Трудность выделения текстового содержимого из загруженных файлов связана в первую очередь с тем, что документы имеют различные форматы представления (например, html, doc, pdf и т.д.). Таким образом, для перевода документа в формат plain text («чистый» текст) необходимо произвести процедуру конвертации. Вторая проблема, возникающая на данном этапе, – определение кодировки документов. Она в особенности характерна для текстов на русском и белорусском языках.

Список использованных источников:

1. Индексирование // Большая научная библиотека [Электронный ресурс]. – Режим доступа: http://bse.sci-lib.com/article054017.html. – Дата доступа: 05.03.2019.
2. Новый систематизированный толковый словарь // Государственная публичная научно-техническая библиотека России [Электронный ресурс]. – 1995. – Режим доступа: http://www.gpntb.ru/win/book/. – Дата доступа: 05.03.2019.
3. Майковский, В.В. Обзор подходов и методов индексирования в информационно-поисковых системах / В.В. Майковский // Сб. тр. Всерос. науч. школы-семинара молодых ученых, аспирантов и студентов «Интеллектуализация информационного поиска, скантехнологии и электронные библиотеки». – Таганрог: Изд-во ТТИ ЮФУ, 2010. – С. 77–78.
4. Manning, C. Introduction to Information Retrieval / C. Manning, P. Ragha-van, H. Schütze. – 1 edition. – Cambridge University Press, 2008. – 496 p.
5. Захаров, В.П. Информационные системы (документальный поиск): учеб. пособие / В.П. Захаров. – СПб.: Изд-во СПбГУ, 2002. – 188 с.

КОМПИЛЯТОР ЯЗЫКА ОБЕРОН, РЕАЛИЗУЮЩИЙ ПАРАДИГМУ РАСШИРЯЕМОГО ПРОГРАММИРОВАНИЯ

Шулицкий Д.С.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Сурков К.А. – ст. преподаватель

В настоящее время перед программистами зачастую встаёт задача расширения функционала существующего кода. Зачастую эта задача решается при помощи механизмов наследования и полиморфизма в ООП. Однако это не позволяет модифицировать уже существующие объекты и переопределять статические функции.

Самый простой способ модификации существующей программы – изменение её исходного кода. Однако, это не всегда допустимо. Например, может быть недопустима модификация библиотечного модуля. Модификация существующего кода приводит к необходимости повторного тестирования модулей.

В объектно-ориентированном программировании названные выше проблемы решаются при помощи наследования и полиморфизма. Однако это не позволяет модифицировать данные в существующих объектах. Для модификации кода необходимо создавать объекты классов, что зачастую не удобно.

Эти проблемы можно решить при помощи расширяемого программирования. Расширяемое программирование состоит из расширения кода и расширения данных.

Расширение кода означает, что новый модуль изменяет работу существующих модулей и реализуется при помощи переопределения существующих процедур, которое похоже на переопределение методов в ООП. Существует несколько способов реализовать переопределение кода:

1) замена адреса вызываемой процедуры во всех точках её вызова;
2) вызов процедуры через процедурную переменную (вместо прямого вызова процедуры);
3) генерация компилятором процедуры-переходника, которая выполнит вызов через процедурную переменную;
4) замена кода процедуры по месту.

Расширение данных означает добавление новых полей в существующую запись. Для расширения данных можно вручную ассоциировать оригинальные объекты с объектами-дополнениями, но следствием этого являются следующие проблемы:

- высокая трудоёмкость, так как необходимо поддерживать таблицу соответствий;
- снижение надёжности, так как необходимо уничтожать объекты-дополнения при уничтожении оригинальных объектов, а ошибки при реализации могут привести к утечкам памяти даже при наличии сборщика мусора;
- падение производительности при поиске объекта-дополнения для объекта-оригинала.
- необходимость синхронизации доступа к таблице соответствия в многопоточных приложениях.

Решением этого является поддержка на уровне компилятора списка дополнений у записей (объектов). Дополнения можно добавлять к существующим записям в момент первого обращения к полю расширения, а можно в момент загрузки модуля.

Так как на этапе компиляции доступна информация о переопределённых процедурах и дополнениях записей, существует возможность генерировать динамически подключаемые модули, которые не только добавляют новый функционал в момент загрузки модуля, но и удаляют свои следы в момент выгрузки.

В качестве языка, на основе которого тестировалась данная парадигма, был использован Оберон. Этот язык имеет очень компактное описание, и, при этом, содержит в себе достаточно высокоуровневый функционал. Отчёт об языке содержит описание его синтаксиса, что упрощает разработку компилятора и расширение языка.

Компилятор языка Оберон выбирался исходя из следующих требований:

1) возможность расширения синтаксиса языка;
2) поддержка компиляции под наиболее популярные архитектуры процессоров: x86, x64, arm.

Существующие компиляторы в данное время не поддерживаются или не позволяют компилировать код одновременно под все данные архитектуры. Таким образом, было принято решение разработать собственный компилятор, удовлетворяющий данным требованиям.

В качестве библиотеки для написания компилятора была выбрана LLVM. Данная технология позволила создать эффективный компилятор, генерирующий код для большого количества архитектур.

Разработанный компилятор открыл большие возможности по написанию надёжных расширяемых программ, в том числе сторонними разработчиками.

ПРИМЕНЕНИЕ ТЕОРИИ МАССОВОГО ОБСЛУЖИВАНИЯ В НАТУРНОМ ЭКСПЕРИМЕНТЕ

Шульга Е. С.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Сурков К. А. – ст. преподаватель

Рассмотрено применение теории систем массового обслуживания для моделирования процессов. Математический аппарат теории СМО позволяет рассчитать ключевые характеристики эффективности функционирования системы. Предложена последовательность действий, используя которую можно задать модель для анализируемого процесса. Для расчета характеристик системы в натурном эксперименте необходимо фиксировать промежутки времени между поступающими в систему заявками, а также промежутки времени между двумя обработанными заявками.

Различные виды деятельности человека можно представить в виде систем массового обслуживания (СМО). В таких системах можно выделить следующие элементы: заявка, поступающая на обработку; процессор, главный вычислитель, занимающийся обработкой заявок; очередь, в которой накапливаются заявки, ожидающие обработки [1]. В системе в общем случае может быть один или несколько обработчиков, причём они могут быть настроены на различные дисциплины обслуживания.

Теория СМО может быть применена к различным видам деятельности человека. Например, в работе [2] рассматривается применение данной теории к работе преподавателя университета. Сам преподаватель выступает в роли обслуживающего прибора, он занимается обработкой заявок – проверкой результатов выполнения индивидуальных заданий студентами.

Математический аппарат теории массового обслуживания позволяет рассчитать характеристики эффективности работы системы [1]:

– пропускную способность;
– вероятность того, что обработчик будет занят и очередная заявка будет помещена в очередь или отброшена (в соответствии с дисциплиной обслуживания системы);
– среднее число заявок в очереди или во всей системе во время ее работы;
– среднее время ожидания заявки в очереди.

Чтобы для задачи создать модель на основе теории массового обслуживания, необходимо выполнить следующие шаги. Для начала, нужно определить, что является заявкой, поступающей в систему. Затем необходимо задать обслуживающее устройство, количество параллельно