Лабораторная работа № 4. «Компиляция объектно-ориентированного языка»

Коновалов А.В.

29 апреля 2023 г.

1 Цель работы

Целью данной работы является ознакомление с компиляцией средств объектно-ориентированного программирования.

2 Задание

Требуется (в зависимости от выбранного варианта) написать компилятор базовой версии НУИЯПа++ или НУИЯПа-П и расширить его одним из предложенных средств (на выбор студента).

Интерфейс командной строки компилятора:

```
refgo MyCompiler+LibraryEx source.txt dest.txt
```

где `source.txt` — исходный текст на НУИЯПе, `dest.txt` — сгенерированный код на языке модельного ассемблера. Исходный текст должен загружаться при помощи `LoadExpr`.

Для каждого из добавленных средств нужно написать пример кода, демонстрирующий его работу.

Допустимо выполнять лабораторную работу и не на Рефале-5. В этом случае нужно будет самостоятельно спроектировать структуры данных для представления синтаксического дерева НУИЯПа на выбранном языке программирования (например, data-классы на Python, case-классы на Scala и т.д.).

2.1 Варианты заданий

1. НУИЯП++ с проверкой типов, реализованной через односвязные списки виртуальных таблиц.
2. НУИЯП++ с проверкой типов, реализованной через массивы дескрипторов типов (виртуальных таблиц) в стиле Оберона.
3. НУИЯП++ с множественным наследованием.
4. НУИЯП++ с виртуальным множественным наследованием.
5. НУИЯП-Go с ООП без наследования в стиле Go. Определения классов имеют синтаксис

   ```
   t.Definition ::= …
     | (class s.ClassName
     (fields (s.Name t.ConstExpr)*)?
     (method s.Name (s.Name+) t.LocalVars? e.Code)+
       )
   s.ClassName ::= s.Name
   ```

   Разные классы могут иметь одноимённые методы.

   Определения интерфейсов имеют синтаксис:

   ```
   t.Definition ::= … | (interface s.InterfaceName e.Methods)
   e.Methods ::= s.Name+
   s.InterfaceName ::= s.Name
   ```

   Различные интерфейсы не могут иметь одноимённые методы.

   Оператор присваивания интерфейсу имеет вид

   ```
   t.Statement ::= …
     | (assign-interface t.Expr ":" s.InterfaceName ":=" t.Expr ":" s.ClassName)
   ```

   Интерфейсы компилируются в структуры, содержащие указатель на экземпляр класса и указатели на методы интерфейса.

   Выражения пополняются вызовами методов классов (`cmcall`) и интерфейсов (`imcall`):

   ```
   t.Expr ::= …
     | (cmcall t.ObjectPtr ":" s.ClassName s.MethodName t.Expr*)
     | (imcall t.ObjectPtr s.MethodName t.Expr*)
   ```

6. (Повышенной сложности.) Предложить реализацию нисходящего приведения типов для множественного наследования.
7. Реализовать обычное одиночное ООП как синтаксический сахар поверх прототипного ООП. Определение `class` описывает глобальную переменную типа `PObject` с проинициализированными слотами для методов, инициализатор `init` преобразуется в оператор клонирования `clone`.
8. Множественное прототипирование. Объект может содержать несколько слотов `proto__`, оператор клонирования, соответственно, принимает вид `(clone t.Address t.ObjectPtr*)`. (`(clone t.Address)` является альтернативной записью для `(clone t.Address 0)` — создания объекта без прототипа).

3 Отчёт по лабораторной работе

Отчёт выполняется в разметке Markdown по следующему шаблону:

```
% Лабораторная работа № 4. «Компиляция объектно-ориентированного языка»
% 29 апреля 2023 г.
% Вася Пупкин, ИУ9-63Б
# Цель работы
‹переписываете цель работы из задания›
# Индивидуальный вариант
‹переписываете выбранные расширения языка›
# Реализация и тестирование
```
xxxxx
```
# Вывод
‹пишете, чему научились›
```

В отчёте приведён лишь необходимый минимум. Можно писать больше и интереснее — интересные и вдумчивые отчёты поощряются дополнительным баллом.

Шаблон отчёта

Ваш отчёт будет конвертирован в PDF при помощи pandoc следующей командой:

```
pandoc \
  --pdf-engine=xelatex \
  -V 'mainfont:Liberation Serif' \
  -V 'monofont:Liberation Mono' \
  "$SOURCE" -o "$PDF"
```

Язык реализации: Markdown

Код решения

Из файла

Отправить