C++ Reserved Keywords[*]

Kenneth Leroy Busbee

Abstract

A brief explanation and list of C++ reserved keywords for use in a programming fundamentals course.

All programming languages have "reserved words". There are usually less than 50 of these reserved words in any given programming language. They are reserved because they have been pre-assigned a specific meaning within that programming language, thus the compiler recognizes those words to mean a specific thing or action. Within C++ the reserved words are also known as "keywords".

Programmers use identifier names for a variety of items, to include: functions, variables, named constants, alias names, etc. But, they can't use as identifier names the words that are "reserved to the language".

For the C++ language all "reserved keywords" are typed in lower case. The list that follows includes the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) lists of reserved words for the C++ programming language. The ISO reserved words may not be implemented in the compiler that you are using, however they may be adopted in future releases of C++ compilers. Wisdom dictates to avoid using them at this point so that there will not be a problem compiling your source code in future releases of compilers. There has been no distinction made in the ANSI or ISO reserved word lists. A search of the Internet for C++ reserved words will reveal several different lists. Some are more unique to a specific compiler. Some will be incomplete because the list has been enlarged. The talbe that follows should work for any beginning programming course using C++. The reserved keywords are:

[*] Version 1.6: Jan 14, 2010 5:57 am +0000

[†] http://creativecommons.org/licenses/by/2.0/

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

Table 1