Object-Oriented Programming Language

11/21/2018

Homework Assignment No. 9

Due 11:59 pm, Thursday November 29, 2018

Late submission within 24 hours: score*0.9;

Late submission within one week: score*0.8.

The solutions will be posted after one week of the due date.

(Total 60%)

1. **(60%)**: Use the following skeleton code to test the performances of sequential containers:

```
int main(){
  const int UPPER_LIMIT = 10000000;
  clock_t t1, e1, t2, e2, t3, e3;

  t1 = clock();
  //vector<int> origion(UPPER_LIMIT,-1);
  e1 = clock();
  cout << "first time range: " << (e1 - t1)*1000.0/CLOCKS_PER_SEC
<< endl;

  t2 = clock();
  copy(origion.begin(), origion.end(),
back_inserter(destination1));
  e2 = clock();
  cout << "first time range: " << (e2 - t2)*1000.0/CLOCKS_PER_SEC
<< endl;

  t3 = clock();
  copy(origion.begin(), origion.end(),
back_inserter(destination2,destination2.begin()));
  e3 = clock();
  cout << "first time range: " << (e3 - t3)*1000.0/CLOCKS_PER_SEC
<< endl;

}
```

In the code fragments, the `ctime` library provides the utilities for time measuring. The `clock()` function represents clock ticks from the processor. The value returned is expressed in *clock ticks*, which are units of time of a constant but system-specific length (with a relation of CLOCKS_PER_SEC *clock ticks* per second).

You should compare the performance of the following setups:

```
vector<int> origion(UPPER_LIMIT,-1);
```

```
vector<int> destination1;
copy(origion.begin(), origion.end(),
back_inserter(destination1));
```

```
vector<int> destination2;
copy(origion.begin(), origion.end(),
             inserter(destination2,destination2.begin())
);
```

and also `front_inserter` for the appropriate containers.

You will discover in this HW problem the performance of different STL containers, and also the performance of their insertion, as well as inserting at different positions (front, back).

You will write a report on your findings, and upload the report in pdf format along with your C++ files in a zipped file.