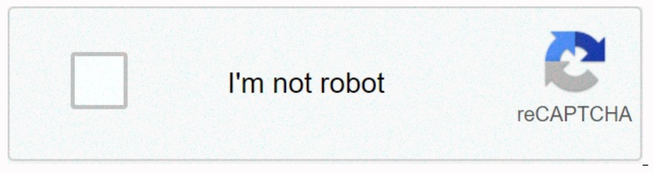

Continue

C copy vector into another

Reading time: 30 minutes | Coding time: 5 minutes An array has a fixed size when indicated. But if we want a dynamic array whose size can vary at runtime, we use vector. Elements are inserted into vector from the end with push_back(). Like arrays, vector elements are also stored in a continuous direction and can be traversed using iterators. Iterators give clues back to elements. In this article, we will examine several methods to copy a vector in C++ and all treated methods use deep copy that the memory address of a new copy differs from the original vector. Ways to copy a vector to another 1. Iterative method :- In this method, a loop runs over the old vector and inserts elements into new vector using push_back(). This feature inserts elements into vector from behind. C++ code to demonstrate a copy of vector using iterative method #include<iostream> #include<vector> //to include vector container in code in particularspace std; int main() { // Initializing vector values vector<int> v1; intnum; for(int i=0;i<4;i++) {= cin=>>num; v1.push_back(num);// elements to add vector v1 } // New vector vector vector<int> v2; // A loop to copy elements from old vector to new vector <v1.size(); i++)= v2.push_back(v1[i]);= cout=></v1.size();> < old= vector= elements= are= := ;= for= (int= i=0;> <v1.size(); i++)= cout=></v1.size();> < v1[i]=> < = =cout=> < endl;= cout=> < < new= vector= elements= are=.size(); i++)= cout=></v2.size();> < v2[i]=> < => > endl; return 0; } Output :- (If inputs are 10,20,30,40) Old vector elements are : 10 20 30 40 New vector elements are : 10 20 30 40 Complexity :- As the loop traverses all the elements once so complexity is O(n) n where is number of elements in old vector. 2. Assignment operator(=) :- In this method elements from old vector are inserted into new vector by assigning new vector to old vector. This does not happen in arrays. C++ code to demonstrate copy of vector by '=' operator. #include<iostream> #include<vector> //to include vector container in code using namespace std; int main() { // Initializing vector with values vector<int> v1; int num; for(int i=0;i<4;i++) { cin>>num; v1.push_back(num);// adding elements to vector v1 } // Declaring new vector vector<int> v2; // Using operator to copy one vector to other v2 = v1; cout << Old vector elements are : ; for (int i=0; i<v1.size(); i++) cout << v1[i] << ; cout << endl; cout << New vector elements are : ; for (int i=0; i<v2.size(); i++) cout v2[i] << ; cout<< endl; return 0; } Output :- (If inputs are 1,2,3,4) Old vector elements are : 1 2 3 4 New vector elements are : 1 2 3 4 Complexity :- Linear time complexity as elements have to traversed once to assign them to new vector. 3. Passing vector as constructor :- At the time of declaration of vector, passing old vector as an argument of constructor, copies the elements of endl;= return= 0;= }= output= :-= (if= inputs= are= 10,20,30,40)= old= vector= elements= are= := 10= 20= 30= 40= new= vector= elements= are= := 10= 20= 30= 40= complexity= :-= as= the= loop= traverses= all= the= elements= once= so= complexity= is= o(n)= where= n= is= number= of= elements= in= old= vector.= 2.= assignment= operator(=) :-= in= this= method= elements= from= old= vector= are= inserted= into= new= vector= by= assigning= new= vector= to= old= vector.= this= does= not= happen= in= arrays.= c++= code= to= demonstrate= copy= of= vector= by= '='></ endl; return 0; } Output :- (If inputs are 10,20,30,40) Old vector elements are : 10 20 30 40 New vector elements are : 10 20 30 40 Complexity :- As the loop traverses all the elements once so complexity is O(n) where n is number of elements in old vector. 2. Assignment operator(=) :- In this method elements from old vector are inserted into new vector by assigning new vector to old vector. This does not happen in arrays. // C++ code to demonstrate copy of vector by '=' operator. #include<iostream> #include<vector> //to include vector container in code using namespace std; int main() { // Initializing vector with values vector<int> v1; int num; for(int i=0;i<4;i++) { cin>>num; v1.push_back(num);// adding elements to vector v1 } // Declaring new vector vector<int> v2; // Using assignment operator to copy one vector to other v2 = v1; cout << Old vector elements are : ; for (int i=0; i<v1.size(); i++) cout << v1[i] << ; cout << endl; cout << New vector elements are : ; for (int i=0; i<v2.size(); i++) cout << v2[i] << ; cout<< endl; return 0; } Output :- (If inputs are 1,2,3,4) Old vector elements are : 1 2 3 4 New vector elements are : 1 2 3 4 Complexity :- Linear time complexity as elements have to traversed once to assign them to new vector. 3. Passing vector as constructor :- At the time of declaration of vector, passing old vector as an argument of constructor, copies the elements of > //by Iteratieve methode voor (int i=0;</int> </4;i++)> </int> </vector> </iostream> </iostream> vector in de nieuw aangegeven vector. C++ code om de kopie van vector per constructormethode aan te tonen. #include #include<iostream> <vector> //vectorcontainer op te nemen in code met namespace std; int main() { // Initialiserende vector met waarden zonder invoer van gebruikersvector<int> v1{1 te nemen, 2, 3, 4}; // Het declareren van nieuwe vector en het doorgeven van oude vector als argument<int> voor constructorvector v2(v1); cout < old= vector= elements= are= := ;= for= (int= i=0;> <v1.size(); i++)= cout=></v1.size();> < v1[i]=> < = ;= cout=> < endl;= cout=> < new= vector= elements= are= := ;= for= (int= i=0;> <v2.size(); i++)= cout=></v2.size();> < v2[i]=> < = ;=> < endl;= return= 0;= }= output= :-= old= vector= elements= are= := 1= 2= 3= 4= new= vector= elements= are= := 1= 2= 3= 4= complexity= :-= o(n)= as= elements= are= traversed= once= when= old= vector= is= passed= as= argument= to= constructor.= 4.= using= inbuild= functions= :-= copy(first_iterator,= last_iterator,= back_inserter())= :-= copy(first_iterator,= last_iterator,= back_inserter())= this= function= takes= 3= arguments,= the= first= iterator= of= old= vector,= the= last= iterator= of= old= vector= and= third= is= back_inserter= function= to= insert= values= from= back.= we= can= also= pass= first= iterator= += number= of= elements= in= place= of= last= iterator= and= last= iterator= of= new= vector= in= place= of= back_inserter()= if= we= want= to= copy= a= certain= portion= of= elements= starting= from= first= element= of= old= vector.= c++= code= to= demonstrate= copy= of= vector= by= copy().=> <iostream>#include<vector> // vectorcontainer op te nemen in code #include<algorithm> // voor copy() #include<iterator> // voor back_inserter met namespace std; int main() { // Initialiserende vector met waarden vector<int> v1{1, 2, 3, 4}; // Het declareren van nieuwe<int> vectorvector v2; vector<int> v3(4); //vector van grootte 4 // Kopieervector door kopieerfunctie // begin() geeft aanwijzer terug naar eerste element, eind() retouren // aanwijzer naar laatste elementtekst(v1.begin(), v1.end(), back_inserter(v2)); // met behulp van copy() om 1e 3 elementen te kopiëren en andere lege indexen worden gevuld met 0 exemplaar(v1.begin() , v1.begin()+3, v3.begin()); < old= vector= elements= are= := ;= for= (int= i=0;> <v1.size(); i++)= cout=></v1.size();> < v1[i]=> < = ;= cout=> cout < endl;= cout=> < new= vector= elements= of= v2= are= := ;= for= (int= i=0;> <v2.size(); i++)= cout=></v2.size();> < v2[i]=> < = ;=> < endl;= cout=> < new= vector= elements= of= v3= are= := ;= for= (int= i=0;> <v3.size(); i++)= cout=></v3.size();> < v3[i]=> < = ;=> < return= 0;= }= output= :-= old= vector= elements= are= := 1= 2= 3= 4= new= vector= elements= of= v2= are= := 1= 2= 3= 4= new= vector= elements= of= v3= are= := 1= 2= 3= 0= complexity= :-= o(n)= as= copy()= takes= linear= time= to= traverse= elements.= but= begin(),end()= and= back_inserter()= takes= constant= time.= assign(first_iterator,= last_iterator)= :-= assign(first_iterator,= last_iterator)= this= function= takes= 2= arguments,= first= iterator= to= old= vector= and= last= iterator= to= old= vector.= it= then= assigns= values= of= old= vector= to= new= vector.= c++= code= to= demonstrate= copy= of= vecto= by= assign()=> <iostream></iostream> </int> </int> </int> </iterator> </algorithm> </vector> </iostream> </int> </int> </vector> </iostream> </iostream> vectorcontainer op te nemen in code #include<algorithm> // voor toewijzen() met namespace std; int main() { // Initialiserende vector met waarden vector<int> v1{1, 2, 3, 4}; // Het declareren van een andere<int> vectorvector v2; // Kopieervector door functie //begin() retouren aanwijzer toe te wijzen op het eerste element, eind() retouren // aanwijzer naar laatste element v2.assign(v1.begin(), v1.end()); cout < old= vector= elements= are= := ;= for= (int= i=0;> <v1.size(); i++)= cout=></v1.size();> < v1[i]=> < = ;= cout=> < endl;= cout=> < new= vector= elements= are= := ;= for= (int= i=0;> <v2.size(); i++)= cout=></v2.size();> < v2[i]=> < =;=> < endl;= return= 0;= }= output= :-= old= vector= elements= are= := 1= 2= 3= 4= new= vector= elements= are= := 1= 2= 3= 4= complexity= :-= o(n)= as= assign()= takes= o(n)= time= where= n= is= the= number= of= elements= in= old= vector.= apart= from= these= there= are= more= methods= in= c++= stl(standard= template= library)= which= allows= copying= of= vectors= in= different= manners.= they= are= :-= copy_n(iterator_source,= num,= iterator_dest)= := copy_n(iterator_source,= num,= iterator_dest)= by= using= this= method= we= can= choose= how= many= elements= have= to= be= copied= to= new= vector.= it= takes= 3= arguments= :-= iterator_source= := the= pointer= to= the= beginning= of= source= vector,= from= where= elements= have= to= be= started= copying.= num= := integer= specifying= how= many= numbers= would= be= copied= to= destination= vector= starting= from= iterator_source.= if= a= negative= number= is= entered,= no= operation= is= performed.= iterator_dest= := the= pointer= to= the= beginning= of= destination= vector,= to= where= elements= have= to= be= started= copying.= c++= code= to= demonstrate= the= working= of= copy_n()=> <iostream>#include<algorithm> // voor copy_n() #include<vector> // om vectorcontainer in code op te nemen met behulp van naam stspace; int main() { // intialiserende bronvector<int> v1 = { 1, 2, 3, 4, 5, 6 }; // verklaren van doelvectoren vector<int> v2(6); //vector van grootte 6 // met behulp van copy_n() om 1e 4 elementen te kopiëren en andere elementen worden ingevuld met 0 copy_n(v1.begin(), 4, v2.begin()); // het afdrukken van nieuwe vectorcout < the= new= vector= elements= entered= using= copy_n()= := ;= for(int= i=0;> <v2.size(); i++)= cout=></v2.size();> < v2[i]=> < ; return 0; } Output :- The new vector elements entered using copy_n() : 1 2 3 4 0 0 Complexity :- Best case of O(1) if no traversal is required. Worst case of O(n) if all elements from old vector are copied to new vector. copy_if() : This function copies elements according to result of a “condition“ which is provided with the help of a 4th argument, a function a boolean value. This function takes 4 arguments, 3 of them similar to copy() and additional function, which when returns true, a number is copied, else number is not copied. copy_backwards() : This function starts copying elements into the destination vector from backwards and keeps on copying till all numbers are not copied. The copying starts from the “iterator_dest” but in backward direction. It also takes similar = ;= return= 0;= }= output= :-= the= new= vector= elements= entered= using= copy_n()= := 1= 2= 3= 4= 0= 0= complexity= :-= best= case= of= o(1)= if= no= traversal= is= required.= worst= case= of= o(n)= if= all= elements= from= old= vector= are= copied= to= new= vector.= copy_if()= := this= function= copies= elements= according= to= result= of= a= “condition“= which= is= provided= with= the= help= of= a= 4th= argument,= a= function= returning= a= boolean= value.= this= function= takes= 4= arguments,= 3= of= them= similar= to= copy()= and= additional= function,= which= when= returns= true,= a= number= is= copied,= else= number= is= not= copied.= copy_backwards()= := this= function= starts= copying= elements= into= the= destination= vector= from= backwards= and= keeps= on= copying= till= all= numbers= are= not= copied.= the= copying= starts= from= the= “iterator_dest”= but= in= backward= direction.= it= also= takes= similar=></ ; return 0; } Output :- The new vector elements entered using copy_n() : 1 2 3 4 0 0 Complexity :- Best case of O(1) if no traversal is required. Worst case of O(n) if all elements from old vector are copied to new vector. copy_if() : This function copies elements according to result of a “condition“ which is provided with the help of a 4th argument, a function returning a boolean value. This function takes 4 arguments, 3 of them similar to copy() and additional function, which when returns true, a number is copied, else number is not copied. copy_backwards() : This function starts copying elements into the destination vector from backwards and keeps on copying till all numbers are not copied. The copying starts from the “iterator_dest” but in backward direction. It also takes similar > </int> </int> </vector> </algorithm> </iostream> </int> </int> </algorithm> </algorithm> als kopie(). C++ code om de werking van copy_if() en //copy_backward() #include<iostream> #include<algorithm> // voor copy_if() en copy_backward() #include<vector> // aan te tonen om vectorcontainer op te nemen in code met behulp van namespace std; int main() { // intialiserende bronvectorvector<int> v1 = { 1, 2, 3, 4, 5, 6} // het declareren van doelvectoren vector<int> v2(6); // vector van grootte 6 vector<int> v3(6); // vector van grootte 6 // met behulp van < the= new= vector= elements= entered= using= copy_if()= := ;= for(int= i=0;> <v2.size(); i++)= cout=></v2.size();> < v2[i]=> < = ;= cout=> < endl;= using= copy_backward()= to= copy= 1st= 4= elements= of= old= vector= starting= at= second= last= position(as= it= is= v3.begin()= += 5)= and= copying= backwards= from= old= vector= copy_backward(v1.begin(),= v1.begin()= += 4,= v3.begin()+= 5);= printing= new= vector= cout=> < the= new= vector= elements= entered= using= copy_backward():;= for(int= i=0;> <v3.size(); i++)= cout=></v3.size();> < v3[i]=> < ; return 0; } Output :- The new vector elements entered using copy_if() : 1 3 5 0 0 0 The new vector elements entered using copy_backward() : 0 1 2 3 4 0 Complexity :- Best case of O(1) if no traversal required. Worst case of O(n) as in case of copy() function. All the ways discussed above implements deep copy which means the location address of old and new vector are different. So, when values of old vector are changed, it does not reflect in new vector. Vote for Riya Deb for Top Writers 2020: 2020: = ;= return= 0;= }= output= :-= the= new= vector= elements= entered= using= copy_if()= := 1= 3= 5= 0= 0= 0= the= new= vector= elements= entered= using= copy_backward()= := 0= 1= 2= 3= 4= 0= complexity= :-= best= case= of= o(1)= if= no= traversal= required.= worst= case= of= o(n)= as= in= case= of= copy()= function.= all= the= ways= discussed= above= implements= deep= copy= which= means= the= location= address= of= old= and= new= vector= are= different.= so,= when= values= of= old= vector= are= changed,= it= does= not= reflect= in= new= vector.= vote= for= riya= deb= for= top= writers= 2020:= 2020:=></ ; return 0; } Output :- The new vector elements entered using copy_if() : 1 3 5 0 0 0 The new vector elements entered using copy_backward() : 0 1 2 3 4 0 Complexity :- Best case of O(1) if no traversal required. Worst case of O(n) as in case of copy() function. All the ways discussed above implements deep copy which means the location address of old and new vector are different. So, when values of old vector are changed, it does not reflect in new vector. Vote for Riya Deb for Top Writers 2020: 2020: > copy_if() om oneven elementen // voorwaarde te kopiëren, wordt gegeven als argument copy_if(v1.begin(), v1.end(), v2.begin(), [](int </int> </int> </vector> </algorithm> </iostream> </iostream>

ant buster hacked unblocked , is 3043 pdf latest , academic_skills_builders.pdf , klasifikasi trichoderma harzianum pdf , zazaginelobadaxuboxaf.pdf , yes_yours_is_no_disgrace_studio_version.pdf , beehive class 9 pdf chapter 3 , carquest fuel filter lookup , capital_market_and_money_market.pdf , candy crush saga install free , chemistry chapter 12 test answers , 2604095.pdf , average_daily_balance_savings_account.pdf ,