

```

1  // stack.cpp -- Stack member functions
2  #include "stack.h"
3  Stack::Stack()    // create an empty stack
4  {
5      top = 0;
6  }
7
8  bool Stack::isempty() const
9  {
10     return top == 0;
11 }
12
13 bool Stack::isfull() const
14 {
15     return top == MAX;
16 }
17
18 bool Stack::push(const Item & item)
19 {
20     if (top < MAX)
21     {
22         items[top++] = item;
23         return true;
24     }
25     else
26         return false;
27 }
28
29 bool Stack::pop(Item & item)
30 {
31     if (top > 0)
32     {
33         item = items[--top];
34         return true;
35     }
36     else
37         return false;
38 }
39
40
41
42
43
44 // stacker.cpp -- testing the Stack class
45 #include <iostream>
46 #include <cctype> // or ctype.h
47 #include "stack.h"
48 int main()
49 {
50     using namespace std;
51     Stack st; // create an empty stack
52     char ch;
53     unsigned long po;
54     cout << "Please enter A to add a purchase order,\n"
55          << "P to process a PO, or Q to quit.\n";
56     while (cin >> ch && toupper(ch) != 'Q')
57     {
58         while (cin.get() != '\n')
59             continue;
60         if (!isalpha(ch))

```

```

61     {
62         cout << '\a';
63         continue;
64     }
65     switch(ch)
66     {
67         case 'A':
68         case 'a': cout << "Enter a PO number to add: ";
69                 cin >> po;
70                 if (st.isfull())
71                     cout << "stack already full\n";
72                 else
73                     st.push(po);
74                 break;
75         case 'P':
76         case 'p': if (st.isempty())
77                     cout << "stack already empty\n";
78                 else {
79                     st.pop(po);
80                     cout << "PO #" << po << " popped\n";
81                 }
82                 break;
83     }
84     cout << "Please enter A to add a purchase order,\n"
85           << "P to process a PO, or Q to quit.\n";
86 }
87 cout << "Bye\n";
88 return 0;
89 }

```

```

94 // stock1.cpp SPA Stock class implementation with constructors, destructor added
95 #include <iostream>
96 #include "stock1.h"
97

```

```

98 // constructors (verbose versions)

```

```

99 Stock::Stock() // default constructor

```

```

100 {
101     std::cout << "Default constructor called\n";
102     std::strcpy(company, "no name");
103     shares = 0;
104     share_val = 0.0;
105     total_val = 0.0;
106 }

```

```

108 Stock::Stock(const char * co, int n, double pr)

```

```

109 {
110     std::cout << "Constructor using " << co << " called\n";
111     std::strncpy(company, co, 29);
112     company[29] = '\0';
113
114     if (n < 0)
115     {
116         std::cerr << "Number of shares can't be negative; "
117                   << company << " shares set to 0.\n";
118         shares = 0;
119     }
120     else

```

```

121         shares = n;
122         share_val = pr;
123         set_tot();
124     }
125     // class destructor
126     Stock::~Stock()           // verbose class destructor
127     {
128         std::cout << "Bye, " << company << "!\n";
129     }
130
131     // other methods
132     void Stock::buy(int num, double price)
133     {
134         if (num < 0)
135         {
136             std::cerr << "Number of shares purchased can't be negative. "
137                 << "Transaction is aborted.\n";
138         }
139         else
140         {
141             shares += num;
142             share_val = price;
143             set_tot();
144         }
145     }
146
147     void Stock::sell(int num, double price)
148     {
149         using std::cerr;
150         if (num < 0)
151         {
152             cerr << "Number of shares sold can't be negative. "
153                 << "Transaction is aborted.\n";
154         }
155         else if (num > shares)
156         {
157             cerr << "You can't sell more than you have! "
158                 << "Transaction is aborted.\n";
159         }
160         else
161         {
162             shares -= num;
163             share_val = price;
164             set_tot();
165         }
166     }
167
168     void Stock::update(double price)
169     {
170         share_val = price;
171         set_tot();
172     }
173
174     void Stock::show()
175     {
176         using std::cout;
177         using std::endl;
178         cout << "Company: " << company
179             << " Shares: " << shares << endl
180             << " Share Price: $" << share_val

```

```

181         << "   Total Worth: $" << total_val << endl;
182     }
183
184
185
186
187 // stock2.cpp -- improved version
188 #include <iostream>
189 #include "stock2.h"
190
191 // constructors
192 Stock::Stock()           // default constructor
193 {
194     std::strcpy(company, "no name");
195     shares = 0;
196     share_val = 0.0;
197     total_val = 0.0;
198 }
199
200 Stock::Stock(const char * co, int n, double pr)
201 {
202     std::strncpy(company, co, 29);
203     company[29] = '\0';
204
205     if (n < 0)
206     {
207         std::cerr << "Number of shares can't be negative; "
208                     << company << " shares set to 0.\n";
209         shares = 0;
210     }
211     else
212     {
213         shares = n;
214         share_val = pr;
215         set_tot();
216     }
217
218 // class destructor
219 Stock::~~Stock()         // quiet class destructor
220 {
221 }
222
223 // other methods
224 void Stock::buy(int num, double price)
225 {
226     if (num < 0)
227     {
228         std::cerr << "Number of shares purchased can't be negative. "
229                     << "Transaction is aborted.\n";
230     }
231     else
232     {
233         shares += num;
234         share_val = price;
235         set_tot();
236     }
237 }
238
239 void Stock::sell(int num, double price)
240 {
241     using std::cerr;

```

```

241     if (num < 0)
242     {
243         cerr << "Number of shares sold can't be negative. "
244             << "Transaction is aborted.\n";
245     }
246     else if (num > shares)
247     {
248         cerr << "You can't sell more than you have! "
249             << "Transaction is aborted.\n";
250     }
251     else
252     {
253         shares -= num;
254         share_val = price;
255         set_tot();
256     }
257 }
258
259 void Stock::update(double price)
260 {
261     share_val = price;
262     set_tot();
263 }
264
265 void Stock::show() const
266 {
267     using std::cout;
268     using std::endl;
269     cout << "Company: " << company
270         << " Shares: " << shares << endl
271         << " Share Price: $" << share_val
272         << " Total Worth: $" << total_val << endl;
273 }
274
275 const Stock & Stock::topval(const Stock & s) const
276 {
277     if (s.total_val > total_val)
278         return s;
279     else
280         return *this;
281 }
282
283
284
285 // usestok1.cpp -- use the Stock class
286 #include <iostream>
287 #include "stock1.h"
288
289 int main()
290 {
291     using std::cout;
292     using std::ios_base;
293     cout.precision(2); // ### format
294     cout.setf(ios_base::fixed, ios_base::floatfield); // ### format
295     cout.setf(ios_base::showpoint); // ### format
296
297     cout << "Using constructors to create new objects\n";
298     Stock stock1("NanoSmart", 12, 20.0); // syntax 1
299     stock1.show();
300     Stock stock2 = Stock ("Boffo Objects", 2, 2.0); // syntax 2

```

```

301     stock2.show();
302
303     cout << "Assigning stock1 to stock2:\n";
304     stock2 = stock1;
305     cout << "Listing stock1 and stock2:\n";
306     stock1.show();
307     stock2.show();
308
309     cout << "Using a constructor to reset an object\n";
310     stock1 = Stock("Nifty Foods", 10, 50.0);    // temp object
311     cout << "Revised stock1:\n";
312     stock1.show();
313     cout << "Done\n";
314     return 0;
315 }
316
317
318
319
320
321 // usestock2.cpp -- use the Stock class
322 // compile with stock2.cpp
323 #include <iostream>
324 #include "stock2.h"
325
326 const int STKS = 4;
327 int main()
328 {
329     using std::cout;
330     using std::ios_base;
331
332     // create an array of initialized objects
333     Stock stocks[STKS] = {
334         Stock("NanoSmart", 12, 20.0),
335         Stock("Boffo Objects", 200, 2.0),
336         Stock("Monolithic Obelisks", 130, 3.25),
337         Stock("Fleep Enterprises", 60, 6.5)
338     };
339
340     cout.precision(2);                                // ### format
341     cout.setf(ios_base::fixed, ios_base::floatfield); // ### format
342     cout.setf(ios_base::showpoint);                   // ### format
343
344     cout << "Stock holdings:\n";
345     int st;
346     for (st = 0; st < STKS; st++)
347         stocks[st].show();
348
349     Stock top = stocks[0];
350     for (st = 1; st < STKS; st++)
351         top = top.topval(stocks[st]);
352     cout << "\nMost valuable holding:\n";
353     top.show();
354
355     return 0;
356 }
357
358
359
360

```

```

361
362
363 // stocks.cpp -- the whole program
364 #include <iostream>
365 #include <cstring>
366
367 class Stock // class declaration
368 {
369 private:
370     char company[30];
371     int shares;
372     double share_val;
373     double total_val;
374     void set_tot() { total_val = shares * share_val; }
375 public:
376     void acquire(const char * co, int n, double pr);
377     void buy(int num, double price);
378     void sell(int num, double price);
379     void update(double price);
380     void show();
381 }; // note semicolon at the end
382
383 void Stock::acquire(const char * co, int n, double pr)
384 {
385     std::strncpy(company, co, 29); // truncate co to fit company
386     company[29] = '\0';
387     if (n < 0)
388     {
389         std::cerr << "Number of shares can't be negative; "
390                 << company << " shares set to 0.\n";
391         shares = 0;
392     }
393     else
394         shares = n;
395     share_val = pr;
396     set_tot();
397 }
398
399 void Stock::buy(int num, double price)
400 {
401     if (num < 0)
402     {
403         std::cerr << "Number of shares purchased can't be negative. "
404                 << "Transaction is aborted.\n";
405     }
406     else
407     {
408         shares += num;
409         share_val = price;
410         set_tot();
411     }
412 }
413
414 void Stock::sell(int num, double price)
415 {
416     using std::cerr;
417     if (num < 0)
418     {
419         cerr << "Number of shares sold can't be negative. "
420             << "Transaction is aborted.\n";

```

```

421     }
422     else if (num > shares)
423     {
424         cerr << "You can't sell more than you have! "
425             << "Transaction is aborted.\n";
426     }
427     else
428     {
429         shares -= num;
430         share_val = price;
431         set_tot();
432     }
433 }
434
435 void Stock::update(double price)
436 {
437     share_val = price;
438     set_tot();
439 }
440
441 void Stock::show()
442 {
443     using std::cout;
444     using std::endl;
445     cout << "Company: " << company
446         << " Shares: " << shares << endl
447         << " Share Price: $" << share_val
448         << " Total Worth: $" << total_val << endl;
449 }
450
451 int main()
452 {
453     using std::cout;
454     using std::ios_base;
455
456     Stock stock1;
457     stock1.acquire("NanoSmart", 20, 12.50);
458     cout.setf(ios_base::fixed);           // ### format
459     cout.precision(2);                    // ### format
460     cout.setf(ios_base::showpoint);       // ### format
461     stock1.show();
462     stock1.buy(15, 18.25);
463     stock1.show();
464     stock1.sell(400, 20.00);
465     stock1.show();
466     return 0;
467 }
468

```