```
 1: // $Id: demangle.cpp,v 1.3 2019-02-14 15:48:36-08 - - $
 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 = 0;
21:    using deleter = void (*) (void*);
22:    unique_ptr<char,deleter> result {
23:       abi::__cxa_demangle (name, nullptr, nullptr, &status),
24:       std::free,
25:    };
26:    return status == 0 ? result.get() : name;
27: }
28:
29: class foo { };
30: class bar: foo { };
31: class baz: bar { };
32: template <typename T> class tmpl { T x; };
33:
34: template <typename type>
35: void print_demangled (const string &str) {
36:    type obj {};
37:    cout << str << " => " << sizeof obj << endl;
38:    cout << "   mangled:   " << typeid(obj).name() << endl;
39:    cout << "   demangled: " << demangle (obj) << endl;
40: }
41:
42: template <typename type>
43: void print_demangled (const string &str, const type& obj) {
44:    print_demangled<type> (str);
45:    cout << "   value: " << obj << endl;
46: }
47:
```

```
48:
49: #define DEMANGLE_T(X) print_demangled<X> (#X)
50: #define DEMANGLE_V(X) print_demangled (#X, X)
51: int main() {
52:    using map_string_int = map<string,int>;
53:    DEMANGLE_T (bool);
54:    DEMANGLE_T (char);
55:    DEMANGLE_T (signed char);
56:    DEMANGLE_T (unsigned char);
57:    DEMANGLE_T (short);
58:    DEMANGLE_T (signed short);
59:    DEMANGLE_T (unsigned short);
60:    DEMANGLE_T (int);
61:    DEMANGLE_T (signed int);
62:    DEMANGLE_T (unsigned int);
63:    DEMANGLE_T (long);
64:    DEMANGLE_T (signed long);
65:    DEMANGLE_T (unsigned long);
66:    DEMANGLE_T (long long);
67:    DEMANGLE_T (float);
68:    DEMANGLE_T (double);
69:    DEMANGLE_T (long double);
70:    DEMANGLE_T (size_t);
71:    cout << "\f" << endl;
72:    DEMANGLE_T (foo);
73:    DEMANGLE_T (bar);
74:    DEMANGLE_T (baz);
75:    DEMANGLE_T (tmpl<int>);
76:    DEMANGLE_T (vector<string>);
77:    DEMANGLE_T (vector<int>);
78:    DEMANGLE_T (list<vector<long>>);
79:    DEMANGLE_T (map_string_int);
80:    DEMANGLE_V (__FILE__);
81:    DEMANGLE_V (__LINE__);
82:    DEMANGLE_V (__DATE__);
83:    DEMANGLE_V (__TIME__);
84:    DEMANGLE_V (__func__);
85:    DEMANGLE_V (__PRETTY_FUNCTION__);
86:    return 0;
87: }
88:
89: //TEST// alias grind="valgrind --leak-check=full --show-reachable=yes"
90: //TEST// grind --log-file=demangle.log demangle >demangle.out 2>&1
91: //TEST// mkpspdf demangle.ps demangle.cpp* demangle.out demangle.log
92:
```

```
    1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting demangle.cpp
    2: checksource demangle.cpp
    3: ident demangle.cpp
    4: demangle.cpp:
    5:      $Id: demangle.cpp,v 1.3 2019-02-14 15:48:36-08 - - $
    6: cpplint.py.perl demangle.cpp
    7: Done processing demangle.cpp
    8: g++ -Wall -Wextra -Wpedantic -Wshadow -fdiagnostics-color=never -std=gnu
++2a -Wold-style-cast -g -O0 demangle.cpp -o demangle -lm
    9: rm -f demangle.o
   10: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished demangle.cpp
```

```
 1: bool => 1
 2:    mangled:   b
 3:    demangled: bool
 4: char => 1
 5:    mangled:   c
 6:    demangled: char
 7: signed char => 1
 8:    mangled:   a
 9:    demangled: signed char
10: unsigned char => 1
11:    mangled:   h
12:    demangled: unsigned char
13: short => 2
14:    mangled:   s
15:    demangled: short
16: signed short => 2
17:    mangled:   s
18:    demangled: short
19: unsigned short => 2
20:    mangled:   t
21:    demangled: unsigned short
22: int => 4
23:    mangled:   i
24:    demangled: int
25: signed int => 4
26:    mangled:   i
27:    demangled: int
28: unsigned int => 4
29:    mangled:   j
30:    demangled: unsigned int
31: long => 8
32:    mangled:   l
33:    demangled: long
34: signed long => 8
35:    mangled:   l
36:    demangled: long
37: unsigned long => 8
38:    mangled:   m
39:    demangled: unsigned long
40: long long => 8
41:    mangled:   x
42:    demangled: long long
43: float => 4
44:    mangled:   f
45:    demangled: float
46: double => 8
47:    mangled:   d
48:    demangled: double
49: long double => 16
50:    mangled:   e
51:    demangled: long double
52: size_t => 8
53:    mangled:   m
54:    demangled: unsigned long
```

```
   55:
   56: foo => 1
   57:    mangled:   3foo
   58:    demangled: foo
   59: bar => 1
   60:    mangled:   3bar
   61:    demangled: bar
   62: baz => 1
   63:    mangled:   3baz
   64:    demangled: baz
   65: tmpl<int> => 4
   66:    mangled:   4tmplIiE
   67:    demangled: tmpl<int>
   68: vector<string> => 24
   69:    mangled:   St6vectorISsSaISsEE
   70:    demangled: std::vector<std::string, std::allocator<std::string> >
   71: vector<int> => 24
   72:    mangled:   St6vectorIiSaIiEE
   73:    demangled: std::vector<int, std::allocator<int> >
   74: list<vector<long>> => 16
   75:    mangled:   St4listISt6vectorIlSaIlEESaIS2_EE
   76:    demangled: std::list<std::vector<long, std::allocator<long> >, std::a
llocator<std::vector<long, std::allocator<long> > > >
   77: map_string_int => 48
   78:    mangled:   St3mapISsiSt4lessISsESaISt4pairIKSsiEEE
   79:    demangled: std::map<std::string, int, std::less<std::string>, std::al
locator<std::pair<std::string const, int> > >
   80: __FILE__ => 13
   81:    mangled:   A13_c
   82:    demangled: char [13]
   83:    value: demangle.cpp
   84: __LINE__ => 4
   85:    mangled:   i
   86:    demangled: int
   87:    value: 81
   88: __DATE__ => 12
   89:    mangled:   A12_c
   90:    demangled: char [12]
   91:    value: Feb  3 2020
   92: __TIME__ => 9
   93:    mangled:   A9_c
   94:    demangled: char [9]
   95:    value: 17:09:08
   96: __func__ => 5
   97:    mangled:   A5_c
   98:    demangled: char [5]
   99:    value: main
  100: __PRETTY_FUNCTION__ => 11
  101:    mangled:   A11_c
  102:    demangled: char [11]
  103:    value: int main()
```

```
    1: ==7434== Memcheck, a memory error detector
    2: ==7434== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
    3: ==7434== Using Valgrind-3.14.0 and LibVEX; rerun with -h for copyright i
nfo
    4: ==7434== Command: demangle
    5: ==7434== Parent PID: 7433
    6: ==7434==
    7: ==7434==
    8: ==7434== HEAP SUMMARY:
    9: ==7434==     in use at exit: 0 bytes in 0 blocks
   10: ==7434==   total heap usage: 96 allocs, 96 frees, 3,092 bytes allocated
   11: ==7434==
   12: ==7434== All heap blocks were freed -- no leaks are possible
   13: ==7434==
   14: ==7434== For counts of detected and suppressed errors, rerun with: -v
   15: ==7434== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```