(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 113608748 A
(43)申请公布日 2021.11.05

(21)申请号 202110812055.2

(22)申请日 2021.07.19

(71)申请人 上海浦东发展银行股份有限公司
地址 200001 上海市黄浦区中山东一路12号

(72)发明人 陈晗 张晶晶 吴德柱

(74)专利代理机构 广州华进联合专利商标代理有限公司 44224
代理人 黄丽霞

(51)Int.Cl.
G06F 8/51(2018.01)

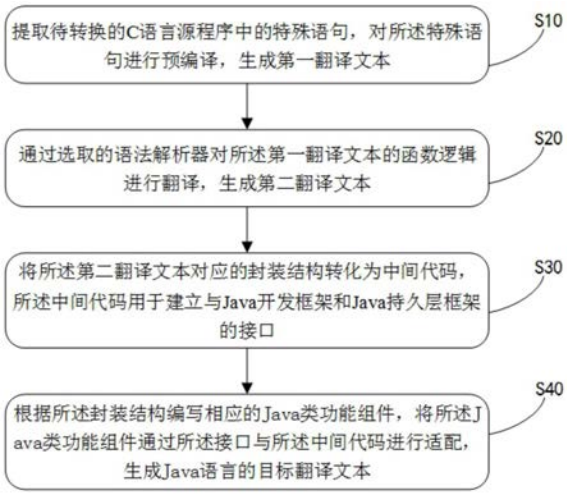
权利要求书2页 说明书9页 附图4页

(54)发明名称

C语言转换Java语言的数据处理方法、装置及设备

(57)摘要

本公开涉及一种C语言转换Java语言的数据处理方法、装置及设备，方法包括提取待转换的C语言源程序中的特殊语句，对所述特殊语句进行预编译，生成第一翻译文本；所述特殊语句包括所述C语言源程序中的非标准C语言语法的语句，所述预编译包括将所述特殊语句替换为标准C语言语句的处理；通过选取的语法解析器对所述第一翻译文本的函数逻辑进行翻译，生成第二翻译文本；将所述第二翻译文本对应的封装结构转化为中间代码，所述中间代码用于建立与Java开发框架和Java持久层框架的接口；将Java类功能组件通过所述接口与所述中间代码进行适配，生成Java语言的目标翻译文本。本公开提高了代码转化的准确性，降低了代码转化过程的膨胀率。



CN 113608748 A

1.一种C语言转换Java语言的数据处理方法,其特征在于,包括如下步骤:

提取待转换的C语言源程序中的特殊语句,对所述特殊语句进行预编译,生成第一翻译文本;所述特殊语句包括所述C语言源程序中的非标准C语言语法的语句,所述预编译包括将所述特殊语句替换为标准C语言语句的处理;

通过选取的语法解析器对所述第一翻译文本的函数逻辑进行翻译,生成第二翻译文本;

将所述第二翻译文本对应的封装结构转化为中间代码,所述中间代码用于建立与Java开发框架和Java持久层框架的接口;

根据所述封装结构编写相应的Java类功能组件,将所述Java类功能组件通过所述接口与所述中间代码进行适配,生成Java语言的目标翻译文本。

2.如权利要求1所述的方法,其特征在于,

所述对所述特殊语言进行预编译之前,所述方法还包括:

提取所述C语言源程序中的注释内容以及所述注释内容的第一位置信息,并将所述注释内容和第一位置信息存放至数据库表中;所述第一位置信息包括所述注释内容的文件名和注释行号;

在所述生成第二翻译文本之后,所述方法还包括:

读取所述数据库表中的所述注释内容和第一位置信息,根据所述注释内容和第一位置信息将所述注释内容插入到所述第二翻译文本或插入到生成的所述目标翻译文本中。

3.如权利要求1所述的方法,其特征在于,

所述对所述特殊语言进行预编译之后,所述方法还包括:

提取所述C语言源程序中的全局符号以及所述全局符号的第二位置信息,并将所述全局符号和第二位置信息存放至数据库表中;所述第二位置信息包括所述全局符号的定义和链接;

在所述生成第二翻译文本之后,所述方法还包括:

读取所述数据库表中的所述全局符号和第二位置信息,根据所述全局符号和第二位置信息将所述全局符号插入到所述第二翻译文本或插入到生成的所述目标翻译文本中。

4.如权利要求1所述的方法,其特征在于,所述通过选取的语法解析器对所述第一翻译文本的函数逻辑进行翻译,生成第二翻译文本包括:

基于所述语法解析器的开发环境定义C语言语法规则并构建抽象语法树;

根据定义的C语言语法规则并生成对应的遍历方法;

遍历所述抽象语法树生成所述第二翻译文本。

5.如权利要求4所述的方法,其特征在于,所述语法解析器采用ANTLR工具,基于ANTLR环境创建g4文件和BaseVisitor文件,所述g4文件用于定义C语言语法规则并构建抽象语法树,所述BaseVisitor文件用于为所述g4文件中的语法规则生成对应的遍历方法。

6.如权利要求1所述的方法,其特征在于,所述通过语法解析器对所述第一翻译文本中的函数进行逻辑翻译生成第二翻译文本之后还包括:

提取所述第二翻译文本中的转向语句和单表达式语句,根据Java语法规则进行转换。

7.如权利要求1所述的方法,其特征在于,所述中间代码为JDBC代码,所述Java开发框架为Spring框架,所述Java持久层框架为Mybatis框架。

8.如权利要求1所述的方法，其特征在于，所述Java类功能组件包括字符串库、数学计算库、系统函数库、公共函数类、数组类和结构体类。

9.如权利要求1所述的方法，其特征在于，所述方法还包括：

读取所述C语言源程序中与简单开放平台SOP相适配的相关函数，将所述相关函数翻写使得所述相关函数与分布式联机服务平台OCP相适配；

在所述目标翻译文本中写入线程上下文类，所述线程上下文类用于管理全局变量；

基于目标翻译文本创建与所述分布式联机服务平台OCP的数据源的JDBC连接。

10.一种C语言转换Java语言的数据处理装置，其特征在于，包括：

预编译模块，用于提取待转换的C语言源程序中的特殊语句，对所述特殊语句进行预编译，生成第一翻译文本；所述特殊语句包括所述C语言源程序中的非标准C语言语法的语句，所述预编译包括将所述特殊语句替换为标准C语言语句的处理；

函数逻辑翻译模块，用于获取所述预编译模块生成的第一翻译文本，并对所述第一翻译文本的函数逻辑进行翻译，生成第二翻译文本；

封装构建翻写模块，用于获取所述函数逻辑翻译模块生成的第二翻译文本，并将所述第二翻译文本对应的封装结构转化为中间代码，所述中间代码用于建立与Java开发框架和Java持久层框架的接口；

功能组件适配模块，用于根据所述封装结构编写相应的Java类功能组件，将所述Java类功能组件通过所述接口与所述中间代码进行适配，生成Java语言的目标翻译文本。

11.一种计算机设备，包括存储器和处理器，所述存储器存储有计算机程序，其特征在于，所述处理器执行所述计算机程序时实现权利要求1至9中任一项所述的方法的步骤。

12.一种计算机程序产品，所述计算机程序产品中包括指令，其特征在于，所述指令被执行时，能够执行权利要求1至9中任一项所述的方法的步骤。

13.一种计算机可读存储介质，其上存储有计算机程序，其特征在于，所述计算机程序被处理器执行时实现权利要求1至9中任一项所述的方法的步骤。

C语言转换Java语言的数据处理方法、装置及设备

技术领域

[0001] 本申请涉及编程语言转换的计算机数据处理技术领域，特别是涉及一种C语言转换Java语言的数据处理方法、装置及设备。

背景技术

[0002] 随着编程语言的不断的发展和变化，编程语言的自动转换被广泛运用在软件移植、维护、升级等领域，可以缩短大量的开发周期，节省软件的开发成本。尤其是C语言和Java语言作为广为使用的高级汇编语言，C语言到Java代码的转换十分必要。目前，国内外对C语言到Java代码转换的工作研究很少，大多采用的转换工具为早期开发的C To Java Converter工具，在产生的Java代码可读性和代码膨胀率等方面具有局限性。

[0003] 在将在C代码到Java代码的移植方案中，较为主流的方式有两种，其一是使用JNI技术将C代码整合到Java中，但是在进行频繁的数据类型转换时代码性能有待进一步提高。另一种是通过GCC编译器，产生Java虚拟机的字节码文件，并将C运行时的内存处理正确映射到Java虚拟机，但是通过GCC编译器实现C语言到字节码文件的转换时，无法处理复杂情况下C运行内存到Java虚拟机的映射。并且两种方案都需要通过单独的接口处理C和Java间的通信，容易造成代码臃肿。

发明内容

[0004] 基于此，有必要针对上述技术问题，提供一种C语言转换Java语言的数据处理方法、装置及设备，其中，一种C语言转换Java语言的数据处理方法，包括如下步骤：

[0005] 提取待转换的C语言源程序中的特殊语句，对所述特殊语句进行预编译，生成第一翻译文本；所述特殊语句包括所述C语言源程序中的非标准C语言语法的语句，所述预编译包括将所述特殊语句替换为标准C语言语句的处理；

[0006] 通过选取的语法解析器对所述第一翻译文本的函数逻辑进行翻译，生成第二翻译文本；

[0007] 将所述第二翻译文本对应的封装结构转化为中间代码，所述中间代码用于建立与Java开发框架和Java持久层框架的接口；

[0008] 根据所述封装结构编写相应的Java类功能组件，将所述Java类功能组件通过所述接口与所述中间代码进行适配，生成Java语言的目标翻译文本。

[0009] 在一个实施例中，所述对所述特殊语言进行预编译之前，所述方法还包括：

[0010] 提取所述C语言源程序中的注释内容以及所述注释内容的第一位置信息，并将所述注释内容和第一位置信息存放至数据库表中；所述第一位置信息包括所述注释内容的文件名和注释行号；

[0011] 在所述生成第二翻译文本之后，所述方法还包括：

[0012] 读取所述数据库表中的所述注释内容和第一位置信息，根据所述注释内容和第一位置信息将所述注释内容插入到所述第二翻译文本或插入到生成的所述目标翻译文本中。

[0013] 在一个实施例中,所述对所述特殊语言进行预编译之后,所述方法还包括:

[0014] 提取所述C语言源程序中的全局符号以及所述全局符号的第二位置信息,并将所述全局符号和第二位置信息存放至数据库表中;所述第二位置信息包括所述全局符号的定义和链接;

[0015] 在所述生成第二翻译文本之后,所述方法还包括:

[0016] 读取所述数据库表中的所述全局符号和第二位置信息,根据所述全局符号和第二位置信息将所述全局符号插入到所述第二翻译文本或插入到生成的所述目标翻译文本中。

[0017] 在一个实施例中,所述通过选取的语法解析器对所述第一翻译文本的函数逻辑进行翻译,生成第二翻译文本包括:

[0018] 基于所述语法解析器的开发环境定义C语言语法规则并构建抽象语法树;

[0019] 根据定义的C语言语法规则并生成对应的遍历方法;

[0020] 遍历所述抽象语法树生成所述第二翻译文本。

[0021] 在一个实施例中,所述语法解析器采用ANTLR工具,基于ANTLR环境创建g4文件和BaseVisitor文件,所述g4文件用于定义C语言语法规则并构建抽象语法树,所述BaseVisitor文件用于为所述g4文件中的语法规则生成对应的遍历方法。

[0022] 在一个实施例中,所述通过语法解析器对所述第一翻译文本中的函数进行逻辑翻译生成第二翻译文本之后还包括:

[0023] 提取所述第二翻译文本中的转向语句和单表达式语句,根据Java语法规则进行转换。

[0024] 在一个实施例中,所述中间代码为JDBC代码,所述Java开发框架为Spring框架,所述Java持久层框架为Mybatis框架。

[0025] 在一个实施例中,所述Java类功能组件包括字符串库、数学计算库、系统函数库、公共函数类、数组类和结构体类。

[0026] 在一个实施例中,所述方法还包括:

[0027] 读取所述C语言源程序中与简单开放平台SOP相适配的相关函数,将所述相关函数翻写使得所述相关函数与分布式联机服务平台OCP相适配;

[0028] 在所述目标翻译文本中写入线程上下文类,所述线程上下文类用于管理全局变量;

[0029] 基于目标翻译文本创建与所述分布式联机服务平台OCP的数据源的JDBC连接。

[0030] 本公开还提供了一种C语言转换Java语言的数据处理装置,包括:

[0031] 预编译模块,用于提取待转换的C语言源程序中的特殊语句,对所述特殊语句进行预编译,生成第一翻译文本;所述特殊语句包括所述C语言源程序中的非标准C语言语法的语句,所述预编译包括将所述特殊语句替换为标准C语言语句的处理;

[0032] 函数逻辑翻译模块,用于获取所述预编译模块生成的第一翻译文本,并对所述第一翻译文本的函数逻辑进行翻译,生成第二翻译文本;

[0033] 封装构建翻写模块,用于获取所述函数逻辑翻译模块生成的第二翻译文本,并将所述第二翻译文本对应的封装结构转化为中间代码,所述中间代码用于建立与Java开发框架和Java持久层框架的接口;

[0034] 功能组件适配模块,用于根据所述封装结构编写相应的Java类功能组件,将所述

Java类功能组件通过所述接口与所述中间代码进行适配，生成Java语言的目标翻译文本。

[0035] 本公开还提供了一种计算机设备，包括存储器和处理器，所述存储器存储有计算机程序，其特征在于，所述处理器执行所述计算机程序时实现上述的一种C语言转换Java语言的数据处理方法的步骤。

[0036] 本公开还提供了一种计算机程序产品，所述计算机程序产品中包括指令，其特征在于，所述指令被执行时，能够执行上述的一种C语言转换Java语言的数据处理方法的步骤。

[0037] 本公开还提供了一种计算机可读存储介质，其上存储有计算机程序，其特征在于，所述计算机程序被处理器执行时实现上述的一种C语言转换Java语言的数据处理方法的步骤。

[0038] 上述C语言转换Java语言的数据处理方法、装置、设备、计算机程序产品及存储介质的有益效果如下：

[0039] 本公开在将C语言转换为Java语言时，通过在函数逻辑翻译之前进行预编译，替换非标准C语言语法的特殊语句，提高了翻译文本的可读性，减少了翻译文本的后续优化处理步骤，提高了函数逻辑翻译的准确性；在对封装结构进行转换时，通过翻写中间代码，建立与Java开发框架和Java持久层框架的接口，通过中间代码实现C语言中封装结构至Java类功能组件的过渡，降低了代码转化过程的膨胀率，且最后生成的目标翻译文本不依赖C语言源程序的C语言源程序和C语言源程序原平台的操作环境。

附图说明

[0040] 为了更清楚地说明本申请实施例或传统技术中的技术方案，下面将对实施例或传统技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

[0041] 图1为一实施例中提供的C语言转换Java语言的数据处理方法的流程示意图；

[0042] 图2为一实施例中提供的注释提取流程图示意图；

[0043] 图3为一实施例中提供的全局符号提取流程图示意图；

[0044] 图4为一实施例中提供的C语言转换Java语言的数据处理方法的流程示意图；

[0045] 图5为一实施例中提供的函数逻辑进行翻译生成第二翻译文本的流程示意图；

[0046] 图6为一实施例中提供的构建抽象语法树的流程示意图；

[0047] 图7为一实施例中提供的C语言转换Java语言的数据处理装置的结构示意图。

具体实施方式

[0048] 为了便于理解本申请，下面将参照相关附图对本申请进行更全面的描述。附图中给出了本申请的实施例。但是，本申请可以以许多不同的形式来实现，并不限于本文所描述的实施例。相反地，提供这些实施例的目的是使本申请的公开内容更加透彻全面。

[0049] 除非另有定义，本文所使用的所有的技术和科学术语与属于本申请的技术领域的技术人员通常理解的含义相同。本文中在本申请的说明书中所使用的术语只是为了描述具体的实施例的目的，不是旨在于限制本申请。

[0050] 需要说明的是，本公开的说明书和权利要求书及上述附图中的术语“第一”、“第二”等是用于区别类似的对象，而不必用于描述特定的顺序或先后次序。应该理解这样使用的数据在适当情况下可以互换，以便这里描述的本公开的实施例能够以除了在这里图示或描述的那些以外的顺序实施。以下示例性实施例中所描述的实施方式并不代表与本公开相一致的所有实施方式。相反，它们仅是与如所附权利要求书中所详述的、本公开的一些方面相一致的装置和方法的例子。术语“包括”、“包含”或者其任何其它变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、产品或者设备不仅包括那些要素，而且还包括没有明确列出的其它要素，或者是还包括为这种过程、方法、产品或者设备所固有的要素。在没有更多限制的情况下，并不排除在包括所述要素的过程、方法、产品或者设备中还存在另外的相同或等同要素。例如若使用到第一，第二等词语用来表示名称，而并不表示任何特定的顺序。

[0051] 本公开所提供的C语言转换Java语言的数据处理方法，可以应用于在编程语言在软件移植，维护等领域中的自动转换，主要用于将C语言转换为Java语言。不同的编程语言间表达方式和标准库不尽相同。现以涉及大型分布式架构系统下的代码的自动翻译和适配的应用场景为例，请参阅图1，本实施例提供一种C语言转换Java语言的数据处理方法，包括如下步骤：

[0052] 步骤S10：提取待转换的C语言源程序中的特殊语句，对特殊语句进行预编译，生成第一翻译文本；特殊语句包括C语言源程序中的非标准C语言语法的语句，预编译包括将特殊语句替换为标准C语言语句的处理。

[0053] 预编译通常在代码翻译中做些代码文本的替换工作。通过解析C语言源程序，以预设的规则可以提取出非标准C语言语法的特殊语句，这里特殊语句包括宏定义和SQL语句，例如筛选以“#”符号开头的宏定义进行替换。又例如针对SQL语句，由于SQL语句在执行时往往会重复执行或者每次执行时只有个别值不同，因此可以用占位符替换，即将SQL语句模板化，实现一次编译、多次运行，减少了解析优化的过程。经过步骤S10的预编译，经过代码替换的C语言源程序形成了第一翻译文本，后续处理在第一翻译文本的基础上进行处理。

[0054] 步骤S20：通过选取的语法解析器对第一翻译文本的函数逻辑进行翻译，生成第二翻译文本。

[0055] 将第一翻译文本通过语法解析器翻译成第二翻译文本，这里语法解析器主要用于将第一翻译文本的函数逻辑进行翻译。在C语言源程序经过预编译后，第一翻译文本中的语句均为标准的C语言语法，便于通过语法解析器进行语法分析，方便对第一翻译文本中的业务逻辑代码进行逻辑翻译。

[0056] 步骤S30：将第二翻译文本对应的封装结构转化为中间代码，中间代码用于建立与Java开发框架和Java持久层框架的接口。

[0057] 在C语言中往往会应用大量的封装结构，例如使用C语言对各个库表的增删改查操作的封装，是模板化的代码。针对封装结构，本实施例采用了转化为中间代码的技术方案。中间代码不仅对封装结构进行了翻写，还用于建立与Java开发框架和Java持久层框架的接口，通过将中间代码封装，可以适配Java开发框架和Java持久层框架。

[0058] 步骤S40：根据封装结构编写相应的Java类功能组件，将Java类功能组件通过接口与中间代码进行适配，生成Java语言的目标翻译文本。

[0059] 根据封装结构编写与封装结构对应的Java类功能组件，将Java类功能组件通过步骤S30的接口与中间代码进行适配，便于Java类功能组件通过该接口访问、调用、查询、更新中间代码，最终实现第二翻译文本中的封装结构与Java类功能组件之间的翻译适配。

[0060] 本实施例提供的C语言转换Java语言的数据处理方法通过对C语言程序的预编译、函数逻辑翻译和封装结构的翻写实现了C语言翻译成Java语言的目的，其中通过对C语言源程序的C语言规范化处理，提高了翻译文本的可读性，减少了翻译文本的后续优化处理步骤；以及通过中间代码实现C语言中封装结构至Java类功能组件的过渡，降低了代码转化过程的膨胀率，且最后生成的目标翻译文本不依赖C语言源程序的C语言源程序和C语言源程序原平台的操作环境。

[0061] 应该理解的是，虽然图1的流程图中的各个步骤按照箭头的指示依次显示，但是这些步骤并不是必然按照箭头指示的顺序依次执行。除非本文中有明确的说明，这些步骤的执行并没有严格的顺序限制，这些步骤可以以其它的顺序执行。而且，图1中的至少一部分步骤可以包括多个步骤或者多个阶段，这些步骤或者阶段并不必然是在同一时刻执行完成，而是可以在不同的时刻执行，这些步骤或者阶段的执行顺序也不必然是依次进行，而是可以与其它步骤或者其它步骤中的步骤或者阶段的至少一部分轮流或者交替地执行。

[0062] 在一个实施例中，结合图2所示的注释提取流程示意图，对特殊语言进行预编译之前，上述方法还包括：

[0063] 步骤A10：提取C语言源程序中的注释内容以及注释内容的第一位置信息，并将注释内容和第一位置信息存放至数据库表中；第一位置信息包括注释内容的文件名和注释行号。

[0064] 在C语言源程序中，C语言源程序中往往存在注释内容，对C语言源程序进行解释说明。注释内容有助于提高程序代码的可读性，方便后续对程序的理解和维护。本实施例通过提取C语言源程序的C语言源程序中的注释内容以及注释内容的第一位置信息，将注释内容进行存储。

[0065] 在生成第二翻译文本之后，上述方法还包括：

[0066] 步骤A20：读取数据库表中的注释内容和第一位置信息，根据注释内容和第一位置信息将注释内容插入到第二翻译文本或插入到生成的目标翻译文本中。

[0067] 在C语言源程序向Java语言的初步转换生成的第二翻译文本，或者C语言源程序向Java语言的最终转换生成的目标翻译文本中，将预先存储的注释内容根据第一位置信息插入对应位置。本实施例避免了不被计算机编译的注释内容丢失，确保经过翻译后的代码具有注释内容，提高了程序代码的可读性。

[0068] 在一个实施例中，结合图3所示的全局符号提取流程图示意图，对特殊语言进行预编译之后，上述方法还包括：

[0069] 步骤B10：提取C语言源程序中的全局符号以及全局符号的第二位置信息，并将全局符号和第二位置信息存放至数据库表中；第二位置信息包括全局符号的定义和链接。

[0070] 在C语言源程序中，C语言源程序中往往存在重复出现的全局符号，例如结构体定义、函数定义和全局变量，将C语言源程序的C语言源程序中的全局符号进行提取和存储，同时存储全局符号定义和链接。

[0071] 在生成第二翻译文本之后，上述方法还包括：

[0072] 步骤B20:读取数据库表中的全局符号和第二位置信息,根据全局符号和第二位置信息将全局符号插入到第二翻译文本或插入到生成的目标翻译文本中。

[0073] 在C语言源程序向Java语言的初步转换生成的第二翻译文本,或者C语言源程序向Java语言的最终转换生成的目标翻译文本中,将预先存储的全局符号根据第二位置信息插入对应位置。

[0074] 在上述实施例中,结合图4,C语言转换Java语言的数据处理方法的流程为先进行注释提取,再进行预编译获得第一翻译文本,在将第一翻译文本进行函数逻辑翻译之间进行全局符号的提取。第二翻译文本经过封装结构转化和Java类功能组件适配,最后生成Java语言的目标翻译文本。

[0075] 在一个实施例中,结合附图5,上述步骤S20包括:

[0076] 步骤S202:基于语法解析器的开发环境定义C语言语法规则并构建抽象语法树。

[0077] 结合附图6,语法解析器通常是指进行语法检查、并构建由输入的单词组成的数据结构(一般是语法分析树、抽象语法树等层次化的数据结构)。通过语法解析器将第一翻译文本进行词法分析,分离成一个个“单词”,将单词流作为输入进行语法分析,并根据C语言语法规则构建抽象语法树。

[0078] 步骤S204:根据定义的C语言语法规则并生成对应的遍历方法,将步骤S202中的每一条C语言语法规则对应生成遍历方法。

[0079] 步骤S206:遍历抽象语法树生成第二翻译文本。根据遍历方法对抽象语法树进行遍历,获得第二翻译文本。

[0080] C语言代码翻写时,不同的符号在不同的位置具有完全不同的意义。同样的代码,在不同的上下文中,也具有不同意义,通过多次遍历抽象语法树,提高翻译精确度,最后生成第二翻译文本。

[0081] 在上述实施例中,语法解析器可以采用ANTLR工具,基于ANTLR环境创建g4文件和BaseVisitor文件,g4文件用于定义C语言语法规则并构建抽象语法树,BaseVisitor文件用于为g4文件中的语法规则生成对应的遍历方法。

[0082] 在一个实施例中,上述步骤S20之后还包括:

[0083] 提取第二翻译文本中的转向语句和单表达式语句,根据Java语法规则进行转换。

[0084] 在得到第二翻译文本后,可以对其进行语法修复。针对第二翻译文本中的转向语句,例如return语句、break语句、goto语句等,以及单表达式语句,根据Java语法规则进行转换,再对代码进行格式化,调整各个语句的缩进、空格,使代码更美观和易于阅读。

[0085] 在一个实施例中,步骤S30中采用的中间代码为JDBC代码,Java开发框架为Spring框架,Java持久层框架为Mybatis框架。JDBC通常是指Java数据库连接,全称为Java Database Connectivity,是Java语言中用来规范客户端程序如何来访问数据库的应用程序接口,提供了诸如查询和更新数据库中数据的方法。

[0086] 在一个实施例中,Java类功能组件包括字符串库、数学计算库、系统函数库、公共函数类、数组类和结构体类。部分Java类功能组件的适配转换情况如下表:

[0087]

Java类功能组件	功能描述
StringLib.Java	字符串库，适配C语言底层字符串函数，如strcmp(),strncpy(), strlen()等。
MathLib.Java	数学计算库，适配C语言中的计算类型函数，如abs(), pow(), sqrt(), floor()等。
SystemLib.Java	系统函数库，适配C语言的系统函数，如open(), close(), mkdir(), write()等。
CommonFunction.Java	公共函数类，针对老核心中的公共函数进行适配，如读写交易公共字段函数ReadComfld()，WriteComfld()以及错误返回函数ReturnError()等。
CArray.Java	数组类，针对C语言的数组和指针，在Java中统一采用数组结构进行适配。
CStruct.Java	结构体类，针对C语言中的结构体，定制特殊的Java类进行适配。

[0088] 除了上表中的Java类功能组件，还包括其他Java类功能组件。针对C语言中出现的基本数据类型，还分别编写了CChar、CDouble和CLong等Java类功能组件进行转换，并封装JDBC以适配Spring、Mybatis框架。

[0089] 在一个实施例中，上述方法还包括：

[0090] 步骤S50：读取C语言源程序中与简单开放平台SOP相适配的相关函数，将相关函数翻写使得相关函数与分布式联机服务平台OCP相适配。

[0091] 简单开放平台SOP是基于Spring Cloud实现的一个开放平台解决方案项目，能够让开发团队快速得搭建起自己的开放平台。简单开放平台SOP提供了两种接口调用方式，通过配置后，使得项目具备接口提供能力。这里的C语言源程序以作为简单开放平台SOP的项目为例，需要对相关函数进行翻译，相关函数可以指代配置接口调用的函数，例如ReadComfld、WriteComfld、ReturnError等函数。

[0092] 在实际应用中的大型业务场合，往往基于分布式联机服务平台OCP。分布式联机服务平台OCP中的应用程序分布在不同计算机上，通过网络来共同完成一项任务，通常为服务器/客户端模式。当将应用程序实现从C语言至Java的转换过程中，还需要将转换后的程序通过简单开放平台SOP提供的接口适配分布式联机服务平台OCP。

[0093] 步骤S60：在目标翻译文本中写入线程上下文类，线程上下文类用于管理全局变量。例如写入开发的ThreadContext类，用于交易级全局变量管理。

[0094] 步骤S70：基于目标翻译文本创建与分布式联机服务平台OCP的数据源的JDBC连

接。

[0095] 在目标翻译文本中通过分布式联机服务平台OCPOCP管理的数据源，创建JDBC连接，使得转换后的程序与底层数据库完全解耦，可以连informix，也可以连DDF或者mysql。

[0096] 本实施例可以实现翻译后的Java程序代码可以自动适配分布式架构系统，无需依赖翻译前的源码和平台框架，便于部署docker容器。

[0097] 基于上述C语言转换Java语言的数据处理方法的描述，本公开还提供C语言转换Java语言的数据处理装置。装置可以包括使用了本说明书实施例方法的系统(包括分布式系统)、软件(应用)、模块、组件、服务器、客户端等并结合必要的实施硬件的装置。基于同一创新构思，本公开实施例提供的一个或多个实施例中的装置如下面的实施例。由于装置解决问题的实现方案与方法相似，因此本说明书实施例具体的装置的实施可以参见前述方法的实施，重复之处不再赘述。以下所使用的，术语“单元”或者“模块”可以实现预定功能的软件和/或硬件的组合。尽管以下实施例所描述的装置较佳地以软件来实现，但是硬件，或者软件和硬件的组合的实现也是可能并被构想的。

[0098] 在一个实施例中，请参阅图7，提供了一种C语言转换Java语言的数据处理装置，包括：

[0099] 预编译模块，用于提取待转换的C语言源程序中的特殊语句，对特殊语句进行预编译，生成第一翻译文本；特殊语句包括C语言源程序中的非标准C语言语法的语句，预编译包括将特殊语句替换为标准C语言语句的处理。

[0100] 函数逻辑翻译模块，用于获取预编译模块生成的第一翻译文本，并对第一翻译文本的函数逻辑进行翻译，生成第二翻译文本。

[0101] 封装构建翻写模块，用于获取函数逻辑翻译模块生成的第二翻译文本，并将第二翻译文本对应的封装结构转化为中间代码，中间代码用于建立与Java开发框架和Java持久层框架的接口。

[0102] 功能组件适配模块，用于根据封装结构编写相应的Java类功能组件，将Java类功能组件通过接口与中间代码进行适配，生成Java语言的目标翻译文本。

[0103] 关于C语言转换Java语言的数据处理装置的具体限定可以参见上文中对于C语言转换Java语言的数据处理方法的限定，在此不再赘述。上述C语言转换Java语言的数据处理装置中的各个模块可全部或部分通过软件、硬件及其组合来实现。上述各模块可以硬件形式内嵌于或独立于计算机设备中的处理器中，也可以以软件形式存储于计算机设备中的存储器中，以便于处理器调用执行以上各个模块对应的操作。需要说明的是，本申请实施例中对模块的划分是示意性的，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式。

[0104] 基于前述方法实施例描述，本公开提供的装置的另一个实施例中，提供了一种计算机设备，包括存储器和处理器，存储器存储有计算机程序，处理器执行计算机程序时实现上述的一种C语言转换Java语言的数据处理方法的步骤。

[0105] 在一个实施例中，还提供了一种计算机程序产品，计算机程序产品中包括指令，指令被执行时，能够执行上述的一种C语言转换Java语言的数据处理方法的步骤。

[0106] 在一个实施例中，还提供了一种计算机可读存储介质，其上存储有计算机程序，计算机程序被处理器执行时实现上述的一种C语言转换Java语言的数据处理方法的步骤。

[0107] 本领域普通技术人员可以理解实现上述实施例方法中的全部或部分流程，是可以

通过计算机程序来指令相关的硬件来完成,的计算机程序可存储于一非易失性计算机可读取存储介质中,该计算机程序在执行时,可包括如上述各方法的实施例的流程。其中,本申请所提供的各实施例中所使用的对存储器、存储、数据库或其它介质的任何引用,均可包括非易失性和易失性存储器中的至少一种。非易失性存储器可包括只读存储器(Read-Only Memory,ROM)、磁带、软盘、闪存或光存储器等。易失性存储器可包括随机存取存储器(Random Access Memory,RAM)或外部高速缓冲存储器。作为说明而非局限,RAM可以是多种形式,比如静态随机存取存储器(Static Random Access Memory,SRAM)或动态随机存取存储器(Dynamic Random Access Memory,DRAM)等。

[0108] 在本说明书的描述中,参考术语“有些实施例”、“其他实施例”、“理想实施例”等的描述意指结合该实施例或示例描述的具体特征、结构、材料或者特征包含于本发明的至少一个实施例或示例中。在本说明书中,对上述术语的示意性描述不一定指的是相同的实施例或示例。

[0109] 上所述实施例的各技术特征可以进行任意的组合,为使描述简洁,未对上述实施例各个技术特征所有可能的组合都进行描述,然而,只要这些技术特征的组合不存在矛盾,都应当认为是本说明书记载的范围。

[0110] 本说明书中上述方法、设备及存储介质的各个实施例均采用递进的方式描述,各个实施例之间相同相似的部分互相参见或参照对应的方法实施例描述即可,每个实施例重点说明的都是与其它实施例的不同之处。相关之处参见方法实施例的部分说明即可。具体的可以根据前述方法实施例的描述的可以得到,且都应属于本申请所保护的实施范围之内,在此不做逐个实施例实现方案的赘述。

[0111] 以上所述实施例仅表达了本申请的几种实施方式,其描述较为具体和详细,但并不能因此而理解为对申请专利范围的限制。应当指出的是,对于本领域的普通技术人员来说,在不脱离本申请构思的前提下,还可以做出若干变形和改进,这些都属于本申请的保护范围。因此,本申请专利的保护范围应以所附权利要求为准。

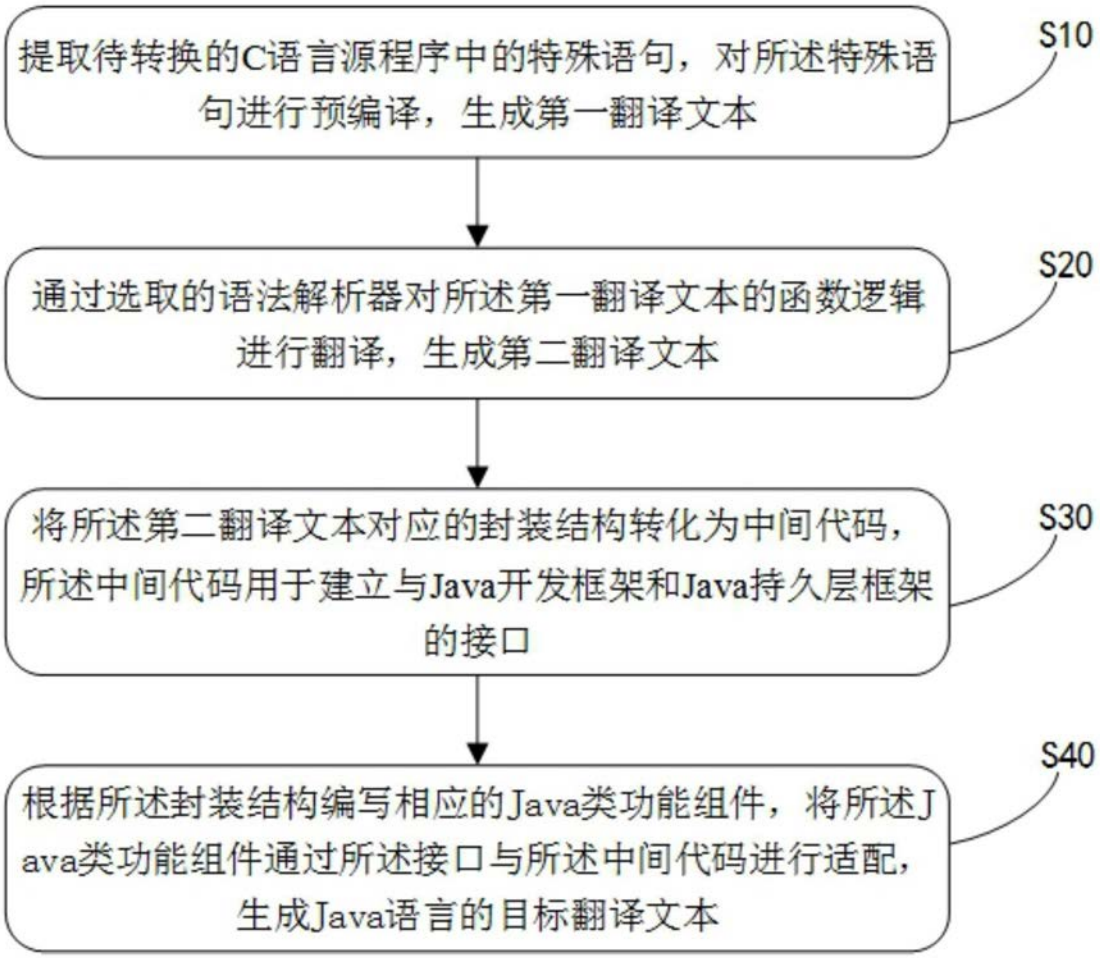

提取待转换的C语言源程序中的特殊语句，对所述特殊语句进行预编译，生成第一翻译文本
S10
通过选取的语法解析器对所述第一翻译文本的函数逻辑进行翻译，生成第二翻译文本
S20
将所述第二翻译文本对应的封装结构转化为中间代码，所述中间代码用于建立与Java开发框架和Java持久层框架的接口
S30
根据所述封装结构编写相应的Java类功能组件，将所述Java类功能组件通过所述接口与所述中间代码进行适配，生成Java语言的目标翻译文本
S40

图1

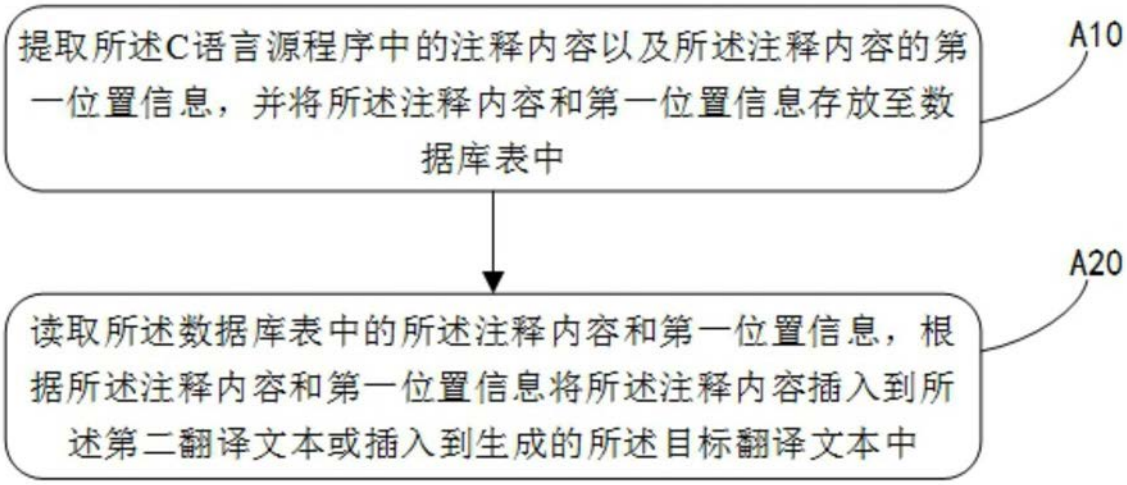

提取所述C语言源程序中的注释内容以及所述注释内容的第一位置信息，并将所述注释内容和第一位置信息存放至数据库表中
A10
读取所述数据库表中的所述注释内容和第一位置信息，根据所述注释内容和第一位置信息将所述注释内容插入到所述第二翻译文本或插入到生成的所述目标翻译文本中
A20

图2

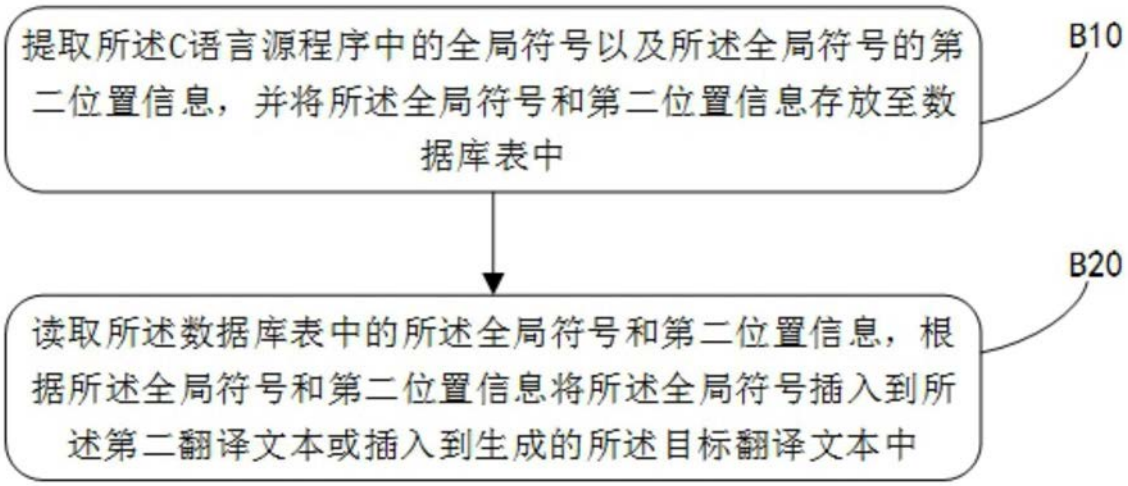

提取所述C语言源程序中的全局符号以及所述全局符号的第二位置信息，并将所述全局符号和第二位置信息存放至数据库表中
B10
读取所述数据库表中的所述全局符号和第二位置信息，根据所述全局符号和第二位置信息将所述全局符号插入到所述第二翻译文本或插入到生成的所述目标翻译文本中
B20

图3

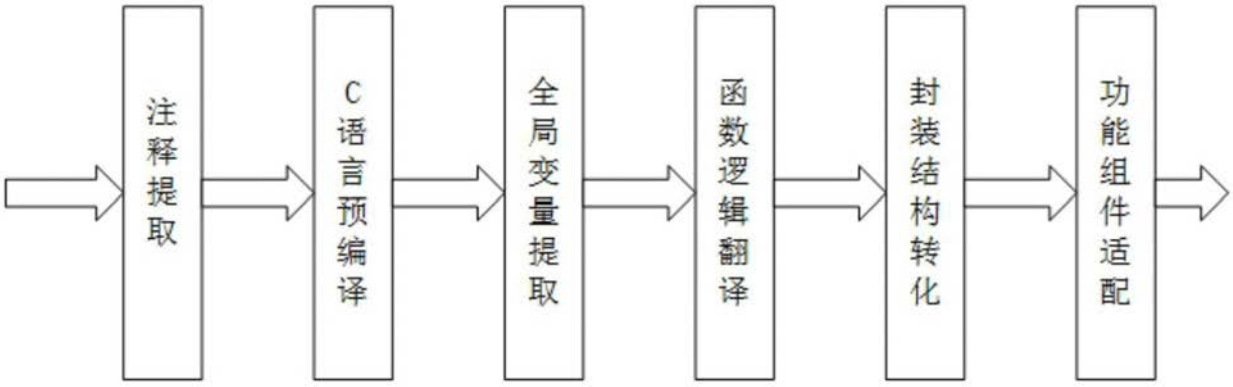

注释提取
C语言预编译
全局变量提取
函数逻辑翻译
封装结构转化
功能组件适配

图4

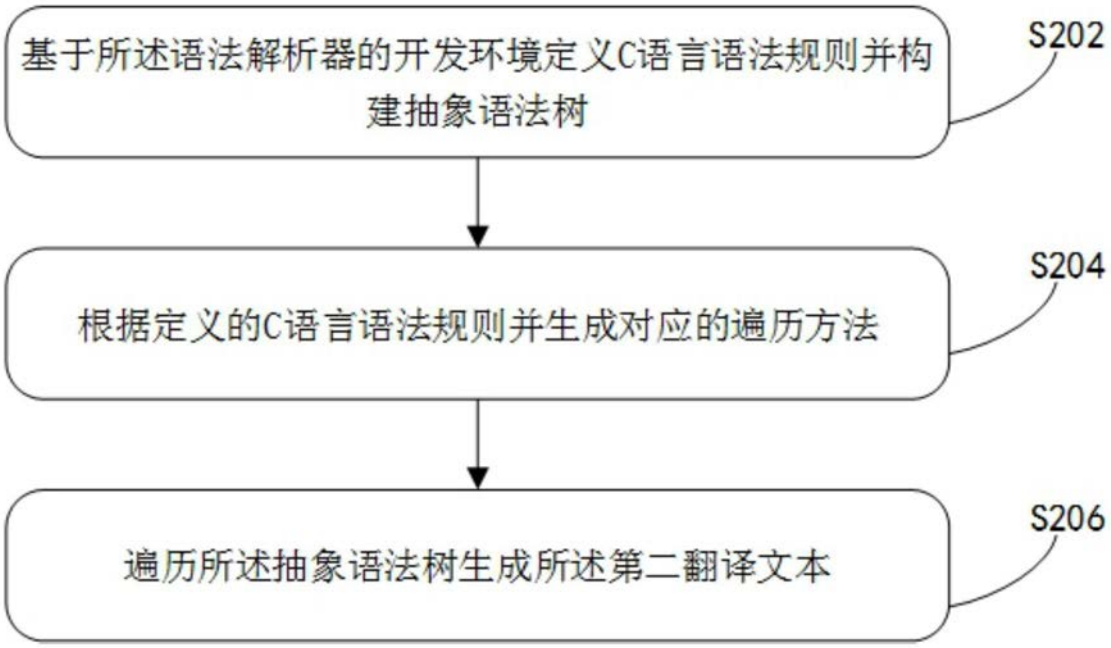

基于所述语法解析器的开发环境定义C语言语法规则并构建抽象语法树
S202
根据定义的C语言语法规则并生成对应的遍历方法
S204
遍历所述抽象语法树生成所述第二翻译文本
S206

图5

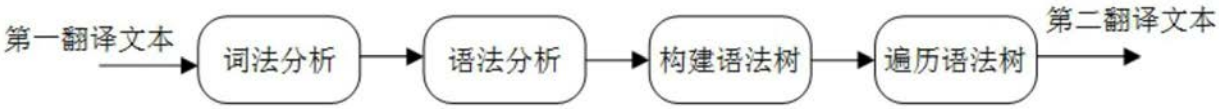

第一翻译文本
词法分析
语法分析
构建语法树
遍历语法树
第二翻译文本

图6

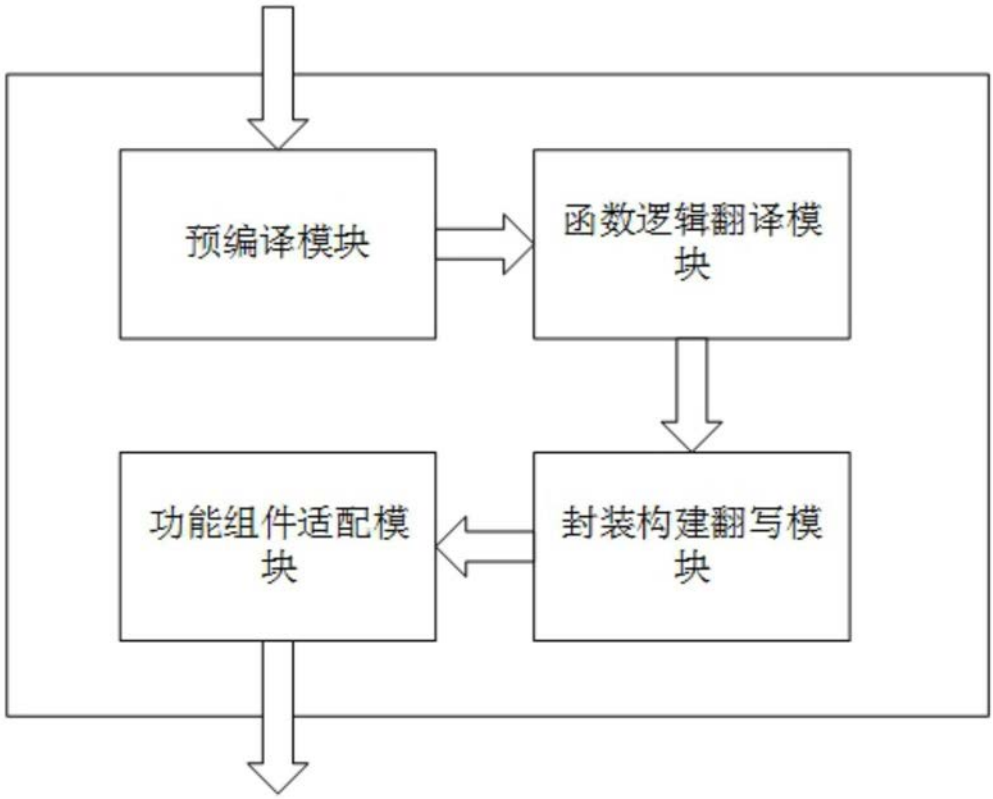


图7