УДК 004.432

С. В. ГРИНЧЕНКО

НАСТРОЙКА ВВОДА-ВЫВОДА РУССКОГО ТЕКСТА В КОНСОЛЬНЫХ ПРИЛОЖЕНИЯХ VISUAL C++ 2010

Зроблено зіставлення символів кодових таблиць cp1251 і cp866, що дозволяє аналізувати помилки некоректного введення-виведення символів рядків. Наведено способи русифікації введення та виведення в консольних додатках Visual C++ в середовищі програмування Microsoft Visual Studio 2010 із використанням: 1) функції `setlocale`; 2) функцій перекодування `CharToOemA` та `OemToCharA`; 3) функцій `SetConsoleCP` та `SetConsoleOutputCP`; 4) функцій `wcout.imbue()` та `wcin.imbue()` в програмах введення та виведення широкосимвольних літералів. Виклад матеріалу супроводжується прикладами вихідних кодів.

Ключові слова: Visual C++, консольний додадок, кодова сторінка.

Произведено сопоставление символов кодовых таблиц cp1251 и cp866, позволяющее анализировать ошибки некорректного ввода-вывода символов строк. Приведены способы русификации ввода-вывода в консольных приложениях Visual C++ в среде программирования Microsoft Visual Studio 2010 с использованием: 1) функции `setlocale`; 2) функций перекодировок `CharToOemA` и `OemToCharA`; 3) функций `SetConsoleCP` и `SetConsoleOutputCP`; 4) функций `wcout.imbue()` и `wcin.imbue()` в программах ввода-вывода широкосимвольных литералов. Изложение материала сопровождается примерами исходных кодов.

Ключевые слова: Visual C++, консольное приложение, кодовая страница.

Comparison of characters by cp1251 and cp866 code pages for the purpose to analyze the errors of incorrect input/output of them is made. There are presented the different methods of input/output Russian localization in Visual C++ console applications in environment Microsoft Visual Studio 2010 with the use: 1) function `setlocale`; 2) conversion functions `CharToOemA` and `OemToCharA`; 3) functions `SetConsoleCP` and `SetConsoleOutputCP`; 4) functions `wcout.imbue()` and `wcin.imbue()` for input/output of wide character strings. The paper is accompanied by the examples of source codes.

Keywords: Visual C++, console application, code page.

Постановка проблемы. Хотя количество книг на русском языке, связанных с тематикой C++, исчисляется тысячами наименований, число источников, где в какой-то мере освещалось решение национальных особенностей C++ (в частности, ввода-вывода русского текста), очень ограничено. Методы решения этих проблем не стандартизированы, поэтому решение одной и той же задачи в различных компиляторах требует разного подхода.

Целью статьи является анализ и классификация средств русификации текста консольных приложений Visual C++ 2010.

Необходимость русификации также обусловлена тем, что на языках C, C++ написаны многие модели нейтральной атмосферы, магнитного поля Земли, ионосферных электрических полей, использующиеся в теоретических расчётах ионосферных параметров. Этими программами пользуются научные работники бывших стран СНГ, где русскоязычная часть является значительной.

Общие сведения о кодировках символов. Соответствие между символами и кодирующими их байтами называется кодировкой символов. Понятно, что, во-первых, что каждая кодировка разрабатывается для конкретного национального языка (точнее, для конкретной письменности), и, во-вторых, что для любого языка таких кодировок можно придумать сколько угодно.

Кодировка ASCII – однобайтная кодировка, используемая для представления в компьютере текстовых данных. Семи- или восьмибитная таблица ASCII позволяет закодировать 128 или 256 разных знаков. Стандартная кодировка ASCII (ASCII-7) использует 7 битов для представления всех прописных и строчных букв, чисел от 0 до 9, знаков препинания и специальных управляющих символов, применяемых в английской раскладке для США. Расширенная кодировка ASCII (ASCII-8) использует восьмой бит каждого кода для представления 128 дополнительных специальных символов, букв различных алфавитов и графических знаков.

Стандартная кодировка ASCII – это семибитная кодовая таблица (коды символов $00\text{-}7F_{16}$ или $0\text{-}127_{10}$). В ней байты с шестнадцатеричными кодами $00\text{-}1F_{16}$ ($0\text{-}31_{10}$) и $7F_{16}$ (127_{10}) используются для кодирования управляющих (неотображаемых) символов, а остальные кодируют различные другие символы.

128 символов с кодами от $128_{10} = 10000000_2 = 80_{16}$ до $255_{10} = 11111111_2 = FF_{16}$ – альтернативная часть таблицы ASCII. Альтернативная часть таблицы расширенной кодировки ASCII называется кодовой страницей (128 кодов, начиная с $128_{10} = 10000000_2$ и кончая $255_{10} = 11111111_2$). Кодовая страница может иметь различные варианты, каждый вариант имеет свой номер.

Кодовые страницы в первую очередь используется для размещения национальных алфавитов, отличных от латинского. В русских национальных кодировках в этой части таблицы размещаются символы русского алфавита.

Среди наиболее распространённых кодировок Microsoft Windows можно назвать следующие: Windows-1250 (ASCII-7 и cp1250) – для языков Центральной Европы, которые используют латинское написание букв (польский, чешский, словацкий, венгерский, словенский, хорватский, румынский и албанский); Windows-1251 (ASCII-7 и cp1251) – для кириллических алфавитов; Windows-1252 (ASCII-7 и cp1252) – для западноевропейских языков;

Windows-1253 (ASCII-7 и cp1253) – для греческого языка; Windows-1254 (ASCII-7 и cp1254) – для турецкого языка; Windows-1255 (ASCII-7 и cp1255) – для иврита.

Стандартные кодировки русских версий Microsoft Windows и её консоли. Кодировка Windows-1251, состоящая из стандартной кодировки ASCII-7 и альтернативной части – кодовой страницы cp1251, является стандартной (т.е. используемой по умолчанию) кодировкой для всех русских версий Microsoft Windows. Однако в консоли русифицированных систем семейства Windows используется кодовая страница cp866. В таблице 1 приведено сравнение символов кодовых таблиц cp1251 и cp866.

Таблица 1 – Символы кодовых таблиц cp1251 и cp866

10-тичное значение байта	16-ричное значение байта	Символ cp866	Символ cp1251
128	80	А	Ђ
129	81	Б	Ѓ
130	82	В	‚
131	83	Г	ѓ
132	84	Д	„
133	85	Е	…
134	86	Ж	†
135	87	З	‡
136	88	И	€
137	89	Й	‰
138	8a	К	Љ
139	8b	Л	‹
140	8c	М	Њ
141	8d	Н	Ќ
142	8e	О	Ћ
143	8f	П	Џ
144	90	Р	ђ
145	91	С	‘
146	92	Т	’
147	93	У	“
148	94	Ф	”
149	95	Х	•
150	96	Ц	–
151	97	Ч	—
152	98	Ш	b
153	99	Щ	™
154	9a	Ъ	љ
155	9b	Ы	›
156	9c	Ь	њ
157	9d	Э	ќ
158	9e	Ю	ћ
159	9f	Я	џ

Продолжение таблицы 1

160	a0	а	NBSP
161	a1	б	Ў
162	a2	в	ў
163	a3	г	J
164	a4	д	¤
165	a5	е	Ґ
166	a6	ж	¦
167	a7	з	§
168	a8	и	Ё
169	a9	й	©
170	aa	к	Є
171	ab	л	«
172	ac	м	¬
173	ad	н	-
174	ae	о	®
175	af	п	Ї
176	b0	░	°
177	b1	▒	±
178	b2	▓	I
179	b3	│	i
180	b4	┤	ґ
181	b5	╡	µ
182	b6	╢	¶
183	b7	╖	·
184	b8	╕	ё
185	b9	╣	№
186	ba	║	є
187	bb	╗	»
188	bc	╝	j
189	bd	╜	S
190	be	╛	s
191	bf	┐	ї
192	c0	└	А
193	c1	┴	Б
194	c2	┬	В
195	c3	├	Г
196	c4	─	Д
197	c5	┼	Е
198	c6	╞	Ж
199	c7	╟	З
200	c8	╚	И
201	c9	╔	Й
202	ca	╩	К
203	cb	╦	Л
204	cc	╠	М
205	cd	═	Н
206	ce	╬	О
207	cf	╧	П

Завершение таблицы 1

208	d0	╨	Р
209	d1	╤	С
210	d2	╥	Т
211	d3	╙	У
212	d4	╘	Ф
213	d5	╒	Х
214	d6	╓	Ц
215	d7	╫	Ч
216	d8	╪	Ш
217	d9	┘	Щ
218	da	┌	Ъ
219	db	█	Ы
220	dc	▄	Ь
221	dd	▌	Э
222	de	▐	Ю
223	df	▀	Я
224	e0	р	а
225	e1	с	б
226	e2	т	в
227	e3	у	г
228	e4	ф	д
229	e5	х	е
230	e6	ц	ж
231	e7	ч	з
232	e8	ш	и
233	e9	щ	й
234	ea	ъ	к
235	eb	ы	л
236	ec	ь	м
237	ed	э	н
238	ee	ю	о
239	ef	я	п
240	f0	Ё	р
241	f1	ё	с
242	f2	Є	т
243	f3	є	у
244	f4	Ї	ф
245	f5	ї	х
246	f6	Ў	ц
247	f7	ў	ч
248	f8	°	ш
249	f9	∙	щ
250	fa	·	ъ
251	fb	√	ы
252	fc	№	ь
253	fd	¤	э
254	fe	■	ю
255	ff		я

Ошибки вывода русского текста без использования средств русификации. При попытке вывода русского текста в C++ без использования специальных средств русификации происходят ошибки вывода. Демонстрацией возникающих проблем вывода являются следующий пример программы:

Код программы:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
string name;
cout<<"Input your English name: ";
cin>>name;
cout<<"Hello, "<<name<<"!"<<endl;
cout<<"Введите ваше русское имя: ";
cin>>name;
cout<<"Привет, "<<name<<"!"<<endl;
return 0;
}
```

Выполнение программы:

```
Input your English name: John
Hello, John!
┬тхфшЄх тр°х Ёєёёъюх шь : Женя
╧Ёштхє, Женя!
```

Значения строк "Введите ваше имя: " и "Привет, " закодированы в Windows-1251 (ASCII-7 & ср1251). Вывод этих строк осуществился с помощью кодовой страницы ср866. Поэтому из-за несоответствия кодировок (т.е. некорректной интерпретации кодов символов в их графические представления) вывод строк оказался некорректным.

Так, вместо Привет появилась абракадабра ╧Ёштхє. Символу П кодовой страницы ср1251 соответствует символ ╧ кодовой страницы ср866: $П_{1251} \rightarrow ╧_{866}$. Аналогично, $р_{1251} \rightarrow Ё_{866}$, $и_{1251} \rightarrow ш_{866}$, $в_{1251} \rightarrow т_{866}$, $е_{1251} \rightarrow х_{866}$, $т_{1251} \rightarrow Є_{866}$.

Для корректного вывода упомянутых строк необходимо было задать кодировку строки Windows-1251.

Использование функции `setlocale` для задания русской локальности. Для управления представлением и обработкой данных используется заголовочный файл `clocale`. Информацию, описывающую представление и правила обработки данных для определённого географического региона, называют локальным контекстом (или локальностью). Процесс преобразования данных в локальное представление называют локализацией.

Таким образом, локальный контекст представляет собой набор параметров и функций, обеспечивающих поддержку национальных или культурных стандартов. В зависимости от локального контекста выбираются разные форматы вещественных чисел, дат, денежных сумм и т.п.

Обычно локальный контекст определяется строкой в формате: язык_зона.код. Здесь язык – обозначение языка (например, английский или немецкий), а зона – страна, географический регион или культура, в которой используется этот язык. В частности, квалификатор зона позволяет поддерживать национальные стандарты даже в том случае, если на одном языке говорят в разных странах. Квалификатор код определяет кодировку символов.

В табл. 2 приведены примеры типичных определений локальных контекстов [1, 2].

Следующий пример показывает пример использования локальности для указания использования требуемой таблицы кодировки при выводе текста. Для установки локальности используется функция `setlocale` [1, 2, 3, 4].

Код программы:
```
#include <iostream>
#include <clocale>
//необязательная строка в MVS
using namespace std;
int main()
{
setlocale(LC_CTYPE,"rus");
cout<<"АБВабв"<<endl;
return 0;
}
```
Выполнение программы:
```
АБВабв
```
Возможны несколько вариантов вызова `setlocale`, например:
```
setlocale(LC_CTYPE,"rus");
setlocale(LC_CTYPE,"Rus");
setlocale(LC_CTYPE,"russian");
setlocale(LC_CTYPE,"Russian");
setlocale(LC_CTYPE,".1251");
```
Во всех случаях происходит настройка функций обработки символов на кодовую страницу 1251.

Таблица 2 – Примеры определений локальных контекстов

язык_зона	Описание
de_De	немецкий язык (Германия)
de_AT	немецкий язык (Австрия)
de_CH	немецкий язык (Швейцария)
en_US	английский язык (США)
en_GB	английский язык (Великобритания)
en_AU	английский язык (Австралия)
en_CA	английский язык (Канада)
fr_FR	французский язык (Франция)
fr_CA	французский язык (Канада)

Приведём ещё несколько примеров использования функции `setlocale`. В нижеследующем примере вместо параметра `LC_CTYPE` поставлен `LC_ALL`.

Код программы:
```
#include <iostream>
using namespace std;
int main()
{
setlocale(LC_ALL,"rus");
cout<<"АБВабв"<<endl;
return 0;
}
```
Выполнение программы:
```
АБВабв
```
В следующей программе используется библиотека функций `cstring` стандартной библиотеки C++, наследующей библиотеку `string.h` из стандартной библиотеки C.

Код программы:
```
#include <iostream>
using namespace std;
int main()
{
setlocale(LC_CTYPE, "Russian");
char str[]="Текст русскими буквами";
cout<<str<<endl;
return 0;
}
```
Выполнение программы:
```
Текст русскими буквами
```
В случае переопределения значения строки библиотеки `cstring` можно воспользоваться указателем.

Код программы:
```
#include <iostream>
using namespace std;
int main()
{
char *str;
setlocale(LC_CTYPE, "");
str="Текст русскими буквами";
cout<<str<<endl;
str="АБВабв";
cout<<str<<endl;
return 0;
}
```
Выполнение программы:
```
Текст русскими буквами
АБВабв
```
В случае переопределения значения строки, обозначенной одним и тем же идентификатором, можно строку описать типом `string`.

Код программы:
```
#include <iostream>
#include <string>
using namespace std;
int main()
{
string str;
setlocale(LC_CTYPE, "");
str="Текст русскими буквами";
cout<<str<<endl;
str="АБВабв";
cout<<str<<endl;
return 0;
}
```
Выполнение программы:
```
Текст русскими буквами
```

АБВабв

Средства языка C++ для работы с локальными контекстами объявлены в заголовке clocale. Настройку программы на конкретный локальный контекст выполняет функция, прототип которой имеет вид: `char *setlocale(int category, const char *locale);`.

Аргумент category определяет раздел локальности, содержащий информацию, связанную по смыслу или назначению. Другими словами, аргумент category определяет категорию функций, на которые установка локальности с помощью функции setlocate оказывает влияние. Разделы локальности нумеруются. Номер раздела локальности называется локальной категорией.

В заголовочном файле clocale (ранее locale.h) определены макросы локальных категорий (табл. 3), имена которых используются в качестве аргументов функции setlocate [1, 2].

Аргумент locale является указателем на строку, задающую имя локального контекста.

Если locale указывает на пустую строку, то используется локальный контекст с кодовой таблицей, получаемой из операционной системы. Локальность "" обозначает местную локальность (native locale).

Локальность "C" используется по умолчанию.

При указании аргумента NULL действующий локальный контекст не изменяется. Если в параметре locale установить значение NULL, то функция вернёт указатель на строку, описывающую локальную категорию, заданную параметром category для текущей локальности.

В конкретной реализации могут быть также определены и другие локальности.

Перед передачей управления программе среда исполнения устанавливает локальность "C".

Таблица 3 – Макросы локальных категорий

Имя макроса локальной категории	Описание макроса
LC_ALL	действие на все локальные категории
LC_COLLATE	локальная категория, связанная с работой функций сравнения и преобразования строк
LC_CTYPE	локальная категория, связанная с работой функций классификации и обработки символов
LC_MONETARY	форматирование денежных значений
LC_NUMERIC	форматированием чисел с плавающей точкой
LC_TIME	форматирование времени

В случае успешного завершения функция возвращает указатель на строку, описывающую установленную локальность, а в случае неудачи – NULL.

Пример организации ввода и вывода русского текста в MVS с использованием функции setlocale. Следующая программа иллюстрирует организацию ввода и вывода русского текста в MVS с использованием функции setlocale.

Код программы:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
string name;
cout<<"Input your English name: ";
cin>>name;
cout<<"Hello, "<<name<<"!"<<endl;
setlocale(LC_CTYPE,"rus");
cout<<"Введите ваше русское имя: ";
cin>>name;
cout<<"Привет, ";
setlocale(LC_CTYPE,"C");
cout<<name<<"!"<<endl;
return 0;
}
```

Выполнение программы:

```
Input your English name: John
Hello, John!
Введите ваше русское имя: Женя
Привет, Женя!
```

Текст программы, в том числе и строки с кириллицей "Введите ваше русское имя: " и "Привет, "; набраны с кодовой страницей cp1251. Для правильного вывода этих строк в консоли задаётся кодировка строк ASCII-7 & cp1251. Переменная name вводится с кодировкой ASCII-7 & cp866. Поэтому для её вывода нужно возвратиться к кодовой странице cp866.

Вывод текста с использованием функций перекодировок CharToOemA и OemToCharA. Функция CharToOemA позволяет перевести строки из ANSI-кодировки (ASCII-7 & cp1251) в OEM-кодировку (ASCII-7 & cp866). Функция OemToCharA выполняет обратную перекодировку. Обе функции имеют по два параметра типа char. Первый параметр является входным и содержит исходную строку. Второй является выходным и содержит перекодированный вариант исходной строки. При использовании функций CharToOemA и OemToCharA необходимо подключать библиотеку Windows.h.

Код программы:

```
#include <iostream>
#include <Windows.h>
using namespace std;
int main()
{
char str[]="АБВабв";
cout<<str<<endl;
CharToOemA(str,str);
cout<<str<<endl;
return 0;
}
```

Выполнение программы:
```
└┴┬рст
АБВабв
```

Текст программы, в том числе и строка с кириллицей "АБВабв" набран с кодовой страницей ср1251. Для правильного вывода строки с кириллицей она перекодируется в символы с кодовой страницей ср866.

Использование функции перекодировки можно осуществить с применением указателей.

Код программы:
```
#include <iostream>
#include <Windows.h>
using namespace std;
int main()
{
char *str;
str="АБВабв";
cout<<str<<endl;
char *str1=new char[10];
CharToOemA(str,str1);
cout<<str1<<endl;
return 0;
}
```

Выполнение программы:
```
└┴┬рст
АБВабв
```

Организации ввода и вывода русского текста с помощью функций консоли. Функция `SetConsoleOutputCP()` устанавливает кодовую страницу вывода данных, используемую консолью. Синтаксис функции `SetConsoleOutputCP`: `bool SetConsoleOutput(uint кодовая страница);`. Параметр кодовая страница – идентификатор кодовой страницы, которая будет установлена. Функция `SetConsoleCP()` устанавливает кодовую страницу ввода, используемую консолью. Синтаксис функции `SetConsoleCP`: `bool SetConsoleCP(uint кодовая страница);`. Здесь кодовая страница – идентификатор кодовой страницы, которая будет установлена.

Заголовочные файлы функций включены в библиотеку `Windows.h`.

Указанные функции работают только при установленном в окне командной строки шрифте Lucida Console (либо каким-либо другим TrueType моноширинным (непропорциональным) шрифтом). Поэтому следует предварительно изменить текущий шрифт окна командной строки.

Чтобы выяснять текущую кодовую страницу ввода консоли, используют функцию `GetConsoleCP`. Для получения кодовой страницы вывода данных консоли используют функцию `GetConsoleOutputCP`. Упомянутые функции относятся к группе так называемых функций консоли. Список функций консоли намного шире перечисленных четырёх функций.

Примером программы, работающей с использованием функций `SetConsoleOutputCP` и `SetConsoleOutputCP` в Visual Studio с компилятором Visual C++2010 являются следующая программа:

Код программы:
```
#include <iostream>
#include <string>
#include <Windows.h>
using namespace std;
int main()
{
string name;
cout<<"Input your English name: ";
cin >> name;
cout<<"Hello, ";
cout<<name<<"!"<<endl;
SetConsoleOutputCP(1251);
cout<<"Введите ваше русское имя: ";
SetConsoleCP(1251);
cin >> name;
cout<<"Привет, ";
cout<<name<<"!"<<endl;
return 0;
}
```

Выполнение программы:
```
Input your English name: John
Hello, John!
Введите ваше русское имя: Женя
Привет, Женя!
```

Примеры использования функций `wcout.imbue()` и `wcin.imbue()` в программах ввода-вывода русского текста. Следующая программа является примером ввода-вывода широкосимвольных литералов [5, 6], содержащих текст, как латиницей, так и кириллицей.

Код программы:
```
#include <iostream>
#include <sstream>
using namespace std;
int main()
{
wstring name;
wcout<<L"Input your English name: ";
wcin>>name;
wcout<<L"Hello, "<<name<<L"!"<<endl;
wcout.imbue(locale("rus_rus.866"));
wcout<<L"Введите ваше русское имя: ";
wcin.imbue (locale("rus_rus.866"));
wcin>>name;
wcout<<L"Привет, "<<name<<L"!"<<endl;
return 0;
}
```

Выполнение программы:
```
Input your English name: John
Hello, John!
Введите ваше русское имя: Женя
Привет, Женя!
```

Текст программы, в том числе и строки с кириллицей "Введите ваше имя: " и "Привет, "; набраны с кодовой страницей ср1251. Для перекодировки «короткого» значения строк в ср866 с целью корректного вывода русского текста используется оператор

`wcout.imbue(locale("rus_rus.866"));`

Для отмены его используется оператор
`wcin.imbue (locale("rus_rus.866"));`

После этого значение переменной `name` вводится с кодировкой ср866 и выводится с этой кодировкой.

Выводы. Приведены различные средства настройки ввода-вывода русского текста в консольных приложениях Visual C++ 2010 в среде MVS 2010:

- с использованием функции `setlocale`;
- с использованием функций перекодировок `CharToOemA` и `OemToCharA`;
- с помощью функций `SetConsoleCP` и `SetConsoleOutputCP`;
- с помощью функций `wcout.imbue()` и `wcin.imbue()` в программах ввода-вывода широкосимвольных литералов.

Использование функции `CharToOemA` представляется наименее удобным, так как эту функцию следует вызывать при каждом выводе строки с русскими буквами.

Список литературы

1. *Josuttis N. M.* C++ Standard Library: A Tutorial and Reference / *N. M. Josuttis*. Reading: Addison Wesley Longman, Inc., 1999. – 640 p.
2. *Josuttis N. M.* C++ Standard Library: A Tutorial and Reference. Second Edition / *N. M. Josuttis*. Ann Arbor: Edwards Brothers Malloy, 2012. – 1161 p.
3. *Schildt H.* C++: Complete Reference. Third Edition / *H. Schildt*. Berkeley, New York, St. Louis, etc.: McGraw-Hill, 1998. – 1008 p.
4. *Stroustrup B.* The C++ Programming Language. Fourth Edition / *B. Stroustrup*. Ann Arbor: Edwards Brothers Malloy, 2013 – 1346 p.
5. *Solter N. A.* Professional C++ / *N. A. Solter, S. J. Kleper*. Indianapolis: Wiley Publishing, Inc., 2005. – 838 p.
6. *Gregoire M.* Professional C++. Second Edition / *M. Gregoire, N. A. Solter, S. J. Kleper*. Indianapolis: John Wiley & Sons, Inc., 2011. – 1072 p.

Bibliography (transliterated):

1. Josuttis N. M. *C++ Standard Library: A Tutorial and Reference*. Reading: Addison Wesley Longman, Inc., 1999, 640 p.
2. Josuttis N. M. *C++ Standard Library: A Tutorial and Reference. Second Edition*. Ann Arbor: Edwards Brothers Malloy, 2012, 1161 p.
3. Schildt H. *C++: Complete Reference. Third Edition*. Berkeley, New York, St. Louis, etc.: McGraw-Hill, 1998, 1008 p.
4. Stroustrup B. *The C++ Programming Language. Fourth Edition*. Ann Arbor: Edwards Brothers Malloy, 2013, 1346 p.
5. Solter N. A., Kleper S. J. *Professional C++*. Indianapolis: Wiley Publishing, Inc., 2005, 838 p.
6. Gregoire M., Solter N. A., Kleper S. J. *Professional C++. Second Edition*. Indianapolis: John Wiley & Sons, Inc., 2011, 1072 p.

Поступила (received) 05.09.2016

Бібліографічні описи / Библиографические описания / Bibliographic descriptions

Налаштування введення та виведення російського тексту в консольних додатках Visual C ++ 2010 / **С. В. Грінченко** // Вісник НТУ «ХПІ». Серія: Радіофізика та іоносфера. – Х. : НТУ «ХПІ», 2016. – № 34 (1206). – С. 34–40. – Бібліогр.: 6 назв. – ISSN 2078-9998.

Настройка ввода-вывода русского текста в консольных приложениях Visual C++ 2010 / **С. В. Гринченко** // Вестник НТУ «ХПИ». Серия: Радиофизика и ионосфера. – Х. : НТУ «ХПИ», 2016. – № 34 (1206). – С. 34–40. – Библиогр.: 6 назв. – ISSN 2078-9998.

Input and output adjustment of Russian text in Visual C ++ 2010 console applications / **S. V. Grinchenko** // Bulletin of NTU "KhPI". Series: Radiophysics and ionosphere. – Kharkiv : NTU "KhPI", 2016. – No. 34 (1206). – P. 34–40. – Bibliogr.: 6. – ISSN 2078-9998.

Відомості про автора / Сведения об авторе / About the Author

Грінченко Сергій Володимирович – Інститут іоносфери НАН та МОН України, м. Харків, науковий співробітник; тел.: (066) 963-18-85; e-mail: svgrinchenko@gmail.com.

Гринченко Сергей Владимирович – Институт ионосферы НАН и МОН Украины, г. Харьков, научный сотрудник; тел.: (066) 963-18-85; e-mail: svgrinchenko@gmail.com.

Grinchenko Sergii Volodymyrovych – Institute of ionosphere of NAS and MES of Ukraine, Kharkiv, researcher; tel.: (066) 963-18-85; e-mail: svgrinchenko@gmail.com.