Русификация языка Си++
проект

Русскоязычные эквиваленты для служебных слов Си++

ЗАО "МЛ:Компиляторы"
Email: info@mlc.ru
Tel: (095) 939-2638

Общие положения

Одной из особенностей проекта С11 является возможность использовать символы кириллицы в программах на Си++. Эта возможность проявляется в следующих трех аспектах:

* задание русскоязычных текстов в комментариях;
* использование символов кириллицы с идентификаторах;
* допущение русскоязычных эквивалентов служебных слов.

Первый аспект допускается Стандартом, при реализации не требует каких-либо дополнительных усилий и обеспечивается уже в нынешнем компиляторе.

Для поддержки символов кириллицы в идентификаторах необходима очень небольшая модификация компилятора. Очевидно, что необходимо обеспечить одновременное присутствие в программе как полностью кириллических и полностью латинских идентификаторов, а также идентификаторов, в составе которых имеются символы из обоих множеств.

Введение русскоязычных эквивалентов для служебных слов Си++ представляет собой более ответственную задачу; прежде всего это относится к качеству их перевода.

Общий список служебных слов Си++ согласно Стандарту

(73 слова в алфавитном порядке)

and	const_cast	friend	private	this
and_eq	continue	goto	protected	throw
asm	default	if	public	true
auto	delete	inline	register	try
bitand	do	int	reinterpret_cast	typedef
bitor	double	long	return	typeid
bool	dynamic_cast	mutable	short	typename
break	else	namespace	signed	union
case	enum	new	sizeof	unsigned
catch	explicit	not	static	using
char	extern	not_eq	static_cast	virtual
class	false	operator	struct	void
compl	float	or	switch	volatile
const	for	or_eq	template	wchar_t
				while
				xor
				xor_eq

Принципы перевода служебных слов

Основные принципы, которыми следует руководствоваться при выборе переводов конкретных слов:

1. Перевод не обязательно должен быть буквальным; прежде всего он должен обеспечивать прежде всего смысловую идентичность.
2. Перевод должен быть единообразным: например, если используются сокращения, то стиль сокращений должен быть единым.
3. Необходимо учесть особенности русского языка, прежде всего, наличие родов для прилагательных. Например, в объявлении вида

```
int a;
```

описатель типа `int` можно трактовать как "целая переменная", "целый объект", "целое данное". В таких случаях можно как отбрасывать родовые окончания, так и допускать несколько вариантов. Общее решение в данном случае состоит в следующем: основной (официальный) перевод подобных служебных слов представляет собой единообразное сокращение, полученное отбрасыванием окончаний. Наряду с официальным имеется ряд альтернативных переводов, которые поддерживаются комплятором, но не рекомендуются для активного использования.

4. Некоторые служебные слова можно переводить различными грамматическими формами: как существительные, как глаголы, как предлоги. Например, для `goto` допустимы следующие переводы:

```
иди
идти
на
переход
```

В подобных случаях предпочтение дается **существительным**, как наиболее нейтральным формам. Это решение позволит избежать неопределенности в случае глаголов и ненужной эмоциональной окраски (например, для варианта "иди"). Так, перевод слова `continue` в глагольной форме вызовет дилемму вроде "продолжать" или "продолжить", или "продолж" (не говоря о не вполне точном соответствии этого глагола семантике оператора `continue`).

5. В некоторых случаях допускаются отступления от перечисленных правил, если они приводят к неудобочитаемым, неблагозвучным или семантически неточным переводам, а также если для данного слова имеется традиционная русскоязычная нотация.

Перевод служебных слов Си++

Ниже приводится предлагаемый перевод служебных слов. Служебные слова сгруппированы в соответствии со смысловыми связями.

В левой колонке дается оригинальное служебное слово Си++, во второй колонке - официальный перевод компании МедиаЛингва:Компиляторы. Третья колонка содержит альтернативные переводы. Наконец, в последней колонке представлены варианты переводов, которые рассматривались, но были отброшены по каким-либо причинам. Здесь же содержатся краткие объяснения этих причин и прочие комментарии.

Оригинальное служебное слово	Официальный перевод	Альтернативные переводы	Рассматривавшиеся и отброшенные варианты
and	**лог_и**	нет; традиционное обозначение - **&&**	Перевод обозначения операции **and** как **и** был отвергнут ввиду недопустимости служебных слов, состоящих из одной буквы. Поэтому для единообразия и другие операции - **or**, **xor**, **not** - переводятся несколько более громоздко, однако единообразно и в стиле языка Си/Си++. Следует иметь в виду, что в Стандарте данные обозначения для логических операций введены как альтернативы известным лексемам &&, ||, !, ^ и, скорее всего, не будут широко использоваться.
and_eq	**и_присв**	нет; традиционное обозначение **&=**	
asm	**ассем**	**ассемблер**	
auto	**автом**	**автоматич** **автоматический** **автоматическая** **автоматическое** **автоматические**	**авто** Предпочтение было отдано варианту **автом**, так как прямая транслитерация вызывает ненужные ассоциации (**авто** - распространенное сокращение для "автомобиль").
bitand	**бит_и**	нет; традиционное обозначение **&**	
bitor	**бит_или**	нет; традиционное обозначение **|**	
bool	**лог**	**логич** **логический** **логическая** **логическое** **логические**	**бул** **булев** Обычно, в знак уважения в Д.Булю, основателю алгебры логики, логические объекты называют "булевскими". Однако, в русском переводе сокращения **бул**, **булев** выглядят не слишком благозвучно, поэтому был выбран нейтральный вариант **лог**.
break	**прервать**		**прерывание** **заверш** Для перевода слова **break** был выбран глагольный перевод **прерывание**, **прервать**. Предпочтение отдано глагольной форме, так как существительное **прерывание** несет специфическую нагрузку, подразумевая аппаратное прерывание.
case	**вариант**		**вар**
catch	**перехват**		См. **throw**.

char	**сим**	**симв** **символьный** **символьная** **символьное** **символьные**	**симв** **литер** Описатель **сим** выглядит так же наглядно, как и **симв**, поэтому был выбран более компактный вариант. (Более точный вариант **литер** был отвергнут из-за множественных толкований этого слова.)
class	**класс**		Для слова **class**, обозначающего ключевое в Си++ понятие, невозможно предложить ничего, кроме прямого перевода.
compl	**бит_не**	нет; традиционное обозначение **~**	
const	**конст**	**константа** **константный** **константная** **константное** **константные**	
const_cast	**конст_прив**		**конст_привед** На редкость неуклюжее имя; перевод неизбежно оказался таким же неуклюжим. Единственное утешение: эта конструкция используется относительно редко.
continue	**итерация**		**еще_итерация** **продолж** Как и во многих других случаях, вариант с существительными для слов **continue**, **return** снимает глагольную неопределенность. Кроме того, для слова **continue** буквальный перевод вида **продолжить**, **продолжать** и т.п. не позволяет выявить смысл соответствующего оператора; вариант **итерация** кажется наиболее предпочтительным, будучи одновременно недвусмысленным и семантически точным.
default	**проч**	**прочие** **прочее**	**умолч** Аналогично **switch**, для слова **default** предпочтительным кажется не буквальный перевод - **умолчание**,- а более точный в данном контексте вариант **прочее** или **проч**. Последний вариант прямо соотносится с известной традицией и поэтому выглядит вполне естественно.
delete	**удалить**		**удл** **удал** Комментарий см. в **new**.
do	**цикл**		**выполн** **вып** Попытка перевести **do** буквально, как **выполнить**, **выполн**, **вып** (последнее вообще неприемлемо, так как вызывает ассоциации с "выпить", "выплюнуть" и т.д.) кажется существенно хуже по сравнению с семантически более точным существительным **цикл**, которое, к тому же, ввиду своей краткости не нужается в сокращении.
double	**двойн**	**двойной** **двойная** **двойное** **двойные** **двойн_точн** **двойной_точности**	**удв** Вариант был отброшен ввиду своей неочевидности.
dynamic_cast	**дин_прив**		**дин_привед** На редкость неуклюжее имя; перевод неизбежно оказался таким же неуклюжим. Единственное утешение: эта конструкция используется относительно редко.

else	**иначе**		Все достаточно очевидно и понятно.
enum	**перечисл**	**перечисление**	Не слишком элегантный, но точный перевод. Полный вариант - **перечисление** - кажется громоздким, а все остальные варианты сокращений, кроме предложенного, - **пер**, **переч** и т.д. - явно неблагозвучны. Заметим, что практически во всех языках программирования, имеющих перечислимые типы, они вводятся без использования специальных служебных слов.
explicit	**явн**	**явная** **явный**	
extern	**внеш**	**внешний** **внешняя** **внешнее** **внешние**	**внешн** Вариант **внеш** выглядит более элегантно по сравнению с более громоздким **внешн**.
false	**ложь**		**нет** **ложь** - единственно возможный перевод. Вариант **нет** отдает любительщиной. См. также **true**.
float	**плав**	**плавающий** **плавающая** **плавающее** **плавающие**	**плв** См. комментарий для **long**
for	**для**		**цикл_для** Точный перевод в данном случае согласуется в русскоязычной традицией в программировании, идущей от первых российских компиляторов Алгола-60. В принципе, можно было бы предложить вариант **цикл_для**, дополнительно проясняющий семантику данного оператора, однако он был отвергнут из-за большей громоздкости.
friend	**друж**	**друг** **дружеств** **дружественный** **дружественная** **дружественные**	**дружеств** Соблазн заменить буквальный перевод каким-либо более наглядным (хотелось дополнительно прояснить понятие дружественных функций и классов в Си++) был преодолен, так как все популярные описания Си++ оперируют именно этим названием. Введение другого, даже более наглядного обозначения привело бы к необходимости ссылок типа "в оригинале это свойство определяется описателем **friend**, что в русскоязычных книгах переводится как "дружественный".
goto	**переход**		**на** **иди** **идти** Выбор сделан в пользу существительного. Помимо общего принципа предпочтительности существительных, относительно длинное слово для явно дискуссионного и для многих сомнительного оператора перехода кажется более оправданным, позволяя легче обнаружить его в тексте программы.
if	**если**		Все достаточно очевидно и понятно.

inline	**подставл**	**подставляемая** **подставляемые**	**подст откр открыт** Предложенный вариант перевода нельзя считать окончательным ввиду его громоздкости (более компактный вариант **подст** отвергнут из-за ненаглядности); однако другого подходящего слова предложить пока не удалось. Заметим, что в русскоязычной программистской литературе, начиная с 60-х г.г. для подпрограмм данного вида использовался термин "открытая".
int	**цел**	**целый** **целая** **целое** **целые**	
long	**длин**	**длинный** **длинная** **длинное** **длинные**	**длн** Дополнительные, помимо отбрасывания окончаний, сокращения внутри слова (**длн**, **плв**), кажутся неоправданными, даже если они продиктованы стремлением "втиснуть" все обозначения стандартных типов в три символа.
mutable	**изменч**	**изменчивый** **изменчивая** **изменчивое** **изменчивые**	**измен** Описатель **mutable** говорит именно об *изменчивости* соответствующего члена класса: даже если объект этого класса объявлен как константный, данный его член может изменять свое значение, в отличие от других членов. Вариант **измен** был отвергнут из-за своей неочевидности.
namespace	**область**		**обл** Полный перевод слова **namespace** - **область** - кажется более наглядным и предпочтительным по сравнению с **обл** (служебные слова заголовочного характера, каким является **namespace**, не стоит делать слишком короткими).
new	**создать**		**нов новый** Точный перевод операций **new** и **delete** как **новый** (**нов**) и **удалить** несимметричен, что маскирует очевидную симметричность семантики этих операций. В русском переводе (то есть для русскоязычного пользователя языка) эта несимметричность проявится со всей очевидностью. Поэтому решение перевести **new** и **delete** в виде пары "взаимоисключающих" глаголов кажется более предпочтительным.
not	**лог_не**	нет; традиционное обозначение **!**	**не**; см. комментарий к **and**
not_eq	**не_равно**	нет; традиционное обозначение **!=**	

operator	**операция**		Английское слово **operator** в данном контексте однозначно переводится как "операция". Например, выражение a+b задает **операцию** сложения (обозначенную **знаком операции** - символом "+"); соответственно, языковая конструкция, позволяющая определить собственную версию **операции** сложения, содержит служебное слово **операция**. Вульгарный буквальный перевод **operator** как "оператор" затемняет смысл соответствующей конструкции языка и приводит к путанице, так как в русскоязычной программистской литературе понятием "оператор" исторически обозначается конструкция языка, задающая некоторое элементарное действие без образования нового значения (англоязычный прототип - statement). Именно такая трактовка слов operator и statement была принята в русском переводе книги Эллис и Страуструпа.
or	**лог_или**	нет; традиционное обозначение **||**	**или**; см. комментарий к **and**
or_eq	**или_присв**	нет; традиционное обозначение **|=**	
private	**себе**	**скрытый** **скрытая** **скрытое** **скрытые**	**для_себя** **прив** **скрыт**
protected	**своим**	**защищ** **защищенный** **защищенная** **защищенное** **защищенные**	**для_своих** **защищ** **защ**
public	**всем**	**доступный** **доступная** **доступное** **доступные**	**для_всех** **пуб** **публ** **общедост** Для часто используемых в оъектно-ориентированном программировании спецификаторов **private**, **public** и **protected** в качестве официальных предлагаются несколько нетривиальные по стилю переводы в виде притяжательных местоимений **себе**, **всем** и **своим**. Представляется, что такие трактовки явно точнее оригинальных, заметно повышают наглядность этих спецификаторов и позволяют легче запомнить их назначение. Их необычность кажется допустимой, кроме того, по той причине, что они и используются не вполне стандартным образом: не как модификаторы для каждого отдельного объявления (как в языке Java), а как своего рода заголовки внутри тела класса. Варианты **для_себя**, **для_всех**, **для_своих** были отброшены ввиду своей неоправданной громоздкости. В списке представлены также традиционные (буквальные) переводы этих спецификаторов; кажется очевидным, что все они существенно хуже по тем или иным причинам.

register	**рег**	**регистровый** **регистровая** **регистровое** **регистровые**	**регистр** Был выбран сокращенный вариант перевода, который не вызывает неверных ассоциаций с регистрами аппаратуры. В противном случае объявление вида **регистр** `A;` можно было бы трактовать как задание некоторого аппаратного регистра с обозначением `A`. Представляется, что сокращение **рег** менее прямолинейно и поэтому более соответствует семантике описателя как просто подсказке компилятору.
reinterpret_cast	**тип_прив**		**тип_привед** На редкость неуклюжее имя; перевод неизбежно оказался таким же неуклюжим. Единственное утешение: конструкция используется относительно редко. Для самого неуклюжего слова из группы **xxx-cast-reinterpret_cast** предложен чуть менее громоздкий, но несколько более точный перевод, нежели буквальный.
return	**возврат**		**вернуть** **возвратить** Как и во многих других случаях, вариант с существительными для слова **return** снимает глагольную неопределенность.
short	**кор**	**короткий** **короткая** **короткое** **короткие**	
signed	**со_знаком**	**знаковый** **знаковая** **знаковое** **знаковые**	**знак** Спецификаторы **signed** и **unsigned** имеют двойной смысл: во-первых, они обозначают знаковый (беззнаковый) целочисленные типы и в этом качестве могут использоваться независимо; во-вторых, они могут использоваться как дополнительные спецификаторы к спецификатору некоторого целочисленного типа, явно обозначая его (знаковый или беззнаковый) вариант, например, **unsigned long** `i;` Перевод этих спецификаторов как прилагательных, очевидно, затруднителен: так, вариант **знак** не воспринимается как сокращение от "знаковый", а, скорее, может пониматься как некоторый отдельный тип **знак**, обозначающий, например, однобайтовые литеральные символы. Поэтому был выбран более громоздкий, но совершенно недвусмысленный вариант перевода.
sizeof	**размер**		Кто придумает что-то другое?
static	**стат**	**статический** **статическая** **статическое** **статические**	
static_cast	**стат_прив**		**стат_привед** На редкость неуклюжее имя; перевод неизбежно оказался таким же неуклюжим. Единственное утешение: эта конструкция используется относительно редко.
struct	**структ**	**структура**	Варианты с полным и сокращенным переводом слова **struct** примерно одинаково приемлемы с точки зрения наглядности.

switch	**выбор**	**перекл** **переключатель** Для оператора выбора было решено отказаться от буквального перевода служебных слов. Использование для **switch** перевода типа **перекл** кажется несколько неуклюжим (не говоря уже о **переключатель**) и искажающим смысл. Традиционный перевод - **выбор** - обращается к русскоязычной традиции (идущей от Паскаля), заметно понятнее и компактнее. См. также **default**.
template	**шаблон**	Можно было бы проявить оригинальность и выбрать вариант вроде **образец**, однако это противоречило бы складывающейся традиции обозначения данного понятия в русскоязычной литературе.
this	**этот**	
throw	**исключение**	**искл исключ передать передача ситуация** В данном случае буквальный перевод служебных слов, предполагающий глагольную форму ("пытаться" для **try**, "поймать" для **catch**, "бросить" для **throw**) приводит к нелепым конструкциям. Вариант с существительными выглядит гораздо более стройным, естественным и не вызывающим посторонних ассоциаций. Кроме того, полный вариант слова **исключение** заметно лучше неуклюжих сокращений **искл**, **исключ**. Остаются сомнения, стоит ли конструкцию, обозначающую некоторое (вполне нетривиальное) **действие** - возбуждение исключительной ситуации,- обозначать несколько аморфным словом **исключение** (которое, строго говоря, и не обозначает никакого действия). Быть может, слово **передать** или **передача** выглядит естественнее. (Варианты **бросить**, **возбудить**, **возбуждение**, равно как и их сокращения, были отброшены как абсолютно неприемлемые.)
true	**истина**	**да** **истина** - единственно возможный перевод. Вариант **да** отдает любительщиной.
try	**контроль**	Перевод слова **try** как **контроль** семантически более точен и соответствует переводу понятия *try-block* как "блок-с-контролем", предложенному в Зеленой книге (Эллис, Страуструп. "Справочное руководство по языку Си++ с комментариями"). См. также **catch**, **throw**.
typedef	**тип**	**оптип**

typeid	**есть_тип**		**тип естьтип** Альтернативный, более буквальный перевод: **typedef** - **оптип**, **typeid** - **тип** был отвергнут по причине очень широкой распространенности конструкции **typedef** и, наоборот, относительной редкости **typeid**. Поэтому для частой конструкции был выбран более простой вариант **тип**, а для редкой операции **typeid** - более громоздкий, но более точный перевод **есть_тип**.
typename	**имя_типа**		**имятипа**
union	**совмещ**	**совмещение**	**союз объед объединение** Все варианты буквальных переводов слова **union** (союз, объединение) отвергнуты из-за того, что они скрывают назначение конструкции (конструкция **union** не "объединяет" свои члены, а, скорее, допускает "совмещение" этих членов в памяти). Так что в этом отношении предпочтительнее перевод **совмещение** или его сокращенный вариант.
unsigned	**без_знака**	**беззнаковый** **беззнаковая** **беззнаковое** **беззнаковые**	**беззнак** См. также **signed**
using	**использ**	**использовать**	**исп** Сокращение вида **исп** может приводить к двусмысленности ("использовать" или "исполнить").
virtual	**вирт**	**виртуальный** **виртуальная** **виртуальное** **виртуальные**	**виртуал**
void	**пуст**	**пустой** **пустая** **пустое** **пустые**	
volatile	**ненадеж**	**ненадежный** **ненадежная** **ненадежное** **ненадежные**	**подвиж** Обычно следует избегать использования прилагательных с приставкой "не". Однако в данном случае использование слова с "не" выглядит оправданно, так как сама семантика данного описателя описывается им более точно: считается, что значение объекта, объявленного с данным описателем, может изменяться асинхронно по отношению к потоку управления программы; в этом смысле объект является ненадежным, то есть нельзя делать никаких предположений о его значении в любой момент времени. В то же время он не является "подвижным": его положение в памяти не может изменяться.
wchar_t	**шсим**	**широкий_симв** **широкая_симв** **широкое_симв** **широкие_симв**	**ш_сим шир_сим** Вариант **шсим** кажется естественнее буквального **ш_сим** или **шир_сим** (последнее неприемлемо ввиду аналогий типа "ширяться").
while	**пока**		

xor	**искл_или**	нет; традиционное обозначение **^**	**бит_искл_или** **мод2** Перевод обозначений операций ^ и ^= сделан несколько отличным от других аналогичных операций ввиду сложного названия данной операции - "исключающее или" в западной терминологии или "сложение по модулю 2" в российской математической традиции. Полный вариант **бит_искл_или**, хотя он и сочетается с другими операциями **бит_и**, **бит_или**, **бит_не**, был отброшен ввиду своей громоздкости, тем более, что "логического" варианта этой операции нет.
xor_eq	**искл_присв**	нет; традиционное обозначение **^=**	

Перевод служебных слов препроцессора Си++

Оригинальное служебное слово	Официальный перевод	Альтернативные переводы	Рассматривавшиеся и отброшенные варианты
#define	**#макрос**		**#опред** **#определить** В данном случае предпочтение было отдано существительным, в первую очередь, потому, что это позволило избежать сокращений и получить краткие и выразительные названия. Заметим, что "макрос" - корректная форма единственного числа для данного понятия. Именно такой перевод английского macro был принят в отечественной профессиональной литературе, начиная с 70-х гг. (см., например, М.Кемпбел-Келли, "Введение в макросы": Пер. с англ./ Под ред. Э.З.Любимского.- М.: Сов. Радио, 1978). Общеупотребительный в устной речи (иногда проникающий и в печатные тексты) вульгаризм "макро" возник только в последнее время.
defined	**есть**		Если директива **#ifdef** переводится как **#если_есть**, то некоторую ее модификацию естественно перевести аналогично. Пример: **#если** (**есть** M1 || **есть** M2) ...
#elif	**#инес**	**#иначе_если**	**#ин_если** **#ин_ес** Перевод директивы **#elif** как **#инес** выглядит наиболее кратко и вместе с тем читабельно. Кроме того, именно так традиционно переводится служебное слово **elif** в тех языках программирования, в которых оно имеется (например, в Алгол-68).
#else	**#иначе**		Было бы странно, если бы директивы **#if** и **#else** были бы переведены как-либо иначе, имея ввиду одноименные служебные слова самого языка.
#endif	**#конес**	**#конец_если**	**#кон_если** **#конесли** **#илсе** Директива **#endif** переведена как **#конес**,- в духе **#если** и **#инес**. "Стильный" перевод как **#илсе** - перевернутой начальной директивы, сам по себе очень привлекательный (и используемый, в частности, для аналогичного служебного слова в Алголе-68), был отвергнут, так как в данном случае он противоречил бы общему стилю данных предложений.

#error	**#ошибка**		
#if	**#если**		
#ifdef	**#если_есть**	**#еслиесть**	**#если_опред**
#ifndef	**#если_нет**	**#еслинет**	**#если_не_опред**
			Предлагаемый перевод **#ifdef** **#ifndef** как, соответственно, **#если_есть** и **#если_нет**, представляется наиболее точным из возможных: директива "срабатывает", если ранее уже было (еще не было) определение макроса с указанным именем.
#include	**#вставка**	**#вставить**	**#внести**
			Здесь также использовано существительное; быть может, перевод **#вставить** так же нагляден и точен, но для единообразия (чтобы директивы определения макроса, его отмены и использования представлялись существительными) была выбрана указанная форма.
#line	**#строка**		
#pragma	**#прагмат**	**#прагма**	Практически единственный возможный перевод. Вариант **#опция**, в принципе, тоже допустим, но он несет оттенок жаргона, используемого в устной речи. К предложенному варианту перевода полностью относится комментарий для директивы **#макрос** (имея ввиду распространенность вульгарного перевода "прагма").
#undef	**#отмена**	**#отмен**	**#отменить**
			Предпочтение было отдано существительным; см. **#define**.
__DATE__	__ДАТА__		
__FILE__	__ФАЙЛ__		
__LINE__	__СТРОКА__		
__TIME__	__ВРЕМЯ__		
__cplusplus	__СИПЛЮСПЛЮС		
__STDC__	__СТАНДСИ__	__СТДСИ__	Имена предопределенных макросов либо имеют совершенно однозначный и краткий перевод, либо представляют собой неудобопроизносимые сокращения, возникшие исторически по произволу создателей языков Си и Си++. Поэтому не следует надеяться исправить дело, пытаясь ввести для них некие "красивые" эквиваленты на русском языке.

■