Reserve good keywords for floating point types

Document #:	PXXXXRX
Date:	2019-10-13
Project:	Programming Language C++ Evolution Working Group
Reply-to:	Lyberta <lyberta@lyberta.net>

1 In a nutshell

Reserve the following keywords:

— `iec60559_binary16`
— `iec60559_binary32`
— `iec60559_binary64`
— `iec60559_binary128`

2 Rationale

[P1468R1] introduces new distinct fixed-layout floating point types but makes them available as aliases under namespace `std`. The author thinks that some of those types are so widely used that putting them behind the namespace makes teachability and long-term evolution of C++ hindered.

IEC 60559 `binary32` and `binary64` are the most widely used and teached floating point formats. Most code assumes that `float` and `double` correspond to those types respectively and will break in surpsrising ways if they are not. Yet, C++ gives little portable guarantees about those types. A good code can `static_assert` on `numeric_limits::is_iec559` but this is relatively obscure feature and beginners can't be relied upon to know that. Also, seeing types `float` and `double` in code doesn't specify the intention of said code with regards to data precision. The types are simply too ambiguous.

Modern code emphasizes data portability. It is crucial to be able to serialize data on one computer, send it over the network and deserialize it on another computer with different operating system and/or hardware architecture. `float` and `double` do not give enough guarantees to do that.

C++20 moves closer towards data portability by requiring all integers to be two's complement and providing `std::endian` so that it is possible to write integer serialization code portably. Having fixed layout floating point keywords will make portable serialization code much easier to write, will make it easier to teach beginners portable floating point types and will emphasize that IEC 60559 formats are first-class citizens that should be preferred instead of legacy floating point types.

This paper only reserves the keywords and leaves introduction of new fundamental types to a later paper, most likely later revision of [P1468R1].

3 Impact on existing code

No mentions of proposed identifiers are found after searching on codesearch.isocpp.org and GitHub.

4 Wording

All text is relative to [N4835].

Modify **Table 5** [**tab:lex.key**] as follows:

alignas	constinit	false	nullptr	template
alignof	const_cast	float	operator	this
asm	continue	for	private	thread_local
auto	co_await	friend	protected	throw
bool	co_return	goto	public	true
break	co_yield	iec60559_binary16	register	try
case	decltype	iec60559_binary32	reinterpret_cast	typedef
catch	default	iec60559_binary64	requires	typeid
char	delete	iec60559_binary128	return	typename
char8_t	do	if	short	union
char16_t	double	inline	signed	unsigned
char32_t	dynamic_cast	int	sizeof	using
class	else	long	static	virtual
concept	enum	mutable	static_assert	void
const	explicit	namespace	static_cast	volatile
consteval	export	new	struct	wchar_t
constexpr	extern	noexcept	switch	while

Modify paragraph 1 of **§5.11** [**lex.key**] as follows:

The identifiers shown in Table 5 are reserved for use as keywords (that is, they are unconditionally treated as keywords in phase 7) except in an *attribute-token* (9.11.1):

[*Note*: The `iec60559_binary16`, `iec60559_binary32`, `iec60559_binary64`, `iec60559_binary128` and `register` keywords ~~is~~are unused but ~~is~~are reserved for future use. — *end note*]

5 References

[N4835]
https://wg21.link/n4835

[P1468R1] Michał Dominiak, Boris Fomitchev, Sergei Nikolaev. 2019. Fixed-layout floating-point type aliases.
https://wg21.link/p1468r1