УДК 681.3.06

ПРОМЕЖУТОЧНОЕ ПРЕДСТАВЛЕНИЕ ЯЗЫКА ПРОЦЕДУРНО-ПАРАМЕТРИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Бовкун А.Я., Пинчук А.А.
Научный руководитель – д.т.н., профессор Легалов А.И.
Сибирский федеральный университет

Процесс компиляции программы состоит из следующих фаз:

- фаза лексического анализа;
- фаза синтаксического анализа (распознавание синтаксической структуры и семантический разбор);
- фаза генерации кода.

Промежуточное представление – это отражение структуры и информационных связей между отдельными элементами программы. Оно является результатом семантического анализа и используется впоследствии для удобства генерации кода и/или проведения различных оптимизаций. Промежуточное представление может в различной степени быть близким либо к исходной программе, либо к машине. Например, промежуточное представление может содержать адреса переменных, и тогда его уже нельзя перенести на другую машину. С другой стороны, промежуточное представление может содержать раздел описаний программы, и тогда информацию об адресах можно извлечь из обработки описаний. Операторы управления в промежуточном представлении могут быть представлены в исходном виде (в виде операторов языка if, for, while и т.д.), а могут содержаться в виде переходов. В первом случае некоторая информация может быть извлечена из самой структуры (например, для оператора for – информация о переменной цикла, которую, может быть, разумно хранить на регистре, для оператора case – информация о таблице меток и т.д.). Во втором случае представление проще и унифицированней.

В ходе разработки языка программирования, являющегося модификацией языка O2M, возникла необходимость сохранения промежуточного представления, полученного при выходе компилятора. Требование обусловлено возможностью оптимизации и исследования промежуточного представления с точки зрения эволюционного расширения.

В качестве контейнера для хранения структуры представления выбран формат XML, который представляет собой структурированный документ, предназначенный для обмена данными между программами. Визуально структуру XML-документа можно представить в виде дерева, что позволяет отразить иерархию и зависимости объектов компилируемой программы. Впоследствии на основе промежуточного представления можно выстраивать графическую модель (граф), воссоздавать структуру объектов в памяти или осуществлять генерацию кода. Кроссплатформенность XML обеспечивает возможность использования на любой операционной системе или языке программирования.

Исходя из используемых технологий разработки (язык программирования C++), необходим соответствующий обработчик XML. Рассмотрим следующие обработчики:

1. Xerces-C++. Поставляется в виде библиотек, которые необходимо устанавливать в систему. Обладает поддержкой кроссплатформенности и большим набором возможностей. Есть библиотеки для различных языков программирования. Удобнее всего использовать при разработке больших программных систем.

2. TinyXml. Поставляется в виде нескольких файлов .h/.cpp, которые достаточно подключить к проекту C++. Поддержка кроссплатформенности. Обладает рядом ограничений, но прост в освоении.

3. Libxml2. Поставляется в виде большого количества файлов .h/.cpp, которые необходимо собирать с помощью утилиты make. Обладает поддержкой кроссплатформенности и версий для различных языков.

4. PugXml. Поставляется в виде нескольких файлов .hpp/.cpp, которые достаточно подключить к проекту. Поддержка кроссплатформенности. Обладает наименьшей скоростью обработки.

В качестве средства реализации был выбран обработчик TinyXml за приемлемую скорость обработки, быструю возможность освоения и удобный API.

Рассмотрим следующую реализацию программы:

```
// Модуль, описывающий прямоугольник
MODULE MRect;
 IMPORT In, Out;
 TYPE
   PRectangle* = POINTER TO Rectangle;
   Rectangle* = RECORD
     x*, y* : INTEGER //стороны прямоугольника
   END;
 // Процедура вывода
 PROCEDURE Output*(VAR r: Rectangle);
 BEGIN
   Out.String("Rectangle: x = "); Out.Int(r.x, 0);
   Out.String(", y = "); Out.Int(r.y, 0);
   Out.Ln;
 END Output;
END MRect.
```

В результате работы обработчика для каждого модуля компилируемой программы формируется отдельный xml-файл. Для модуля, приведенного выше, результирующий XML будет выглядеть так:

```
<?xml version="1.0" ?>
<Module Name="MRect">
    <ImportedModule Name="In" />
    <ImportedModule Name="Out" />
    <Type Name="PRectangle" Type="POINTER" TO="Rectangle" />
    <Type Name="Rectangle" Type="RECORD" >
          <Field Name="x" Type="INTEGER"/>
          <Field Name="y" Type="INTEGER"/>
    </Type>
    <Procedure Name="Output" >
          <FormalParameter Name="r" Type="Rectangle"/>
    </Procedure>
</Module>.
```

В промежуточном представлении сохраняется информация обо всех объектах модуля. Структурированность файла позволяет проследить зависимости между про-

граммными объектами. Перенос представления осуществляется простым копированием файла.

В качестве выводов следует отметить:

- задача сохранения промежуточного представления программы решена за счет использования формата XML;
- в ходе реализации были исследованы различные обработчики XML, основой анализа был выбран наиболее подходящий для решения поставленной задачи вариант;
- сохранение промежуточного представления в формате XML дает возможность детально представить зависимости между объектами, воссоздавать структуру объектов в памяти или осуществлять генерацию кода;
- в дальнейшем использование XML дает возможность представить структуру программы в виде графа, что обеспечит большую наглядность (в отличие от текстового представления).