```cpp
 1: // $Id: inherit.cpp,v 1.7 2018-07-11 13:45:06-07 - - $
 2:
 3: // Simple example using inheritance.
 4:
 5: #include <cmath>
 6: #include <iostream>
 7: #include <memory>
 8: #include <sstream>
 9: #include <vector>
10: using namespace std;
11:
12: ////////////////////////////////////////////////////////////////
13: // class shape
14: ////////////////////////////////////////////////////////////////
15:
16: class shape {
17:    private:
18:       static size_t next_id;
19:       size_t id {++next_id};
20:    public:
21:       virtual double area() const = 0;
22:       virtual double circum() const = 0;
23:       virtual ostream& show (ostream&) const;
24:       friend ostream& operator<< (ostream&, const shape&);
25: };
26:
27: size_t shape::next_id {0};
28:
29: ostream& shape::show (ostream& out) const {
30:    return out << "shape(" << id << ")";
31: }
32:
33: ostream& operator<< (ostream& out, const shape& sh) {
34:    return sh.show (out);
35: }
36:
37: ////////////////////////////////////////////////////////////////
38: // class nothing
39: ////////////////////////////////////////////////////////////////
40:
41: class nothing: public shape {
42:    public:
43:       double area() const override { return 0; }
44:       double circum() const override { return 0; }
45:       ostream& show (ostream&) const override;
46: };
47:
48: ostream& nothing::show (ostream& out) const {
49:    return shape::show (out) << ": nothing ";
50: }
51:
```

```
52:
53: /////////////////////////////////////////////////////////////
54: // class circle
55: /////////////////////////////////////////////////////////////
56:
57: class circle: public shape {
58:    private:
59:       double radius {};
60:    public:
61:       circle (double r = 0): radius(r) {}
62:       double area() const override { return M_PI * pow (radius, 2); }
63:       double circum() const override { return 2 * M_PI * radius; }
64:       ostream& show (ostream&) const override;
65: };
66:
67: ostream& circle::show (ostream& out) const {
68:    return shape::show (out) << ": circle radius " << radius;
69: }
70:
71: /////////////////////////////////////////////////////////////
72: // class square
73: /////////////////////////////////////////////////////////////
74:
75: class square: public shape {
76:    private:
77:       double side {};
78:    public:
79:       square (double s): side(s) {}
80:       double area() const override { return pow (side, 2); }
81:       double circum() const override { return 4 * side; }
82:       ostream& show (ostream&) const override;
83: };
84:
85:
86: ostream& square::show (ostream& out) const {
87:    return shape::show (out) << ": square side " << side;
88: }
89:
```

```
 90:
 91:
 92: ///////////////////////////////////////////////////////////
 93: // main function
 94: ///////////////////////////////////////////////////////////
 95:
 96: using shape_ptr = shared_ptr<shape>;
 97: int main (int argc, char** argv) {
 98:     vector<string> args (&argv[1], &argv[argc]);
 99:     vector<shape_ptr> vsp;
100:     for (const auto& arg: args) {
101:         cout << arg << ":" << endl;
102:         istringstream argstream (arg);
103:         char type {}; double num {};
104:         argstream >> type >> num;
105:         shape_ptr obj = nullptr;
106:         switch (arg[0]) {
107:            case 'c': obj = make_shared<circle> (num); break;
108:            case 's': obj = make_shared<square> (num); break;
109:            default:  obj = make_shared<nothing>(); break;
110:         }
111:         vsp.push_back (obj);
112:     }
113:     for (const auto& item: vsp) {
114:         cout << *item << ", area " << item->area() << ", circum "
115:              << item->circum() << endl;
116:     }
117:     return 0;
118: }
119:
120: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
121: //TEST// grind inherit c c10 s s15 x >inherit.out 2>&1
122: //TEST// mkpspdf inherit.ps inherit.cpp* inherit.out
123:
```

```
    1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting inherit.cpp
    2: checksource inherit.cpp
    3: ident inherit.cpp
    4: inherit.cpp:
    5:       $Id: inherit.cpp,v 1.7 2018-07-11 13:45:06-07 - - $
    6: cpplint.py.perl inherit.cpp
    7: Done processing inherit.cpp
    8: g++ -g -O0 -Wall -Wextra -Wpedantic -Wshadow -fdiagnostics-color=never -
std=gnu++17 -Wold-style-cast inherit.cpp -o inherit -lm
    9: rm -f inherit.o
   10: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished inherit.cpp
```

```
 1: ==31710== Memcheck, a memory error detector
 2: ==31710== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al
.
 3: ==31710== Using Valgrind-3.14.0.GIT and LibVEX; rerun with -h for copyri
ght info
 4: ==31710== Command: inherit c c10 s s15 x
 5: ==31710==
 6: c:
 7: c10:
 8: s:
 9: s15:
10: x:
11: shape(1): circle radius 0, area 0, circum 0
12: shape(2): circle radius 10, area 314.159, circum 62.8319
13: shape(3): square side 0, area 0, circum 0
14: shape(4): square side 15, area 225, circum 60
15: shape(5): nothing , area 0, circum 0
16: ==31710==
17: ==31710== HEAP SUMMARY:
18: ==31710==     in use at exit: 0 bytes in 0 blocks
19: ==31710==   total heap usage: 22 allocs, 22 frees, 854 bytes allocated
20: ==31710==
21: ==31710== All heap blocks were freed -- no leaks are possible
22: ==31710==
23: ==31710== For counts of detected and suppressed errors, rerun with: -v
24: ==31710== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```