УДК 004.432

Г.І. РАДЕЛЬЧУК
Хмельницький національний університет

МЕТОДИ ВИРІШЕННЯ ПРОБЛЕМ РОБОТИ З КИРИЛИЦЕЮ У КОНСОЛЬНИХ С-ПРОГРАМАХ

В статті проведено комплексне дослідження методів вирішення проблем, які виникають під час опрацювання україномовного тексту у консольних С-програмах. Встановлено, що причиною цих проблем є різні кодові сторінки, які використовуються у середовищах розробки і виконання програм. Розглянуто два підходи до вирішення цих проблем – забезпечення перетворення кодування символів до середовища виконання або зміна кодової сторінки у консольному вікні. На основі проведеного дослідження сформовано рекомендації щодо використання запропонованих засобів для різних версій Microsoft Visual Studio.

***Ключові слова**: консоль, консольна програма, мова програмування С, середовище розробки, кодова сторінка, локалізація, перекодування.*

G.I. RADELCHUK
Khmelnytsky National University

METHODS OF SOLUTION TO THE PROBLEMS OF WORK WITH CYRILLICS IN CONSOLE C-PROGRAMS

The purpose of the article is to study, analyze and systematize methods of solution to the problems with the Ukrainian-language text in console C-programs. The article provides a comprehensive study of methods for solution to the problems that occur during input, output and processing of the Ukrainian-language text in console C-programs, oriented on the development of Microsoft Visual Studio environment in different versions. It is found that the causes of these problems are different code pages used in the development environment (Microsoft Visual Studio) and the program execution environment (console) – 1251 and 866, respectively. Two approaches to the solution to these problems are considered – ensuring the conversion of character encoding to the execution environment, or the change of the code page in the runtime environment (console window). With the first approach, you can save the C-program file in the 866 encoding, which is not expedient. In the second approach, the possibilities of the programming language C and the Windows operating system are analyzed resulting in the suggestion of the means for ensuring the correct elaboration of the Ukrainian-language text: the standard function of the language C – setlocale(); functions WinAPI – SetConsoleCP() and SetConsoleOutputCP(); the system chcp command and the system() function. The mechanisms of the influence of these functions on the input/output streams are described, their advantages and disadvantages are identified, analyzed and experimentally confirmed. The presentation of the material is accompanied by fragments of program codes. On the basis of the study, recommendations for the use of the proposed tools in console C-programs for various versions of the Microsoft Visual Studio development environment were generated.

***Keywords**: console, console program, programming language C, development environment, code page, localization, transcoding.*

Постановка проблеми

Найкращий спосіб почати навчатись програмуванню – це складання консольних програм. Тому на першому курсі при вивченні дисципліни ”Основи програмування” студенти спеціальності ”Інженерія програмного забезпечення” зазвичай створюють саме консольні С-програми, використовуючи середовище програмування Microsoft Visual Studio (MVS) різних безкоштовних версій (2010, 2012, 2013, 2015 або 2017).

Структура консольного проекту максимально спрощена, оскільки немає графічного режиму; окрім цього, консольний режим використовується у випадках, коли основними вимогами до програми є мінімізація часу обчислень та витрат оперативної пам’яті. На підготовку таких програм потрібно менше часу, тому консольний режим зручно використовувати для швидкої перевірки та налагодження окремих алгоритмів.

Консольне вікно ОС Windows є інтерфейсом, який надає операційна система для введення/виведення даних додаткам, які працюють у текстовому режимі, тобто таким, що не мають власного графічного інтерфейсу. С-програми, які створюються на початковому етапі навчання, виконуються у консольному вікні. Вони використовують стандартні функції введення і виведення: **printf()**, **scanf()**, **getchar()** та інші.

Компілюючи у Windows консольні С-програми, користувач (програміст) постійно стикається з проблемою кодувань, а саме: символи кирилиці, на відміну від латиниці, обробляються і виводяться некоректно. Це пов’язано з тим, що в середовищі розробки (MVS) і в середовищі виконання програм (консольне вікно) використовуються дві різні кодові сторінки (*cp – Code Page*) з різними кодами для символів кирилиці. У зв’язку з цим консольні програми повинні вводити/виводити повідомлення лише англійською мовою, що дуже незручно, особливо для студентів-першокурсників, які лише починають вивчати програмування.

Аналіз останніх досліджень та публікацій

На форумах програмістів, кіберфорумах постійно з'являються запитання типу "у мене виводяться кракозябри", "замість тексту – ієрогліфи", "як виводити українською", “не сортуються українські слова” тощо, причому переважно ці питання пов'язані з мовою С/С++. На цих же форумах можна знайти багато порад, але значна частина цих порад далеко не завжди допомагає у вирішенні проблеми.

Хоча кількість книг, пов’язаних з тематикою мови С, обчислюється тисячами найменувань, число джерел, де в якійсь мірі висвітлювалося б вирішення національних особливостей С, дуже обмежена. В публікаціях деяких авторів ([1, 2]) розглядаються питання русифікації консольних С++-додатків, але лише

частково; до того ж, деякі методи, запропоновані авторами, не працюють у консольних С-програмах.

Методи вирішення цих проблем нестандартизовані, тому розв'язання однієї і тієї ж задачі в різних середовищах програмування (і, навіть, в одному середовищі різних версій) вимагає різних підходів.

Таким чином, необхідний комплексний підхід до зазначеної проблеми в цілому – яка інформація і як буде вводитись, як вона буде зберігатися в оперативній пам'яті комп'ютера і оброблятися, як вона буде передаватися, виводитися, яке середовище програмування використовується і які засоби для роботи з різними кодуваннями воно надає тощо. Марно намагатися українізувати введення/виведення "цього рядка, цих даних" без розуміння відповідних механізмів, а також без розуміння того, що буде з цими даними відбуватися надалі.

Отже, **метою статті є:** дослідження, аналіз і систематизація методів вирішення проблем роботи з україномовним текстом у консольних С-програмах з орієнтацією на середовище розробки Microsoft Visual Studio (різних версій), а також надання відповідних рекомендацій тим, хто вивчає програмування мовою С.

Виклад основного матеріалу

При написанні консольних С-програм рядки, що містять символи кирилиці, можуть зустрічатися в двох різних кодуваннях у наступних випадках: у вихідних текстах програм (у вигляді символьних і текстових літералів); при виведенні на консоль; при введенні з консолі; при виведенні у файл; при введенні з файла.

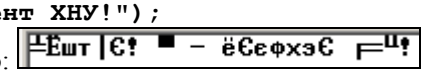
Основне правило при роботі з національними алфавітами: усі рядки повинні бути в єдиному кодуванні. При недотриманні цього правила неможливо порівнювати та сортувати рядки (і символи), а також, як було вже зазначено, введення/виведення рядків на консоль або у файл буде некоректним.

Хоча вже давно розроблений єдиний стандарт кодування – Unicode, ОС Windows і досі використовує кодову сторінку Windows-1251. І при створенні С-проекту у MVS цей стандарт кодування успадковується проектом (а, отже, і програмою). У свою чергу, консольні програми виконуються у консольних вікнах, де ні Unicode, ні UTF-8, ні мультибайтові рядки безпосередньо не підтримуються, а використовується лише кодова сторінка ср866 (ASCII). Таким чином, утворюється "ланцюжок", у якому фігурують дві різні кодові сторінки:

середовище розробки –> текст програми – > програма – > середовище виконання –> екран (консоль)

У ср1251 і ср866 для символів кирилиці виділені різні діапазони кодів, отже, програма, створена із застосуванням ср1251, буде виводити замість символів кирилиці символи ср866, що мають ті ж коди, що і символи кирилиці у ср1251. Так, наприклад, в результаті виконання оператора:

```
printf("Привіт! Я - студент ХНУ!");
```

у консольному вікні буде виведено: 

Таким чином, програма передає коди символів стандарту ср1251; командний рядок приймає ці коди і переводить їх у символи, але вже за стандартом ср866 (тобто здійснюється динамічне перекодування), бо іншого стандарту не знає. У результаті символи кирилиці, передані у консоль, інтерпретуються невірно.

Для того, щоб консольна С-програма "розуміла" кирилицю, потрібно забезпечити перетворення кодування символів до середовища виконання або "змусити" командний рядок Windows повертати текст у кодуванні ср1251 замість ср866.

При першому підході для забезпечення перетворення кодування символів до середовища виконання можна зберегти файл С-програми у кодуванні ср866 (у вікні збереження файла слід вибрати кодову сторінку 866). Однак, цей спосіб далеко не завжди є прийнятним, оскільки вирішує проблеми роботи з кирилицею лише локально, тобто для окремої програми. Так, наприклад, при записі введеного з клавіатури тексту у файл, текст у файлі також буде збережено у ср866. При використанні ж цього файла іншою С-програмою виникають ті ж самі проблеми, якщо кодування для цієї програми не змінено на ср866 (а ймовірність такої зміни є досить малою). Тому очевидно, що використання цього способу "повернення у минуле" навряд чи є доцільним.

Іншим підходом до узгодження середовища розробки С-програм і середовища їх виконання є забезпечення введення, виведення та оброблення текстових даних у "рідному" для MVS кодуванні – ср1251.

Розглянемо спершу засоби реалізації цього підходу, наявні у мові С.

Для врахування особливостей, пов'язаних з конкретною країною (регіоном), у мові С використовуються спеціальні середовища, які називаються ***локальними контекстами*** (*locale* – місце дії) [1]. Вони включають набір параметрів та функцій, що забезпечують підтримку національних стандартів.

Локальний контекст визначається рядком формату: **мова [_зона [.код]]**. Тут **мова** – позначення мови (наприклад, англійська, німецька, українська), а **зона** – країна, в якій використовується мова. Цей кваліфікатор дозволяє підтримувати національні стандарти для різних країн, що використовують одну мову [3]. Кваліфікатор **код** визначає кодову сторінку.

Приклади рядків, що визначають локальні контексти для нашого регіону, наведені у таблиці 1.

Налаштування програми на конкретний локальний контекст у мові С виконує функція **setlocale()** [4], яка змінює поведінку стандартних функцій С для роботи з рядками у відповідності до локалі та категорії:

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

Таблиця 1

Приклади рядків, що визначають локальні контексти

Позначення локалі	Пояснення
Ukr (або ukr) Ukrainian (або ukrainian) Ukrainian_Ukraine	Українська мова (Україна). Кодова сторінка за замовчуванням.
Ukrainian_Ukraine.1251	Українська мова (Україна). Кодова сторінка 1251.
Ukrainian_Ukraine.866	Українська мова (Україна). Кодова сторінка 866.
Rus (або rus) Russian (або russian) Russian_Russia	Російська мова (Росія). Кодова сторінка за замовчуванням.
Russian_Russia.1251	Російська мова (Росія). Кодова сторінка 1251.
Russian_Russia.866	Російська мова (Росія). Кодова сторінка 866.

Аргумент **category** визначає категорію функцій, на які **setlocale()** має вплив. Він може приймати шість різних значень, але найчастіше використовуються два значення: **LC_ALL** – всі категорії; **LC_CTYPE** – функції обробки символів. Аргумент **locale** є вказівником на рядок, що задає ім'я локального контексту. Якщо **locale** вказує на порожній рядок, використовується кодова сторінка, яку одержують із операційної системи. Якщо **locale** дорівнює нулю (NULL), діючий локальний контекст не змінюється.

Можливі різні варіанти виклику функції **setlocale()**, наприклад:

setlocale (LC_ALL, "Ukrainian") – налаштування всіх функцій на Україну;

setlocale (LC_CTYPE, "Ukr") – налаштування функцій обробки символів на Україну;

setlocale (LC_CTYPE, ".1251") – налаштування функцій обробки символів на кодову сторінку 1251;

setlocale(LC_ALL, "") – установка локального контексту з налаштуваннями ОС Windows (ср1251).

Якщо країною і мовою за замовчуванням операційної системи є, наприклад, "Україна" і "українська", наступні два виклики **setlocale()** є функціонально еквівалентними:

`setlocale( LC_ALL, ".1251" );` **`setlocale( LC_ALL, "Ukrainian_Ukraine.1251").`**

Перший аргумент функції можна замінити на **0** або **NULL**: **setlocale(0, "")**; **setlocale(NULL, "Ukr")**.

Замість номера кодової сторінки можна вживати **".OCP**" або **".ACP**" (для використання кодових сторінок 866 та 1251 відповідно):

`setlocale( LC_ALL, ".OCP" )` – явне задання мовного стандарту згідно поточної кодової сторінки ср866, отриманої від операційної системи;

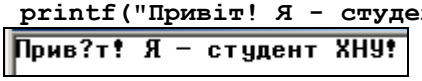
`setlocale( LC_ALL, ".ACP" )` – явне задання мовного стандарту згідно поточної кодової сторінки ср1251, отриманої від операційної системи.

Набір доступних імен мовного стандарту, назв мов, кодів країн та кодових сторінок підтримується Windows API (*Application Programming Interface)*, окрім кодових сторінок, які потребують два і більше байт на один символ (таких, наприклад, як UTF-7 чи UTF-8). Двобайтовою версією функції **setlocale()** є функція **_wsetlocale()**, аргументами та повернеными значеннями якої є рядки двобайтових символів.

Функція **locale()** визначена у стандартному заголовному файлі **locale.h**, який необхідно підключити до С-програми (**#include <locale.h>**). В останніх версіях MVS (2015, 2017) ця функція визначена у файлі **stdlib.h**.

З врахуванням цих відомостей в результаті виконання операторів:

`setlocale (LC_ALL, "Ukr"); printf("Привіт! Я - студент ХНУ!");`

у консольному вікні буде виведено: 

Як бачимо, некоректно виводиться лише український символ "і" (замість нього виводиться символ "?"). Цьому можна запобігти, використовуючи в тексті замість українського "і" англійське "i". Решта символів українського алфавіту (в т. ч. "є", "ї") відображаються коректно.

Але найголовнішим є те, що один раз застосувавши функцію **setlocale()**, весь подальший текст, заданий у С-програмі, у консольному вікні буде виведений коректно.

Особливістю функції **setlocale()** є те, що вона працює ***лише з потоком виведення***, тобто при налаштуванні локалі на ср1251 текст, введений у консольному вікні з клавіатури (наприклад, на запит функції **scanf()** або **gets()**), буде виводитись некоректно (рис. 1).

Як бачимо, текст з потоку виведення (тобто заданий у програмі рядковим літералом) виведений коректно. Коректно відобразились і символи кирилиці у процесі їх набору з клавіатури. Але введений рядок виведено некоректно. Це пов'язано з тим, що функція **setlocale()** не має впливу на потік введення, тобто при введені рядків з клавіатури використовується ср866. Отже, для коректного виведення символів кирилиці, введених у консольному вікні, потрібно знову змінити локальний контекст, але вже на ср866 викликом функції **setlocale (LC_ALL, ".866")**, тим самим повернувши початкові налаштування локалі (рис. 2).

```
char str[30];
// зміна локального контексту на Україну (ср1251)
setlocale(LC_ALL,"Ukr");
printf("Введіть рядок символів:\n");
/* введення рядка з клавіатури в консолі:
   Привіт! Я - студент ХНУ! */
gets(str);  // читання введеного рядка
// виведення введеного рядка
printf("\n%s\n", str);
```

```
Введіть рядок символів:
Привіт! Я - студент ХНУ!

?aЁўів! ? - бвг*?-в ?"!
```

а) **б)**

Рис. 1. Некоректне виведення символів кирилиці з потоку введення: а) фрагмент програмного коду; б) результат

```
char str[30];
// зміна локального контексту на Україну (ср1251)
setlocale(LC_ALL,"Ukr");
printf("Введіть рядок символів:\n");
/* введення рядка з клавіатури в консолі:
   Привіт! Я - студент ХНУ! */
gets(str);  // читання введеного рядка
// повернення початкового налаштування локалі (ср866)
setlocale(LC_ALL, ".OCP");
// виведення введеного рядка
printf("\n%s\n\n", str);
```

```
Введіть рядок символів:
Привіт! Я - студент ХНУ!

Привіт! Я - студент ХНУ!
```

а) **б)**

Рис. 2. Коректне виведення символів кирилиці з потоку введення: а) фрагмент програмного коду; б) результат

Таким чином, для виведенні кирилиці, заданої безпосередньо у програмі у вигляді рядкових літералів, слід за допомогою функції **setlocale()** змінити локальний контекст на ср1251, а перед виведенням кирилиці, введеної з клавіатури, необхідно повернути початкові налаштування локалі, тобто встановити ср866. Якщо ж безпосередньо у програмі немає рядків кирилиці, але програма вимагає введення їх з клавіатури, то це не потребує зміни локального контексту, оскільки введення/виведення здійснюється у ср866.

Однак, цей спосіб стає незручним, якщо програма працює з багатьма потоками введення/виведення (потрібно відповідним чином періодично міняти налаштування локалі).

Окрім цього, цей спосіб не вирішує проблем роботи з кирилицею під час спільного оброблення символів та рядків, заданих як програмно, так і введених з клавіатури. Так, наприклад, представлений на рис. 3, а) вірний (синтаксично і логічно) фрагмент програми пошуку у введеному з клавіатури рядку програмно заданого символу дає невірний результат, причому, виведення результату є не зовсім зрозумілим (рис. 3, б)). Переналаштування локалі не дає бажаних результатів.

Звісно, можна окремо виводити рядки, що задаються програмно, і які вводяться з клавіатури (з відповідними переналаштуваннями локалі). Однак, проблема полягає в тому, що у будь-якому випадку програма працює невірно. Як неважко помітити, символ, який шукається (**‘д’**), у введеному рядку є, проте, програма “каже”, що немає. З урахуванням викладеного вище причина цього є зрозумілою: код символу ‘д’ відповідає ср1251 (оскільки він заданий програмно у вигляді символьного літерала), а символи введеного рядка відповідають ср866 (оскільки вони введені з клавіатури у консольному вікні).

Таким чином, застосування функції **setlocale()** вирішує проблеми введення/виведення символів кирилиці, але не завжди вирішує проблем їх програмної обробки. Для узгодження середовища розробки та консолі слід встановити єдиний стандарт кодування – ср1251 – як для потоку введення, так і для потоку виведення, що за допомогою лише функції **setlocale()** реалізувати неможливо.

Подальший аналіз показав, що, окрім локалізації, можливість якої закладена у стандарті мови С, існують нестандартні засоби підтримки роботи з кирилицею у консольних С-програмах – функції Win32 API для управління консольним вікном.

Win32 API (далі – WinAPI) – це набір функцій, що працюють під управлінням ОС Windows [5].

Функції WinAPI можуть бути додані до С-проекту включенням заголовного файла **windows.h** (**#include <windows.h>**). **Windows.h** – це специфічний заголовний файл Windows для мов програмування C/C++, який містить декларації (описи) всіх функцій WinAPI, всі загальні макроси, що використовуються програмістами Windows, і всі типи даних, що використовуються різними функціями та підсистемами [6]. Окрім цього, **windows.h** містить директиви включення значної кількості інших (дочірніх) заголовних файлів, надаючи непрямий доступ до їх вмісту (напряму ці файли не можуть бути включеними у С-програму, оскільки не є самодостатніми). Зокрема, у **windows.h** включений заголовний файл **wincon.h**, який містить

структури даних, типи даних та описи функцій, призначених виключно для роботи в консолі.

```
char str[30], ch = 'д';
setlocale(LC_ALL,".1251");
printf("Введіть рядок символів:\n");
/* введення рядка з клавіатури в консолі:
   Привіт! Я - студент ХНУ! */
gets(str);
setlocale(LC_ALL, ".866");
//Пошук заданого символа у рядку
if (strchr(str,ch) != NULL)
   printf ("\nУ рядку '%s' символ '%c' є.\n", str, ch);
     else
       printf ("\nУ рядку '%s' символа '%c' немає.\n", str, ch);
```

а)

```
Введіть рядок символів:
Привіт! Я - студент ХНУ!

ц Ё фъє 'Привіт! Я - студент ХНУ!' ёшьтюыр 'ф' эхьр||.

Введіть рядок символів:
Привіт! Я - студент ХНУ!

У рядку '?аЁўів! ? - бвг*?-в ?"!' символа 'д' немає.
```

б)

Рис. 3. Некоректне виведення рядків кирилиці: а) фрагмент програмного коду; б) результати

Файл **wincon.h** місить прототип функції **SetConsoleCP()** [7]:

BOOL WINAPI SetConsoleCP (UINT wCodePageID);

Ця функція встановлює кодову сторінку, яка використовується консоллю, асоційовану із запущеним процесом під час введення символів з клавіатури (тобто в ***потік введення***). Параметр **wCodePageID** є ідентифікатором встановлюваної кодової сторінки (866, 1251).

Установка ср1251 в потік виведення за допомогою функції **setlocale()** і установка ср1251 в потік введення за допомогою функції **SetConsoleCP(1251)** (рис. 4, а)) дає можливість вірно опрацьовувати символи і рядки кирилиці. Однак, експериментально встановлено, що при цьому виведення тексту знову є некоректним (рис. 4, б)). Це пояснюється особливістю роботи функції **setlocale()**: ця функція не змінює кодову сторінку в потоці виведення, якщо така сторінка в консолі вже встановлена. Оскільки ср1251 в консолі встановлена функцією **SetConsoleCP(1251)**, то функція **setlocale()** її не змінює. А так як ср1251 встановлена лише на потік введення, то виведення здійснюється в ср866. Єдиний спосіб вирішити цю проблему – після завершення потоку введення повернути в консолі ср866 за допомогою функції **SetConsoleCP(866)**, щоб в потоці виведення функція **setlocale()** її змінила на ср1251 (рис. 5, а), що й забезпечує коректне виведення результатів (рис. 5, б).

```
char str[30], ch = 'д';
// Установка ср1251 в потік виведення
setlocale(LC_ALL,".1251");
printf("Введіть рядок символів:\n");
// Установка ср1251 в потік введення
SetConsoleCP(1251);
gets(str);
// Пошук заданого символа у рядку
if (strchr(str,ch) != NULL)
printf ("\nУ рядку '%s' символ '%c' є.
\n", str, ch);
  else
    printf ("\nУ рядку '%s' символа '%c'
немає.\n", str, ch);
```

а)

```
Введіть рядок символів:
дорога

ц Ё фъє 'фюЁюур' ёшьтюы 'ф' ||.

Введіть рядок символів:
мама

ц Ё фъє 'ьрьр' ёшьтюыр 'ф' эхьр||.
```

б)

Рис. 4. Некоректне виведення рядків кирилиці: а) фрагмент програмного коду; б) результати

```
char str[30], ch = 'д';
// Установка ср1251 в потік виведення
setlocale(LC_ALL,".1251");
printf("Введіть рядок символів:\n");
// Установка ср1251 в потік введення
SetConsoleCP(1251);
gets(str);
// Повернення консолі у ср866
SetConsoleCP(866);
// Пошук заданого символа у рядку
if (strchr(str,ch) != NULL)
printf ("\nУ рядку '%s' символ '%c' є.
\n", str, ch);
  else
     printf ("\nУ рядку '%s' символа '%c'
немає.\n", str, ch);
```

а)

```
Введіть рядок символів:
дорога

У рядку 'дорога' символ 'д' є.
```

```
Введіть рядок символів:
мама

У рядку 'мама' символа 'д' немає.
```

б)

Рис. 5. Коректне опрацювання та виведення рядків кирилиці: а) фрагмент програмного коду; б) результати

Таким чином, метод спільного використання функцій **setlocale()** та **SetConsoleCP()**, хоча й вирішує практично всі проблеми роботи з кирилицею, однак також є незручним, якщо програма працює з багатьма потоками введення/виведення. Окрім того, для використання кожної з цих функцій до програми має бути підключена своя окрема бібліотека, що не є оптимальним.

У заголовному файлі **windows.h** (точніше, **wincon.h**) визначена функція, яка змінює кодову сторінку в ***потоці виведення*** – **SetConsoleOutputCP()** [8]:

BOOL WINAPI SetConsoleOutputCP(UINT wCodePageID);

Параметр **wCodePageID** є ідентифікатором встановлюваної кодової сторінки.

Спільне використання функцій **SetConsoleCP(1251)** і **SetConsoleOutputCP(1251)** (тобто одноразове встановлення ср1251 як в потік введення, так і в потік виведення) виключає необхідність використання функції **setlocale()** та заголовного файла **locale.h** і, як наслідок, не потребує періодичних переналаштувань консолі.

Однак, за допомогою функції **SetConsoleOutputCP(1251)** символи кирилиці виводяться коректно лише за умови, якщо у консольному вікні встановлений векторний шрифт (шрифт TrueType).

Проведений аналіз різних версій ОС Windows показав, що консольне вікно може відкриватись як з точковими (наприклад, у Windows XP, Windows 7), так і з векторним (наприклад, у Windows 10) шрифтами, встановленими за замовчуванням. Для MVS останніх версій (2015 та 2017), які можуть бути встановлені в операційному середовищі Windows 10, консольне вікно відкривається зі шрифтом TrueType – **Consolas**. Для більш ранніх версій MVS (2010, 2012, 2013), які можуть бути встановлені у середовищі Windows XP чи Windows 7, консольне вікно відкривається з точковими шрифтами, що й призводить до некоректного виведення україномовного тексту. Виходом з цієї ситуації є ручне встановлення в консолі шрифту TrueType під час виконання С-програми (у вікні властивостей вікна) – зазвичай шрифту **Lucida Console**. Шрифт можна змінити або лише для поточного вікна (тоді цю процедуру треба буде повторювати при кожному виконанні програми) або зберегти властивості для інших вікон (тоді процедуру зі зміни поточного шрифта повторювати не прийдеться).

Отже, при спільному використанні функцій **SetConsoleCP(1251)** і **SetConsoleOutputCP(1251)** перекодування символів у процесі їх введення/виведення здійснюються автоматично. Недоліком цього способу можна вважати ручне встановлення TrueType-шрифту в консолі для MVS 2010, 2012, 2013 (які функціонують у середовищі ОС Windows XP або 7); при цьому у самій консолі відображення кирилиці при введенні здійснюється коректно навіть при використанні растрових шрифтів. Також окремо слід зауважити, що включення заголовного файла **windows.h** у С-програму, враховуючи об'ємність інтерфейсу WinAPI, суттєво збільшує час компіляції навіть невеликих програм (хоча цей недолік можна усунути за допомогою механізму прекомпіляції).

І, нарешті, ще одну можливість вирішення проблем роботи з кирилицею у С-програмах надає системна утиліта ОС Windows – **chcp.com**, яка застосовується для перегляду або зміни поточної кодової сторінки у вікні командного рядка Windows (у консольному вікні) [9]. Команда "**chcp** номер_кодової_сторінки" запускається із С-програми за допомогою функції **system()** [10], яка визначена у заголовних файлах **windows.h**, **stdlib.h** і **process.h** (один з них слід підключити до С-програми):

int system(const char *str);

Функція **system()** передає рядок, що адресується параметром **str** (у нашому випадку це, як було вже зазначено, "**chcp** номер_кодової_сторінки") як команду для командного процесора операційної системи.

Отже, щоб змінити кодову сторінку в консолі на ср1251, достатньо включити у С-програму функцію:

system("chcp 1251");

Однак, слід пам'ятати, що за допомогою цієї функції, як і функції **SetConsoleOutputCP(1251)**, символи кирилиці виводяться коректно лише за умови, якщо у консольному вікні встановлений шрифт TrueType.

Висновки

Проблема української мови у консольних С-програмах полягає в тому, що консоль і редактор коду MVS підтримують різні кодові сторінки (cp866 і cp1251 відповідно), у яких для символів кирилиці виділені різні діапазони кодів. З метою вирішення цієї проблеми для коректного опрацювання кирилиці в консолі необхідно забезпечити перетворення кодування символів до середовища виконання або поміняти кодову сторінку в консолі (щоб вона відповідала кодовій сторінці редактора MVS).

При першому підході для забезпечення перетворення кодування символів до середовища виконання можна зберегти файл С-програми у кодуванні cp866 (що не є доцільним). У рамках другого підходу проаналізовані та запропоновані наступні засоби: стандартна функція мови С – **setlocale()**; функції WinAPI – **SetConsoleCP()** та **SetConsoleOutputCP()**; системна команда **chcp** та функція **system()**.

На основі проведеного дослідження можна зробити висновок, що для MVS версій 2015 та 2017 (встановлених у середовищі ОС Windows 10) оптимальним варіантом забезпечення коректної роботи з кирилицею є використання в С-програмах функції **system("chcp 1251")**, яка визначена у стандартному заголовному файлі **stdlib.h** (що не вимагає підключення до С-програм об'ємного WinAPI). Вибір засобів забезпечення коректної роботи з кирилицею (з врахуванням їх переваг та недоліків) для MVS нижчих версій залишається за програмістом, в залежності від версії та налаштувань ОС Windows і типу вирішуваних задач. Зокрема, якщо програма обробляє текстові дані тільки в потоці виведення, то достатньо використовувати лише функцію **setlocale()**. Загальним же способом є конвертування вхідних даних та їх оброблення у cp1251.

Література

1. Тарасов. В. Л. Локализация консольных приложений в языке C++ / В. Л. Тарасов // Вестник Нижегородского университета имени Н. И. Лобачевского. – 2011. – № 3(2). – С. 301–307.

2. Гринченко С. В. Настройка ввода-вывода русского текста в консольных приложениях Visual C++ 2010 / С. В. Гринченко // Вісник НТУ «ХПІ». Серія: Радіофізика та іоносфера. – 2016. – № 34 (1206). – С. 34–40.

3. Microsoft Locale ID Values // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/library/ms912047(v=winembedded.10).aspx. – Title from the screen.

4. Setlocale, _wsetlocale // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/library/x99tb11d(v=vs.140).aspx. – Title from the screen.

5. Windows API // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/library/cc433218(VS.85).aspx. – Title from the screen.

6. Windows.h // WikipediA. The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Windows.h. – Title from the screen.

7. SetConsoleCP function // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/vstudio/ms686013 (v=vs.110). – Title from the screen.

8. SetConsoleOutputCP function // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/vstudio/ms686036 (v=vs.110). – Title from the screen.

9. Chcp [Электронный ресурс] // Библиотека TechNet Microsoft. – Режим доступа : https://technet.microsoft.com/library/bb490874.aspx. – Название с экрана.

10. System, _wsystem // Microsoft Developer Network. URL: https://msdn.microsoft.com/ru-ru/library/windows/desktop/277bwbdz(v=vs.120).aspx. – Title from the screen.

References

1. Tarasov V. L. Localization of console applications in C++ language / V. L. Tarasov // Bulletin of the Nizhny Novgorod University named after N. I. Lobachevsky. – 2011. – No. 3(2). – P. 301-307.

2. Grinchenko S. V. 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.

3. Microsoft Locale ID Values // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/library/ms912047(v=winembedded.10).aspx. – Title from the screen.

4. Setlocale, _wsetlocale // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/library/x99tb11d(v=vs.140).aspx. – Title from the screen.

5. Windows API // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/library/cc433218(VS.85).aspx. – Title from the screen.

6. Windows.h // WikipediA. The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Windows.h. – Title from the screen.

7. SetConsoleCP function // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/vstudio/ms686013 (v=vs.110). – Title from the screen.

8. SetConsoleOutputCP function // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/vstudio/ms686036 (v=vs.110). – Title from the screen.

9. Chcp // Library TechNet Microsoft. URL: https://technet.microsoft.com/library/bb490874.aspx. – Title from the screen.

10. System, _wsystem // Microsoft Developer Network. URL: https://msdn.microsoft.com/ru-ru/library/windows/desktop/277bwbdz(v=vs.120).aspx. – Title from the screen.

Рецензія/Peer review : 05.11.2017 р. Надрукована/Printed :24.01.2018 р.
Рецензент: д-р фіз.-мат. наук, проф. Бедратюк Л. П.