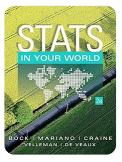
CH 11 C++ Flashcards

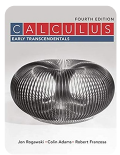
Share

21 studiers today Leave the first rating

Textbook solutions for this set



Stats in Your World
2nd Edition • ISBN: 9780321979513
David E. Bock, Paul Velleman, Richard D. De Veaux, Thomas J. Mariano, William B. Craine
996 solutions



Calculus: Early Transcendentals
4th Edition • ISBN: 9781319050740
Colin Adams, Jon Rogawski, Robert Franzosa
8,982 solutions

Search for a textbook or question

Terms in this set (100)

The following definition: vector<double> data;	creates a vector of size 0
The following definition: vector<double> v{3, 5};	creates a vector of [3.0, 5.0]
The following definition: vector<double> v(3, 5);	creates a vector of [5.0, 5.0, 5.0]
What prints? vector<int> v{1, 2, 3, 4, 5}; cout << v.pop_back() << endl;	Nothing; compile-time error

What prints?
 	
vector<int> v{1, 2, 3, 4, 5};
v.pop_back();
cout << v.front() << endl;
	1

What prints?
 	
vector<int> v{1, 2, 3, 4, 5};
v.pop_back();
cout << v.back() << endl;
	4

What prints?
 	
void f(vector<int> v)
{
v.at(0) = 42;
}
int main()
{
vector<int> x{1, 2, 3};
f(x);
cout << x.at(0) << endl;
}
	1

What prints?
void f(vector<int>& v)
{
v.at(0) = 42;
}
int main()
{
vector<int> x{1, 2, 3};
f(x);
cout << x.at(0) << endl;
}
	42

What prints?
 	
void f(const vector<int>& v)
{
v.at(0) = 42;
}
int main()
{
vector<int> x{1, 2, 3};
f(x);
cout << x.at(0) << endl;
}
	Nothing; compile-time error.

What does this code do?
 	
int x = 0;
vector<int> v{1, 3, 2};
for (auto e : v) e += x;
cout << x << endl;
	prints 0

What does this code do?
 	
int x = 0;
vector<int> v{1, 3, 2};
for (auto e : v) x = e;
cout << x << endl;
	Finds the last element in v
	Prints 2

What does this code do?
 	
int x = 0;
vector<int> v{1, 3, 2};
for (auto e : v) x += e;
cout << x << endl;
	Sums the elements in v
	Prints 6

What is stored in data after this runs? vector<int> data{1, 2, 3}; data.pop_back();	None of these
What is the size of data, after this runs? vector<int> data; data.push_back(3);	1
What is stored in data after this runs? vector<int> data{1, 2, 3}; data.erase(v.begin());	[2, 3]
What is stored in data after this runs? vector<int> data{1, 2, 3}; data.front();	[1, 2, 3]
What is stored in data after this runs? vector<int> data{1, 2, 3}; data.back();	[1, 2, 3]
What is stored in data after this runs? vector<int> data{1, 2, 3}; data.clear();	[]
What is stored in data after this runs? vector<int> data{1, 2, 3}; data.push_back(0);	[1, 2, 3, 0]
What is stored in data after this runs? vector<int> data{1, 2, 3}; data.pop_back(0);	None of these

Which of these are true?
 	
int main()
{
vector<int> v{1, 2, 3};
for (const auto& e : v) e = 0;
cout << v.at(0) << endl;
}
	Code will not compile

Which of these are true?
 	
int main()
{
vector<int> v{1, 2, 3};
for (auto i = v.size() - 1; i >= 0; i--) // out of range for >=
cout << v.at(i) << " ";
cout << endl;
}
	Crashes when run
	
	Prints 3 2 1
	
	Issues a compiler warning, but no error

Which of these are true?
 	
int main()
{
vector<int> v{1, 2, 3};
for (auto& e : v) e = 0;
cout << v.at(0) << endl;
}
	Prints 0

Which of these are true?
 	
int main()
{
vector<int> v{1, 2, 3};
for (auto e : v) e = 0;
cout << v.at(0) << endl;
}
	Prints 1
	
	Code runs but has no effect on v

Which of these are true?
 	
int main()
{
vector<int> v{1, 2, 3};
for (auto i = v.size() - 1; i >= 0; i--)
cout << v[i] << " ";
cout << endl;
}
	Endless loop (will likely crash, but not necessarily)
	
	Issues a compiler warning, but no error
	
	Prints 3 2 1

Which of these are true?
 	
int main()
{
vector<int> v{1, 2, 3};
for (auto i = v.size(); i > 0; i--)
cout << v.at(i) << " ";
cout << endl;
}
	crashes when runs

Which line of code can be added to print the value 4?
 	
int main()
{
struct S {int a, b; };
vector<S> v;
S s{3, 4};
v.push_back(s);
// Add code here
}
	cout << v.at(0).b << endl;

Which defines a vector to store the salaries of ten employees?	vector<double> salaries(10);

Assume vector<double> speed(5); Which line throws a run-time error?
 	
cout << speed[speed.size()];
speed[0] = speed.back()
speed.front() = 12;
speed.erase(speed.begin());
	None of these

The following code is logically correct. What is the semantically correct prototype for mystery()? vector<double> v; mystery(v);	void mystery(vector<int>&);
The following code is logically correct. What is the semantically correct prototype for mystery()? vector<double> v{1, 2, 3}; mystery(v);	Either mystery(const vector<int>&); or mystery(vector<int>&); could be correct.

Which line will not compile?
 	
int main()
{
vector<int> v{1, 2, 3};
auto size = v.size();
 	
cout << v.back() << endl; // 1.
cout << v.front() << endl; // 2.
cout << v.at(0) << endl; // 3.
cout << v.at(size) << endl; // 4.
cout << v.pop_back() << endl; // 5.
}
	5

Which line prints 3?
 	
int main()
{
vector<int> v{1, 2, 3};
auto size = v.size();
 	
cout << v.back() << endl; // 1.
cout << v.front() << endl; // 2.
cout << v.at(0) << endl; // 3.
cout << v.at(size) << endl; // 4.
cout << v.pop_back() << endl; // 5.
}
	1

Which line compiles, but crashes when run?
 	
int main()
{
vector<int> v{1, 2, 3};
auto size = v.size();
 	
cout << v.back() << endl; // 1.
cout << v.front() << endl; // 2.
cout << v.at(0) << endl; // 3.
cout << v.at(size) << endl; // 4.
cout << v.pop_back() << endl; // 5.
}
	4

Which statement is false? The elements in a vector: Are accessed by using an index or subscript Each use the same amount of memory Are are all of the same type Are homogeneous	None of these

Which lines have an identical effect?
 	
int main()
{
vector<int> v{1, 2, 3};
auto size = v.size();
 	
cout << v.back() << endl; // 1.
cout << v.front() << endl; // 2.
cout << v.at(0) << endl; // 3.
cout << v.at(size) << endl; // 4.
cout << v.pop_back() << endl; // 5.
}
	2 and 3

In C++ the parameterized collection classes are called _____?	templates
Classes that contain objects as elements are called?	collections

Assume vector<double> speed(5); Which line throws a runtime error?
 	
None of these
speed.erase(speed.begin());
speed.front() = 12;
speed[0] = speed.back()
	cout << speed.at(speed.size());

vector<int> v;	Creates the empty vector []
vector<int> v(1);	Creates the vector [0]
v.begin()	Points to the first element in v
v.back();	Returns a reference to the last element in v
v.erase(v.begin());	Removes the first element in v and shifts the rest to the left
v.pop_back()	Removes the last element in v
v[3];	Returns a reference to the fourth element in v with no range checking
vector<int>v(2,3);	Creates the vector [3,3]
vector<int>v[2, 3];	Creates the vector [2, 3]
v.push_back(3);	Adds a new element to the end of v
v.at(3);	Safely returns a reference to the fourth element in v
You can create vector objects to store any type of data, but each element in the vector must be the same type.	True

Assume the vector v contains [1, 2, 3]. v.erase(0); changes v to [2, 3].	False

Assume vector<int> v; Writing cout << v.front(); throws a runtime exception.	True
Assume the vector v contains [1, 2, 3]. v.erase(v.begin() + 2); changes v to [1, 2].	True
The declaration: vector<string> v(5, "bob"); creates a vector containing five string objects, each containing "bob".	True
In the declaration: vector<int> v; the word int represents the object's base type.	True
The elements of a vector are allocated contiguously.	True
vector subscripts begin at 0 and go up to the vector size - 1	True
The clear() member function removes all the elements from a vector.	True
The statement v.insert(v.end() + 1, 3) is undefined because end() + 1 points past the last element in the vector.	True
The statement v.insert(v.end(), 3) appends the element 3 to the end of the vector v.	True
Contiguous allocation means that the elements are stored next to each other in memory.	True
The push_back member function adds elements to the end of a vector.	True
Assume the vector v contains [1, 2, 3]. v.erase(v.begin()); changes v to [2, 3].	True
The declaration: vector<int> v(10); creates a vector object containing ten elements initialized to 0.	True
Assume the vector v contains [1, 2, 3]. v.pop_back(); changes v to [1, 2].	True
The term for classes with a base-type specification are parameterized classes.	True
Assume that v contains [1, 2, 3]. The result of writing cout << v[4]; is undefined.	True
The C++ term for classes like vector are template classes.	True
A vector subscript represents the element's offset from the beginning of the vector.	True
The declaration: vector<string> v{"bill", "bob", "sally"}; creates a vector containing three string objects.	True
The declaration: vector<int> v(10, 5); creates a vector object containing ten integers.	True
Assuming that Star is a structure, the declaration: vector<Star> stars(3); creates three default initialized Star objects.	True
The declaration: vector<string> v(5); creates a vector containing five empty string objects.	True
Assume the vector v contains [1, 2, 3]. v.erase(0); is a syntax error.	True
The declaration: vector<int> v; creates a vector object with no elements.	True
A vector represents a linear homogeneous collection of data.	True
Assume vector<double> v; Writing cout << v.back(); is undefined.	True
Elements in a vector are accessed using a subscript.	True
Assume that v contains [1, 2, 3]. The result of writing cout << v.at(4); throws a runtime exception.	True
The statement v.insert(v.begin(), 3) inserts the element 3 into the vector v, shifting the existing elements to the right	True
Vector subscripts begin at 1 and go up to the vector size.	False
The statement v.insert(v.end(), 3) is undefined because end() points past the last element in the vector.	False
Assume that v contains [1, 2, 3]. The result of writing cout << v.at(4); is undefined.	False
The C++ term for classes like vector are generic classes.	False
The statement v.insert(v.begin(), 3) inserts the element 3 into the vector v, overwriting the exiting element at index 0.	False

The push_back member function adds elements to the end of a vector as long as there is room for the elements.	False
The declaration: vector<int> v(10); creates a vector object containing uninitialized elements.	False
The declaration: vector<int> v(10, 5); creates a vector object containing five integers.	False
The declaration: vector<string> v(5); creates a vector containing five null pointers.	False
In the declaration: vector<int> v; the word vector represents the object's base type.	False
The declaration: vector<int> v; creates a vector variable but no vector object.	False
Assume that v contains [1, 2, 3]. The result of writing cout << v.at(4); is a compiler error.	False
Vector subscripts begin at 1 and go up to the vector size.	False
A vector consists of named members.	False
The declaration: vector<int> v(10, 5); is illegal.	False
Assume vector<double> v; Writing cout << v.back(); throws a runtime exception.	False
Assume that v contains [1, 2, 3]. The result of writing cout << v[4]; is a compiler error.	False
The declaration: vector<int> v = new vector<>(); creates a vector object with no elements.	False
The pop_back member function adds elements to the end of a vector.	False