CH 13 C++ Flashcards

Share

16 studiers today ★ Leave the first rating

Terms in this set (83)

[1401] Which of these lines correctly prints 3?

```
struct S {
int a = 3;
double b = 2.5;
};

S obj, *p = &obj;

cout << p.a << endl;
cout << *p.a << endl;
cout << *(p).a << endl;
cout << *(p.a) << endl;
cout << (*p).a << endl;
```

cout << (*p).a << endl;

[1402] Which of these lines correctly prints 2.5?

```
struct S {
int a = 3;
double b = 2.5;
};

S obj, *p = &obj;

cout << *(p).b << endl;
cout << *p.b << endl;
cout << p->b << endl;
cout << *(p.b) << endl;
cout << *p->b << endl;
```

cout << p->b << endl;

[1403] Which of these lines displays the eighth element of a?

```
int a[15];

cout << a[8] << endl;
cout << a(7) << endl;
cout << a.at(7) << endl;
cout << a[7] << endl;
```

cout << a[7] << endl;

[1404] Which prints the number of elements in a?

```
int a[] = {1, 2, 3};

cout << a.length << endl;
cout << sizeof(a[0]) << endl;
cout << a.size() << endl;
cout << sizeof(a) << endl;
```

None of these

None of these

[1405] What is stored in the last element of nums?

```
int nums[3] = {1, 2};
```

Undefined value
2
Syntax error in array declaration
0
1

0

[1406] Which line throws and out_of_range exception?

```
double speed[5] = {. . .};
```

None of these

```
cout << speed[4] << endl;
cout << speed[5] << endl;
cout << speed[0] << endl;
cout << speed[1] << endl;
```

None of these

[1407] Which line has undefined output?

```
double speed[5] = {. . .};

cout << speed[5] << endl;
cout << speed[0] << endl;
```

cout << speed[5] << endl;

int[5] d;
int b[5];
int a[4];
None of these
int[] c[5];	

[1409] What is printed?
 	
int a[] = {1, 2, 3};
int b[] = {1, 2, 3};
 	
if (a == b) cout << "a == b" << endl;
else cout << "a != b" << endl;
 	
a != b
Undefined behavior
a == b
Syntax error; does not compile.
	a != b

[1410] What does the array a contain after this runs?
 	
int a[] = {1, 2, 3};
int b[] = {4, 5, 6};
a = b;
 	
Syntax error; does not compile.
{4, 5, 6}
{1, 2, 3}
Undefined behavior
	Syntax error; does not compile.

[1411] Which assigns a value to the first position in letters?
 	
char letters[26];
 	
letters[0] = 'a';
letters[0] = "a";
letters[1] = 'b';
letters.front() = 'a';
letters = 'a';
	letters[0] = 'a';

[1412] Which assigns a value to the first position in letters?
 	
char letters[26];
 	
*letters = 'a';
*letters = "a";
*letters[0] = 'a';
*(letters + 1) = 'a';
*letters + 1 = 'b';
	*letters = 'a';

[1413] What does this loop do?
 	
int a[] = {6, 1, 9, 5, 1, 2, 3};
int x(0);
for (auto e : a) x += e;
cout << x << endl;
 	
Counts the elements in a
Selects the largest value in a
Has no effect
Selects the smallest value in a
Sums the elements in a
	Sums the elements in a

[1414] What is the address of the first pixel in the last row of this image?
 	
Pixel *p; // address of pixel data
int w, h; // width and height of image
 	
p + w + h
p + w + (h - 1)
p + w * h
p + w * (h - 1)
None of these are correct
	p + w * (h - 1)

[1415] Which returns the last pixel on the first row of this image?
 	
Pixel *p; // address of pixel data
int w, h; // width and height of image
 	
*p + w - 1
None of these are correct
*(p + w) - 1
	*(p + w - 1)

Pixel *p; // address of pixel data
int w, h; // width and height of image
 	
p[w - 1]
*p[w - 1]
None of these are correct
p[w] - 1
p + w - 1	

[1417] What is the equivalent array notation? int dates[10]; cout << (*dates + 2) + 2 << endl; dates[0] + 4 dates[2] + 2 dates[2] dates[0] + 2 &dates[2]	dates[0] + 4
[1418] What is the equivalent array notation? int dates[10]; cout << (dates + 2) << endl; dates[2] + 2 &dates[2] dates[0] + 2 dates[2] dates[0] + 4	&dates[2]
[1419] What is the equivalent array notation? int dates[10]; cout << *(dates + 2) << endl; dates[2] + 2 dates[0] + 4 dates[2] &dates[2] dates[0] + 2	dates[2]
[1420] What is the equivalent array notation? int dates[10]; cout << (*dates) + 2 << endl; &dates[2] dates[0] + 2 dates[0] + 4 dates[2] dates[2] + 2	dates[0] + 2
[1421] What is the equivalent array notation? int dates[10]; cout << *dates + 2 << endl; &dates[2] dates[2] + 2 dates[0] + 4 dates[2] dates[0] + 2	dates[0] + 2
[1422] What is the equivalent array notation? int dates[10]; cout << *(dates + 2) + 2 << endl; &dates[2] dates[0] + 4 dates[0] + 2 dates[2] dates[2] + 2	dates[2] + 2

int a[] = {1, 2, 3, 4, 5, 6, 7}; int *p = a; cout << a[1] * 2 << endl; None of these *p + 1 * 2 p + 1 * 2 (*p + 1) * 2 *(p + 1) * 2	

[1424] What prints?
 	
int a[] = {1, 3, 5, 7, 9};
int *p = a;
cout << *p++;
cout << *p << endl;
 	
13
None of these
33
22
12
	13

[1425] What prints?
 	
int a[] = {1, 3, 5, 7, 9};
int *p = a;
cout << *++p;
cout << *p << endl;
 	
33
13
None of these
22
12
	33

[1426] What prints?
 	
int a[] = {1, 3, 5, 7, 9};
int *p = a;
cout << ++*p;
cout << *p << endl;
 	
13
12
None of these
22
33
	22

[1427] Which pointer initialization is illegal?
 	
int a[] = {1, 3, 5, 7, 9};
int *p3 = &a[1];
None of these
int *p1 = a;
int *p4 = &a;
int *p2 = a + 3;
	int *p4 = &a;

[1428] Which expression returns the number of countries?
 	
string countries[] = {"Andorra", "Albania", . . . };
 	
len(countries)
countries.length
sizeof(countries) * sizeof(countries[0])
sizeof(countries)
None of these
	None of these

[1429] Which expression returns the number of countries?
 	
string countries[] = {"Andorra", "Albania", . . . };
 	
sizeof(countries)
len(countries)
sizeof(countries) / sizeof(string)
None of these
sizeof(countries) * sizeof(countries[0])
	sizeof(countries) / sizeof(string)

string countries[] = {"Andorra", "Albania", . . . };
len(countries)
sizeof(countries) * sizeof(countries[0])
sizeof(countries)
None of these
sizeof(countries) / sizeof(countries[0])	

[1431] Which array definition is illegal?
int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
a2
a3
None of these
a1
a5
	a1

[1432] Which array definition contains undefined values?
int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
a3
a1
None of these
a5
a2
	a2

[1433] Which array definition is initialized to all zeros?
int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
a5
a2
None of these
a3
a1
	a3

[1434] Which array definition produces {0, 1, 2}?
int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
a5
a3
None of these
a2
a1
	None of these

[1435] Which array definition is illegal?
const int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[2] = {1, 2, 3};
a2
a5
a3
None of these
a1
	a5

int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
 	
a3
a5
a2
a1
None of these	

An incomplete type and a forward reference generally mean the same thing.	True
Explicitly initializing an array like this: int a[3] = {1, 2, 3}; requires the size and the number of elements supplied to be the same.	False
In C++ using == to compare one array to another is permitted (if meaningless).	True
You must use an integral constant or literal to specify the size of a built-in C++ array.	True
The reinterpret_cast instruction changes way that a pointer's indirect value is interpreted.	True
If p is a pointer to a structure, and the structure contains a data member x, you can access the data member by using the notation: (*p).x	True

C++ arrays have no support for bound-checking.	True
In C++ assigning one array to another is illegal	True
The allocated size of a built-in C++ array cannot be changed during runtime.	True
The size of the array is not stored along with its elements.	True
If img is a pointer to the first byte in an image loaded into memory, Pixel is a structure as defined in your textbook, you can create a Pixel pointer pointing to the image by writing: Pixel **p = reinterpret_cast<Pixel** >(img);	True
The subscripts of a C++ array range from 0 to the array size - 1.	True
C++ arrays have no built-in functions for inserting and deleting.	True
A forward reference can be used when you want to use a pointer to a structure as a data member without first defining the entire structure.	True
The elements of a C++ array created in a function are allocated on the stack.	True
The elements of a C++ array created outside of a function are allocated in the static-storage area.	True

The elements of a C++ string array with no explicit initialization, created in a function will be set to the empty string.	True
Explicitly initializing an array like this: int a[3] = {1, 2, 3}; requires the size to be the same or larger than the number of elements supplied.	True
In C++ printing an array name prints the address of the first element in the array.	True
In C++ there is no separate array variable. The array name is a symbolic representation of the address of the first element in the array.	True
In C++ initializing an array with the contents of another is illegal.	True
C++ arrays produce undefined results if you access an element outside the array.	True
Explicitly initializing an array like this: int a[] = {1, 2, 3}; works in all versions of C++.	True
You may use any kind of integral variable to specify the size of a built-in C++ array.	False
The elements of a C++ string array with no explicit initialization, created in a function	False

In C++ using == to compare one array to another is illegal.	False
The allocated size of a built-in C++ array may be changed during runtime	False
If img is a pointer to the first byte in an image loaded into memory, Pixel is a structure as defined in your textbook, you can create a Pixel pointer pointing to the image by writing: Pixel **p = static_cast<Pixel** >(img);	False
The reinterpret_cast instruction produces a temporary value by converting its argument.	False
In C++ initializing an array with the contents of another is permitted.	False
C++ arrays use bound-checking when you access their elements with the at() member function.	False
The elements of a C++ array created in a function are allocated on the heap.	False
In C++ assigning one array to another is permitted.	False
C++ arrays throw an out_of_bounds exception if you access an element outside the array.	False
In C++ an array variable and the array elements are separate. The array variable contains the address of the first element in the array.	False
In C++ printing an array name prints the value of the first element in the array.	False
The elements of a C++ int array with no explicit initialization, created in a function will be set to zero.	False
C++ arrays can be allocated with a size of 0.	False
The static_cast instruction changes way that a pointer's indirect value is interpreted.	False
The size of the array is stored along with its elements.	False
The allocated size of a built-in C++ array may be changed during runtime	False
A forward reference can be used when you want to use a structure as a data member without first defining the entire structure.	False
The elements of a C++ array created outside of a function are allocated on the stack.	False
If p is a pointer to a structure, and the structure contains a data member x, you can access the data member by using the notation: *p->x	False
C++ arrays offer built-in member functions for inserting and deleting.	False
Explicitly initializing an array like this: int a[] = {1, 2, 3}; only works in C++ 11.	False