Notes on Variable Names (identifiers) in C

- Valid identifiers
  - Sequence of lower and/or upper case letters, digits, and underscores
    - Note, case sensitive! So, **`my_var`** and **`My_var`** are distinct.
  - Initial character may not be a digit
  - Cannot use reserved words as identifiers (see tables below)

- Strategy for names (for maximum portability and usability)
  - Keep length to 31 or fewer characters
  - Make the first 8 characters unique
  - Avoid names used by run-time libraries (e.g., **`log`**, which is used in math.h)
  - Avoid using names beginning with underscores
  - Try to use lowercase letters for variable names, and UPPER CASE letters for macro names (e.g., #define PI 3.14159)
  - Choose names that are meaningful (e.g., **`light_level`** instead of just **`output`**)

Reserved C Keywords[1] (You may not use these as variable names)

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Reserved C++ Keywords[2] (You may not use these as variable names in Clean C[3] programs)

asm	inline	static_cast
bol	mutable	template
catch	namespace	this
class	new	throw
const_cast	operator	true
delete	private	try
dynamic_cast	protected	typeid
false	public	using
friend	reinterpret_cast	virtual

[1] Adapted from Darnell, P. A., and Margolis, P. E., C: A Software Engineering Approach, 3rd ed., Springer, New York, 1996, p. 42.
[2] ibid
[3] 'Clean C' is a subset of ANSI C that leaves out older features of C that might not compile with C++ compilers, and thus maximizes portability of program code.