Sample Assignment 3

Discussed on 2020-09-24
Not graded

Question 1 (Storing numbers in a queue)
Data structure `std::vector` is very popular; it allows `push_back(...)` method: you can add new elements at the end, but not at the beginning.
Double ended queue (*deque*) is another data structure (as fast and efficient as vector); it allows pushing elements at either end: `push_front(...)` and `push_back(...)`.

Source file **File1.cpp**:

```
1  #include <iostream>
2  #include <string>
3  #include <deque>
4
5  using namespace std;
6  int main() {
7    deque<int> queue1;
8    int n;
9
10   while (cin >> n) {
11     queue1.push_front(n);
12   }
13
14   string sep; // initialized as empty
15   for (int m : queue1) {
16       cout << sep << m; sep = ",";
17   }
18   cout << endl;
19
20   int head = queue1.at(0);
21   queue1.pop_front();
22   cout << head << " -> ";
23   for (int m : queue1) {
24       cout << m << ",";
25   }
26 }
```

Input file **input.txt**:

```
1 3 5
2 4 6
```

What is the output from **File1.cpp** on this input?

(A)

```
1,3,5,2,4,6
6 -> 4,2,5,3,1,
```

(B)

```
1,3,5,2,4,6
1 -> 3,5,2,4,6,
```

(C)

```
6,4,2,5,3,1
1 -> 3,5,2,4,6,
```

(D)

```
6,4,2,5,3,1
6 -> 4,2,5,3,1,
```

Question 2 (Reading Line by Line)

Source file **File2.cpp**:

```
#include <iostream>
#include <sstream>
#include <map>
#include <string>
#include <vector>

using namespace std;
int main() {
  map<int, vector<int>> mymap;

  vector<int> vect;
  string line;
  while (getline(cin, line)) {
    istringstream sstr(line);
    int n;
    while (sstr >> n) {
      vect.push_back(n);
    }

    mymap.insert(
      make_pair(vect.at(0), vect));
    vect.clear();
  }

  map<int,vector<int>>::iterator it=
      mymap.begin();
  while (it != mymap.end()) {
    cout << it ->first << ": ";
    for (int m : it -> second) {
      cout << m << ",";
    }
    it++; cout << endl;
  }
}
```

Input file **input.txt**:

```
1 3 5
7 8 9
2 4 6
```

Mark true/false statements about this code:

(A) The output is 3 lines

(B) The output is 1 line

(C) Iterator `it` visits pairs from `mymap` in the same order they were inserted.

(D) Iterator `it` visits pairs from `mymap` in a random order.

(E) Iterator `it` visits pairs from `mymap` in increasing order of the key (`it -> first`).

Question 3: This code inserts some elements in a `std::set`, then tries to find (element `5` is there, but `4` is not). Finally, we iterate over the set in two different ways: as in C++11 (for-loop syntax for an iterator) or an older construct with an explicit iterator.

Source file **File3.cpp**:

```
1  #include <iostream>
2  #include <set>
3
4  using namespace std;
5  int main() {
6    set<int> myset;
7    myset.insert(11);
8    myset.insert(13);
9    myset.insert(5);
10   myset.insert(7);
11   myset.insert(5);
12   bool b4,b5;
13   b4 = myset.find(4) == myset.end();
14   b5 = myset.find(5) == myset.end();
15   cout << "(b4,b5)=(" << b4 <<
16     "," << b5 << ")" << endl;
17   for (int u: myset) {
18       cout << u << "; ";
19   }
20   cout << endl;
21   set<int>::iterator it;
22   it = myset.begin();
23   while (it != myset.end()) {
24     cout << (*it) << "; ";
25     it++;
26   }
27 }
```

Mark which statements about this code are true/false.

(A) 1st line in output is `(b3,b4) = (0,1)`.

(B) Lines 17–19 and 23–26 iterate over `myset` in the same way.

(C) Iterator on Lines 17–19 visits elements in increasing order.

(D) Iterator on Lines 17–19 visits elements in random order.

Question 4 (Sets/Vectors with Custom Classes)
The code below does the task of EX02: it reads student data from STDIN; outputs the smallest and the largest student compared by age; or by height (if ages are equal).

Source file **File4.cpp**:

```
1  #include <iostream>
2  #include <iomanip>
3  #include <set>
4
5  using namespace std;
6  struct Student {
7    int age;
8    double height;
9    Student(int aa = 1, double hh = 1):
10           age(aa), height(hh) {}
11
12   friend istream &operator>>(
13     istream  &input, Student &S ) {
14       input >> S.age >> S.height;
15       return input;
16   }
17
18   friend ostream &operator<<(
19     ostream &output,
20         const Student &S ) {
21       output << "Student(" <<
22         S.age << "," << std::fixed <<
23         std::setprecision(5) <<
24                 S.height << ")";
25       return output;
26   }
27
28   friend bool operator<(
29     const Student &left,
30     const Student &right) {
31       return (left.age<right.age) ||
32       (left.age == right.height &&
33         left.height<right.height);
34     }
35 };
36
37 int main() {
38     int n; cin >> n;
39     set<Student> myset;
40     for (int i = 0; i < n; i++) {
41         Student student;
42         cin >> student;
43         myset.insert(student);
44     }
45     cout << *(myset.begin()) << endl;
46     cout << *(--myset.end());
47     return 0;
48 }
```

Write short answers to these questions:

(A) What happens, if Line 43 is rewritten as follows: `cout << *(myset.end());`

(B) How would you overload the comparison operation, if you only look at `age`, but order by age alphabetically (as in a dictionary). Namely, age "17" comes before age "2" (since digit "1" alphabetically precedes digit "2"). *(Just show how the Lines 31–33 would look, if you order alphabetically.)*

(C) Assume that the class `Student` had a destructor. On which line of your code (if at all) is it called, when you read the input containing data for a few students and insert them all in a set.

(D) Assume that you want to use `std::vector` instead of a set; and output the first and the last element you inserted into a vector. What would you write on Line 43, Line 45 and 46? (How to add something to an end of a vector? How to get the first element? The last element?)

Solutions

Question 1. Answer: D

Question 2. Answer: `true`, `false`, `false`, `false`, `true`.

Question 3. Answer: `false`, `true`, `true`, `false`.

Question 4. TBD