```
 1: // $Id: demangle.cpp,v 1.9 2015-02-05 19:05:17-08 - - $
 2:
 3: // Demangle a typeid(X).name() string
 4:
 5: #include <cstdlib>
 6: #include <iostream>
 7: #include <list>
 8: #include <map>
 9: #include <memory>
10: #include <string>
11: #include <typeinfo>
12: #include <vector>
13:
14: using namespace std;
15:
16: #include <cxxabi.h>
17: template <typename type>
18: string demangle (const type &object) {
19:    const char *const name = typeid (object).name();
20:    int status;
21:    char* demangled = abi::__cxa_demangle (name, nullptr, 0, &status);
22:    if (demangled == nullptr) return name;
23:    string result {status == 0 ? demangled : name};
24:    free (demangled);
25:    return result;
26: }
27:
28: class foo { };
29: class bar: foo { };
30: class baz: bar { };
31: template <typename T> class tmpl { T x; };
32:
33: template <typename type>
34: void print_demangled (const string &str, const type &obj) {
35:    cout << str << " => " << sizeof obj << endl;
36:    cout << "   mangled:   " << typeid(obj).name() << endl;
37:    cout << "   demangled: " << demangle (obj) << endl;
38: }
39:
40: #define DEMANGLE(X) print_demangled (#X, X())
41: int main() {
42:    using schar = signed char;
43:    using uchar = unsigned char;
44:    using ushort = unsigned short;
45:    using uint = unsigned int;
46:    using ulong = unsigned long;
47:    using map_string_int = map<string,int>;
48:    DEMANGLE (bool);
49:    DEMANGLE (char);
50:    DEMANGLE (uchar);
51:    DEMANGLE (schar);
52:    DEMANGLE (short);
53:    DEMANGLE (ushort);
54:    DEMANGLE (int);
55:    DEMANGLE (uint);
56:    DEMANGLE (long);
57:    DEMANGLE (ulong);
58:    DEMANGLE (float);
```

```
   59:    DEMANGLE (double);
   60:    DEMANGLE (size_t);
   61:    DEMANGLE (foo);
   62:    DEMANGLE (bar);
   63:    DEMANGLE (baz);
   64:    DEMANGLE (tmpl<int>);
   65:    DEMANGLE (vector<string>);
   66:    DEMANGLE (vector<int>);
   67:    DEMANGLE (list<vector<long>>);
   68:    DEMANGLE (map_string_int);
   69:    return 0;
   70: }
   71:
   72: //TEST// demangle >demangle.out 2>&1
   73: //TEST// mkpspdf demangle.ps demangle.cpp* demangle.out
   74:
```

```
   1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting demangle.cpp
   2: demangle.cpp:
   3:       $Id: demangle.cpp,v 1.9 2015-02-05 19:05:17-08 - - $
   4: g++ -g -O0 -Wall -Wextra -rdynamic -std=gnu++11 demangle.cpp -o demangle
-lglut -lGLU -lGL -lX11 -lrt -lm
   5: rm -f demangle.o
   6: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished demangle.cpp
```

```
bool => 1
   mangled:   b
   demangled: bool
char => 1
   mangled:   c
   demangled: char
uchar => 1
   mangled:   h
   demangled: unsigned char
schar => 1
   mangled:   a
   demangled: signed char
short => 2
   mangled:   s
   demangled: short
ushort => 2
   mangled:   t
   demangled: unsigned short
int => 4
   mangled:   i
   demangled: int
uint => 4
   mangled:   j
   demangled: unsigned int
long => 8
   mangled:   l
   demangled: long
ulong => 8
   mangled:   m
   demangled: unsigned long
float => 4
   mangled:   f
   demangled: float
double => 8
   mangled:   d
   demangled: double
size_t => 8
   mangled:   m
   demangled: unsigned long
foo => 1
   mangled:   3foo
   demangled: foo
bar => 1
   mangled:   3bar
   demangled: bar
baz => 1
   mangled:   3baz
   demangled: baz
tmpl<int> => 4
   mangled:   4tmplIiE
   demangled: tmpl<int>
vector<string> => 24
   mangled:   St6vectorISsSaISsEE
   demangled: std::vector<std::string, std::allocator<std::string> >
vector<int> => 24
   mangled:   St6vectorIiSaIiEE
   demangled: std::vector<int, std::allocator<int> >
list<vector<long>> => 16
```

```
   59:    mangled:   St4listISt6vectorIlSaIlEESaIS2_EE
   60:    demangled: std::list<std::vector<long, std::allocator<long> >, std::a
llocator<std::vector<long, std::allocator<long> > > >
   61: map_string_int => 48
   62:    mangled:   St3mapISsiSt4lessISsESaISt4pairIKSsiEEE
   63:    demangled: std::map<std::string, int, std::less<std::string>, std::al
locator<std::pair<std::string const, int> > >
```