Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра “Компьютерные технологии”

Е.А. Лысенко, П.С. Скаков

Транслитерация между различными системами записи японских слов

Программирование с явным выделением состояний

Проектная документация

Проект создан в рамках
"Движения за открытую проектную документацию"
http://is.ifmo.ru

Санкт-Петербург
2004

Содержание

Введение

Развитие компьютерной техники и сети Интернет породило ряд серьезных проблем, связанных с поддержкой языков программным обеспечением.

Первая из них состоит в отображении национальных символов и относительно просто решается разработкой соответствующих шрифтов.

Вторая проблема связана с внутренним представлением символов. Трудности в этой области во многом уменьшились с появлением единого стандарта Unicode (http://www.unicode.org).

Однако наиболее сложная проблема - ввод символов – актуальна и по сей день. К примеру, представьте себе клавиатуру, с несколькими тысячами значащих клавиш и десятком регистровых – именно столько потребуется для прямого ввода символов китайского языка. В подобных случаях прибегают к так называемому косвенному вводу, который, как правило, основан на фонетике. При этом необходимая единица языка распознается по звучанию, переданному средствами некоторого языка. Если языки совпадают, то это транскрипция. В общем случае (при разных языках) процесс носит название транслитерации. Английский как международный язык используется для нее наиболее часто, но это, впрочем, не обязательно.

Транслитерация необходима изучающим иностранные языки и всегда лежит в основе создания словарей-разговорников. С ее помощью можно вводить текст на практически любом языке даже при работе на компьютере с минимальной поддержкой языков. Использования транслитерации не избежать при употреблении имен собственных или, например, при выборе имени домена в сети Интернет.

Японский язык представляет особенный интерес, не только как язык высокотехнологичной и загадочной Страны Восходящего Солнца, но и как один из наиболее сложных в отношении транслитерации.

Дело в том, что в современном японском языке для записи слов одновременно используют *кандзи* (иероглифы), *хирагану* и *катакану*. *Кандзи* записывают корни слов, *хираганой* - окончания, союзы и постфиксы, *катаканой* - заимствованные слова и междометия. Допускается заменять *кандзи хираганой* (так пишут дети и не очень грамотные люди). *Хирагана* и *катакана* - слоговые азбуки, использующиеся в японском языке. В них каждый символ означает не звук, а слог. Азбуки содержат по 46 символов и дополнительно несколько устаревших.

Ромадзи - система письменности в японском языке, основанная на латинице.

Однозначность транслитерации призваны обеспечить стандарты.

Для записи японских слов и названий латиницей в Японии была разработана система транскрипции *кунрэй* ("официальная"). В настоящее время ее можно встретить в японских учебниках, а также в правительственных и парламентских документах.

Во второй половине XIX века американский миссионер Д. К. Хепберн (J. C. Hepburn) предложил собственную систему *ромадзи*-транслитерации, основанную на английском способе записи согласных и латинском способе записи гласных.

Система Хепберна стала фактическим стандартом транслитерации японского языка сначала в англоязычных странах, а потом и во всех государствах, использующих латиницу. Во времена оккупации после Второй мировой войны *система Хепберна* была объявлена официальной и в Японии. Она до сих пор продолжает применяться японцами, когда они ведут переписку с иностранцами. Именно транслитерацию по *системе Хепберна* чаще всего можно встретить в Интернете.

Киридзи или *россиядзи* - система записи японских слов и предложений, основанная на кириллице. Используется японистами в России. Первоначальный вариант *россиядзи* был предложен в 1917 году востоковедом Е. Д. Поливановым, поэтому ее часто называют *системой Поливанова*.

1. Постановка задачи

Разработанная программа, представленная в виде `Java`-апплета, осуществляет транслитерацию между различными системами записи слов японского языка.

Апплет демонстрирует однопроходный перевод между *кунрэй-ромадзи*, *Хепберн-ромадзи*, *киридзи*, *хираганой* и *катаканой* через промежуточный буфер. Формат хранения информации в буфере почти полностью совпадает с *кунрэй-ромадзи*. Перевод может осуществляться как одновременно из всех возможных кодировок, так и только из некоторых из них по выбору. При этом во входной строке допустимо наличие смайлов, цифр и других посторонних символов, которые на выходе сохраняются на тех же позициях.

Кроме японского текста апплет воспринимает и русский текст, приводя его к фонетическим правилам японского языка и пытаясь при этом как можно полнее сохранить его звучание.

Выбор языка `Java` для реализации описанной программы обусловлен необходимостью корректной работы с форматом `Unicode`, мультиплатформенностью, отвечающей

современным стандартам, и желанием сделать транслитератор удобным для интерактивного использования в сети Интернет.

На рис.1 приведен внешний вид апплета. В левое поле введены японские слова, записанные *хираганой.* Выбран режим перевода “в *киридзи*” – при транслитерации слова должны быть записаны символами кириллицы. После исполнения программы в правом поле появляется результат транслитерации. Программа может применяться таким образом, например, для выяснения произношения японских слов.

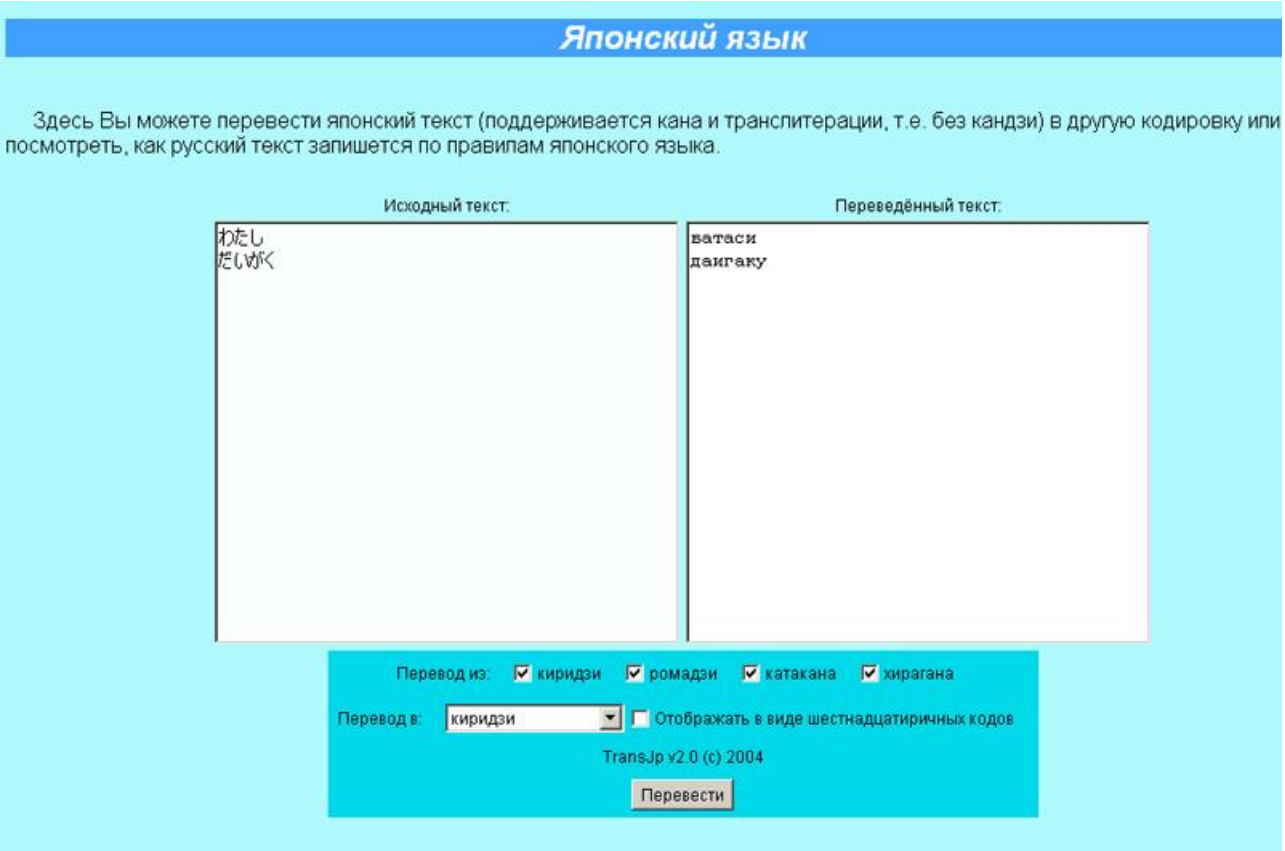

Рис. 1. Внешний вид апплета

2. Технические особенности

Несмотря на то, что в стандарте технологии `Java` заявлена полная поддержка `Unicode`, на практике в некоторых случаях возникают трудности с его использованием, в частности при работе с символами японского языка.

Кроме того, в конкретных реализациях `Java` под `Windows` степень поддержки `Unicode` также различается.

В среде Windows 2003 Server Enterprise Edition каких-либо проблем в функционировании апплета не выявлено. Для его работы необходимо лишь наличие Java-машины. Также рекомендуется в качестве кодовой страницы для не-Unicode программ поставить японскую.

Предыдущие версии Windows хуже поддерживают стандарты, поэтому с ними не все так хорошо. Необходимо заметить, что изложенное ниже, как правило, не решает проблемы взаимодействия английской и русской старых версий Windows с Java, а лишь обходит их. При использовании японской версии Windows трудностей такого рода быть не должно.

Авторам не удалось добиться нормальной работы ввода и отображения символов японских алфавитов на Java 2 Plugin от корпорации Sun. Рекомендуется его выключить, так как без него системная Java лучше работает с японским.
Java от корпорации Microsoft поддерживает ввод на японском, если его поддерживает сама Windows, то есть под Windows 2000+ требуется установить поддержку восточных языков (это стандартная часть системы) и соответствующую языковую раскладку.

Отображение настроить несколько труднее. Дело в том, что язык Java не позволяет точно выбрать имя шрифта для выводимых символов, а указать можно только его семейство. В программе выбрано семейство Arial, и Java почему-то использует стандартный шрифт Arial, который не поддерживает японские символы, а не подходящий для этого Arial Unicode. Решить проблему можно следующим образом: шрифт Arial Unicode (его можно взять, например, из Microsoft Office 2000) переименовывается в Arial, помещается в систему вместо обычного Arial. После этого японские символы отображаются правильно. Правда, необходимо предупредить, что шрифт Arial Unicode немного, но всё-таки отличается от Arial, поэтому надписи в различных программах, которые подразумевают стандартный Arial, после описанных манипуляций могут выглядеть несколько странно и даже не умещаться в нужные границы.

Также авторами предусмотрен специальный режим ввода/вывода символов в шестнадцатеричных кодах Unicode, который позволяет использовать все возможности программы на **абсолютно любой** системе, даже без специфических поддержек восточных языков.

3. Описание подхода

Посимвольная обработка входного потока, в ходе которой в зависимости от языковых свойств текущего символа совершаются различные действия, говорит в пользу “автоматного” подхода к решению задачи [1]:

1. На основе анализа предметной области выделяется множество состояний и множество событий, влияющих на переходы между состояниями.
2. Формируется один или система взаимосвязанных автоматов.
3. Для каждого автомата создается словесное описание в форме перечня решаемых задач.
4. Каждое событие описывается в словесной форме с пояснением условий и причин его возникновения, и необходимой обработки.
5. Для каждого автомата строится схема связей, а при наличии нескольких автоматов – схема их взаимодействия.
6. Для каждого автомата строится граф переходов.
7. Первоначально пишется шаблон, изоморфный графу переходов, проверяется его работоспособность, после чего шаблон конкретизируется добавлением функций, реализующих выходные воздействия.

Рассмотрение работы [2] подтвердило целесообразность выбранного метода решения.

Заметим, что в условиях поставленной задачи (необходимо автоматизировать непосредственно процесс транслитерации) множества событий как такового нет, состояния перебираются внутри бесконечного цикла, а переходы происходят в зависимости от входных переменных.

4. Обозначение входных переменных

x0 – переменная “Не достигнут конец строки”

x1 – переменная “Текущий символ значащий”
Примечание: в *киридзи* ‘:’ обозначает долготу гласной и интерпретируется программой как значащий символ.

x2 – переменная “Вывод следует формировать в виде шестнадцатеричных кодов”

5. Обозначение выходных воздействий

z0 – “Считать следующий символ из входной строки”

z1 – “Перевести символ во внутренний формат и занести в буфер”
Необходимо пояснить, что согласно правилам японского языка обработка осуществляется по слогам, то есть в буфере накапливается информация о текущем слоге, пока поступающие символы могут быть интерпретированы как часть этого слога. Когда обнаруживается начало

нового слога, то ранее накопленный в буфере слог транслитерируется, добавляется в выходной текст и удаляется из буфера, а затем в буфер заносится информация о новом слоге.

z2 – "Дописать в результирующую строку содержимое буфера, транслитерировав его, а текущий символ отправить на вывод без обработки"
Заметим, что результирующая строка формируется по мере поступления транслитерированных символов, а выводится, готовая, целиком.

z3 – "Дописать в результирующую строку содержимое буфера, транслитерировав его"

z4 – "Вывести результирующую строку на экран в виде символов"

z5 – "Вывести результирующую строку на экран в виде шестнадцатеричных кодов"
(в соответствии с пожеланием пользователя)

6. Автомат «управление транслитерацией» (A0)

6.1. Граф переходов

Алгоритм транслитерации представляется с помощью графа переходов конечного автомата, последовательность построения которого описана в разд. 2. Граф приведен на рис.2.

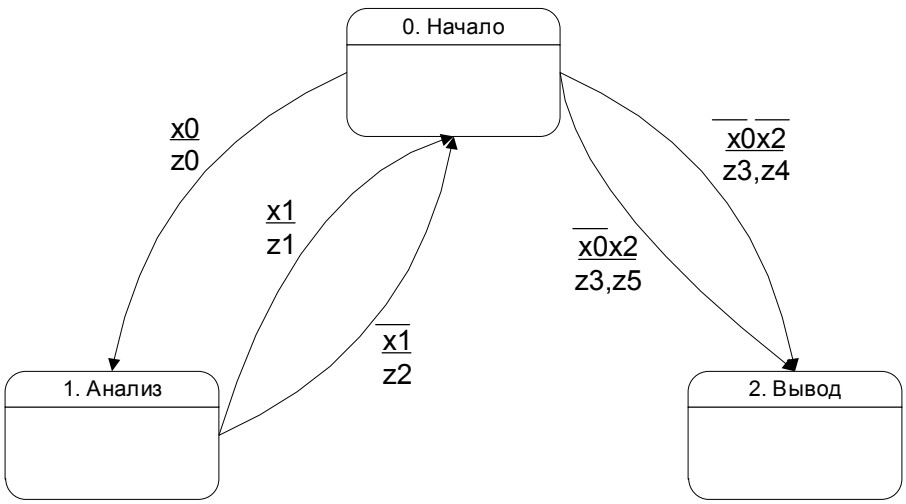

Рис. 2. Граф переходов автомата A0

В этом графе вершины – это состояния, в которых может находиться процесс. Исходно автомат – в нулевом состоянии.

Дуги графа показывают, из каких состояний в какие могут быть выполнены переходы. Дуги помечены условиями, при которых происходят переходы, и действиями, которые выполняются при этом.

6.2. Программная реализация автомата

Код автоматной функции, выстроенный согласно SWITCH-технологии [3], приведен в Листинге 1.

Листинг 1. Автоматная функция, реализующая A0

```
private void Translate()
    {
      int state = 0;
      while (true)
          {
             switch (state)// automata
                {
                   case 0:
                      if(x0())        {z0();          state=1;}
                      else if(!x2())  {z3(); z4();    state=2;}
                         else         {z3(); z5();    state=2;}
                   break;

                   case 1:
                      if (x1())       {z1();          state=0;}
                      else            {z2();          state=0;}
                   break;

                   case 2:
                   break;
                }
             if (state==2)      break;
          }
     }
```

Листинги 2,3 демонстрируют программную реализацию входных переменных и выходных воздействий соответственно.

Листинг 2. Входные переменные автомата A0

```
private boolean x0()
{
   return pos<src.length();
}

private boolean x1()
{
   int chrn=curchar;
   if (!(chrn>0x60 && chrn<0x7B && chrn!=0x78 && srcRom.getState() ||
         ((chrn>0x42F && chrn<0x450) || chrn==0x451 || chrn==0x3A) &&
         srcCyr.getState()||chrn>0x30A0 && chrn<0x30FD && srcKat.getState() &&
         destCode.getSelectedIndex()!=3 || chrn>0x3040 && chrn<0x3095 &&
         srcHir.getState() && destCode.getSelectedIndex()!=4))
            {
               goodvar=false;
            }
```

```
   else goodvar=true;
   return goodvar;
}

private boolean x2()
{
      return hex;
}
```

Листинг 3. Выходные воздействия автомата A0

```
private int z0()
{
      int chrn = (int) Character.toLowerCase(src.charAt(pos));
      curchar = chrn;
      pos++;
      return chrn;
}

private void z1()
{
      AddToBuf(curchar);
}

private void z2()
{
      FlushBuf();
      dest+=(char)curchar;
}


private void z3()
{
      FlushBuf();
}

private void z4()
{
      txtDest.setText(dest);
      translating=false;
}

private void z5()
{
      toHex(txtDest, dest);
      translating=false;
}
```

В силу специфики фонетических законов японского языка, служебные функции `AddToBuf` и `FlushBuf` довольно сложны и велики по объему. Их назначение и полные листинги приведены в Приложении.

7. Заключение

Разработанная программа может служить иллюстрацией, подтверждающей удобство рассмотренного в работе [2] подхода к решению задачи о транслитерации.

Использование автоматного подхода обеспечивает простоту модификации программы. Действительно, убедившись в работоспособности кода с “заглушками”, его можно дополнить функциями, соответствующими входным переменным и выходным воздействиям, для транслитерации между другими выбранными языками. Для этого необходимо продумать корректную буферизацию и разбираться в фонетических законах рассматриваемых языков.

В рамках движения “За открытую проектную документацию” [4] исходный код и документация программы доступны на сайте www.is.ifmo.ru , раздел “Проекты”. Знакомство с работой [2] и с данным проектом, позволяет, по мнению авторов, упростить задачу создания транслитератора для других языков.

Литература

1. *Шалыто А.А.* Технология автоматного программирования (http//is.ifmo.ru)

2. *Бабаев А.А.* Транслитерация и как правильно ее надо программировать (http//is.ifmo.ru)

3. *Шалыто А.А., Туккель Н.И.* SWITCH-технология - автоматный подход к созданию программного обеспечения "реактивных" систем (http//is.ifmo.ru)

4. *Шалыто А.А.* "Новая инициатива в программировании. Движение за открытую проектную документацию" (http//is.ifmo.ru)

Приложение

Логика построения служебных функций `AddToBuf` и `FlushBuf` выходных воздействий диктуется фонетическими законами японского языка. С этим же связана сложность рассматриваемых функций, а с множеством способов записи японских слов - большой объем кода.

Функция `AddToBuf` осуществляет перевод текущего символа в единый внутренний формат, анализирует содержимое промежуточного буфера, если возможно “освобождает” буфер функцией `FlushBuf`, а затем помещает внутреннее представление текущего символа в буфер. Полный текст функции `AddToBuf` приведен в Листинге П1.

Функция `FlushBuf` удаляет из буфера накопленный слог и собственно транслитерирует его, добавляя полученное в выходную строку. Текст функции `FlushBuf` приведен в Листинге П2.

Листинг П1. Служебная функция `AddToBuf`

```
private void AddToBuf(int n)
{
	char p, t='\u0000', d='\u0000';
	p=buf.length()>0?buf.charAt(buf.length()-1):'\u0000';

	switch(n) //Перевод текущего символа во внутренний формат
	{
	   case 0x44A:		FlushBuf(); return;
	   case 0x44C:		return;

	   case 0x3063:
	   case 0x30C3:		FlushBuf(); dbl=true; return;

	   case (int)':':
	   case 0x30FC:		 t=':';
				 if (buf.length()>0)
				 {
				    char c=buf.charAt(buf.length()-1);
				    if (c=='a' || c=='i' || c=='u' || c=='e' || c=='o')
				    break;
				 }
				 FlushBuf();
				 dest+=":";
				 return;

	   case (int)'a':
	   case 0x430:
	   case 0x3042:
	   case 0x30A2:		t='a'; break;

	   case (int)'e':
	   case 0x435:
	   case 0x44D:
	   case 0x3048:
	   case 0x30A8:		t='e'; break;

	   case (int)'i':
```

```
case 0x438:
case 0x439:
case 0x44B:
case 0x3044:
case 0x30A4:      t='i'; break;

case (int)'o':
case 0x43E:
case 0x304A:
case 0x30AA:      t='o'; break;

case (int)'u':
case 0x443:
case 0x3046:
case 0x30A6:      t='u'; break;

case (int)'b':
case 0x431:       t='b'; break;

case (int)'d':
case 0x434:       t='d'; break;

case (int)'f':
case 0x444:       t='f'; break;

case (int)'h':
case 0x445:       t='h'; break;

case (int)'g':
case 0x433:       t='g'; break;

case (int)'k':
case 0x43A:       t='k'; break;

case (int)'l':
case (int)'r':
case 0x43B:
case 0x440:       t='r'; break;

case (int)'m':
case 0x43C:       t='m'; break;

case (int)'n':
case 0x43D:
case 0x3093:
case 0x30F3:      t='n'; break;

case (int)'p':
case 0x43F:       t='p'; break;

case (int)'s':
case 0x441:
case 0x448:
case 0x449:       t='s'; break;

case (int)'c':    t='c'; break;

case (int)'t':
case 0x442:       t='t'; break;

case 0x446:
case 0x447:       t='t'; d='y'; break;

case (int)'v':    t='v'; break;

case (int)'w':
case 0x432:       t='w'; break;
```

```
case (int)'z':
case 0x436:
case 0x437:       if (p!='d')
                      t='z';
                  else
                      buf.replace('d', 'z');
                  break;

case (int)'y':    t='y'; break;

case (int)'j':    t='z'; d='y'; break;
case 0x451:
case 0x3088:
case 0x30E8:      t='y'; d='o'; break;

case 0x44E:
case 0x3086:
case 0x30E6:      t='y'; d='u'; break;

case 0x44F:
case 0x3084:
case 0x30E4:      t='y'; d='a'; break;


case 0x304B:
case 0x30AB:      t='k'; d='a'; break;

case 0x304C:
case 0x30AC:      t='g'; d='a'; break;

case 0x304D:
case 0x30AD:      t='k'; d='i'; break;

case 0x304E:
case 0x30AE:      t='g'; d='i'; break;

case 0x304F:
case 0x30AF:      t='k'; d='u'; break;

case 0x3050:
case 0x30B0:      t='g'; d='u'; break;

case 0x3051:
case 0x30B1:      t='k'; d='e'; break;

case 0x3052:
case 0x30B2:      t='g'; d='e'; break;

case 0x3053:
case 0x30B3:      t='k'; d='o'; break;

case 0x3054:
case 0x30B4:      t='g'; d='o'; break;

case 0x3055:
case 0x30B5:      t='s'; d='a'; break;

case 0x3056:
case 0x30B6:      t='z'; d='a'; break;

case 0x3057:
case 0x30B7:      t='s'; d='i'; break;

case 0x3058:
case 0x30B8:      t='z'; d='i'; break;
```

```
case 0x3059:
case 0x30B9:      t='s'; d='u'; break;

case 0x305A:
case 0x30BA:      t='z'; d='u'; break;

case 0x305B:
case 0x30BB:      t='s'; d='e'; break;

case 0x305C:
case 0x30BC:      t='z'; d='e'; break;

case 0x305D:
case 0x30BD:      t='s'; d='o'; break;

case 0x305E:
case 0x30BE:      t='z'; d='o'; break;

case 0x305F:
case 0x30BF:      t='t'; d='a'; break;

case 0x3060:
case 0x30C0:      t='d'; d='a'; break;

case 0x3061:
case 0x30C1:      t='t'; d='i'; break;

case 0x3062:
case 0x30C2:      t='d'; d='i'; break;

case 0x3064:
case 0x30C4:      t='t'; d='u'; break;
case 0x3065:

case 0x30C5:      t='d'; d='u'; break;

case 0x3066:
case 0x30C6:      t='t'; d='e'; break;

case 0x3067:
case 0x30C7:      t='d'; d='e'; break;

case 0x3068:
case 0x30C8:      t='t'; d='o'; break;

case 0x3069:
case 0x30C9:      t='d'; d='o'; break;

case 0x306A:
case 0x30CA:      t='n'; d='a'; break;

case 0x306B:
case 0x30CB:      t='n'; d='i'; break;

case 0x306C:
case 0x30CC:      t='n'; d='u'; break;

case 0x306D:
case 0x30CD:      t='n'; d='e'; break;

case 0x306E:
case 0x30CE:      t='n'; d='o'; break;

case 0x306F:
case 0x30CF:      t='h'; d='a'; break;

case 0x3070:
```

```
case 0x30D0:      t='b'; d='a'; break;

case 0x3071:
case 0x30D1:      t='p'; d='a'; break;

case 0x3072:
case 0x30D2:      t='h'; d='i'; break;

case 0x3073:
case 0x30D3:      t='b'; d='i'; break;

case 0x3074:
case 0x30D4:      t='p'; d='i'; break;

case 0x3075:
case 0x30D5:      t='h'; d='u'; break;

case 0x3076:
case 0x30D6:      t='b'; d='u'; break;

case 0x3077:
case 0x30D7:      t='p'; d='u'; break;

case 0x3078:
case 0x30D8:      t='h'; d='e'; break;

case 0x3079:
case 0x30D9:      t='b'; d='e'; break;

case 0x307A:
case 0x30DA:      t='p'; d='e'; break;

case 0x307B:
case 0x30DB:      t='h'; d='o'; break;

case 0x307C:
case 0x30DC:      t='b'; d='o'; break;

case 0x307D:
case 0x30DD:      t='p'; d='o'; break;

case 0x307E:
case 0x30DE:      t='m'; d='a'; break;

case 0x307F:
case 0x30DF:      t='m'; d='i'; break;

case 0x3080:
case 0x30E0:      t='m'; d='u'; break;

case 0x3081:
case 0x30E1:      t='m'; d='e'; break;

case 0x3082:
case 0x30E2:      t='m'; d='o'; break;

case 0x3089:
case 0x30E9:      t='r'; d='a'; break;

case 0x308A:
case 0x30EA:      t='r'; d='i'; break;

case 0x308B:
case 0x30EB:      t='r'; d='u'; break;

case 0x308C:
case 0x30EC:      t='r'; d='e'; break;
```

```
        case 0x308D:
        case 0x30ED:      t='r'; d='o'; break;

        case 0x308F:
        case 0x30EF:      t='w'; d='a'; break;

        case 0x3092:
        case 0x30F2:      t='w'; d='o'; break;

        case 0x3094:
        case 0x30F4:      t='v'; d='u'; break;

        case 0x3083:
        case 0x30E3:      t='a';

        case 0x3085:
        case 0x30E5:      if (t=='\u0000') t='u';
        case 0x3087:
        case 0x30E7:       if (t=='\u0000') t='o';
                           if (buf.endsWith(new Character('i').toString()))
                                 buf=buf.replace('i', 'y')+t;
                           return;

        case 0x3041:
        case 0x30A1:      t='a';
        case 0x3043:
        case 0x30A3:      if (t=='\u0000') t='i';
        case 0x3047:
        case 0x30A7:      if (t=='\u0000') t='e';
        case 0x3049:
        case 0x30A9:       if (t=='\u0000') t='o';
                           if (buf.endsWith(new Character('u').toString()))
                           {
                              if (buf.length()==1)
                                 buf="w"+t;
                              else
                              {
                                 if (buf.charAt(0)=='h') buf.replace('h', 'f');
                                    buf.replace('u', t);
                              }
                           }

                           return;

    default:          FlushBuf(); dest+=(char)n; return;
   }
// Анализ содержимого буфера

   if (t=='u' && p=='w') buf="";

   if (t=='u' && p=='f')
   {
    p='h';
    buf.replace('f', 'h');
   }

   if (p=='c')
   {
    if (t=='h')
    {
     buf="t";
     p='t';
    }
    else FlushBuf();
   }
```

```
      if (t=='h' && (p=='s' || p=='t'))
       t='y';

      if (t=='s' && p=='t')
      {
       t=d;
       d='\u0000';
      }

      if (p=='y' && t!='a' && t!='i' && t!='u' && t!='e' && t!='o') FlushBuf();

      if ((p=='m' || p=='n') && (t=='b' || t=='m' || t=='p'))
      {
       buf.replace('m', 'n');
       lng=true;
       FlushBuf();
       p='\u0000';
      }

      if (p==t)
       if (t=='b' || t=='d' || t=='g' || t=='h' || t=='k' || t=='m' || t=='p' ||
           t=='r' || t=='s' || t=='t' || t=='v' || t=='w' || t=='z')
        if (dbl) FlushBuf();
        else
        {
         dbl=true;
         t=d;
         d='\u0000';
        }

      if (p!='\u0000' && t!='\u0000')
      {
       if (p=='a' || p=='i' || p=='u' || p=='e' || p=='o')
       {
        if (t=='u' && p=='o' || (t==p || t=='o') && p!='o' || t==':')
        {
         if (lng) FlushBuf();
         else
         {
          lng=true;
          return;
         }
        }
         else FlushBuf();
       } else if (t!='a' && t!='i' && t!='u' && t!='e' && t!='o' && t!='y' && p!='c')
FlushBuf();
      }

      if (t!='\u0000') buf+=t;
      if (d!='\u0000') buf+=d;
 }
```

Листинг П2. Служебная функция `FlushBuf`

```
private void FlushBuf()
  {
      char s, s2='\u0000', s3='\u0000';

      if (buf.length()==0) return;
      s=buf.charAt(0);
      if (buf.length()>1) s2=buf.charAt(1);
      if (s=='c' || s=='y' && s2=='\u0000')
      {
          dest+=s;
          buf="";
          dbl=false;
          lng=false;
```

```
    return;
}

 if (buf.length()>2) s3=buf.charAt(2);

if (s=='y' && s2=='i')
{
 s='i';
 s2=s3;
}
if (s2=='y' && s3=='i')
{
 s2='i';
 s3='\u0000';
}
if (s=='w' && s2=='\u0000')
 s='u';

if (s2=='\u0000' && s!='a' && s!='i' && s!='u' && s!='e' && s!='o'
                 && s!='y' && s!='n')
 s2='u';

if (s3=='\u0000' && s2!='a' && s2!='i' && s2!='u' && s2!='e' && s2!='o'
                 && s2!='\u0000')
 s3='u';

switch (destCode.getSelectedIndex())
{
 case 0:
         if (s=='n' && s2=='\u0000' && lng)
         {
          s='m';
          lng=false;
         }
         buf=E2R(s, s2);
         if (s!='y') buf+=E2R(s2, s3);
         if (dbl) buf=buf.charAt(0)+buf;
         if (lng) buf+=":";
         dest+=buf;
         break;
 case 2:
         if ((s=='s' || s=='t') && s2=='y') s2='h';
         if (s=='s' && s2=='i')
         {
          s2='h'; s3='i';
         }
         if (s=='t' && s2=='i')
         {
          s='c'; s2='h'; s3='i';
         }
         if (s=='t' && s2=='u')
         {
          s2='s'; s3='u';
         }
         if (s=='z' && s2=='y')
         {
          s='j'; s2=s3; s3='\u0000';
         }
         if (s=='z' && s2=='i') s='j';
         if (s=='h' && s2=='u') s='f';

         if (s=='n' && s2=='\u0000' && lng)
         {
          s='m';
          lng=false;
         }
 case 1:
```

```
        if (s=='n' && s2=='\u0000' && lng)
         lng=false;
        buf=new Character(s).toString();
        if (dbl) buf+=buf;
        if (s2!='\u0000') buf+=s2;
        if (s3!='\u0000')
        {
         buf+=s3;
         s2=s3;
        }
        if (lng)
        {
         if (s2=='\u0000') s2=s;
         if (s2!='o') buf+=s2;
         else buf+="u";
        }
        dest+=buf;
        break;
case 3:
case 4:
        if (s=='n' && s2=='\u0000' && lng)
         lng=false;
        if (s2=='y'){ if (s!='w' && s!='v') s2='i'; else s2='u';}
        switch (s)
        {
         case 'k': switch(s2)
                   {
                    case 'a':     buf="\u304B"; break;
                    case 'i':     buf="\u304D"; break;
                    case 'u':     buf="\u304F"; break;
                    case 'e':     buf="\u3051"; break;
                    case 'o':     buf="\u3053"; break;
                   }
                   break;
         case 'g': switch(s2)
                   {
                    case 'a':     buf="\u304C"; break;
                    case 'i':     buf="\u304E"; break;
                    case 'u':     buf="\u3050"; break;
                    case 'e':     buf="\u3052"; break;
                    case 'o':     buf="\u3054"; break;
                   }
                   break;
         case 's': switch(s2)
                   {
                    case 'a':     buf="\u3055"; break;
                    case 'i':     buf="\u3057"; break;
                    case 'u':     buf="\u3059"; break;
                    case 'e':     buf="\u305B"; break;
                    case 'o':     buf="\u305D"; break;
                   }
                   break;
         case 'z': switch(s2)
                   {
                    case 'a':     buf="\u3056"; break;
                    case 'i':     buf="\u3058"; break;
                    case 'u':     buf="\u305A"; break;
                    case 'e':     buf="\u305C"; break;
                    case 'o':     buf="\u305E"; break;
                   }
                   break;
         case 't': switch(s2)
                   {
                    case 'a':     buf="\u305F"; break;
                    case 'i':     buf="\u3061"; break;
                    case 'u':     buf="\u3064"; break;
                    case 'e':     buf="\u3066"; break;
```

```
            case 'o':     buf="\u3068"; break;
           }
           break;
case 'd': switch(s2)
           {
            case 'a':     buf="\u3060"; break;
            case 'i':     buf="\u3062"; break;
            case 'u':     buf="\u3065"; break;
            case 'e':     buf="\u3067"; break;
            case 'o':     buf="\u3069"; break;
           }
           break;
case 'n': switch(s2)
           {
            case '\u0000':buf="\u3093"; break;
            case 'a':     buf="\u306A"; break;
            case 'i':     buf="\u306B"; break;
            case 'u':     buf="\u306C"; break;
            case 'e':     buf="\u306D"; break;
            case 'o':     buf="\u306E"; break;
           }
           break;
case 'f': switch(s2)
           {
            case 'a':     buf="\u3075\u3041"; break;
            case 'i':     buf="\u3075\u3043"; break;
            case 'u':     buf="\u3075"; break;
            case 'e':     buf="\u3075\u3047"; break;
            case 'o':     buf="\u3075\u3049"; break;
           }
           break;
case 'h': switch(s2)
           {
            case 'a':     buf="\u306F"; break;
            case 'i':     buf="\u3072"; break;
            case 'u':     buf="\u3075"; break;
            case 'e':     buf="\u3078"; break;
            case 'o':     buf="\u307B"; break;
           }
           break;
case 'b': switch(s2)
           {
            case 'a':     buf="\u3070"; break;
            case 'i':     buf="\u3073"; break;
            case 'u':     buf="\u3076"; break;
            case 'e':     buf="\u3079"; break;
            case 'o':     buf="\u307D"; break;
           }
           break;
case 'p': switch(s2)
           {
            case 'a':     buf="\u3071"; break;
            case 'i':     buf="\u3074"; break;
            case 'u':     buf="\u3077"; break;
            case 'e':     buf="\u307A"; break;
            case 'o':     buf="\u307C"; break;
           }
           break;
case 'm': switch(s2)
           {
            case 'a':     buf="\u307E"; break;
            case 'i':     buf="\u307F"; break;
            case 'u':     buf="\u3080"; break;
            case 'e':     buf="\u3081"; break;
            case 'o':     buf="\u3082"; break;
           }
           break;
```

```
 case 'r': switch(s2)
           {
            case 'a':     buf="\u3089"; break;
            case 'i':     buf="\u308A"; break;
            case 'u':     buf="\u308B"; break;
            case 'e':     buf="\u308C"; break;
            case 'o':     buf="\u308D"; break;
           }
           break;
 case 'w': switch(s2)
           {
            case 'a':     buf="\u308F"; break;
            case 'i':     buf="\u3046\u3043"; break;
            case 'u':     buf="\u3046"; break;
            case 'e':     buf="\u3046\u3047"; break;
            case 'o':     buf="\u3092"; break;
           }
           break;
 case 'v': switch(s2)
           {
            case 'a':     buf="\u3094\u3041"; break;
            case 'i':     buf="\u3094\u3043"; break;
            case 'u':     buf="\u3094"; break;
            case 'e':     buf="\u3094\u3047"; break;
            case 'o':     buf="\u3094\u3049"; break;
           }
           break;
 case 'y': switch(s2)
           {
            case 'a':     buf="\u3084"; break;
            case 'u':     buf="\u3086"; break;
            case 'o':     buf="\u3088"; break;
           }
           break;
 case 'a': buf="\u3042"; break;
 case 'i': buf="\u3044"; break;
 case 'u': buf="\u3046"; break;
 case 'e': buf="\u3048"; break;
 case 'o': buf="\u304A"; break;
 default: buf="~"; break;
}
if (s3=='a') buf+="\u3083";
 else if (s3=='u') buf+="\u3085";
  else if (s3=='o') buf+="\u3087";
if (destCode.getSelectedIndex()==3)
{
 if (buf.length()==1)
  buf=new Character((char)((int)buf.charAt(0)+0x60)).toString();
 else
  buf=new Character((char)((int)buf.charAt(0)+0x60)).toString()
     +(char)((int)buf.charAt(1)+0x60);
 if (dbl) buf="\u30C3"+buf;
 if (lng) buf+="\u30FC";

} else
{
 if (dbl) buf="\u3063"+buf;
 if (lng)
 {
  if (s3=='o' || s2=='o')
   buf+="\u3046";
  else
   buf+="\u304A";
 }
}
dest+=buf;
break;
```

```
  default:
            dest+="{";
            if (dbl) dest+="*";
            dest+=buf;
            if (lng) dest+=":";
            dest+="}";
 }

 buf="";
 dbl=false;
 lng=false;
}
```