Question: We are going to dive deep into a few of the stl containers bec...

We are going to dive deep into a few of the stl containers because it is cool. But in real life the whole point of stl is to keep you from having to write them yourselves. So the first data structure homework is about *using* stl.

List, Vector, Stack, Queue, Set, Map, UnorderedMap, PriorityQueue

1. Make a Student class that has Name, GPA, and StudentDebt as properties
2. In main, make a List of ints. Add some numbers to it
3. Make a Vector of Students. Add some Student objects to it
4. Make a Stack of Student Pointers. Push and pop some dynamically created Students to it. Make sure you don't leak memory
5. Make a Set of ints. Add some numbers to it
6. Make a Map of strings to ints. Put some movie titles and the year they cam out in it
7. Do #6 with an unordered map

And then the hard ones

1. Make a priority queue that uses a Student's GPA for the sort criteria. Add some Student objects and show that the next one popped is the highest GPA
2. Make a priority queue that uses a Student's Debt for the sort criteria. Add some Student **pointers** for dynamically created Students and show the next one popped is the one with the least debt.

Show transcribed image text

Expert Answer

AshishShinde answered this
220 answers

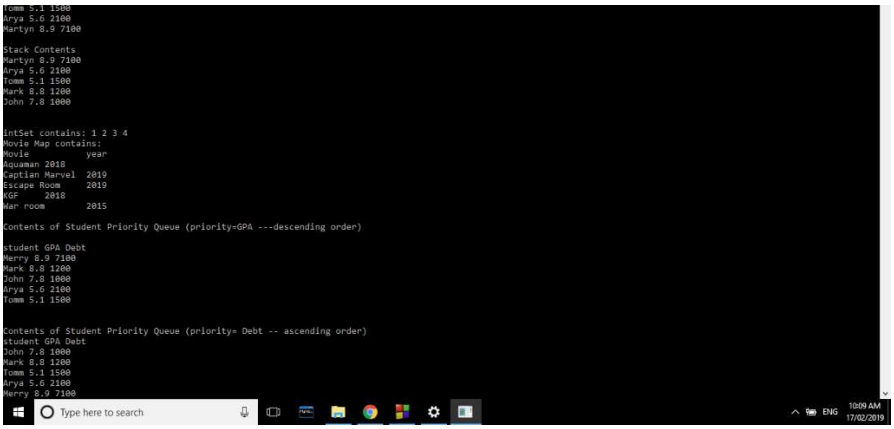
Was this answer helpful? 1 0

```
#include <iostream>
#include<list>
#include<vector>
#include<stack>
#include <set>
#include<map>
//#include <unordered_map>
#include <queue>
#include<iterator>
using namespace std;

class Student{
public:
string name;
double StudentDebt;
double GPA;
Student(){}
Student(string a,double b,double c)
{
name=a;
GPA=b;
StudentDebt=c;
}
void print()
{
cout<<name<<" "<<GPA<<" "<<StudentDebt<<endl;
}

};////class

// operator < overloading
bool operator < ( const Student& p1,const Student& p2)
{
// this will return true when second Student
// has greater GPA.
return p1.GPA < p2.GPA;
}
```

```
bool operator()(const Student* a, const Student* b)
{
return (a->StudentDebt > b->StudentDebt);
}
};
//-----------------------------------------
int main()
{
///declare list of int
list<int> intList;
///add integers to list at backward
for(int i=0;i<5;i++)
intList.push_back(i);
///add integers to front of list
for(int i=5;i<10;i++)
intList.push_front(i);

///declare iterator to display output
cout<<" List = ";
list <int> :: iterator it;
for(it = intList.begin(); it != intList.end(); ++it)
cout << " " << *it;


//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------

/// 3. create vector of Students
vector<Student> studentVector;
///adding student objects into it
studentVector.push_back( Student("John",7.8,1000));
studentVector.push_back( Student("Mark",8.8,1200));
studentVector.push_back( Student("Tomm",5.1,1500));
studentVector.push_back( Student("Arya",5.6,2100));
studentVector.push_back( Student("Martyn",8.9,7100));

///display contents of vector
cout<<" vector =";
cout<<" Name GPA Debt ";
for(int i=0;i<studentVector.size();i++)
{
studentVector.at(i).print();
}

/// 4. Stack of Student pointers
stack<Student*> studentStack;
///add student pointers dynamically
studentStack.push( new Student("John",7.8,1000));
studentStack.push( new Student("Mark",8.8,1200));
studentStack.push( new Student("Tomm",5.1,1500));
studentStack.push( new Student("Arya",5.6,2100));
studentStack.push( new Student("Martyn",8.9,7100));

///display Stack
cout<<" Stack Contents ";
while (!studentStack.empty())
{
studentStack.top()->print();
studentStack.pop();
}
cout << ' ';

//-----------------------------------------------
// 5 set of ints
set <int> intSet;
set <int>::iterator intSetIterator;

intSet.insert(1);
intSet.insert(2);
intSet.insert(2); //2 will be inserted only once
intSet.insert(3);
intSet.insert(4);

cout << " intSet contains:";
for (intSetIterator = intSet.begin(); intSetIterator!=intSet.end(); ++intSetIterator)
cout << ' ' << *intSetIterator;


/// 6) .Map string(movieName) to int (year)
```

```
movieMap.insert(pair<string,int>("KGF",2018));
movieMap.insert(pair<string,int>("Aquaman",2018));
movieMap.insert(pair<string,int>("Escape Room",2019));
movieMap.insert(pair<string,int>("Captian Marvel",2019));
movieMap.insert(pair<string,int>("War room",2015));

cout<<" Movie Map contains: ";
cout<<" Movie year";
for(movieItarator=movieMap.begin() ; movieItarator !=movieMap.end() ; movieItarator++)
{
cout<<" "<<movieItarator->first;
cout<<" "<<movieItarator->second;
}
//uncomment Unordered Map
/*
/// 7 Unordered map
unordered_map<string, int> movie;
unordered_map<string, int>::iterator movieItr;
// inserting values by using [] operator
movie["KGF"] = 2018;
movie["Aquaman"]=2018;
movie["Escape room"]=2019;
movie["captian Marvel"]=2019;
movie["war room"]=2015;

for(movieItr=movie.begin() ; movieItr!=movie.end() ; movieItr++)
{
cout<<" "<<movieItr->first;
cout<<" "<<movieItr->second;
}
*/
/// Priority Queue
priority_queue <Student> studentPQ;
///add students into PQ
studentPQ.push( Student("John",7.8,1000));
studentPQ.push( Student("Mark",8.8,1200));
studentPQ.push( Student("Tomm",5.1,1500));
studentPQ.push( Student("Arya",5.6,2100));
studentPQ.push( Student("Merry",8.9,7100));

///display PQ in descending order of GPA
cout<<" Contents of Student Priority Queue (priority=GPA ---descending order) ";
cout<<" student GPA Debt ";
while (!studentPQ.empty()) {
Student p = studentPQ.top();
studentPQ.pop();
p.print();
}

/// Priority Queue for Student Pointers using comaparator to compare Debt of student
priority_queue <Student*,vector<Student*>,Comparator> studentPtrPQ;

/// insert Student pointers into PQ
studentPtrPQ.push( new Student("John",7.8,1000));
studentPtrPQ.push( new Student("Mark",8.8,1200));
studentPtrPQ.push( new Student("Tomm",5.1,1500));
studentPtrPQ.push( new Student("Arya",5.6,2100));
studentPtrPQ.push( new Student("Merry",8.9,7100));
///display PQ in ascending order of Debt
cout<<" Contents of Student Priority Queue (priority= Debt -- ascending order)";
cout<<" student GPA Debt ";
while (!studentPtrPQ.empty()) {
Student* p = studentPtrPQ.top();
studentPtrPQ.pop();
p->print();
}

return 0;
}
```

OUTPUT



Comment

Up next for you in Computer Science

link
http://cs.ucsb.edu/~cspens...

See answer

C++ language List, Vector, Stack, Queue, Set, Map, UnorderedMap, PriorityQueue Make a Student class that has Name,
GPA, and StudentDebt as properties In main, make

See answer

See more questions for subjects you study

Questions viewed by other students

Q: I need help with understanding my c++ hw. Its about stl containers We are going to dive deep into a few of the stl containers because it is cool. But in real life the whole point of stl is to keep you from having to write them yourselves. So the first data structure homework is about using stl. List, Vector, Stack, Queue, Set, Map, UnorderedMap, PriorityQueue -Make a Student class...

A: See answer

Q: List, Vector, Stack, Queue, Set, Map, UnorderedMap, PriorityQueue Make a Student class that has Name, GPA, and StudentDebt as properties In main, make a List of ints. Add some numbers to it Make a Vector of Students. Add some Student objects to it Make a Stack of Student Pointers. Push and pop some dynamically created Students to it. Make sure you don't leak memory Make a Set of...

A: See answer

Show more

CHEGG NETWORK

CUSTOMER SERVICE