Сами Кобилов, Ильес Раббимов

Самаркандский государственный университет
Самарканд, Республика Узбекистан

О создании системы программирования с адаптируемым интерфейсом

Обсуждаются вопросы создания программных продуктов с адаптируемым интерфейсом. В качестве таких продуктов рассматриваются системы программирования. Показываются пути применения систем в образовательной информатике.

Sami Kobilov, Ilyos Rabbimov

Samarkand State University
Samarkand city, Republic of Uzbekistan

About creation of programming system with the adapted interface

The question of creation of software with the adapted interface are discussed. As such products the programming system are considered. The ways of application of system in educational informatics are shown.

Введение

Программное обеспечение создаётся и развивается по двум направлениям: проектирование нового и локализация существующего. Здесь под **локализацией** программного обеспечения подразумевается любые изменения и компоновки, которые не меняют его сущности.

Новые программные продукты и технологии их реализации подробно описываются в научных трудах и технических документациях разработчиков. Исследование проблем и задач, связанных с локализацией показывает, что эта тема, которой зачастую не находится места в книгах, посвященных программированию. Однако вопросы локализации и применения программных продуктов для **образовательной информатики** является очень важными.

Как правило, локализация программ начинается после их разработки. Для этого решаются вопросы обоснования необходимости локализации, согласования с разработчиками продукта и модификации программного обеспечения. Всё это занимает много времени, сил и средств. Возникает вопрос: **нельзя ли создать программные продукты, изначально предназначенные и готовые для дальнейшей локализации?** Именно здесь появляется ещё одна актуальная задача системного программирования – это проектирование программного обеспечения с адаптируемым интерфейсом. Её решение на порядок упрощает процесс локализации и применения программ.

В данной работе обсуждается один из возможных подходов к решению задачи **создания программных продуктов с адаптируемым интерфейсом.** В качестве таких продуктов рассматриваются современные интегрированные системы программирования. Работа состоит из двух разделов. В первом разделе описана структура программного обеспечения с адаптируемым интерфейсом на примере системы программирования. Второй раздел посвящен вопросам технологии реализации таких программных продуктов. Предложенный подход к созданию, базовая структура и приёмы локализации могут быть использованы и для разработки других типов программного обеспечения.

Процесс локализации является итеративным и трудоёмким, требует специальных знаний и участия разработчика продукта. Удовлетворит всем требованиям этого сложного процесса часто очень сложно. Чтобы упростить

это и дать возможность самим пользователям локализовать программные продукты необходимо разработать новые подходы и технологии их создания.

На наш взгляд, одним из перспективных подходов является проектирование программных продуктов с адаптируемым интерфейсом. Обсудим структуру, особенности и некоторые технологические аспекты реализации таких продуктов.

Структура программного обеспечения с адаптируемым интерфейсом

Система программирования является важной компонентой системного программного обеспечения, играет ключевую роль в образовательной информатике при изучении основ алгоритмики и программирования. Поэтому для описания структурны программных продуктов с адаптируемым интерфейсом мы будем рассматривать современные интегрированные системы (среды) программирования (ИСП).

Любая ИСП состоит из множества подсистем и стандартных модулей. Основными подсистемами являются транслятор, экранный редактор, отладчик, компоновщик и подсистема помощи. Среда имеет определенную (базовую) лексику общения. Она состоит из двух типов: лексика общения пользователя со средой и лексика входного языка программирования. Поэтому в системе с адаптируемым интерфейсом должны быть включены следующие дополнительные подсистемы (утилиты):

а) Изменения лексики общения пользователя с системой. Подсистема меняет названия главного меню, опций (подопций), окон и тексты сообщения процесса диалога;

б) Изменения лексики входного языка программирования. Утилита в диалоговом режиме даёт возможность изменения зарезервированных слов языка с одной лексики на другую;

в) Модификации сообщений об ошибках времени трансляции программ. Утилита связана также со справочной службой, которая выводить детальную информацию об ошибке и рекомендации по её устроению;

г) Модификации сообщений об ошибках периода исполнения программ. Эти ошибки делятся на несколько категорий;

д) Модификации сообщений об информационной помощи. Подсистема помощи содержит справочную информацию о трансляторе, зарезервированных словах, модулях и входном языке. Утилита переводит всю справочную информацию с одного языка на другой;

е) Наряду с этими подсистемами для работы локализируемой среды, при необходимости, используется драйвер консоли на базе соответствующего национального алфавита.

Таким образом, эти подсистемы обеспечивают интерфейсную локализацию среды программирования.

Пути реализации программного продукта

В состав большинства современных ИСП помимо самой среды (например, среда Turbo Pascal, файл turbo.exe) поставляется также автономный компилятор (например, tpc.exe), который для работы требует меньше памяти, чем среда в целом и является простым для применения. Поэтому процесс локализации ИСП можно упростить, т.е. на первом этапе локализовать не всю среду, а только её систему программирования. Мы будем учитывать это обстоятельство, и для решения своих задач используем автономный компилятор среды программирования.

При реализации системы программирования с адаптируемым интерфейсом нами приняты следующие обоснования и технологические решения.

Лексика общения. Система программирования имеет базовую лексику общения и входного языка программирования. Если нет необходимости адаптации интерфейса, то система используется как обычная. Иначе эта

лексика может быть изменена пользователем с помощью вышеуказанных программных утилит.

Входной язык. В качестве входного языка программирования разрабатываемой системы был выбран Паскаль/Р[1], который имеет русскую нотацию стандартного Паскаля[2]. Выбор обосновывается следующими соображениями. Во-первых, язык Паскаль был разработан в качестве учебного языка. Во-вторых, в нем имеются минимально необходимые и традиционные конструкции, которые достаточны для начального этапа обучения алгоритмике и программированию. В-третьих, линия паскалеобразных языков быстро развивается. Разработаны и с успехом используются такие языки и системы, как Модула, Оберон и Зонон.

Техника трансляции. Транслятор системы разрабатывается в виде **конвертора.** Такой выбор сделан для упрощения его реализации и использования готовых подпрограмм стандартных моделей промежуточного языка и среды программирования Borland Pascal[3]. Поэтому система работает в двухэтапном режиме. На первом этапе конструкции входного языка переводятся в соответствующие правильные конструкции промежуточного языка в виде готовой программы. На втором этапе выполняется эта программа. Если были допущены ошибки в исходном тексте программы, то транслятор заканчивает свою работу, при обнаружении первой из них и выдаёт соответствующее сообщение на базе выбранной лексики.

Технология программирования. Известно, что одним из классических подходов к созданию трансляторов является **метод синтаксических подпрограмм**[4]. Общая идея этого метода заключается в разделении программы транслятора на две части. Одна часть содержит набор синтаксических подпрограмм. Каждая подпрограмма соответствует определенной синтаксической конструкции входного языка и предназначена для обработки именно этой конструкции. Например, подпрограмма оператора цикла обрабатывает только оператор цикла, а подпрограмма

оператора присваивания только оператор присваивания. Другая часть транслятора – управляющая программа – выделяет и распознаёт отдельные конструкции входной программы и вызывает каждый раз нужную синтаксическую подпрограмму. Следовательно, управляющая программа выполняет только синтаксический анализ.

При разработке нашего транслятора метод синтаксических подпрограмм был расширен идеями объектно-ориентированного программирования. В основе такого расширения лежит объединение синтаксических подпрограмм (метод – обработчик конструкций) и данных (данные – характеристики конструкций) в единое целое – объект[5].

Например, конструкции вида

если <условие> **то <**оператор1**> иначе** <оператор2>;

в трансляторе соответствует объект типа

type = **object**(obj)

Cond: **string;**{Содержит условие}

Point1, Point2: objpoint;

{Указатели на оператор1 и оператор2}

Constructor Init; Destructor Done;

{Создание и удаление экземпляра объекта}

Procedure Parsing; **Virtual;**

{Процедура синтаксического разбора конструкции}

Procedure Synthes; **Virtual;**

{Процедура синтеза промежуточной конструкции}

Function Nextobj: objpoint;

{Адрес следующего объекта}

end;

Объекты разделены на два типа:

а) Интерфейсные объекты.

б) Объекты – отображения конструкций входного языка.

К первому типу относятся объекты, обеспечивающие связь транслятора с внешними подсистемами (утилитами). Такими объектами являются.

а.1) Сканер-объект, считывающий исходную программу из внешнего файла.

а.2) Объект обработки ошибок, выводящий на монитор код и место обнаруженной ошибки.

а.3) Объект, записывающий промежуточную программу во внешний файл.

Второй тип разделяется на два подтипа.

б.1) Обобщенные объекты. Они сконструированы по одной схеме, т.е. по одним и тем же правилам. Эти объекты – описания операторов входного языка.

б.2) Необобщенные объекты, правила построения которых носят произвольный характер. Они описывают разделы паскаль-программы.

В конверторе инициатором начала разбора исходной программы является метод **Parsing.** Объект данного метода описывает общую конструкцию входного языка, т.е. конструкцию **Program** языка Паскаль. Далее данный объект инициализирует объекты, соответствующие разделам структуры исходной программы. Отсюда вытекает, что схема работы конвертора основывается методу нисходящего разбора.

Заключение

В работе были рассмотрены вопросы локализации и создания программного обеспечения с адаптируемым интерфейсом. Процесс проектирования программного обеспечения с адаптируемым интерфейсом описывается на примере системы программирования. Уточнены компоненты, предложены технологические решения и принципы программной реализации таких систем.

Литература

1. Кобилов С. Система программирования Паскаль/Р//Современные проблемы прикладной математики и экономики. – Самарканд: СамГУ, 1997. – С. 73-79.
2. Йенсен К., Вирт Н. Паскаль. Руководство пользователя и описание языка. – М.: ФиС, 1982. – 151 с.
3. Сурков Д. и др. Программирование в среде Borland Pascal для Windows. – Минск: Вышейшая школа, 1996. – 426 с.
4. Лебедев В. Введение в системы программирования. – М.: Статистика, 1975. – 262 с.
5. Кобилов С., Елизаров М. Применение новой технологии к решению одной классической задачи программирования. – Самарканд: Вестник СамГУ, 1999. - №1-2. – С.29-36.