Е.Е. КАНУНОВА

Муромский институт (филиал) ФГБОУ ВПО «Владимирский государственный университет имени А.Г. и Н.Г. Столетовых», г. Муром

Разработка программы – конвертера исходного кода программ

УДК 004.912: 004.421, 004.438

Статья посвящена вопросам разработки программы – конвертера исходного кода программы, написанной на языке C++ в аналогичный код на языке Java и обратно. Описана структура конвертера. Рассмотрены принцы реализации лексического, синтаксического и семантического блоков программы. Приведены результаты тестирования конвертера.

Использование компьютерных и информационных технологий в промышленности [1-7], в медицине [8-11], при защите информации [12], при обработке географических карт [13-15], в музейном и архивном деле [16 - 19] и других сферах человеческой деятельности ускоряет процесс обработки данных, а также предоставляет намного больше функциональных возможностей в отличие от бумажных и ручных технологий. В статье описываются основные вопросы разработки программы – конвертера исходного кода программы, написанной на языке C++ в аналогичный код языка Java, и обратно.

Разрабатываемая программа будет полезным инструментом для людей знающих один из этих языков и желающих изучить другой. Например, данную разработку можно будет использовать при изучении студентами направления «Информационные системы и технологии» (кафедра «Информационные системы» МИВлГУ) таких дисциплин, как «Объектно-ориентированное программирование», «Сравнительный анализ языков программирования» и других.

Рассмотрим схему работы транслятора. Разрабатываемый конвертер это по сути транслятор, с той лишь разницей, что транслятор формирует объектный код, а конвертер аналогичную программу на другом языке программирования. Перевод программы с одного языка на другой, в общем случае, состоит в изменении алфавита, лексики и синтаксиса языка программы с сохранением её семантики. Процесс трансляции исходной программы разбивается на несколько независимых фаз, которые реализуются соответствующими блоками транслятора. Удобно считать основными фазами трансляции лексический анализ, синтаксический анализ, семантический анализ и синтез программы[20]. Схема транслятора представлена на рисунке 1.

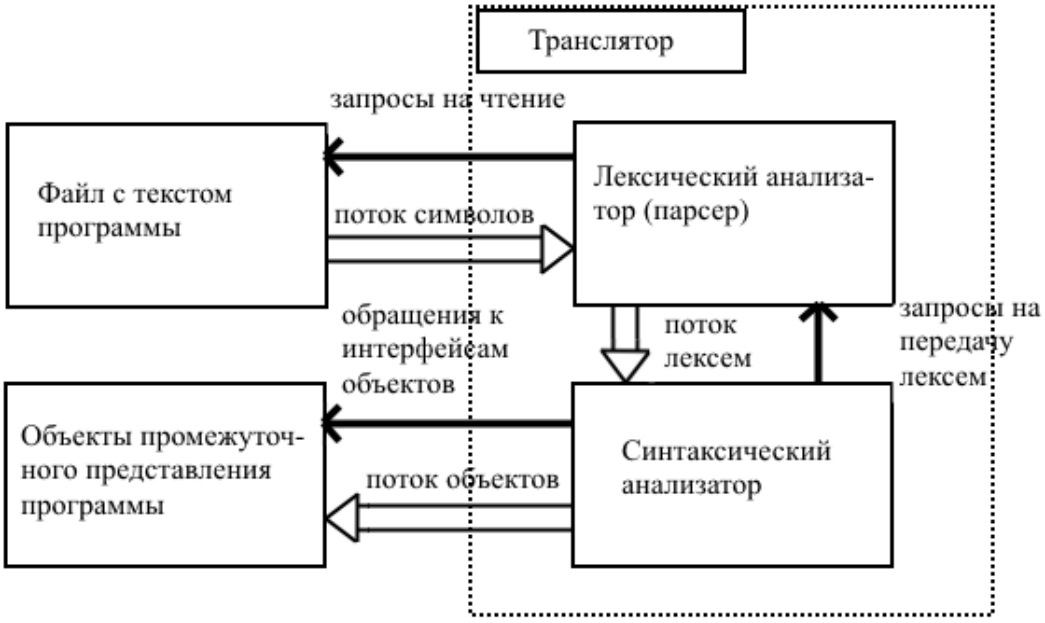

Рис. 1. Упрощенная функциональная модель транслятора.

Цель лексического анализа — перевод исходной программы на внутренний язык транслятора, в котором ключевые слова, идентификаторы, метки и константы приведены к одному формату и заменены условными кодами: числовыми или символьными, которые называются дескрипторами. Каждый дескриптор состоит из двух частей: класса (типа) лексемы и указателя на адрес в памяти, где хранится информация о конкретной лексеме. Одновременно с переводом исходной программы на внутренний язык на этапе

лексического анализа проводится лексический контроль — выявление в программе недопустимых слов [21].

Синтаксический анализатор воспринимает выход лексического анализатора и переводит последовательность образов лексем в форму промежуточной программы. Промежуточная программа является, по существу, представлением синтаксического дерева программы. Последнее отражает структуру исходной программы, т.е. порядок и связи между её операторами. В ходе построения синтаксического дерева выполняется синтаксический контроль — выявление синтаксических ошибок в программе [22].

Синтез программы начинается, как правило, с распределения и выделения памяти для основных программных объектов. Затем производится исследование каждого предложения исходной программы и генерируются семантически эквивалентные предложения. В качестве входной информации здесь используется синтаксическое дерево программы и выходные таблицы лексического анализатора — таблица идентификаторов, таблица констант и другие. Анализ дерева позволяет выявить последовательность генерируемых команд объектной программы, а по таблице идентификаторов определяются типы команд, которые допустимы для значений операндов в генерируемых командах (например, какие требуется породить команды: с фиксированной или плавающей точкой и т.д.).

Непосредственно генерации программы часто предшествует семантический анализ, который включает различные виды семантической обработки. Один из видов — проверка семантических соглашений в программе. Примеры таких соглашений: единственность описания каждого идентификатора в программе, определение переменной производится до её использования и т.д. Семантический анализ может выполняться и на более поздних фазах трансляции, например, на фазе оптимизации программы, которая тоже может включаться в транслятор [20].

Программа, описываемая в данной статье, разрабатывалась под платформу Windows XP\Vista\7\8.1, но так же возможна её сборка под Linux и Android средствами QT.

В основу построения конвертера положен синтаксически - управляемый метод обработки языков. Конвертер включает в свой состав следующие блоки:

- лексического анализа;
- синтаксического анализа;
- семантической обработки и генерации программы;
- визуализации и взаимодействия с пользователем;
- генерации отчётов.

Взаимодействие этих блоков и пользователя представлено на рисунке 2.

При разработке лексического анализатора использовалась теория регулярных языков и конечных автоматов. В рамках этой теории классы однотипных лексем (идентификаторы, константы и т. д.) рассматриваются как формальные языки (язык идентификаторов, язык констант и т.д.), множество предложений которых описывается с помощью соответствующей порождающей грамматики.

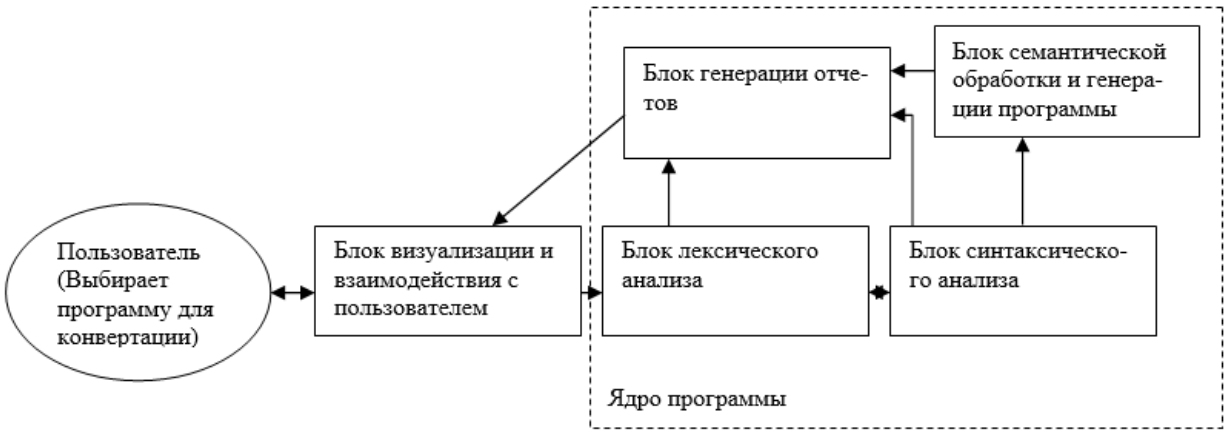

Рис. 2. Взаимодействие блоков программы.

Матрица переходов конечного автомата строится следующим образом: столбцы матрицы соответствуют символам из входного алфавита, строки — символам из алфавита состояний, а элементы матрицы соответствуют состояниям, в которые переходит автомат для данной комбинации входного символа и символа состояния. Для каждого типа лексем используется свой конечный распознаватель. Первоначально в тексте исходной программы лексический анализатор выделяет последовательность символов,

которая по его предположению должна быть лексемой. Затем проводится идентификация лексемы. Она выполняется путём сравнения лексемы с эталонным значением. При успешной идентификации значение лексемы помещается в таблицу идентификации лексем данного класса. Формируется дескриптор лексемы. Для каждого типа идентификатора в таблице хранится специфическая информация. Так, например, для имён переменных: тип (вещественный, целый, строка и т.д.); вид (простая переменная, массив, указатель и т.д.) и т.п.

Синтаксический анализ в конвертере выполнен по нисходящей стратегии. На выходе синтаксического анализа мы имеем набор блоков: блок подключаемых модулей, в нем содержится информация об используемых модулях; блок типов, описываются новые типы; блок переменных, все глобальные переменные; блок констант; блок меток; блок функций; тело программы (обязательная часть). Тело программы или функции представляется в виде синтаксического дерева, в котором присутствуют все операторы и функции. Дерево синтаксического разбора имеет разное количество «веток» и «листьев», все зависит от кода.

При разработке программы-конвертера реализованы следующие взаимосвязанные модули:

1. Модуль импорта - экспорта внешних данных.
2. Модуль генерации отчёта.
3. Модуль представления исходного кода программы во внутренний язык (семантическое дерево). К функциям которого можно отнести:

- разбор исходного кода на лексемы;
- генерации таблицы идентификаторов;
- сопоставление каждой лексемы с таблицей идентификаторов;
- генерация дескрипторного текста из полученных связей каждой лексемы с идентификатором;
- генерация из дескрипторного текста программы синтаксического дерева разбора;
- передача синтаксического дерева разбора в следующий модуль.

4. Модуль семантического анализа модели программы, подсветка синтаксиса и генерация аналогичной программы. В функции которого входит:

- проверка семантики программы по синтаксическому дереву программы;

- подсветка синтаксиса исходного кода программы.

- генерация программы на другом языке программирования с использованием синтаксического дерева и описания выходного языка.

Приведем результаты тестирования разработанной программы при выполнении этапа преобразования кода с языка C++ в аналогичный код на языке Java. Исходный код

```
#include <iostream>
using namespace std;
 int main()
  {char m1 = '=';
  for (int i=0; i < 10;i++ )
    {cout<<m1;}
  cout<<endl;
  for (int i=0; i < 5;i++ )
    {cout<<m1;
    for (int j=0; j < 8;j++ )
      {cout<<" ";}
    cout<<m1<<endl;}
  for (int i=0; i < 10;i++ )
    {cout<<m1;}
  return 0;
  }
```

Дескрипторный текст:

```
<GLOB>
 <title>
  <include>#include</include>
  <file>iostream</file>
 </title>
 <glob_obj>
  <com>using namespace </com>
  <data>std</data>
 </glob_obj>
 <function_body>
  <type>int </type>
  <name> main</name>
 </function_body>
 <skobka>
  <comx>{</comx>
  <variable>
   <type>char </type>
   <name>m1 </name>
   <data> '='</data>
  </variable>
  <FOR>
   <com>for</com>
   <data>int i=0</data>
   <condition> i &lt; 10</condition>
   <operation>i++</operation>
  </FOR>
```

```
  <skobka>
   <comx>{</comx>
   <output_data>
    <com>cout</com>
    <data>m1</data>
   </output_data>
   <comx>}</comx>
  </skobka>
  <output_data>
   <com>cout</com>
   <data>endl</data>
  </output_data>
  <FOR>
   <com>for</com>
   <data>int i=0</data>
   <condition> i &lt; 5</condition>
   <operation>i++</operation>
  </FOR>
  <skobka>
   <comx>{</comx>
   <output_data>
    <com>cout</com>
    <data>m1</data>
   </output_data>
   <FOR>
    <com>for</com>
    <data>int j=0</data>
    <condition> j &lt; 8</condition>
    <operation>j++</operation>
   </FOR>
   <skobka>
    <comx>{</comx>
    <output_data>
     <com>cout</com>
     <data>" "</data>
    </output_data>
    <comx>}</comx>
   </skobka>
   <output_data>
    <com>cout</com>
    <data>m1</data>
    <data>endl</data>
   </output_data>
   <comx>}</comx>
  </skobka>
  <FOR>
   <com>for</com>
   <data>int i=0</data>
   <condition> i &lt; 10</condition>
   <operation>i++</operation>
  </FOR>
  <skobka>
   <comx>{</comx>
   <output_data>
    <com>cout</com>
    <data>m1</data>
   </output_data>
   <comx>}</comx>
  </skobka>
  <return_data>
   <com>return </com>
   <data>0</data>
  </return_data>
  <comx>}</comx>
 </skobka>
</GLOB>
```

После того как этот текст прошёл синтаксический и семантический анализ, с использованием таблиц ключевых слов и разделителей языка C++ программа сгенерировала следующий код:

```
import java.util.Scanner;
import java.lang.Math;
public class test
{public static void main(String[] args) throws InterruptedException
  {
  char m1 ='=';
  for ( int i = 0; i< 10;i++ )
    {
    System.out.print(m1);
    }
  System.out.println();
  for ( int i = 0; i< 5;i++ )
    {
    System.out.print(m1);
    for ( int j = 0; j<8;j++ )
      {
      System.out.print(" ");
      }
    System.out.print(m1);
    System.out.println();
    }
  for ( int i = 0; i< 10;i++ )
    {
    System.out.print(m1);
    }
  }
}
```

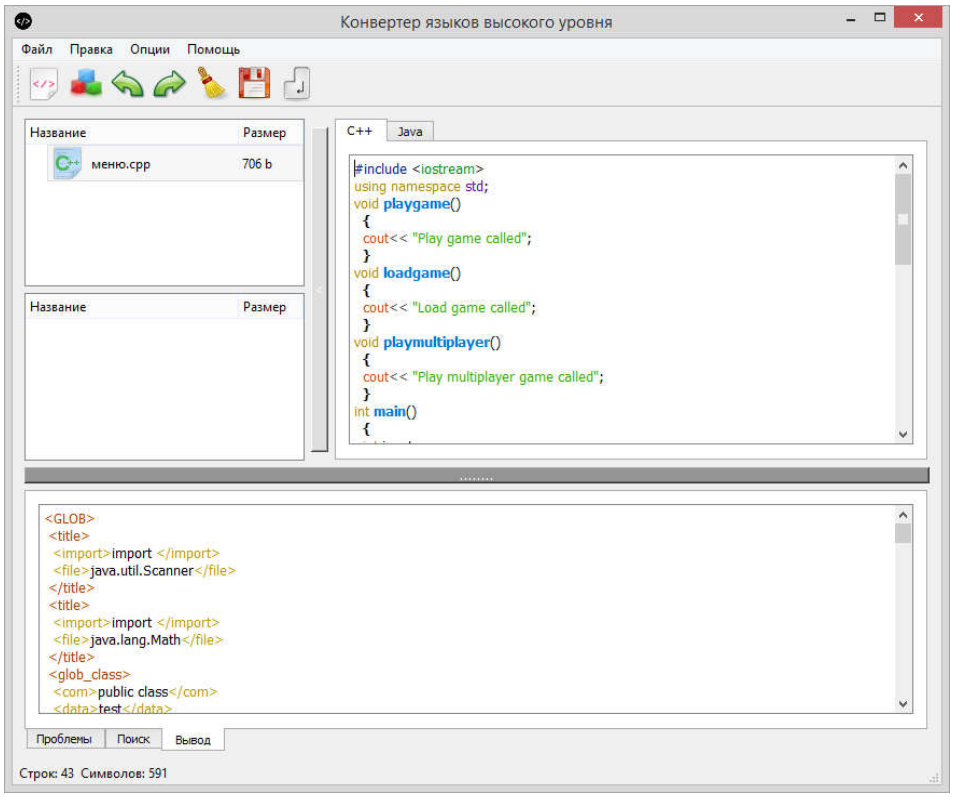

Рис. 3. Пример работы программы – конвертера.

В заключение следует заметить, что разработанная и описанная в статье программа – конвертер прошла успешно всестороннее тестирование при преобразовании кодов программ и сейчас применяется при изучении дисциплины «Сравнительный анализ языков программирования».

Литература

1. Фомин А.А., Данилов С.Д. Многомасштабный анализ объектов изображений. //Алгоритмы, методы и системы обработки данных. 2008. № 13. С. 158-164.

2. Жизняков А.Л., Фомин А.А. Автоматизированная подсистема кратномасштабной обработки рентгенограмм в системах неразрушающего контроля //Автоматизация и современные технологии. 2007. № 12. С. 3-9.

3. Садыков С.С., Савичева С.В. Исследование наложенности плоских объектов в поле зрения системы технического зрения// Известия высших учебных заведений. приборостроение. 2012. т. 55. № 2. с. 14-18.

4. Садыков С.С., Савичева С.В. Предварительная обработка изображений плоских объектов в системах технического зрения//Известия высших учебных заведений. приборостроение. 2012. т. 55. № 2. с. 19-23.

5. Орлов А.А., Стародубов Д.Н. Модель и алгоритм синтеза изображения линейчатой структуры на снимках промышленных материалов// Алгоритмы, методы и системы обработки данных. 2012. № 19. с. 98-104.

6. Орлов А.А., Стародубов Д.Н. Автоматический анализ неметаллических включений в стали// Алгоритмы, методы и системы обработки данных. 2011. № 18. с. 13.

7. Стародубов Д.Н. Определение процентного соотношения феррита и перлита в стали по снимку микроструктуры// Алгоритмы, методы и системы обработки данных. 2011. № 17. с. 14.

8. Садыков С.С., Буланова Ю.А., Захарова Е.А. Компьютерная диагностика новообразований на маммографических снимках// Компьютерная оптика. 2014. т. 38. № 1. с. 131-138.

9. Садыков С.С., Захарова Е.А., Буланова Ю.А. Использование информационных технологий для выявления области кисты молочной железы на маммограммах// Вестник рентгенологии и радиологии. 2013. № 3. с. 015-020.

10. Буланова Ю.А. Использование информационных технологий для локализации области рака молочной железы на маммограммах с преобладанием железистого компонента// Прикаспийский журнал: управление и высокие технологии. 2013. № 3. с. 100-111.

11. Садыков С.С., Буланова Ю.А., Яшков В.С. Предварительная обработка маммографических снимков// Труды международного симпозиума надежность и качество. 2013. т. 1. с. 340-343.

12. Чижов В.С., Ковалев Ю.А., Варламов А.Д. Разработка метода повышения качества поиска лиц на изображениях анализом их биометрических признаков // Алгоритмы, методы и системы обработки данных. 2014. № 2 (27). с. 55-63.

13. Еремеев С.В. Алгоритмы обработки данных в геоинформационной системе для учета земельных участков// Ползуновский вестник. 2012. № 2-1. с. 121-125.

14. Еремеев С.В., Андрианов Д.Е., Баринов А.Е., Титов Д.В. Алгоритмы поиска объектов по пространственным характеристикам в задачах муниципальных Гис// Известия юго-западного государственного университета. 2012. № 2. с. 37.

15. Еремеев С.В. Пространственно-временной анализ муниципальных карт// Алгоритмы, методы и системы обработки данных. 2012. № 22 (4). с. 52-57.

16. Канунова Е.Е., Варламов А.Д. Структурно-функциональная организация системы управления информационными ресурсами регионального музея// Алгоритмы, методы и системы обработки данных. 2011. №2. С. 50-55.

17. Садыков С.С., Канунова Е.Е. Система формирования данных об информационных ресурсах краеведческого музея и управления ими: опыт разработки и использования// Информационные технологии. 2007. №10. С.59-65.

18. Садыков С.С., Канунова Е.Е., Варламов А.Д. Автоматизированная реставрация изображений архивных текстовых и фотографических документов// Автоматизация и современные технологии. 2007. №8. С.10-12.

19. Канунова Е.Е., Полякова Е.В. Особенности распознавания изображений старопечатных текстовых символов// Алгоритмы, методы и системы обработки данных. 2009. №14. С. 55-61.

20. Альфред Ахо, Рави Сети, Джеффри Ульман. Компиляторы. Принципы, технологии, инструменты, 2-е издание, 2008. - 1185 с.

21. Основы разработки трансляторов. URL: http://www.softcraft.ru/translat/lect/content.shtml (дата обращения: 11.10.2014)

22. Синтаксический анализатор грамматического словаря. URL: http://www.solarix.ru/for_developers/docs/rules.shtml (дата обращения: 11.10.2014)

23. Андрианов Д.Е., Штыков Р.А., Уткин Ю.В. Экономия энергии путем управления тепловыми сетями на промышленном предприятии//Промышленная энергетика. -2003. -№ 6. -С. 2-5.

24. Андрианов Д.Е., Садыков С.С., Симаков Р.А. Разработка муниципальных геоинформационных систем. -М.: Мир, 2006. -109 с. ил.

E-MAIL: KANUNOVAEE@LIST.RU