УДК 681.3.06

ЛОКАЛИЗАЦИЯ КОНСОЛЬНЫХ ПРИЛОЖЕНИЙ В ЯЗЫКЕ C++

© 2011 г. ***В.Л. Тарасов***

Нижегородский госуниверситет им. Н.И. Лобачевского

vl-tarasov@yandex.ru

Поступила в редакцию 04.03.2011

Рассматриваются методы локализации консольных приложений, существующие в языке C++, и их реализация в трех средах разработки – Visual Studio, C++Builder и Eclipse. Установлено, что возможность русификации стандартными средствами языка программирования полностью реализована только в Visual Studio. Даются рекомендации по методам локализации.

Ключевые слова: консольные приложения, локализация, C++, кодовые страницы, перекодировка, среда разработки.

При разработке консольных приложений возникает проблема их локализации (русификации), вызванная тем, что в средах разработки и выполнения программ используются разные кодовые страницы с разными кодами для русских букв.

Символы с кодами 0x00 – 0x7F во всех кодовых страницах одинаковы и образуют набор символов ASCII (American Standard Code for Information Interchange – американский стандартный код для обмена информацией). Символы с кодами 0x80 – 0xFF используются для кодирования символов национальных алфавитов. На рис. 1 показана вторая половина кодовой страницы Windows-1251 [1], которую использует операционная система Windows для России. Код символа равен сумме шестнадцатеричных номеров строки и столбца. Консольные приложения выполняются в текстовых окнах, где используется кодовая страница CP866 [2] (рис. 2). В этих страницах для русских букв выделены разные диапазоны кодов, поэтому программа, созданная с применением кодовой страницы 1251, будет выводить вместо русских букв символы кодовой страницы 866, имеющие те же коды, что и русские буквы в странице 1251.

Однобайтовые схемы кодирования позволяют работать только с 256 символами, что гораздо меньше количества используемых символов, поэтому были развиты многобайтовые схемы кодирования, чтобы знаки могли быть представлены в 8-битовых, 16-битовых, 24-битовых,

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
80	Ђ 0402	Ѓ 0403	‚ 201A	ѓ 0453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 0409	‹ 2039	Њ 040A	Ќ 040C	Ћ 040B	Џ 040F
90	ђ 0452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	љ 0459	› 203A	њ 045A	ќ 045C	ћ 045B	џ 045F
A0	NBSP 00A0	Ў 040E	ў 045E	Ј 0408	¤ 00A4	Ґ 0490	¦ 00A6	§ 00A7	Ё 0401	© 00A9	Є 0404	« 00AB	¬ 00AC	- 00AD	® 00AE	Ї 0407
B0	° 00B0	± 00B1	І 0406	і 0456	ґ 0491	µ 00B5	¶ 00B6	· 00B7	ё 0451	№ 2116	є 0454	» 00BB	ј 0458	Ѕ 0405	ѕ 0455	ї 0457
C0	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
D0	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
E0	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
F0	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F

Рис. 1. Кодовая страница Windows 1251

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
80	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
90	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
A0	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
B0	▒ 2591	▓ 2592	▓ 2593	│ 2502	┤ 2524	╡ 2561	╢ 2562	╖ 2556	╕ 2555	╣ 2563	║ 2551	╗ 2557	╝ 255D	╜ 255C	╛ 255B	┐ 2510
C0	└ 2514	┴ 2534	┬ 252C	├ 251C	─ 2500	┼ 253C	╞ 255E	╟ 255F	╚ 255A	╔ 2554	╩ 2569	╦ 2566	╠ 2560	═ 2550	╬ 256C	╧ 2567
D0	╨ 2568	╤ 2564	╥ 2565	╙ 2559	╘ 2558	╒ 2552	╓ 2553	╫ 256B	╪ 256A	┘ 2518	┌ 250C	▉ 2588	▄ 2584	▌ 258C	▐ 2590	▀ 2580
E0	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F
F0	Ё 0401	ё 0451	Є 0404	є 0454	Ї 0407	ї 0457	Ў 040E	ў 045E	° 00B0	∙ 2219	· 00B7	√ 221A	№ 2116	¤ 00A4	■ 25A0	NBSP 00A0

Рис. 2. Кодовая страница OEM 866

или 32-битовых последовательностях. Стандарт Unicode (Юникод) для представления данных [3, 4] задаёт однозначное соответствие символов кодам – элементам кодового пространства, представляющим неотрицательные целые числа, которые называются кодовыми точками. Например, цифре 0 соответствует кодовая точка U+0030. В образцах кодовых страниц, приведенных на рисунках 1 и 2, коды символов в кодировке Юникод указаны под знаками, например, для русских букв отведен диапазон от U+0410 (заглавная А) до U+044F (малая буква я). Коды русских букв образуют последовательность, возрастающую в алфавитном порядке. Исключение составляют буквы Ё(U+0401) и ё(U+0451), которые расположены вне общего алфавитного порядка. Доступные в среде Windows символы и их кодовые точки можно увидеть с помощью программы **Таблица символов** (запускается командой **Пуск, Все программы, Стандартные, Служебные, Таблица символов**).

Чтобы консольное приложение «заговорило» по-русски, нужно обеспечить преобразование кодировки символов к среде выполнения. Далее рассмотрены средства учета национальных особенностей (локализации), имеющиеся в языках C и C++.

Средства локализации языка C

Для учета особенностей, связанных со страной и языком, используются специальные среды, называемые *локальными контекстами* (*locale* – место действия) [5], которые включают набор параметров и функций, обеспечивающих поддержку национальных и культурных стандартов.

Локальный контекст определяется строкой формата:

`язык[_зона[.код]]`

Здесь `язык` – обозначение языка (например, английский, немецкий, русский), а `зона` – страна, в которой используется язык. Этот квалификатор позволяет поддерживать национальные стандарты для различных стран, использующих один язык. Квалификатор `код` определяет кодовую страницу.

В таблице приведены примеры строк, определяющих локальные контексты. Эти строки не стандартизированы, поэтому поддержка тех или иных локальных контекстов зависит от реализации.

Таблица

Имя	Описание
C	Используется по умолчанию: соглашения стандарта ANSI-C, английский язык
de_DE	Немецкий язык (Германия)
de_AT	Немецкий язык (Австрия)
rus	Русский язык (Россия). Кодовая страница по умолчанию
Rus	
russian	
Russian	
Russian_Russia	
Russian_Russia.1251	Русский язык (Россия). Кодовая страница 1251
Russian_Russia.866	Русский язык (Россия). Кодовая страница 866

Средства языка C для работы с локальными контекстами объявлены в заголовке `locale`. Настройку программы на конкретный локальный контекст выполняет функция:

```
char *setlocale( int category, const char *locale );
```

Аргумент category определяет категорию функций, на которые setlocale оказывает влияние:

```
LC_ALL      - все категории
LC_COLLATE  - сравнение и преобразование строк
LC_CTYPE    - обработка символов
LC_MONETARY - форматирование денежных сумм
LC_NUMERIC  - форматирование чисел
LC_TIME     - форматирование времени
```

Аргумент locale является указателем на строку, задающую имя локального контекста. Если locale указывает на пустую строку, используется локальный контекст, использующий кодовую страницу, получаемую из операционной системы. Если locale равен нулю (NULL), действующий локальный контекст не изменяется.

При успешном завершении работы функция setlocale возвращает указатель на статически определенную строку с описанием локального контекста или нулевой указатель при неудаче.

Возможны несколько вариантов вызова setlocale, например:

setlocale(LC_ALL, "Russian") – настройка всех функций на Россию;

setlocale(LC_CTYPE, "Russian") – настройка функций обработки символов на Россию;

setlocale(LC_CTYPE, ".1251") – настройка функций обработки символов на кодовую страницу 1251.

Программа 1. Использование функции setlocale

Программа выводит русское слово, заданное непосредственно в программе и введенное с консоли (клавиатуры) при локальном контексте по умолчанию, локальном контексте с настройками из ОС и явно заданными локальными контекстами, использующими кодовые страницы 866 и 1251.

```
#include <iostream>
#include <locale>
#include <conio.h>
using namespace std;
void main()
{
   string program = "Программа", console;
   cout << "Default locale is\t"
      << setlocale(LC_ALL, NULL) << endl;
   cout << "Enter the Russian word: ";
   cin >> console;
   cout << program << "\t"
      << console << endl;
// Установка локального контекста
// с настройками из ОС
   cout << "OS locale is\t"
      << setlocale(LC_ALL, "") << endl;
   cout << program << "\t"
      << console << endl;
   cout << "DOS locale is\t"
     << setlocale(LC_CTYPE, ".866") << endl;
   cout << program << "\t"
      << console << endl;
   cout << "Locale: "
      << setlocale(LC_CTYPE, ".1251") << endl;
   cout << program << endl;
   cout << "Locale: "
      << setlocale(LC_CTYPE, ".866") << endl;
   cout << console << endl;
   getch();
   return 0;
}
```

При выполнении программы с консоли вводилось слово «Консоль». В среде Visual Studio получаем:

```
Default locale is        C
Enter the Russian word:  Консоль
╧ЁюуЁрьр       Консоль
OS locale is     Russian_Russia.1251
Программа        ?R-бR<м
DOS locale is    Russian_Russia.866
╧ЁюуЁрьр       Консоль
Locale: Russian_Russia.1251
Программа
Locale: Russian_Russia.866
Консоль
```

Видно, что по умолчанию устанавливается стандартный контекст C, при действии которого русские буквы из программы не выводятся, а введенные с консоли выводятся.

Вызов setlocale(LC_ALL, "") устанавливает локальный контекст с настройками ОС, где предусмотрено использование кодовой страницы 1251, в котором русские буквы из программы выводятся правильно, а введенные с консоли – нет.

Поскольку в консольном окне используется кодовая страница 866, для правильного вывода русских букв, введенных из консольного окна, нужно изменять локальный контекст на кодовую страницу 866 вызовом setlocale(LC_CTYPE, ".866"), но при этом русские буквы, заданные в программе, выводятся неправильно.

Таким образом, при выводе русских букв, заданных непосредственно в программе, следует устанавливать кодовую таблицу 1251, а перед выводом русского текста, введенного с клавиатуры, необходимо установить кодовую таблицу 866.

В среде C++ Builder программа выдает:

```
Default locale is            C
Enter the Russian word:      Консоль
╧ЁюуЁрьр        Консоль
```

```
OS locale is LC_MONETARY=Russian_Russia.866
LC_TIME=Russian_Russia.866
LC_NUMERIC=Russian_Russia.866
LC_COLLATE=Russian_Russia.866
LC_CTYPE=Russian_Russia.866
╧ЁюуЁрььр         Консоль
DOS locale is LC_CTYPE=Russian_Russia.866
╧ЁюуЁрььр         Консоль
Locale: LC_CTYPE=Russian_Russia.1251
╧ЁюуЁрььр
Locale: LC_CTYPE=Russian_Russia.866
Консоль
```

Здесь, так же как в Visual Studio, локальным контекстом по умолчанию является C, но локальным контекстом, устанавливаемым по настройкам ОС, является Russian_Russia.866, в то время как в Visual Studio это Russian_Russia.1251. Сведения о локальном контексте здесь выводятся подробно с указанием всех категорий. Установка локального контекста на Russian_Russia.1251 не обеспечивает правильного вывода из программы русских букв.

Та же программа, построенная в среде Eclipse, выводит:

```
Default locale is       C
Enter the Russian word: Консоль
╧ЁюуЁрььр        Консоль
OS locale is     Russian_Russia.1251
╧ЁюуЁрььр        Консоль
DOS locale is    Russian_Russia.866
╧ЁюуЁрььр        Консоль
Locale: Russian_Russia.1251
╧ЁюуЁрььр
Locale: Russian_Russia.866
Консоль
```

Таким образом, в C++ Builder и Eclipse не реализовано преобразование символов из кодовой страницы 1251, используемой в среде разработки, в кодовую страницу 866 среды выполнения программ. Правильно выводятся только строки русского текста, вводимые из консольного окна.

Средства локализации языка C++

В языке C++, так же как в C, используется концепция локального контекста, включающего настройки обработки символов, форматирования чисел, даты, времени, денежных величин. Локальный контекст реализуется библиотечным классом locale, объявленным в заголовке locale. Используя класс locale, можно создать в программе несколько объектов локальных контекстов и использовать их по мере необходимости.

В классе locale есть несколько конструкторов, создающих объекты локальных контекстов.

Конструктор по умолчанию locale() создает объект локального контекста, являющийся копией глобально назначенного локального контекста на момент конструирования.

Конструктор, аргументом которого является пустая строка – locale(""), создает объект локального контекста с параметрами из настроек операционной системы.

Аргументом конструктора класса locale может быть строка, описывающая локальный контекст, такая же, как аргумент функции setlocale.

Статическая функции global класса locale устанавливает заданный локальный контекст как глобальный, после чего он оказывает влияние на все аспекты локализации.

Метод name класса locale возвращает строку типа string с именем локального контекста.

Программа 2. Использование класса locale

Программа аналогична программе 1, но используются средства C++ для управления локальными контекстами.

```
#include <iostream>
#include <locale>
#include <conio.h>
using namespace std;
int main()
{
// Создание копии действующего по умолчанию
// локального контекста
   locale def_loc;
// Вывод имени локального контекста
   cout << "Default locale is\t"
        << def_loc.name() << endl;
   string program = "Программа", console;
   cout << "Enter the Russian word: ";
   cin >> console;
   cout << program << "\t"
        << console << endl;
// Создание локального контекста
// с установками из ОС
   locale os_loc("");
   cout << "OS locale is\t"
        << os_loc.name() << endl;
// Установка для использования лок. контекста
   locale::global(os_loc);
   cout << program << "\t"
        << console << endl;
// Создание локального контекста для России
// с кодовой страницей 866 для вывода русских
// букв, вводимы с консоли в кодировке 866
   locale Output866("Russian_Russia.866");
// Установка для использования лок. контекста
   locale::global(Output866);
   cout << "Locale is\t"
        << Output866.name() << endl;
```

```
   cout << program << "\t"
        << console << endl;
// Создание локального контекста для России
// с кодовой страницей 1251 для вывода русских
// букв, заданных в программе в кодировке 1251
   locale Output1251("Russian_Russia.1251");
// Установка для использования лок. контекста
   locale::global(Output1251);
   cout << "Locale is\t"
        << Output1251.name() << endl;
   cout << program << "\n";
// Повторное использование объекта лок. контекста
   locale::global(Output866);
   cout << "Locale is\t"
        << Output866.name() << endl;
   cout << console << endl;
   getch();
   return 0;
}
```

При запуске в среде Visual Studio программа выводит:

```
Default locale is       C
Enter the Russian word: Консоль
╧ЁюуЁрьр        Консоль
OS locale is    Russian_Russia.1251
Программа       ?R-бR<м
Locale is       Russian_Russia.866
╧ЁюуЁрьр        Консоль
Locale is       Russian_Russia.1251
Программа
Locale is       Russian_Russia.866
Консоль
```

Видно, что русские буквы, заданные в программе, правильно выводятся при установке для использования кодовой страницы 1251, а для правильного вывода русских букв, вводимых с консоли, должна быть установлена кодовая страница 866.

В средах C++ Builder и Eclipse не реализовано преобразование кодировок, то есть русский текст, заданный непосредственно в программе, не выводится правильно на консоль. Удается вывести только вводимые с консоли русские буквы. Далее приведен вывод программы, созданной в Eclipse. Вывод программы, созданной с использованием C++Builder, аналогичен, но более подробен и поэтому не приводится.

```
Default locale is       C
Enter the Russian word: Консоль
╧ЁюуЁрьр        Консоль
OS locale is    C
╧ЁюуЁрьр        Консоль
Locale is       Russian_Russia.866
╧ЁюуЁрьр        Консоль
Locale is       Russian_Russia.1251
╧ЁюуЁрьр
Locale is       Russian_Russia.866
Консоль
```

Таким образом, использование класса `locale` дает те же результаты, что и использование функции `setlocale` языка C.

Нестандартные функции настройки консоли

Кроме локализации, возможность которой заложена в стандартах языков C и C++, существуют нестандартные средства. В Windows имеются функции для управления консольным окном DOS, для доступа к которым в программу следует включить заголовочный файл `Windows.h`.

Функция

```
UINT GetConsoleCP(void);
```

извлекает входную кодовую страницу, используемую консолью, ассоциированной с запущенным процессом (программой) для преобразования клавиатурного ввода в соответствующие символьные значения. Возвращаемое значение является номером кодовой страницы. Тип возвращаемого значения `UINT` определен в заголовочном файле `windef.h` инструкцией

```
typedef unsigned int UINT;
```

Функция

```
BOOL SetConsoleCP(UINT wCodePageID);
```

устанавливает кодовую страницу, используемую консолью, ассоциированной с запущенным процессом, при вводе символов с клавиатуры. Параметр `wCodePageID` является идентификатором устанавливаемой кодовой страницы. При успешном завершении возвращается ненулевое значение. Если работа функции неудачна, возвращается нуль.

Функция

```
UINT GetConsoleOutputCP(void);
```

извлекает выходную кодовую страницу, используемую консолью, ассоциированной с вызванным процессом для перевода символьных значений, записываемых различными выводящими функциями в изображения символов, показываемые в консольном окне. Возвращает номер кодовой страницы.

Функция

```
BOOL SetConsoleOutputCP (UINT wCodePageID);
```

устанавливает выходную кодовую страницу, используемую консолью, ассоциированной с

вызванным процессом. Параметр wCodePageID является идентификатором устанавливаемой кодовой страницы. При успешном завершении возвращается ненулевое значение.

Программа 3. Настройка консоли

Программа использует функции для управления кодовыми страницами при вводе с консоли и при выводе на консоль.

```
#include <windows.h>
#include <iostream>
#include <conio.h>
using namespace std;
void main()
{
   cout << "GetConsoleCP() =\t"
   << GetConsoleCP() << endl;
   cout << "GetConsoleOutputCP() =\t"
      << GetConsoleOutputCP() << endl;
// Кодовая страница 1251 для ввода из консоли
   SetConsoleCP(1251);
// Кодовая страница 1251 для вывода на консоль
   SetConsoleOutputCP(1251);
   char Name[100];
   cout << "Введите Ваше имя\t";
   cin >> Name;
   cout << "Здравствуйте, \t\t" << Name << endl;
   getch() ;
}
```

В программе сначала получаются входные и выходные кодовые страницы, затем для ввода и вывода устанавливается кодовая страница 1251, которая обеспечивает корректный ввод и вывод русских букв.

Далее приведены результаты работы программы.

```
GetConsoleCP() =        866
GetConsoleOutputCP() =  866
Введите Ваше имя        Владимир
Здравствуйте,           Владимир
```

Видно, что по умолчанию для ввода с консоли и вывода на консоль устанавливается кодовая страница 866. Для согласования среды разработки и консоли следует установить для консольного ввода и вывода кодовую страницу 1251.

При использовании функций настройки консоли русские буквы будут отображаться правильно, если для консольного окна выбран шрифт Lucida Console. Для выбора шрифта нужно щелкнуть правой кнопкой мыши по заголовку окна, выполнить команду **Свойства**, а затем на вкладке **Шрифт** выбрать нужный шрифт.

Программа 3 работает одинаково как в среде Visual Studio, так и в средах C++Builder и Eclipse, но только тогда, когда для консоли используется оконный режим. При полноэкранном режиме в консольном окне русские буквы не отображаются.

Прямое преобразование символов

Для преобразования кодировки символов из кодовой страницы, действующей в среде разработки, в кодовую страницу, используемую в среде выполнения, можно вставить в программу заголовочный файл windows.h и использовать функцию:

```
BOOL CharToOem( LPCTSTR lpszSrc, LPSTR lpszDst );
```

Здесь lpszSrc – указатель на строку для перевода, lpszDst – указатель на буфер для преобразованной строки.

Функция CharToOem существует в двух вариантах: для однобайтовых (ANSI) и для широких символов. Если функция CharToOem используется как ANSI функция, строка может быть переведена на месте установкой для параметров lpszDst и lpszSrc одинакового значения. Это не может быть сделано, если CharToOem используется как функция для обработки широких символов.

В качестве аргументов функции CharToOem можно передавать обычные строки символов с нулевым байтом на конце. Возвращаемое функцией CharToOem значение всегда не нуль, за исключением случая, когда передаются одинаковые адреса для lpszSrc и lpszDst при использовании версии функции для широких символов. В этом случае функция возвращает нуль.

Программа 4. Использование CharToOem

Далее приведен пример использования функции CharToOem. Для удобства использования написана вспомогательная функция Rus, возвращающая указатель на строку, преобразованную в кодировку OEM.

```
#include <iostream>
#include <windows.h>
#include <conio.h>
using namespace std;
char* Rus(char* s)
{
   static char buf[500];
   CharToOem(s, buf);
   return buf;
}
```

```
int main()
{
   cout << Rus("Как Вас зовут?\t");
   char Name[100];
   cin >> Name;
   cout << Rus("Здравствуйте,\t");
   cout << Name << endl;
   getch();
   return 0;
}
```

Пример работы программы:

```
Как Вас зовут? Владимир
Здравствуйте, Владимир
```

При вводе строки инструкцией

```
cin >> Name;
```

используется кодовая страница 866 консольного окна, поэтому при обратном выводе на консоль строки `Name` ее не требуется преобразовывать для правильного отображения русских букв.

В средах С++ Buider и Eclipse данная пограмма работет точно так же.

Выводы

Заложенная стандартами языков С и С++ возможность русификации консольных приложений реализована только в Microsoft Visual Studio. Нестандартные средства локализации, реализованные в ОС Windows, обеспечивают локализацию и работают одинаково во всех рассмотренных средах разработки. Выбор средства локализации остается за программистом, но использование функции `CharToOem` представляется менее удобным, так как эту функцию приходится вызывать при выводе каждой созданной в программе строки с русскими буквами. Более удобным представляется локализация, реализуемая путем настройки консоли с использованием функций `SetConsoleCP` и `SetConsoleOutputCP`, продемонстрированная в программе 3, которая переносима между средами разработки.

Список литературы

1. Windows 1251 [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/ru-ru/goglobal/cc305144

2. OEM 866 [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/ru-ru/goglobal/cc305166

3. The Unicode Consortium [Электронный ресурс]. – Режим доступа: http://www.unicode.org

4. Юникод – Википедия [Электронный ресурс]. – Режим доступа: http: http://ru.wikipedia. org/wiki/Unicode

5. Джосьютис Н. С++ Стандартная библиотека. Для профессионалов. СПб.: Питер, 2004. 730 с.

LOCALIZATION OF CONSOLE APPLICATIONS IN C++ LANGUAGE

V.L. Tarasov

The methods of localization of console applications that exist in the language C ++, and their implementation in the three development environments – Visual Studio, C ++ Builder and Eclipse are considered. It is established that the possibility of Russification of the standard language C++ has been fully realized only in Visual Studio. Some recommendations on localization methods are given.

Keywords: console applications, localization, C++, code pages, recoding, development environment.