G++ 2.91.57，cygnus\cygwin-b20\include\g++**stl_stack.h** 完整列表

```
/*
 *
 * Copyright (c) 1994
 * Hewlett-Packard Company
 *
 * Permission to use, copy, modify, distribute and sell this software
 * and its documentation for any purpose is hereby granted without fee,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation.  Hewlett-Packard Company makes no
 * representations about the suitability of this software for any
 * purpose.  It is provided "as is" without express or implied warranty.
 *
 *
 * Copyright (c) 1996,1997
 * Silicon Graphics Computer Systems, Inc.
 *
 * Permission to use, copy, modify, distribute and sell this software
 * and its documentation for any purpose is hereby granted without fee,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation.  Silicon Graphics makes no
 * representations about the suitability of this software for any
 * purpose.  It is provided "as is" without express or implied warranty.
 */

/* NOTE: This is an internal header file, included by other STL headers.
 *   You should not attempt to use it directly.
 */

#ifndef __SGI_STL_INTERNAL_STACK_H
#define __SGI_STL_INTERNAL_STACK_H

__STL_BEGIN_NAMESPACE

#ifndef __STL_LIMITED_DEFAULT_TEMPLATES
template <class T, class Sequence = deque<T> >
#else
template <class T, class Sequence>
#endif
class stack {
  friend bool operator== __STL_NULL_TMPL_ARGS (const stack&, const stack&);
  friend bool operator< __STL_NULL_TMPL_ARGS (const stack&, const stack&);
public:
  typedef typename Sequence::value_type value_type;
  typedef typename Sequence::size_type size_type;
  typedef typename Sequence::reference reference;
  typedef typename Sequence::const_reference const_reference;
```

```
protected:
  Sequence c;     // 底層容器
public:
  // 以下完全利用 Sequence c 的操作，完成 stack 的操作。
  bool empty() const { return c.empty(); }
  size_type size() const { return c.size(); }
  reference top() { return c.back(); }
  const_reference top() const { return c.back(); }
  // deque 是兩頭可進出，stack 是末端進，末端出（所以後進者先出）。
  void push(const value_type& x) { c.push_back(x); }
  void pop() { c.pop_back(); }
};

template <class T, class Sequence>
bool operator==(const stack<T, Sequence>& x, const stack<T, Sequence>& y)
{
  return x.c == y.c;
}

template <class T, class Sequence>
bool operator<(const stack<T, Sequence>& x, const stack<T, Sequence>& y)
{
  return x.c < y.c;
}

__STL_END_NAMESPACE

#endif /* __SGI_STL_INTERNAL_STACK_H */

// Local Variables:
// mode:C++
// End:
```