{};  struct B{  A a = 7;  }; | 报错，对结构体对象，不允许进行赋值操作 | 与预期一致 |
| 20 | 字面量类型检查 | struct B{  short a[2\*3.14];  }; | 报错，数组对象的长度声明只能是整型 | 与预期一致 |
| 21 | 字面量类型检查 | struct B{  short a[99999999];  }; | 报错，数组对象的长度声明应在short范围内 | 与预期一致 |
| 22 | 字面量类型检查 | struct B{  float f = 100000000000000000000000;  }; | 报错，普通类型的对象赋值字面量须在对应范围内 | 与预期一致 |
| 23 | .hxx导出（case1） | module space{  struct A{  short i1=10;  };  }; | 与样例hxx内容一致 | 与预期一致 |
| 24 | .hxx导出（case2） | module space{  struct A{  short i1=10;  };  struct B{  long i2=100;  A i3;  };  }; | 与样例hxx内容一致 | 与预期一致 |
| 25 | .hxx导出（case3） | struct A{  short i1=10;  }; | 与样例hxx内容一致 | 与预期一致 |
| 26 | .hxx导出（case4） | module space{  module inner{  struct A{  short i1=10;  };  };  }; | 与样例hxx内容一致 | 与预期一致 |
| 27 | .hxx导出（case5） | module space{  struct A{  short i1=10;  int16 i2=10;  long i3=100;  int32 i4=100;  long long i5=1000;  int64 i6=1000;  unsigned short i7=10;  uint16 i8=10;  unsigned long i9=100;  uint32 i10=100;  unsigned long long i11=1000;  uint64 i12=1000;  char c0='a';  string c1="abc";  boolean c2=true;  float c3=10.901f;  double c4=23.234d;  long double c5=12.23456432235d;  short arr[10]=[0,1,2,3,4,5,6,7,8,9];  };  }; | 与样例hxx内容一致 | 与预期一致 |
| 28 | .hxx导出 | module a{  struct b{  uint16 c = 17;  uint32 d = 18;  };  }; | 内容正确 | 与预期一致 |
| 29 | .hxx导出 | module space{  struct A{  short i1=10;  float f1=12.5;  };  struct B{  long i2=100;  A i3;  };  }; | 内容正确 | 与预期一致 |
| 30 | .hxx导出 | struct A{  uint32 i1=1;  };  struct B{  float i1=3.14;  };  struct C{  string i1="hello";  }; | 内容正确 | 与预期一致 |
| 31 | .hxx导出 | struct A{  string s[9]=["hello"];  }; | 内容正确 | 与预期一致 |

测试结果分析：

语义分析所要求的三点功能完成了检查并正确报错。

.hxx文件按照模板完成了5个用例中所要求的输出内容，并在扩展测试中保持正确。