# Локалізація

# Список основних джерел

# 1. <https://docs.microsoft.com/en-us/cpp/standard-library/codecvt-class?view=msvc-170>

# 2. <http://cppstudio.com/cat/309/318/>

# 3. <https://docs.microsoft.com/en-us/cpp/standard-library/codecvt-class?view=msvc-170>

# Вступ

Стандарт C++ складається із двох основних частин: ядра мови й стандартної бібліотеки. Стандартна бібліотека C++ увібрала в себе бібліотеку шаблонів STL, що розроблялася одночасно зі стандартом. Зараз назва STL офіційно не вживається, однак у колах програмістів на C++ ця назва використовується для позначення частини стандартної бібліотеки, що містить визначення шаблонів контейнерів, ітераторів, алгоритмів і функторів. Стандарт C++ містить нормативне посилання на стандарт С і не визначає самостійно ті функції стандартної бібліотеки, які запозичаються зі стандартної бібліотеки С. Крім того, існує величезна кількість бібліотек C++, що не входять у стандарт.

У програмах на C++ можна використовувати багато бібліотек С. Стандартизація визначила мову програмування C++, однак за цією назвою можуть ховатися також неповні, обмежені, достандартні варіанти мови. На початку мова розвивалася поза формальними рамками, спонтанно, у міру завдань, що ставилися перед нею. Розвитку мови сприяв розвиток кросскомпілятора cfront. Нововведення в мові відбивалися в зміні номера версії кросс-компілятора.

Мова C++ багато в чому є надмножиною С. Нові можливості C++ включають оголошення у вигляді виразів, перетворення типів у вигляді функцій, оператори new і delete, тип bool, посилання, розширене понятя константності функції, що підставляються, аргументи за замовчуванням, перевизначення, простори імен, класи (включаючи й усі, пов’язані із класами, можливості, такі як спадкування, функції-члени, віртуальні функції, абстрактні конструктори), перевизначення операторів, шаблони, оператор ::, обробку виключень, динамічну ідентифікацію й багато чого іншого. Мова C++ також у багатьох випадках суворіше ставиться до перевірки типів, ніж С. В C++ з’явилися коментарі у вигляді подвійної косої риски (//), які були в попереднику С — мові BCPL. Деякі особливості C++ пізніше були перенесені в С, наприклад, ключові слова const і inline, оголошення в циклах for і коментарі в стилі C++. У більш пізніх реалізаціях С також були представлені можливості, яких немає в C++, наприклад, макроси vararg і поліпшена робота з масивамипараметрами.

Стандартна бібліотека C++ включає стандартну бібліотеку С з невеликими змінами, які роблять її більш підходящою для мови C++. Інша більша частина бібліотеки C++ заснована на Стандартній Бібліотеці Шаблонів (STL). Вона надає такі важливі інструменти, як контейнери (наприклад, вектори й списки) та ітератори (узагальнені покажчики), що надають доступ до цих контейнерів як до масивів. Крім того, STL дозволяє подібним чином працювати й з іншими типами контейнерів, наприклад, асоціативними списками, стеками, чергами. Використовуючи шаблони, можна писати узагальнені алгоритми, здатні працювати з будь-якими контейнерами або послідовностями, обумовленими ітераторами. Так само, як і в С, можливості бібліотек активізуються використанням директиви #include для включення стандартних файлів. Усього в стандарті C++ визначено 50 таких файлів.

C++ — надзвичайно потужна мова, що містить засоби створення ефективних програм практично будь-якого призначення, від низькорівневих утиліт і драйверів до складних програмних комплексів будь-якого призначення.

Класи і об’єкти в С є основними концепціями об’єктно-орієнтованого програмування — ООП. Об’єктно-орієнтоване програмування — розширення структурного програмування, в якому основними концепціями є поняття класів і об’єктів. Основна відмінність мови програмування С++ від С полягає в тому, що в С немає класів, а отже мова С не підтримує ООП, на відміну від С++. **Класи в С++**— це абстракція, що описує методи, властивості, ще не існуючих об’єктів.

**Об’єкти** — конкретне уявлення абстракції, що має свої властивості і методи. Створені об’єкти на основі одного класу називаються екземплярами цього класу. Ці об’єкти можуть мати різну поведінку, властивості, але все одно будуть об’єктами одного класу.

# Локалізація

Для врахування особливостей, повязаних з країною та мовою, використовують спеціальне середовище — локальні контексти локалі (locale— місце дії), які налічують набір параметрів і функцій, що забезпечують підтримку національних та культурних стандартів.

Локалізація

* <locale> визначає класи і оголошує функції, які зберігають і управляють інформацією, що властива [локалі](https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C).
* <codecvt> забезпечує код для перетворення даних у відповідності до різних кодувань.

Мова C підтримує локалізацію певних параметрів, таких як формати дати, часу та грошових одиниць для певної країни.

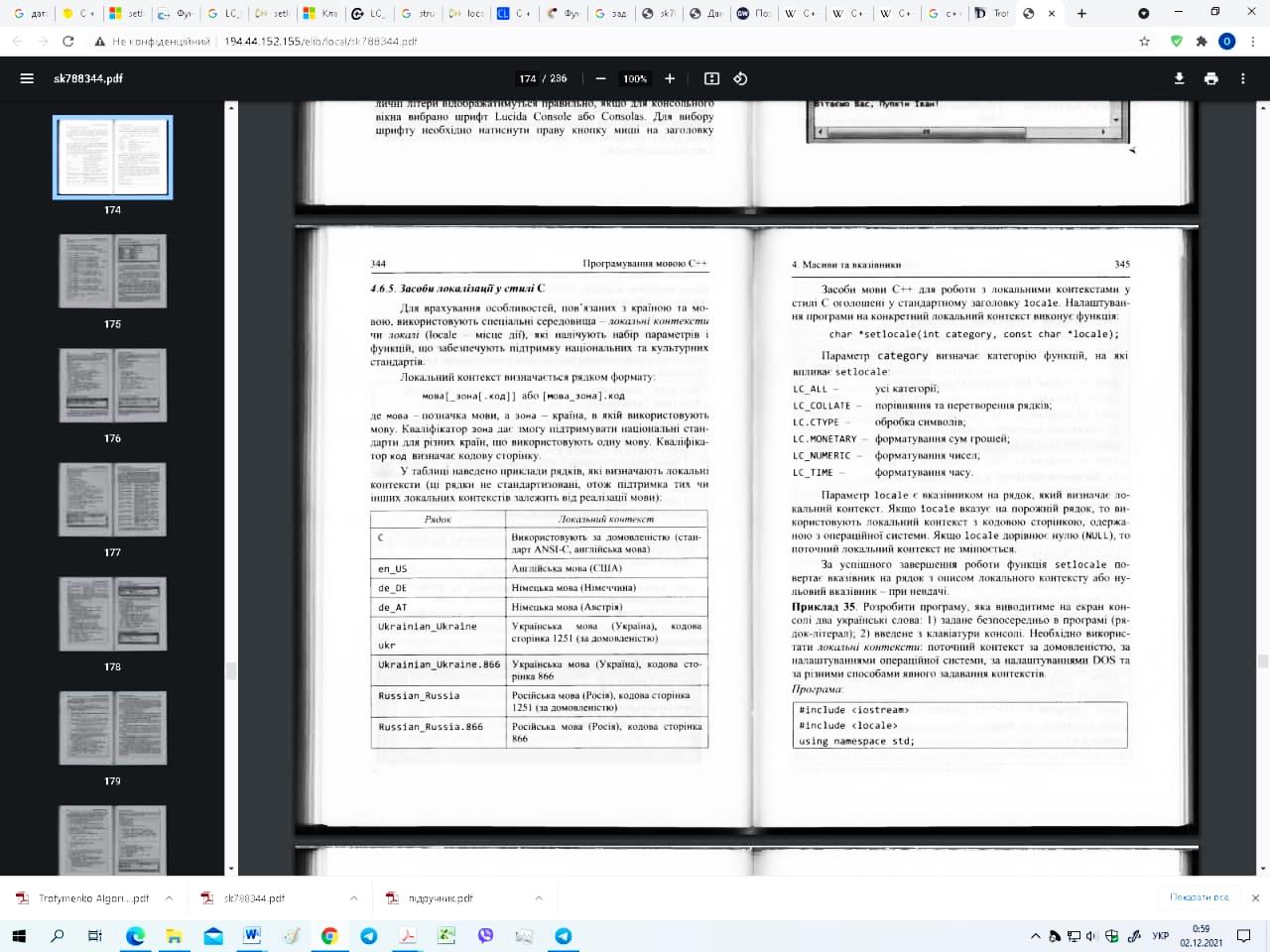
Кожна система і конкретний компілятор можуть надавати різні параметри локалі, які можна вибрати (за допомогою функції), але принаймні дві локалі доступні на вибір для будь-якої програми C: setlocale, C локалі містить мінімум інформації про локалізацію програми. Це нейтральна локалі, яка має однакові параметри на всіх системах і компіляторах. Тому можна легко спрогнозувати точний результат програми. Ця мова використовується за замовчуванням для всіх програм C.

Локальний контекст визначається рядком формату:

мова[\_зона[.код]] або [мова\_зона].код

де мова — позначка мови, а зона — країна, в якій використовують мову. Класифікатор зона дає змогу підтримувати національні стандарти для різних країн, що використовують мову. Класифікатор код визначає кодову сторінку.

У таблиці наведені приклади рядків, які визначають локальні контексти (ці рядки не стандартизовані, тому підтримка тих чи інших локальних контекстів залежить від реалізації мови):



Засоби мови С++ для роботи з локальними контекстами у стилі С оголошені у стандартному заголовку locale. Налаштування програми на конкретний локальний контекст виконує функція:

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

Параметр category визначає категорію функцій, на які впливає setlocale:

LC\_ALL — усі категорії;

LC\_COLLATE — порівняння та перетворення рядків;

LC.STYPE — обробка символів;

LC.MONETARY — форматування сум грошей;

LC\_NUMERIC — форматування чисел;

LC\_TIME — форматування часу.

Наприклад, виведення дати в поточному форматі:

#include <cstdio>

#include <clocale>

#include <ctime>

#include <cwchar>

int main()

{

std::setlocale(LC\_ALL, "en\_US.UTF-8");

std::setlocale(LC\_NUMERIC, "de\_DE.UTF-8");

std::setlocale(LC\_TIME, "ja\_JP.UTF-8");

wchar\_t str[100];

std::time\_t t = std::time(nullptr);

std::wcsftime(str, 100, L"%A %c", std::localtime(&t));

}

Наприклад, я використав функція strftime, яка має такі параметри:

%а Скорочене найменування дня тижня

%А Повне найменування дня тижня

%b Скорочена назва місяця

%В Повна назва місяця

%с Стандартний рядок дати та часу

%d День місяця у десятковому поданні (1—31)

%Н Година дня в межах (0-23)

%I Година дня в межах (1-12)

%j День у році в десятковому представленні (1—366)

%m Місяць у десятковому представленні (1-12)

%М Хвилина в десятковому представленні (0-59)

%р Локальний еквівалент для AM або РМ

%S Секунди в десятковому представленні (0-61)

%U Тиждень у році, неділя є першим днем ​​(0—52)

%w День тижня в десятковому представленні (0-6, відповідає 0)

%W Тиждень у році, понеділок є першим днем ​​(0—53)

%х Стандартний рядок дати

%Х Стандартний рядок часу

%у Рік у десятковому представленні в межах століття (00-99)

%Y Рік, включаючи століття, у десятковому представленні

%Z Назва часового поясу

%% Символ відсотка

Наприклад:

#include <stdio.h>

#include <time.h>

int main ()

{

time\_t rawtime;

struct tm \* timeinfo;

char buffer [80];

time (&rawtime);

timeinfo = localtime (&rawtime);

strftime (buffer,80,"Now it's %I:%M%p.",timeinfo);

puts (buffer);

return 0;

}

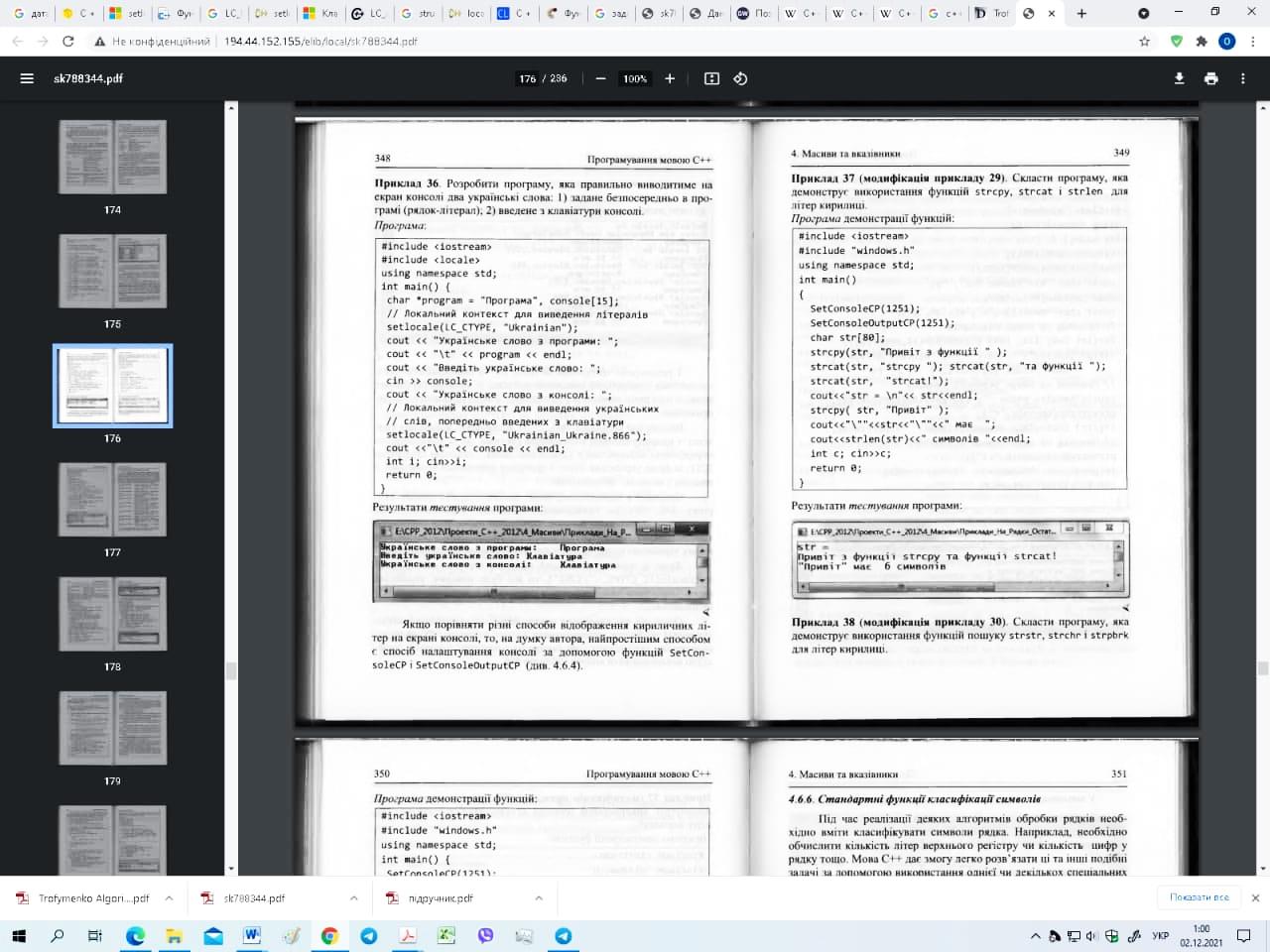
Параметр locale є вказівним на рядок, який визначає локальний контекст. Якщо locale вказує на порожній рядок, то використовують локальний контекст з кодовою сторінкою, одержаною з операційної системи. Якщо locale дорівнює нулю (NULL), то поточний локальний контекст не змінюється.

За успішного завершення роботи функція setlocale повертає вказівник на рядок з описом локального контексту або нульовий вказівник при невдачі.

***Приклад.*** Виклик setlocale (LC\_ALL, “ “) встановлює локальний контекст з налаштуваннями операційної системи, де за домовленістю передбачено використання української мови з кодовою сторінкою 1251, за якою українське слово з програми виводиться правильно, а введене з консолі — неправильно. Якщо, наприклад, в консольному вікні використовують кодову сторінку 866, то для правильного виведення українських слів, попередньо введені з консолі, необхідно змінювати локальний контекст через виклик setlocale(LC\_CTYPE, “.866”), однак при цьому українські слова, задані у програмі, виводяться неправильно.

Якщо ж змінити локальний контекст через виклик функції setlocale(LC\_CTYPE, “.1251”), то все буде навпаки — з консолі неправильно.

Отже, під час виведення українських слів з програми (рядків-літералів) необхідно встановлювати кодову сторінку 1251, а перед виведенням українських слів, попередньо введених з консолі, необхідно встановлювати кодову таблицю 866.



# Клас locale

Клас, що описує об’єкт мовного стандарту, інкапсулює дані щодо відповідної культури як набору аспектів, які збірно визначають те чи інше локалізоване середовище.

**Синтаксис** class locale;

Аспект — це покажчик на об’єкт класу, виведеного похідним від класу [facet](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#facet_class), який має загальний об’єкт у форматі:

static locale::id id;

Можна встановити відкритий набір даних аспектів. Можна також створити об’єкт мовного стандарту, який визначає довільну кількість аспектів.

Зумовлені групи цих аспектів представляють [категорії мовного стандарту](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#category) , якими у стандартній бібліотеці C зазвичай керує функція. setlocale.

|  |  |
| --- | --- |
| *Категорія collate(LC\_COLLATE) включає аспекти:* | collate<char>  collate<wchar\_t> |
| *Категорія ctype(LC\_CTYPE) включає аспекти:* | ctype<char>  ctype<wchar\_t>  codecvt<char, char, mbstate\_t>  codecvt<wchar\_t, char, mbstate\_t>  codecvt<char16\_t, char, mbstate\_t>  codecvt<char32\_t, char, mbstate\_t> |
| *Категорія monetary(LC\_MONETARY) включає аспекти:* | moneypunct<char, false>  moneypunct<wchar\_t, false>  moneypunct<char, true>  moneypunct<wchar\_t, true>  money\_get<char, istreambuf\_iterator<char>>  money\_get<wchar\_t, istreambuf\_iterator<wchar\_t>>  money\_put<char, ostreambuf\_iterator<char>>  money\_put<wchar\_t, ostreambuf\_iterator<wchar\_t>> |
| *Категорія numeric(LC\_NUMERIC) включає аспекти:* | num\_get<char, istreambuf\_iterator<char>>  num\_get<wchar\_t, istreambuf\_iterator<wchar\_t>>  num\_put<char, ostreambuf\_iterator<char>>  num\_put<wchar\_t, ostreambuf\_iterator<wchar\_t>>  numpunct<char>  numpunct<wchar\_t> |
| *Категорія time(LC\_TIME) включає аспекти:* | time\_get<char, istreambuf\_iterator<char>>  time\_get<wchar\_t, istreambuf\_iterator<wchar\_t>>  time\_put<char, ostreambuf\_iterator<char>>  time\_put<wchar\_t, ostreambuf\_iterator<wchar\_t>> |
| *Категорія messages(LC\_MESSAGES) включає аспекти:*  Необхідна для POSIX, але не для стандарту C | messages<char>  messages<wchar\_t> |

Деякі з цих зумовлених аспектів використовуються iostream класами для управління перетворенням числових значень текстові послідовності та з них.

Об’єкт класу ***locale*** також зберігає ім’я мовного стандарту як об’єкт класу [string](https://docs.microsoft.com/ru-ru/cpp/standard-library/string-typedefs?view=msvc-170#string). Використання неприпустимого імені мовного стандарту для створення аспекту мовного стандарту або об’єкта мовного стандарту створює об’єкт класу [runtime\_error](https://docs.microsoft.com/ru-ru/cpp/standard-library/runtime-error-class?view=msvc-170). Ім’я збереженого мовного стандарту має значення, "\*"Якщо об’єкт мовного стандарту не може бути впевнений у тому, що мова C у стилі точно відповідає тому, який представлений об’єктом. В іншому випадку можна встановити відповідний мовний стандарт у стандартній бібліотеці C для деякого об’єкта мовного стандарту locale\_object, викликавши setlocale (LC\_ALL, locale\_object. [name](https://docs.microsoft.com/en-us/cpp/standard-library/locale-class?view=msvc-170" \l "name) ().c\_str()).

У цьому випадку можна також викликати статичну member function:

static locale empty();

створити об’єкт мовного стандарту без аспектів. Це також прозорий мовний стандарт. Якщо функції шаблону [has\_facet](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-functions?view=msvc-170#has_facet) і [use\_facet](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-functions?view=msvc-170#use_facet) не можуть знайти потрібний аспект у прозорому мовному стандарті, вони звертаються до першого глобального мовного стандарту, а потім, якщо вони є прозорими, класичний мовний стандарт. Можна написати:

cout.imbue(locale::empty());

Подальші операції вставки [cout](https://docs.microsoft.com/ru-ru/cpp/standard-library/iostream?view=msvc-170#cout) виправляються поточним станом глобального мовного стандарту. Можна навіть написати:

locale loc(locale::empty(),

locale::classic(),

locale::numeric);

cout.imbue(loc);

Числові правила форматування для наступних вставок [cout](https://docs.microsoft.com/ru-ru/cpp/standard-library/iostream?view=msvc-170#cout) залишаються такими ж, як у мовному стандарті C*,* навіть якщо глобальний мовний стандарт надає інші правила вставки дат та грошових сум.

| **Конструктор** | **Опис** |
| --- | --- |
| [locale](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#locale) | Створює мовний стандарт, копію мовного стандарту або копію мовного стандарту, в якій аспект чи категорія замінені аспектом або категорією іншого мовного стандарту. |

| **Ім’я типу** | Опис |
| --- | --- |
| [category](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#category) | Цілочисленний тип, що містить значення бітової маски для позначення стандартних сімейств аспектів. |

**Member function**

| Функція-член | Опис |
| --- | --- |
| [combine](https://docs.microsoft.com/en-us/cpp/standard-library/locale-class?view=msvc-170#combine) | Вставляє аспект з певного мовного стандарту в цільовий мовний стандарт. |
| [name](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#name) | Повертає ім’я збереженого мовного стандарту. |

**Static Functions (Статичні функції)**

| Ім’я | Опис |
| --- | --- |
| [classic](https://docs.microsoft.com/en-us/cpp/standard-library/locale-class?view=msvc-170#classic) | Ця статична функція-член повертає об’єкт мовного стандарту, що представляє класичний мовний стандарт C. |
| [global](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#global) | Скидання стандартного мовного програмного забезпечення. |

**Оператори**

| Оператор | Опис |
| --- | --- |
| [operator=](https://docs.microsoft.com/en-us/cpp/standard-library/locale-class?view=msvc-170#op_eq) | Призначає мовний стандарт. |
| [operator! =](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#op_neq) | Перевірка двох мовних стандартів на нерівність. |
| [[operator](https://docs.microsoft.com/en-us/cpp/standard-library/locale-class?view=msvc-170#op_eq)()](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#op_call) | Порівнює два об’єкти basic\_string. |
| [[operator==](https://docs.microsoft.com/en-us/cpp/standard-library/locale-class?view=msvc-170#op_eq_eq)](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#op_eq_eq) | Перевірка двох мовних стандартів на рівність. |

**Класи**

| Клас | Опис |
| --- | --- |
| [facet](https://docs.microsoft.com/en-us/cpp/standard-library/locale-class?view=msvc-170#facet_class) | Клас, який використовується як базовий клас для всіх аспектів мовного стандарту. |
| [id](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#id_class) | Клас члена містить унікальний ідентифікатор аспекту, що використовується як індекс для пошуку аспектів у мовному стандарті. |

**Вимоги**

**Заголовок:** <locale>

**Простір імен:** std

## locale::category

Цілочисленний тип, що містить значення bitmask для позначення стандартних сімейств аспектів.

* typedef int category;
* static const int collate = LC\_COLLATE;
* static const int ctype = LC\_CTYPE;
* static const int monetary = LC\_MONETARY;
* static const int numeric = LC\_NUMERIC;
* static const int time = LC\_TIME;
* static const int messages = LC\_MESSAGES;
* static const int all = LC\_ALL;
* static const int none = 0.

Наприклад:

#include <iostream>

#include <ctime>

#include <locale>

using namespace std;

int main (){

time\_t t = time(NULL);

char buffer [80];

cout<<"Local is: "<<setlocale(LC\_ALL,"C")<<endl;

//monetary formatting will be US

setlocale(LC\_MONETARY, "en\_US.UTF-8");

struct lconv \*lc = localeconv();

cout<<"Local Currency Symbol: "

<<lc->currency\_symbol<<endl;

//time formatting will be environment's

//default locale

setlocale(LC\_TIME, "");

strftime (buffer,80,"%c",localtime(&t));

cout<<"Date is: "<<buffer<< endl;

return 0;

}

Тип є синонімом типу **int**, який може представляти групу різних елементів типу bitmask, для класу мовного стандарту або може використовуватися для представлення будь-якої з відповідних категорій стандарту C. Елементи:

* collate, що відповідає категорії C LC\_COLLATE
* ctype, що відповідає категорії C LC\_CTYPE
* monetary, що відповідає категорії C LC\_MONETARY
* numeric, що відповідає категорії C LC\_NUMERIC
* time, що відповідає категорії C LC\_TIME
* messages, що відповідає категорії POSIX LC\_MESSAGES

Два більш корисні значення:

* none, що відповідає одній із категорій C
* all, що відповідає об’єднанню C всіх категорій LC\_ALL.

Можна представити довільну групу категорій з допомогою OR з цими константами, як в monetary | time.

**locale:: classic**

Ця статична функція-член повертає об’єкт мовного стандарту, що представляє класичний мовний стандарт C.

static const locale& classic();

Значення, що повертається: Посилання на мовний стандарт C.

Класична мова C — це мовний стандарт ASCII американської англійської мови у стандартній бібліотеці C. Це мовний стандарт, який неявно використовується у програмах, які не є міжнародними.

## locale::combine

Вставляє аспект з певного мовного стандарту в цільовий мовний стандарт.

template <class Facet>

locale combine(const locale& source\_locale) const;

Параметри

source\_locale (мовний стандарт, що містить аспект для вставки в цільовий мовний стандарт).

Значення, що повертається: Функція-член повертає об’єкт мовного стандарту, який замінює або додає в **\*** цей аспект Facet, вказаному в source\_locale.

## facet Class

Клас, який використовується як базовий клас для всіх аспектів мовного стандарту.

class facet {

protected:

explicit facet(size\_t references = 0);

virtual ~facet();

private:

facet(const facet&) // not defined

void operator=(const facet&) // not defined

};

Не можна скопіювати або призначити об’єкт класу facet. Можна створювати та видаляти об’єкти, похідні від класу locale::facet, але не об’єкти правильного базового класу Зазвичай, створюється об’єкт \_Myfac, похідний від facet, при створенні locale, як у locale loc(locale::classic(), new \_Myfac);

У таких випадках, конструктор базового класу facet повинен мати аргумент з нульовим числом посилань. Якщо об’єкт більше не потрібен, він видаляється, тому аргумент ненульових посилань вказується лише в окремих випадках.

**locale:: Global**

Скидання стандартного мовного стандарту для програми. Цей виклик впливає на глобальний мовний стандарт C і C++.

static locale global(const locale& new\_default\_locale);

Параметри

new\_default\_locale — мовний стандарт, який використовуватиметься програмою за промовчанням.

Значення, що повертається: Мовний стандарт до скидання стандарту мовного стандарту.

При запуску програми глобальний мовний стандарт збігається з класичним мовним стандартом. Функція global() викликає setlocale( LC\_ALL, loc.name. c\_str()) встановлення відповідного мовного стандарту в стандартній бібліотеці C.

## id Class (Клас id)

The member class містить унікальний ідентифікатор аспекту, що використовується як індекс для пошуку аспектів у мовному стандарті.

***Приклад***

class id

{

protected: id();

private: id(const id&)

void operator=(const id&) // not defined

};

The member class описує статичний member object, необхідний для кожного унікального аспекту мовного стандарту. Не можна скопіювати або призначити об’єкт класу id.

## locale::locale

Створює мовний стандарт чи копію мовного стандарту або копію мовного стандарту, в якій аспект чи категорія замінені аспектом або категорією іншого мовного стандарту. Також містить деструктор.

locale();

explicit locale(const char\* locale\_name, category new\_category = all);

explicit locale(const string& locale\_name);

locale(const locale& from\_locale);

locale(const locale& from\_locale, const locale& Other, category new\_category);

locale(const locale& from\_locale, const char\* locale\_name, category new\_category);

template <class Facet>

locale(const locale& from\_locale, const Facet\* new\_facet);

~locale();

Параметри

* locale\_name (ім’я стандарту мови);
* from\_locale (мовний стандарт, який копіюватиметься під час створення нового мовного стандарту);
* Інший (мовний стандарт, з якого вибиратиметься категорія);
* new\_category (категорія для заміни у створеному мовному стандарті;
* new\_facet (аспект для заміни у створеному мовному стандарті).

Перший конструктор ініціалізує об’єкт відповідно глобальному мовному стандарту. Другий та третій конструктори ініціалізують усі категорії мовного стандарту, щоб їхня поведінка відповідала імені локалі locale\_name. Інші конструктори копіюють from\_locale, крім зазначених нижче:

* locale(const locale& from\_locale, const locale& Other, category new\_category);
* замінює інші аспекти, що відповідають категорії С, для якої С & new\_category не дорівнює нулю.
* locale(const locale& from\_locale, const char\* locale\_name, category new\_category);
* locale(const locale& from\_locale, const string& locale\_name, category new\_category);
* замінює з locale(locale\_name, all) ці аспекти, що відповідають категорії locale(locale\_name, all), для яких replace\_category & new\_category ненульове значення;
* template<class Facet> locale(const locale& from\_locale, Facet\* new\_facet);
* замінює в (або додає) from\_locale аспект new\_facet, якщо new\_facet не є порожнім покажчиком.

Якщо ім’я локалі locale\_name є порожнім покажчиком або неприпустимим, функція створює виняток [runtime\_error](https://docs.microsoft.com/ru-ru/cpp/standard-library/runtime-error-class?view=msvc-170).

## locale::name

Повертає ім’я збереженого мовного стандарту.

string name() const;

Значення, що повертається: Рядок, що визначає ім’я мовного стандарту.

## locale::operator =

Призначає мовний стандарт.

const locale& operator=(const locale& other) noexcept;

**locale::operator! =**

Перевірка двох мовних стандартів на нерівність.

bool operator!=(const locale& right) const;

Параметри

right (Один із мовних стандартів для перевірки на нерівність).

Значення, що повертається: Логічне значення, що дорівнює, **true,** якщо мовні стандарти не є копіями одного і того ж мовного стандарту. **false,** якщо мовні стандарти є копіями одного і того ж мовного стандарту.

Два мовні стандарти рівні, якщо вони збігаються, якщо одна з них є копією іншого або якщо вони мають однакові імена.

**locale::operator ()**

Порівнює два basic\_string об’єкти відповідно до правил порівняння лексикографічним порядком, визначеними std::collate<charT>аспектом локалі.

template <class CharType, class Traits, class Allocator>

bool operator()(

const basic\_string<CharType, Traits, Allocator>& left,

const basic\_string<CharType, Traits, Allocator>& right) const;

Параметри

* left (перший порівнюваний рядок);
* right (другий порівнюваний рядок).

Значення, що повертається: **true,** якщо left має лексикографічно значення менше ніж right, інакше — значення **false**.

The member function фактично виконує:

const collate<CharType>& fac = use\_fac<collate<CharType>>(\*this);

return (fac.compare(left.begin(), left.end(), right.begin(), right.end()) < 0);

Це означає, що об’єкт мовного стандарту можна використовувати як об’єкт функції.

**locale::operator = =**

Перевірка двох мовних стандартів на рівність.

bool operator==(const locale& right) const;

Параметри

right (один із мовних стандартів буде перевірись на рівність).

Значення, що повертається: Логічне значення, що дорівнює, **true,** якщо мовні стандарти є копіями одного і того ж мовного стандарту; **false,** якщо мовні стандарти не є копіями одного і того ж мовного стандарту.

Два мовні стандарти рівні, якщо вони збігаються, якщо один з них є копією іншого або якщо вони мають однакові імена.

***Клас codecvt***

Шаблон класу, який описує об’єкт, що може служити в якості аспекта локалі. Шаблон класу, що описує об’єкт, який може бути як аспект мовного стандарту. Він може керувати перетвореннями між послідовністю значень, що використовуються для кодування символів у програмі, і послідовністю значень, що використовуються для кодування символів за межами програми.

Синтаксис:

template <class CharType, class Byte, class StateType>

class codecvt : public locale::facet, codecvt\_base;

Параметри

* *CharType*  (тип, який використовується всередині програми для кодування символів);
* *Byte*  (тип, який використовується для кодування символів поза програмою);
* *StateType* (тип, який можна використовувати для подання проміжних станів перетворення між внутрішніми та зовнішніми типами представлень символів).

Шаблон класу описує об’єкт, який може служити як [аспект мовного стандарту](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#facet_class) для управління перетвореннями послідовності значень типу *CharType* і послідовності значень типу *Byte*. Клас *StateType* характеризує перетворення, а об’єкт класу *StateType* зберігає всі необхідні відомості про стан під час перетворення.

Внутрішнє кодування використовує подання з фіксованою кількістю байтів на символ, зазвичай або тип **char**, або тип **wchar\_t**.

Як і будь-якого іншого аспекту мовного стандарту, початкове збережене значення статичного id об’єкта дорівнює нулю. Перша спроба отримати доступ до збереженого значення збереже унікальне додатнє значення id.

Версії шаблону [do\_in](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_in) і [do\_out](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_out) завжди повертають codecvt\_base: :noconv.

Стандартна бібліотека C++ визначає кілька явних спеціалізацій:

template<>

codecvt<wchar\_t, char, mbstate\_t>

виконує перетворення між послідовностями **wchar\_t** та **char**.

template<>

codecvt<char16\_t, char, mbstate\_t>

виконує перетворення між послідовностями **char16\_t** в кодуванні UTF-16 і послідовностями **char** в кодуванні UTF-8.

template<>

codecvt<char32\_t, char, mbstate\_t>

виконує перетворення між послідовностями **char32\_t** в кодуванні UTF-32 (UCS-4) і послідовностями **char** в кодуванні UTF-8.

| **Конструктор** | **Опис** |
| --- | --- |
| [codecvt](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#codecvt) | Конструктор для об’єктів класу codecvt, який є аспектом мовного стандарту для обробки перетворень. |

| Ім’я типу | Опис |
| --- | --- |
| [extern\_type](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#extern_type) | Тип символу, який використовується для зовнішніх представлень. |
| [intern\_type](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#intern_type) | Тип символу, який використовується для внутрішніх представлень. |
| [state\_type](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#state_type) | Тип символу, який використовується для представлення проміжних станів під час перетворень між внутрішніми та зовнішніми. |

**Функції елементів**

| Функція-член | Опис |
| --- | --- |
| [always\_noconv](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#always_noconv) | Перевіряє, чи не потрібно виконати перетворення. |
| [do\_always\_noconv](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_always_noconv) | Віртуальна функція, що викликається перевіркою того, чи не потрібно виконати перетворення. |
| [do\_encoding](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_encoding) | Віртуальна функція, яка перевіряє, чи залежить кодування потоку від стану, незалежно від того, що Byte співвідношення між Byte значеннями, що використовуються, і CharType отриманими значеннями є константою, і, якщо це так, визначає значення цього співвідношення. |
| [do\_in](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_in) | Віртуальна функція, що викликається для перетворення послідовності внутрішніх Byte значень на послідовність зовнішніх CharType значень. |
| [do\_length](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_length) | Віртуальна функція, що визначає Byte, скільки значень заданої послідовності зовнішніх Byte значень створює не більше заданої кількості внутрішніх CharType значень і повертає цю кількість Byte значень. |
| [do\_max\_length](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_max_length) | Віртуальна функція, що повертає максимальну кількість зовнішніх байтів, необхідних для створення одного внутрішнього CharType. |
| [do\_out](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_out) | Віртуальна функція, що викликається для перетворення послідовності внутрішніх CharType значень на послідовність зовнішніх байтів. |
| [do\_unshift](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_unshift) | Віртуальна функція, що викликається для надання Byte значень, необхідних в перетворені, яке залежить від стану, для завершення останнього символу в послідовності Byte значень. |
| [encoding](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#encoding) | Перевіряє, чи залежить кодування Byte потоку від стану, незалежно від того, яке відношення між Byte значеннями, що використовуються, і CharType, які значення є константами, і, якщо це так, визначає значення цього відношення. |
| [in](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#in) | Перетворює зовнішнє представлення послідовності Byte значень у внутрішнє представлення послідовності CharType значень. |
| [length](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#length) | Визначає, скільки Byte значень із заданої послідовності зовнішніх Byte значень створює не більше заданої кількості внутрішніх CharType значень і повертає цю кількість Byte значень. |
| [max\_length](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#max_length) | Повертає максимальну кількість зовнішніх Byte значень, що необхідні для створення одного внутрішнього об’єкта CharType. |
| [out](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#out) | Перетворює послідовність внутрішніх CharType значень у послідовність зовнішніх Byte значень. |
| [unshift](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#unshift) | Надає зовнішні Byte значення, необхідні для перетворення, яке залежить від стану, для завершення останнього символу в послідовності Byte значень. |

Вимоги!

**Заголовок:**<locale>

**Простір імен:** std

**codecvt::always\_noconv**

Перевіряє, чи потрібно виконувати перетворення.

bool always\_noconv() const throw();

Значення, що повертається: Логічне значення, що дорівнює, **true,** якщо не потрібно виконувати перетворення; **false,** якщо потрібно виконати хоча б одну операцію.

Функція члена повертає значення [do\_always\_noconv](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_always_noconv).

**codecvt::codecvt**

Конструктор для об’єктів класу codecvt, який служить як аспект мовного стандарту обробки перетворень.

explicit codecvt(size\_t refs = 0);

Параметри

*refs* цілочисленне значення, яке використовується для вказівки типу управління пам’яттю для об’єкта.

Можливі значення для *refs* параметра та їх значущість:

* 0: час існування об’єкта управляється мовними стандартами, що його містять 4;
* 1: час існування об’єкта має керуватися вручну;
* 2: ці значення не визначено.

Конструктор ініціалізує свій locale::facet базовий об’єкт за допомогою [locale::facet](https://docs.microsoft.com/ru-ru/cpp/standard-library/locale-class?view=msvc-170#facet_class) (refs)

**codecvt::do\_always\_noconv**

Віртуальна функція, що викликається для перевірки того, чи потрібно виконувати перетворення.

virtual bool do\_always\_noconv() const throw();

Значення, що повертається: Захищена віртуальна функція — член повертає **true** тільки, якщо кожен виклик методу [do\_in](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_in) або [do\_out](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_out) повертає значення nocon v.

Версія шаблону завжди повертає значення **true**.

**codecvt::do\_encoding**

Віртуальна функція, яка перевіряє, чи залежить кодування потоку від стану, незалежно від того, що Byte співвідношення між Byte значеннями, що використовуються, і отриманими CharType значеннями є константою і, якщо це так, визначає значення цього співвідношення.

virtual int do\_encoding() const throw();

Значення, що повертається: захищена функція-член повертає:

* –1, якщо кодування послідовностей типу extern\_type залежить від стану;
* 0, якщо кодування включає послідовності змінної довжини;
* *N*, якщо кодування включає лише послідовності довжини *N.*

**codecvt::do\_in**

Віртуальна функція, що викликається для перетворення послідовності зовнішніх Byte значень на послідовність внутрішніх CharType значень.

virtual result do\_in(

StateType& state,

const Byte\* first1,

const Byte\* last1,

const Byte\*& next1,

CharType\* first2,

CharType\* last2,

CharType\*& next2,) const;

Параметри

* *State* стан перетворення, який підтримується між викликами функції-члена;
* *first1 п*окажчик на початок послідовності перетворення;
* *last1* покажчик на кінець послідовності перетворення;
* *next1* вказівник за межами перетвореної послідовності для першого неперетвореного символу;
* *first2* покажчик початку перетвореної послідовності;
* *last2* покажчик на кінець перетвореної послідовності;
* *next2* покажчик на об’єкт CharType, який надходить після останнього перетворення CharType, до першого незміненого символу послідовності;

Значення, що повертається: Значення, що повертається, вказує успіх, частковий успіх або збій операції. Функція повертає:

* codecvt\_base: :error значення, якщо вихідну послідовність сформовано неправильно.
* codecvt\_base: :nocon v, якщо функція не виконує перетворення;
* codecvt\_base ::ok значення, якщо перетворення виконано;
* codecvt\_base: :partial, якщо джерело недостатнє або місця призначення недостатньо, для успішності перетворення.

Обєкт *state* має представляти початковий стан перетворення на початку нової вихідної послідовності. Функція змінює це збережене значення, щоб відобразити поточний стан успішного перетворення. В іншому випадку збережене значення буде незаданим.

**codecvt::do\_length**

Віртуальна функція, що визначає Byte, скільки значень заданої послідовності зовнішніх Byte значень створює не більше заданої кількості внутрішніх CharType значень і повертає цю кількість Byte значень.

virtual int do\_length(

const StateType& state,

const Byte\* first1,

const Byte\* last1,

size\_t len2) const;

Параметри:

* *state* стан перетворення, який підтримується між викликами функції-члена;
* *first1* покажчик початку зовнішньої послідовності;
* *last1* покажчик на кінець зовнішньої послідовності;
* *len2* максимальна кількість Byte значень, яке може бути повернене функцією члена.

Значення, що повертається:Ціле число, що представляє число перетворень, що не перевищують значення len2, що визначається зовнішньою вихідною послідовністю [ first1, last1].

Захищена віртуальна функція-член фактично викликає do\_in ( state, first 1, last 1, next 1, buf, buf + len2, next 2) (копію стану), деякий буфер buf, покажчики next 1та next 2.

Потім він повертає next 2 - buf. Він підраховує максимальну кількість перетворень, що не перевищують len2, які визначаються вихідною послідовністю [first1, last1].

Версія шаблону завжди повертає менше з *last1* - *first1*і *len2*.

**codecvt::do\_max\_length**

Віртуальна функція, що повертає максимальну кількість зовнішніх Byte значень, необхідних для отримання одного внутрішнього об’єкта CharType.

virtual int do\_max\_length() const throw();

Значення, що повертається: Максимальна кількість значень Byte, яке необхідне для його створення CharType.

Захищена віртуальна функція з членом повертає максимальне допустиме значення, яке може бути повернуто [do\_length](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_length) ( first 1, last 1, 1) для довільних допустимих значень *first 1* та *last 1*.

**codecvt::do\_out**

Віртуальна функція, що викликається для перетворення послідовності внутрішніх CharType значень на послідовність зовнішніх Byte значень.

virtual result do\_out(

StateType& state,

const CharType\* first1,

const CharType\* last1,

const CharType\*& next1,

Byte\* first2,

Byte\* last2,

Byte\*& next2) const;

Параметри

* *State* стан перетворення, який підтримується між викликами функції-члена;
* *first1* покажчик на початок послідовності перетворення;
* *last1* покажчик на кінець послідовності перетворення;
* *next1* посилання на покажчик на перший неперетворений об’єкт CharType після останнього CharType перетворення;
* *first2* покажчик початку перетвореної послідовності;
* *last2* покажчик на кінець перетвореної послідовності;
* *next2* посилання на покажчик на перший неперетворений об’єкт Byte після останнього Byte перетворення.

Значення, що повертається: Функція повертає:

* codecvt\_base: :error значення, якщо вихідну послідовність сформовано неправильно.
* codecvt\_base: :nocon v, якщо функція не виконує перетворення;
* codecvt\_base: :ok значення, якщо перетворення виконано;
* codecvt\_base: :partial, якщо джерело недостатнє або якщо призначення недостатнє для успішності перетворення.

Обєкт  *state*має представляти початковий стан перетворення на початку нової вихідної послідовності. Функція змінює це збережене значення, щоб відобразити поточний стан успішного перетворення. В іншому випадку збережене значення буде незаданим.

**codecvt::do\_unshift**

Віртуальна функція, що викликається для надання Byte значень, необхідних у перетвореннях, які залежать від стану, для завершення останнього символу в послідовності Byte значень.

virtual result do\_unshift(

StateType& state,

Byte\* first2,

Byte\* last2,

Byte\*& next2) const;

Параметри

* *State* стан перетворення, який підтримується між викликами функції-члена;
* *first2* покажчик першої позиції у цільовому діапазоні;
* *last2* покажчик останньої позиції у цільовому діапазоні;
* *next2* покажчик на перший незмінений елемент цільової послідовності.

Значення, що повертається: Функція повертає:

* codecvt\_base: :error, якщо codecvt\_base: :error представляє неприпустимий стан;
* codecvt\_base: :noconv, якщо функція не виконує перетворення;
* codecvt\_base: :ok, якщо перетворення виконано;
* codecvt\_base: :partial, якщо призначення недостатнє для успішного перетворення.

Захищена віртуальна функція-член намагається перетворити вихідний елемент CharType(0) на кінцеву послідовність, яка зберігається в [ first2, last2], за винятком завершального елемента Byte(0). Він завжди зберігає *next2* покажчик на перший незмінений елемент у кінцевій послідовності.

Обєкт *State*має представляти початковий стан перетворення на початку нової вихідної послідовності. Функція змінює це збережене значення так, щоб відобразити поточний стан успішного перетворення. Зазвичай, перетворення вихідного елемента CharType(0) залишає поточний стан вихідного стану перетворення.

**codecvt::encoding**

Перевіряє, чи залежить кодування Byte потоку від стану, незалежно від того, яке відношення між Byte значеннями, що використовуються, і CharType, значення якого є константами, і, якщо це так, визначає значення цього відношення.

int encoding() const throw();

Значення, що повертається: Якщо значення, що повертається додатнє, це значення є константним числом Byte символів, необхідних для створення CharType символу.

Захищена функція-член повертає:

* –1, якщо кодування послідовностей типу extern\_type залежить від стану;
* 0, якщо кодування включає послідовності змінної довжини;
* *N* значення, якщо кодування включає лише послідовності довжини *N*.

Функція-член повертає [do\_encoding](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_encoding) .

**codecvt::extern\_type**

Тип символу, який використовується для зовнішніх представлень.

typedef Byte extern\_type;

Цей тип є синонімом для параметра шаблону Byte.

**codecvt:: in**

Перетворює зовнішнє представлення послідовності Byte значень у внутрішнє представлення послідовності CharType значень.

result in(

StateType& state,

const Byte\* first1,

const Byte\* last1,

const Byte\*& next1,

CharType\* first2,

CharType\* last2,

CharType\*& next2,) const;

Параметри

* *state* стан перетворення, який підтримується між викликами функції-члена;
* *first1* покажчик на початок послідовності перетворення;
* *last1* покажчик на кінець послідовності перетворення;
* *next1* вказівник за межами перетвореної послідовності для першого неперетвореного символу;
* *first2* покажчик початку перетвореної послідовності;
* *last2* покажчик на кінець перетвореної послідовності;
* *next2* покажчик на об’єкт CharType, який надходить після останнього перетворення на Chartype перший незмінений символ у кінцевій послідовності.

Значення, що повертається: Значення, що повертається, яке вказує успіх, частковий успіх або збій операції. Функція повертає:

* codecvt\_base: :error значення, якщо вихідну послідовність сформовано неправильно;
* codecvt\_base: noconv, якщо функція не виконує перетворення;
* codecvt\_base: :ok значення, якщо перетворення виконано;
* codecvt\_base: :partial, якщо джерело недостатнє або якщо призначення недостатньо для успішності перетворення.

Обєкт *state*має представляти початковий стан перетворення на початку нової вихідної послідовності. Функція змінює це збережене значення, щоб відобразити поточний стан успішного перетворення. Після часткового перетворення *state* необхідно встановити, щоб перетворення було відновлено при появі нових символів.

Функція члена повертає [do\_in](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_in) ( state, first1, last1, next1, first2, last2, next2).

**codecvt::intern\_type**

Тип символу, який використовується для внутрішніх представлень.

typedef CharType intern\_type;

Цей тип є синонімом для параметра шаблону CharType.

**codecvt::length**

Визначає, скільки Byte значень із заданої послідовності зовнішніх Byte значень створює не більше заданої кількості внутрішніх CharType значень і повертає цю кількість Byte значень.

int length(

const StateType& state,

const Byte\* first1,

const Byte\* last1,

size\_t len2) const;

Параметри

* *state* стан перетворення, який підтримується між викликами функції-члена;
* *first1* покажчик початку зовнішньої послідовності;
* *last1* покажчик на кінець зовнішньої послідовності;
* *len2* максимальна кількість об’єктів Byte, яка може повернути функцію-член.

Значення, що повертається: Ціле число, що представляє число перетворень, що не перевищують значення *len2*, визначене зовнішньою вихідною послідовністю [first1, last1].

Функція члена повертає [do\_length](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_length) ( state, first1, last1, len2).

**codecvt::max\_length**

Повертає максимальну кількість зовнішніх Byte значень, необхідних для створення одного внутрішнього об’єкта CharType.

int max\_length() const throw();

Значення, що повертається: Максимальна кількість значень, Byte необхідне для його створення CharType.

Функція члена повертає [do\_max\_length](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_max_length).

**codecvt::out**

Перетворює послідовність внутрішніх CharType значень на послідовність зовнішніх Byte значень.

result out(

StateType& state,

const CharType\* first1,

const CharType\* last1,

const CharType\*& next1,

Byte\* first2,

Byte\* last2,

Byte\*& next2) const;

Параметри

* *state* стан перетворення, який підтримується між викликами функції-члена;
* *first1* покажчик на початок послідовності перетворення;
* *last1* покажчик на кінець послідовності перетворення;
* *next1* посилання на покажчик на перший неперетворений об’єкт CharType після останнього CharType перетворення;
* *first2* покажчик початку перетвореної послідовності;
* *last2* покажчик на кінець перетвореної послідовності;
* *next2* посилання на покажчик на перший неперетворений об’єкт Byte після останнього перетворення Byte.

Значення, що повертається: Функція члена повертає [do\_out](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_out) ( state, first1, last1, next1, first2, last2, next2).

**codecvt::state\_type**

Тип символу, який використовується для представлення проміжних станів під час перетворень між внутрішніми та зовнішніми представленнями.

typedef StateType state\_type;

Цей тип є синонімом для параметра шаблону StateType.

**codecvt:: unshift**

Надає Byte значення, необхідні для перетворення, які залежать від стану, для завершення останнього символу в послідовності Byte значень.

result unshift(

StateType& state,

Byte\* first2,

Byte\* last2,

Byte\*& next2) const;

Параметри

* *state* стан перетворення, який підтримується між викликами функції-члена;
* *first2* покажчик першої позиції у цільовому діапазоні;
* *last2* покажчик останньої позиції у цільовому діапазоні;
* *next2* покажчик на перший незмінений елемент цільової послідовності.

Значення, що повертається: Функція повертає:

* codecvt\_base::error, якщо *state* представляє неприпустимий стан;
* codecvt\_base::nocon v, якщо функція не виконує перетворення;
* codecvt\_base::ok значення, якщо перетворення виконано;
* codecvt\_base::partial значення, якщо призначення недостатньо для успішного перетворення.

Захищена віртуальна функція-член намагається перетворити вихідний елемент CharType(0) на кінцеву послідовність, яка зберігається в [ first2, last2], за винятком завершального елемента Byte(0). Він завжди зберігає *next2* покажчик на перший незмінений елемент у кінцевій послідовності.

Обєкт *state*має представляти початковий стан перетворення на початку нової вихідної послідовності. Функція змінює це збережене значення, щоб відобразити поточний стан успішного перетворення. Зазвичай, перетворення вихідного елемента CharType(0) залишає поточний стан вихідного стану перетворення.

Функція члена повертає [do\_unshift](https://docs.microsoft.com/ru-ru/cpp/standard-library/codecvt-class?view=msvc-170#do_unshift) ( state, first2, last2, next2 ).