# Chương : Ôn tập và giới thiệu về C++

1. Ngôn ngữ hướng đối tượng đầu tiên là
   1. C
   2. Java
   3. Pascal
   4. Smalltalk
2. Các ngôn ngữ hướng đối tượng
   1. Eifel, Clos, Loops, Flavors
   2. Object Pascal, Object C
   3. C++, Delphi, Java
   4. a, b, c đều đúng
3. Các hàm có định nghĩa trong C++

A. exp(x) - tính ex

B. sqrt(x) - tính căn bậc hai của x

C. sqr(x) - tính x2

* 1. A, B, C
  2. A, B
  3. A, C
  4. B, C

1. Giá trị lớn nhất của biến kiểu unsigned char là

a) 255

b) 127

c) ‘z’

d) ‘a’

1. Dòng nào không đúng trong C:

a) class aClass{public:int x;};

b) /\* A comment \*/

c) char x=12;

1. Trong hàm có bắt buộc phải có khai báo return không?
   1. Có
   2. Không, nếu như kiểu giá trị trả về không được chỉ ra
   3. Không, nếu như kiểu giá trị trả về là void
   4. Không, nếu hàm không có các đối số
2. Một biến được định nghĩa trong một hàm gọi là biến
   1. vùng
   2. cục bộ
   3. toàn cục
   4. tham chiếu
3. Nếu muốn có một hàm như một constant thì khai báo nào dưới đây là đúng
   1. const int ShowData(void) { /\* statements \*/ }
   2. int const ShowData(void) { /\* statements \*/ }
   3. int ShowData(void) const { /\* statements \*/ }
   4. Cả A và B
4. Cho khai báo như sau

int num[2][3]={ {1,2}, {3,4}, {5, 6} };

Hãy lựa chọn phát biểu đúng:

* 1. Giá trị của num[1][2] là 2
  2. Giá trị của num[1][2] là 4
  3. Chương trình bị lỗi
  4. Giá trị của num[1][2] là 3

1. Hàm nào phải có trong các chương trình C++?

a) start()

b) system()

c) main()

d) program()

1. Kí hiệu dùng để kết thúc một dòng lệnh trong lập trình C++ là

a) .

b) ;

c) :

d) ‘

1. Kí pháp nào sau đây dùng để bắt đầu và kết thúc một đoạn code trong C++?

a) { }

b)→ and ←

c) BEGIN and END

d) ( and )

1. Kiểu dữ liệu nào sau đây không dùng trong C++?

a) float

b) real

c) int

d) Double

1. Kí hiệu nào sau đây dùng để so sánh hai giá trị

a) :=

b) =

c) equal

d) ==

1. Số nào sau đây mang giá trị True trong kiểu dữ liệu Boolean?

a) 1

b) 66

c) 1

d) -1

e) Tất cả đều đúng

1. Kí hiệu của biểu thức và là

a) &

b) &&

c) |

d) |&

1. Kết quả của biểu thức !(1 && !(0 || 1)) là

a) True

b) False

c) Không có giá trị

1. Đoạn code nào dưới đây có thể được dùng để tạo các số nguyên có giá trị từ 1 đến 10
   1. 1 + rand() % (10 - 1 + 1)
   2. 1 + (10 - 1 + 1) % rand()
   3. 10 + rand() % (10 - 1 + 1)
   4. 10 + rand() % (10 + 1)
2. Toán tử “ , ” đươc sử dụng chủ yếu với
   1. Khai báo if-else
   2. Khai báo do-while
   3. Khai báo for
   4. Tất cả các khai báo trên
3. Giá trị của x khi đoạn code sau được thực thi là?

int x;

for(x = 0; x < 10; x++) { }

a) 10

b) 9

c) 0

d) 1

1. Cấu trúc lệnh nào sau đây không phải là cấu trúc lặp trong C++?

a) for

b) do while

c) while

d) repeat until

1. Kiểu khai báo nào sau đây không đúng?

a) int funct(char x, char y);

b) double funct(char x):

c) void funct();

d) char x();

1. Lệnh nào sau đây là lời gọi hàm đúng?

a) funct;

b) funct x, y;

c) funct();

d) int funct();

1. Lệnh nào sau đây dùng để kết thúc một case trong switch - case?

a) end;

b) break;

c) stop;

d) continue;

1. Kết quả đúng cho đoạn chương trình sau là

#include <stdio.h>

void main()

{

int x = 0;

switch(x)

{

case 1: cout<<"One";

case 0: cout<<"Zero”;

case 2: cout<< "Hello World";

}

}

a) Zero Hello Word

b) Zero

c) Hello World

d) ZeroHello World

1. Khai báo đúng cho khai báo kiểu con trỏ là

a) int x;

b) int &x;

c) ptr x;

d) int \*x;

1. Đáp án nào sau đây thể hiện địa chỉ bộ nhớ của biến số nguyên a?

a) \*a;

b) a;

c) &a;

d) address(a);

1. Đáp án nào sau đây thể hiện địa chỉ bộ nhớ của một biến được trỏ đến bởi con trỏ a?

a) a;

b) \*a;

c) &a;

d) address(a);

1. Đáp án nào sau đây cho biết giá trị được lưu trữ tại địa chỉ được trỏ tới bởi con trỏ a?

a) a;

b) val(a);

c) \*a;

d) &a;

1. Đáp án nào sau đây mô tả việc truy cập một biến từ cấu trúc b (structure)?

a) b→var;

b) b.var;

c) b-var;

d) b>var;

1. Đáp án nào sau đây mô tả việc truy cập một biến từ một con trỏ trỏ tới cấu trúc, \*b?

a) b→var;

b) b.var;

c) b-var;

d) b>var;

1. Đáp án nào sau đây đúng khi định nghĩa về struct?

a) struct {int a;}

b) struct a\_struct {int a;}

c) struct a\_struct int a;

d) struct a\_struct {int a;};

1. Đáp án nào khai báo một biến của struct foo?

a) struct foo;

b) struct foo var;

c) foo;

d) int foo;

1. Cách khai báo mảng một chiều đúng trong C++ là

a) int arr[10];

b) int arr;

c) arr{10};

d) array arr[10];

1. Chỉ số của phần tử cuối cùng trong mảng gồm 29 phần tử là:

a) 29

b) 28

c) 0

d) Programmer-defined

1. Cách khai báo nào sau đây là các khai báo mảng hai chiều đúng trong C++?

a) array arr[20][20];

b) int arr[20][20];

c) int arr[20, 20];

d) char arr[20];

1. Đáp án nào sau đây mô tả đúng việc truy cập phần tử thứ bảy được lưu trữ trong “foo” - mảng foo gồm 100 phần tử?

a) foo[6];

b) foo[7];

c) foo(7);

d) foo(6);

1. Đáp án nào sau đây cung cấp địa chỉ bộ nhớ của phần tử đầu tiên trong mảng arr, mảng arr gồm 100 phần tử?

a) arr[0];

b) arr;

c) &arr;

d) arr[1];

1. Kí hiệu kết thúc chuỗi string là?

a) ‘.’

b) ‘ ‘

c) ‘\0’

d) ‘/0’

1. Nếu char s[] = “0123456789”; thì chỉ số cuối cùng của mảng s là:

a) 11

b) 10

c) 9

1. Biểu thức 1/9\*9 cho giá trị:

a) 1

b) 0.0862

c) 0

1. Đâu là lệnh cấp phát mảng:

a) char \*p=new char[20];

b) char p=new char[20];

c) char p=new char(20);

1. Cho đoạn khai báo: int A[10], \*p = A + 4; Khi sử dụng biến p trong biểu thức thì \*(p + 1) là:

a) Giá trị của p

b) Giá trị của A[5]

c) Địa chỉ của A[5]

1. Đâu là lệnh giải phóng mảng p đã cấp phát:

a) delete p;

b) delete p[];

c) delete []p;

1. Dòng lệnh: int c = 0; cout << c++ << c; cho in ra màn hình:

a) 00

b) 01

c) 10

1. Biểu thức trong vòng lặp for(x = 0; x = 3; x++) được thực hiện bao nhiêu lần:

a) 0

b) 3

c) vô cùng

1. Luồng nhập chuẩn, tương ứng với bàn phím là
   1. cin
   2. cout
   3. stin
   4. stout
2. Cho khai báo như sau

int num[4] = {11, -13, 17, 105};

Hãy lựa chọn phát biểu đúng:

* 1. Giá trị của num[2] là -13
  2. Giá trị của num[2] là 17
  3. Chương trình bị lỗi
  4. Giá trị của num[2] là 18

1. cout là một:
   1. operator
   2. function
   3. object
   4. macro
2. Khai báo nào sau đây khởi tạo và gán giá trị ban đầu cho một con trỏ có tên salesPtr
   1. \*salesPtr = "";
   2. float &salesPtr = NULL;
   3. float \*salesPtr = "";
   4. float \*salesPtr = NULL;
3. Địa chỉ của một biến temp kiểu float là
   1. \*temp
   2. &temp
   3. float& temp
   4. float temp&
4. Nếu muốn một hàm không có kết quả trả về thì kiểu trả về của nó phải là
   1. empty
   2. void
   3. barren
   4. không cho phép trong C++
5. Một khối catch mặc định sẽ bắt
   1. Tất cả các đối tượng đã được quăng ra
   2. Không đối tượng nào
   3. Bất cứ đối tượng nào không được bắt bởi các khối catch trước nó
   4. Tất cả các đối tượng đã được bắt lại bởi một khối catch trước đó
6. Tên gọi chung cho các lỗi xuất hiện trong quá trình thực thi chương trình gọi là
   1. Infractions
   2. Exceptions
   3. Deviations
   4. Anomalies
7. Mã có khả năng sinh ra ngoại lệ (exception) cần nằm trong khối:
   1. Sau từ khóa catch
   2. Sau từ khóa try
   3. Sau từ khóa throw;
   4. Trong khối bất kỳ của chương trình
8. Việc gì không thể làm với con trỏ void?

a) Các phép toán số học trên con trỏ.

b) Các hàm trên con trỏ.

c) Cả 2 đáp án đều đúng.

d) Cả 2 đáp án đều sai.

1. C++ hỗ trợ loại comment nào sau đây

a) single line

b) multi line

c) single line and multi line

d) Tất cả đều sai

1. Có bao nhiêu phép biểu diễn trong string?

a) 1

b) 2

c) 3

d) 4

1. file header của lớp string là gì?

a) #include<ios>

b) #include<str>

c) #include<string>

d) Tất cả đều sai

1. Đáp án nào được sử dụng để trả lại số ký tự trong chuỗi?

a) length

b) size

c) cả a & b

d) Không đáp án nào đúng

1. Những phương thức nào chúng ta sử dụng để nối thêm nhiều hơn một ký tự cùng một chuỗi?

a) append

b) operator+=

c) data

d) cả a & b

1. Các vector biểu diễn cho?

a) Static arrays

b) Dynamic arrays

c) Stack

d) Queue

1. Chọn ra định nghĩa chính xác về vector.

a) vector<int> values (5)

b) vector values (5)

c) vector<int> (5)

d) Không đáp án nào đúng.

1. Đáp án nào dưới đây dùng trong việc khai báo vector?

a) Type

b) Name

c) Vector

d) Number\_of\_elements

1. Tập tin tiêu đề (header file) nào được sử dụng cho các phép toán nhập và xuất của C trong C ++?

a) stdio.h

b) cstdio

c) iostream

d) Tất cả đều sai

1. Điều nào sau đây sẽ được sử dụng với các thiết bị vật lý để tương tác từ chương trình C ++?

a) Programs

b) Library

c) Streams

d) Không đáp án nào đúng.

1. Biểu thức 1 & 4 cho giá trị:

a) 0

b) 1

c) 4

1. strcat(aString, “abc”) thực hiện:

a) copy “abc” vào aString

b) Thêm “abc” vào sau aString

c) so sánh “abc” và aString

1. Kiểu dữ liệu float có thể xử lí dữ liệu trong phạm vi nào :

a) 3.4\*10^-38đến 3.4\*10^38.

b) -32768 đến 32767.

c) -128 đến 127.

d) 0…65535

1. Cho đoạn chương trình sau:

int I = 12, J = 0;

while ( I > 8 )

{

I = I – 1;

J = J + I;

}

cout << "I = " << I << endl;

cout << "J = " << J << endl;

* 1. I = 4 và J = 8
  2. I = 8 và J = 38
  3. I = 9 và J = 45
  4. I = 6 và J = 11

1. Cho đoạn chương trình sau:

int j = 2;

while (j!=28)

{

cout << j << endl;

j = j + 3;

}

Vòng lặp được thực hiện bao nhêu lần?

* 1. 8
  2. 10
  3. 11
  4. Tất cả đều sai

1. Trường hợp nào không dùng để định nghĩa kiểu dữ liệu trong các trường hợp sau

1.

struct book

{

char name[10];

float price;

int page;

};

2.

long int l = 2.3;

3.

enum day{Monday, Tuesday, Wednesday};

a) 1

b) 2

c) 3

d) 1 & 2 & 3

1. Kí hiệu nào sau đây cho phép sử dụng để đặt tên biến

a) \* (asterisk)

b) | (pipeline)

c) – (hyphen)

d) \_ (underscore)

1. Cho hàm

int fn(int x) {

if (x == 0 || x == 1) return 1;

if (x % 2 == 0) return (fn(x / 2) + 2);

return (fn(x – 1) + 3);

}

fn(7) trả về giá trị:

a) 7

b) 8

c) 11

1. Cho đoạn chương trình sau:

void main()

{

int i, j, sum;

for (i=1; i <= 3; i++)

{

sum = 0;

for (j=1; j <=3; j++)

sum++;

cout << sum << endl;

}

}

Ở dòng xuất thứ hai, giá trị của sum là:

* 1. 1
  2. 0
  3. 3
  4. 6

1. Cho khai báo sau:

void Fix(int& , float );

int someInt = 42;

float someFloat = 3.14;

Gọi hàm Fix:

A. Fix(24, 6.85);

B. Fix(someInt, 6.85);

C. Fix(24, someFloat);

D. Fix(someInt, someFloat);

E. Fix(someInt + 5, someFloat);

Cách gọi đúng là:

* 1. Tất cả
  2. Chỉ A và C
  3. Chỉ B và D
  4. Không có

1. Cho đoạn chương trình sau:

void DoThis(int& Alpha, int Beta)

void main()

{

int Temp = 15;

int Ben = -5, Jerry = 42;

DoThis(Ben, Jerry);

cout << "Ben = " << Ben << endl;

cout << "Jerry = " << Jerry << endl;

cout << "Temp = " << Temp << endl;

}

void DoThis(int& Alpha, int Beta)

{

int Temp;

Alpha = Alpha + 100;

Temp = Beta;

Beta = 999;

Temp = 12;

}

* 1. Ben = 95, Jerry = 42, Temp = 15
  2. Ben = -5, Jerry = 42, Temp = 15
  3. Ben = 95, Jerry = 999, Temp = 15
  4. Ben = 95, Jerry = 999, Temp = 12

1. Cho biết kết quả của đoạn chương trình sau:

#include <iostream>

using namespace std;

int max(int a, int b )

{

return ( a > b ? a : b );

}

int main()

{

int i = 5;

int j = 7;

cout << max(i, j );

return 0;

}

* 1. 5
  2. 7
  3. a hoặc b đúng
  4. Tất cả đều sai

1. Cho biết kết quả của đoạn chương trình sau

#include <iostream>

using namespace std;

int mult (int x, int y)

{

int result;

result = 0;

while (y != 0) {

result = result + x;

y = y - 1;

}

return(result);

}

int main ()

{

int x = 5, y = 5;

cout << mult(x, y) ;

return(0);

}

* 1. 20
  2. 25
  3. 30
  4. 35

1. Cho biết kết quả đoạn chương trình sau:

#include <iostream>

using namespace std;

int gcd (int a, int b)

{

int temp;

while (b != 0) {

temp = a % b;

a = b;

b = temp;

}

return(a);

}

int main ()

{

int x = 15, y = 25;

cout << gcd(x, y);

return(0);

}

* 1. 15
  2. 25
  3. 375
  4. 5

1. Cho biết kết quả của đoạn chương trình sau:

#include <iostream>

using namespace std;

int add(int a, int b);

int main()

{

int i = 5, j = 6;

cout << add(i, j) << endl;

return 0;

}

int add(int a, int b )

{

int sum = a + b;

a = 7;

return a + b;

}

a) 11

b) 12

c) 13

d) Lỗi biên dịch

1. Cho biết kết quả của chương trình sau:

#include <iostream>

using namespace std;

void Sum(int a, int b, int & c)

{

a = b + c;

b = a + c;

c = a + b;

}

int main()

{

int x = 2, y =3;

Sum(x, y, y);

cout << x << " " << y;

return 0;

}

* 1. 2 3
  2. 6 9
  3. 2 15
  4. Lỗi biên dịch

1. Cho biết kết quả đoạn chương trình sau:

int Limit = 10;

cout << ‘H’;

for (int LoopCount = 8; LoopCount < Limit; LoopCount++)

cout << ‘E’;

cout << "LP";

* 1. HLP
  2. HEELP
  3. HELP
  4. Tất cả đều sai

1. Cần thay thế điều kiện nào vào vị trí “???” để kết quả xuất ra là 8 17 35 71

int N = 8;

do

{

cout << setw(3) << N;

N = N\*2 +1;

}while (???);

* 1. N < 71
  2. N <=71
  3. N >143
  4. N < =35

1. Cho khai báo sau, chỉ số của myArray[] có giá trị từ:

int MaxStuff = 65;

char myArray [MaxStuff + 10];

* 1. 0 đến 75
  2. 0 đến 74
  3. 1 đến 75
  4. 1 đến 74

1. Cho biết kết quả chương trình sau:

void main()

{

int Ray[5] = {100,200,300,400,500};

int k;

for (k = 4; k >= 1; k--)

cout << Ray[k] << ‘ ’;

}

* 1. 100 200 300 400 500
  2. 400 300 200 100 0
  3. 500 400 300 200 100
  4. 500 400 300 200

1. Chọn khai báo đúng
   1. void f(int a, float b =3.5, int c = 2, float d =1.0)
   2. void f(int a, float b =3.5, int c, float d =1.0)
   3. void f(int a, float b=3.5, int c =2, float d)
   4. a và c đều đúng
2. Kết quả của chương trình sau là

#include <stdio.h>

using namespace std;

int main()

{

int array[] = {10, 20, 30};

cout << -2[array];

return 0;

}

* 1. -15
  2. -30
  3. Biên dịch lỗi
  4. Giá trị rác

1. Kết quả của chương trình sau là:

#include <stdio.h>

using namespace std;

int array1[] = {1200, 200, 2300, 1230, 1543};

int array2[] = {12, 14, 16, 18, 20};

int temp, result = 0;

int main()

{

for (temp = 0; temp < 5; temp++) {

result += array1[temp];

}

for (temp = 0; temp < 4; temp++) {

result += array2[temp];

}

cout << result;

return 0;

}

* 1. 6553
  2. 6533
  3. 6522
  4. 12200

1. Kết quả của chương trình sau là

#include <stdio.h>

using namespace std;

int main ()

{

int array[] = {0, 2, 4, 6, 7, 5, 3};

int n, result = 0;

for (n = 0; n < 8; n++) {

result += array[n];

}

cout << result;

return 0;

}

* 1. 25
  2. 26
  3. 27
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau

#include <iostream>

using namespace std;

int main()

{

/\* this is comment\*

cout << "hello world";

return 0;

}

* 1. hello world
  2. hello
  3. Biên dịch lỗi
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau:

#include <iostream>

using namespace std;

long factorial (long a)

{

if (a > 1)

return (a \* factorial (a + 1));

else

return (1);

}

int main ()

{

long num = 3;

cout << num << "! = " << factorial ( num );

return 0;

}

* 1. 6
  2. 24
  3. Lỗi phân đoạn
  4. Biên dịch lỗi

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <cstring>

using namespace std;

int main ()

{

char str1[10] = "Hello";

char str2[10] = "World";

char str3[10];

int len ;

strcpy( str3, str1);

strcat( str1, str2)

len = strlen(str1);

cout << len << endl;

return 0;

}

* 1. 5
  2. 55
  3. 11
  4. 10

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <string>

using namespace std;

int main ()

{

string str ("microsoft");

string::reverse\_iterator r;

for (r = str.rbegin() ; r < str.rend(); r++ )

cout << \*r;

return 0;

}

* 1. microsoft
  2. micro
  3. tfosorcim
  4. tfos

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <string>

using namespace std;

int main ()

{

string str ("nobody does like this");

string key ("nobody");

size\_t f;

f = str.rfind(key);

if (f != string::npos)

str.replace (f, key.length(), "everybody");

cout << str << endl;

return 0;

}

* 1. nobody does like this
  2. nobody
  3. everybody
  4. everybody does like this

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <string>

using namespace std;

int main ()

{

string str ("steve jobs is legend");

string::iterator it;

str.erase (str.begin()+ 5, str.end()-7);

cout << str << endl;

return 0;

}

* 1. jobs is
  2. steve legend
  3. steve
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <string>

using namespace std;

int main ()

{

string str ("Microsoft");

for (size\_t i = 0; i < str.length();)

{

cout << str.at(i-1);

}

return 0;

}

* 1. M
  2. Microsoft
  3. Micro
  4. Lỗi lúc chạy chương trình

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <string>

using namespace std;

int main ()

{

string str ("Ubuntu");

cout << str.capacity();

cout << str.max\_size();

return 0;

}

* 1. 61073741820
  2. 51073741820
  3. 6 và kích thước tối đa phụ thuộc vào trình biên dịch.
  4. Không đáp án nào đúng.

1. Kết quả của chương trình sau là:

#include <stdio.h>

using namespace std;

int main()

{

int a = 5, b = 10, c = 15;

int arr[3] = {&a, &b, &c};

cout << \*arr[\*arr[1] - 8];

return 0;

}

* 1. 15
  2. 18
  3. Giá trị rác
  4. Lỗi biên dịch

1. Khi ngoại lệ xảy ra, trình điều khiển sẽ chuyển chúng tới?
   1. catch
   2. handlers
   3. throw
   4. Tất cả đều sai
2. Từ khóa được sử dụng để kiểm tra ngoại lệ trong đoạn code?
   1. catch
   2. throw
   3. try
   4. Tất cả đều sai
3. Điều gì sẽ xảy ra khi các ngoại lệ không bị xử lí trong các chương trình?
   1. error
   2. Chương trình sẽ thực thi.
   3. Đoạn mã đó sẽ không được thực thi.
   4. Tất cả đều sai
4. Cho biết kết quả của chương trình sau:

#include <iostream>

using namespace std;

int main()

{

int age = 0;

try {

if (age < 0) {

throw "Positive Number Required";

}

cout << age;

}

catch(const char \*Message)

{

cout << "Error: " << Message;

}

return 0;

}

* 1. 0
  2. Error:Positive Number Required
  3. Lỗi biên dịch
  4. Không có kết quả

1. Cho biết kết quả của chương trình sau:

#include <iostream>

using namespace std;

void PrintSequence(int StopNum)

{

int Num;

Num = 1;

while (true) {

if (Num >= StopNum)

throw Num;

cout << Num;

Num++;

}

}

int main(void)

{

try {

PrintSequence(20);

}

catch(int ExNum)

{

cout << "Caught an exception with value: " << ExNum;

}

return 0;

}

advertisements

* 1. Lỗi biên dịch
  2. In ra 19 số đầu tiên
  3. In ra 19 số đầu tiên và lỗi ở số thứ 20
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau

#include <iostream>

using namespace std;

double division(int a, int b)

{

if (b == 0) {

throw "Division by zero condition!";

}

return (a / b);

}

int main ()

{

int x = 50;

int y = 2;

double z = 0;

try {

z = division(x, y);

cout << z;

}

catch(const char \*msg) {

cerr << msg;

}

return 0;

}

* 1. 25
  2. 20
  3. Division by zero condition!
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau

#include <iostream>

using namespace std;

int main()

{

char\* buff;

try {

buff = new char[1024];

if (buff == 0)

throw "Memory allocation failure!";

else

cout << sizeof(buff) << "Byte successfully allocated!"<<endl;

}

catch(char \*strg) {

cout<<"Exception raised: "<<strg<<endl;

}

return 0;

}

* 1. 4 Bytes allocated successfully
  2. 8 Bytes allocated successfully
  3. Memory allocation failure.
  4. Phụ thuộc vào kích thước của kiểu dữ liệu.

1. Chuyện gì sẽ xảy ra trong đoạn code sau đây:

int a = 100, b = 200;

int \*p = &a, \*q = &b;

p = q;

* 1. b được gán vào a.
  2. p trỏ sang b.
  3. a được gán vào b.
  4. q trỏ sang a.

1. Cho biết kết quả của chương trình sau

#include <iostream>

using namespace std;

int main()

{

int a = 5, b = 10, c = 15;

int \*arr[ ] = {&a, &b, &c};

cout << arr[1];

return 0;

}

* 1. 5
  2. 10
  3. 15
  4. Chương trình sẽ trả về một số ngẫu nhiên

1. Cho biết kết quả của chương trình sau:

#include <iostream>

using namespace std;

void square (int \*x)

{

\*x = (\*x + 1) \* (\*x);

}

int main ( )

{

int num = 10;

square(&num);

cout << num;

return 0;

}

* 1. 100
  2. Lỗi biên dịch
  3. 144
  4. 110

1. Cho biết kết quả của đoạn chương trình sau

#include <iostream>

using namespace std;

int main()

{

char arr[20];

int i;

for(i = 0; i < 10; i++)

\*(arr + i) = 65 + i;

\*(arr + i) = '\0';

cout << arr;

return(0);

}

* 1. ABCDEFGHIJ
  2. AAAAAAAAAA
  3. JJJJJJJJ
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau

#include <iostream>

using namespace std;

int main()

{

char \*ptr;

char Str[] = "abcdefg";

ptr = Str;

ptr += 5;

cout << ptr;

return 0;

}

* 1. fg
  2. cdef
  3. defg
  4. abcd

1. Cho biết kết quả của chương trình sau

#include <iostream>

using namespace std;

void Funct();

int main()

{

try {

Funct();

}

catch(double) {

cerr << "caught a double type..." << endl;

}

return 0;

}

void Funct()

{

throw 3;

}

* 1. caught a double type
  2. compile time error
  3. abnormal program termination
  4. Không đáp án nào đúng.

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <exception>

using namespace std;

int main()

{

try {

int \* array1 = new int[100000000];

int \* array2 = new int[100000000];

int \* array3 = new int[100000000];

int \* array4 = new int[100000000];

cout << "Allocated successfully";

}

catch(bad\_alloc&) {

cout << "Error allocating the requested memory." << endl;

}

return 0;

}

* 1. Allocated successfully
  2. error allocating the requested memory
  3. Tùy thuộc vào bộ nhớ của máy tính.
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau:

#include <iostream>

using namespace std;

int func(void \*Ptr);

int main()

{

char \*Str = "abcdefghij";

func(Str);

return 0;

}

int func(void \*Ptr)

{

cout << Ptr;

return 0;

}

* 1. abcdefghij
  2. địa chỉ của chuỗi “abcdefghij”
  3. lỗi biên dịch
  4. lỗi khi chạy chương trình

1. Cho biết kết quả của chương trình sau:

#include <iostream>

using namespace std;

int main()

{

int \*p;

void \*vp;

if (vp == p);

cout << "equal";

return 0;

}

* 1. equal
  2. không có kết quả
  3. lỗi biên dịch
  4. lỗi khi chạy chương trình

1. Cho biết kết quả của chương trình sau:

#include <iostream>

using namespace std;

int main()

{

int a = 5, c;

void \*p = &a;

double b = 3.14;

p = &b;

c = a + b;

cout << c << '\n' << p;

return 0;

}

* 1. 8, địa chỉ bộ nhớ
  2. 8.14
  3. địa chỉ bộ nhớ
  4. Tất cả đều sai

1. Điều gì sẽ xảy ra khi trình xử lý không được khai báo cho ngoại lệ?
   1. Gọi hàm thư viện chuẩn terminate()
   2. thông báo lỗi.
   3. thi hành đoạn mã còn lại.
   4. không đáp án nào đúng.
2. Cho biết kết quả chương trình sau:

void main()

{

int n = 19;

int \*p = &n;

int &r = \*p;

cout << "&n = " << &n << ", n = " << n << endl;

cout << "&r = " << &r << ", r = " << r << endl;

cout << "&p = " << &p << ", p = " << p << endl;

}

* 1. &n = 0xfff4, n = 19

&r = 0xfff4, r = 19

&p = 0xfff2, \*p = 19

* 1. &n = 0xfff4, n = 19

&r = 0xfff4, r = 19

&p = 0xfff4, \*p = 19

* 1. &n = 0xfff4, n = 19

&r = 0xfff2, r = 19

&p = 0xfff2, \*p = 19

* 1. &n = 0xfff4, n = 19

&r = 0xfff4, r = 19

&p = 0xfff2, \*p = 0xfff2

1. Cho biết kết quả chương trình sau:

void main()

{

float a[] = {11, 22, 33, 44, 55};

cout << "a = " << a << ", \*a = " << \*a << endl;

for( float \*p = a; p < a + sizeof(a)/sizeof(float); p++)

cout << "p = " << p << ", \*p = " << \*p << endl;

}

Nếu a bắt đầu từ ô nhớ có địa chỉ 0xffe2 thì:

* 1. a = 0xffe2, \*a = 11

p = 0xffe2, \*p = 11

p = 0xffe3, \*p = 22

p = 0xffe4, \*p = 33

p = 0xffe5, \*p = 44

p = 0xfff6, \*p = 55

* 1. a = 0xffe2, \*a = 11

p = 0xffe3, \*p = 22

p = 0xffe4, \*p = 33

p = 0xffe5, \*p = 44

p = 0xffe6, \*p = 55

p = 0xfff7, \*p = null

* 1. a = 0xffe2, \*a = 11

p = 0xffe2, \*p = 11

p = 0xffe6, \*p = 22

p = 0xffea, \*p = 33

p = 0xffee, \*p = 44

p = 0xfff2, \*p = 55

* 1. a = 11, \*a = 0xffe2

p = 11, \*p = 0xffe2

p = 22, \*p = 0xffe6

p = 33, \*p = 0xffea

p = 44, \*p = 0xffee

p = 55, \*p = 0xfff2

1. Cho biết kết quả sau khi thực hiện chương trình:

int x;

int & setx();

void main()

{

x = 15;

setx() = 19;

cout << "x = " << x << endl;

}

int & setx()

{

return x;

}

* 1. x = 15
  2. x = 19
  3. x = 0xfff4
  4. Tất cả đều sai

1. Cho biết kết quả chương trình sau:

int main()

{

int x = 5;

int \* p = &x;

cout<<p;

}

* 1. Giá trị khác
  2. 5
  3. Chương trình báo lỗi
  4. Chương trình không in ra gì khác

1. Cho biết kết quả chương trình sau:

int main()

{

int\* a = new int[5];

int\* p = a;

for (int i = 0; i < 5; i++)

a[i] = i + 1;

cout<<\*(p+2);

}

* 1. 3
  2. 2
  3. Chương trình in ra giá trị không xác định
  4. Chương trình báo lỗi khi biên dịch

1. Cho biết kết quả chương trình sau:

int main()

{

int n = 7;

int m = n; m = 9;

int &k = m; k++;

cout<<n<<m;

}

* 1. 7 10
  2. 7 8
  3. 8 8
  4. 7 9

1. Cho biết kết quả chương trình sau:

void cap\_phat\_bo\_nho (int\*&a)

{

a = new int[5];

for (int i=0; i<5; i++)

a[i]=i+1;

}

int main()

{

int n=5;

int \*a= &n;

cap\_phat\_bo\_nho(a);

cout<<a[0];

}

* 1. 1
  2. 5
  3. Chương trình in ra địa chỉ của biến n
  4. Chương trình báo lỗi lúc biên dịch

1. Cho biết kết quả chương trình sau:

void int F(int a, int b, int c=0, int d=1)

{

return a\*d + b\*c;

}

void main()

{

cout<<F(1,2,3,4)<<endl;

cout<<F(2,3,4)<<endl;

cout<<F(3,4)<<endl;

}

* 1. 10 14 3
  2. 10 10 3
  3. 1 2 3
  4. Chương trình báo lỗi lúc biên dịch

1. Chọn các khai báo đúng:

A. float \*pi;

pi = new float;

\*pi = 3.1415;

B. float \*pi =&x;

\*pi = 3.1415;

C. const int \*p = new int;

delete p;

D. float x = 3.1415;

float \*pi = &x;

delete pi;

* 1. A và B
  2. A và C
  3. B và D
  4. A, B, C

Cho các khai báo:

int\* p;

int\* const cp;

const int\* pc;

const int\* const cpc;

cout << Fred[i] << endl;

1. Chọn thao tác có hiệu lực:
   1. ++ cp;
   2. ++ (\*p);
   3. ++ (\*cpc);
   4. ++ cpc;
2. Chọn thao tác không có hiệu lực:
   1. ++ p;
   2. ++ (\*cp);
   3. ++ (\*pc);
   4. ++ pc;
3. Cấp phát bộ nhớ cho mảng 2 chiều:

double\*\* A = (double\*\*)malloc( Row\*sizeof(double\*));

Hàm giải phóng bộ nhớ mảng A

* 1. free (\*\*A);
  2. free (\*A);

free (A);

* 1. delete \*\*A;
  2. delete A;

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <vector>

using namespace std;

int main ()

{

unsigned int i;

vector<int> first;

vector<int> second (4, 100);

vector<int> third (second.begin(), second.end());

vector<int> fourth (third);

int myints[] = {16, 2, 77, 29};

vector<int> fifth (myints, myints + sizeof(myints) / sizeof(int) );

for (vector<int> :: iterator it = fifth.begin(); it != fifth.end(); ++it)

cout << ' ' << \*it;

return 0;

}

* 1. 16
  2. 16 2
  3. 16 2 77
  4. 16 2 77 29

1. Cho biết kết quả của chương trình sau

#include <iostream>

#include <vector>

using namespace std;

int main ()

{

vector<int> myvector;

int sum (0);

myvector.push\_back (100);

myvector.push\_back (200);

myvector.push\_back (300);

while (!myvector.empty())

{

sum += myvector.back();

myvector.pop\_back();

}

cout << sum << '\n';

return 0;

}

* 1. 500
  2. 600
  3. 700
  4. Error

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <vector>

using namespace std;

int main ()

{

vector<int> a (3, 0);

vector<int> b (5, 0);

b = a;

a = vector<int>();

cout << "Size of a " << int(a.size()) << '\n';

cout << "Size of b " << int(b.size()) << '\n';

return 0;

}

* 1. Size of a 0

Size of b 3

* 1. Size of a 3

Size of b 5

* 1. Error
  2. Tất cả đều sai

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <vector>

using namespace std;

int main ()

{

vector<int> first;

first.assign (7,100);

vector<int>::iterator it;

it=first.begin()+1;

int myints[] = {1776,7,4};

cout << int (first.size()) << '\n';

return 0;

}

* 1. 10
  2. 9
  3. 8
  4. 7

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <vector>

using namespace std;

int main ()

{

vector<int> myvector (5);

int\* p = myvector.data();

\*p = 10;

++p;

\*p = 20;

p[2] = 100;

for (unsigned i = 0; i < myvector.size(); ++i)

cout << ' ' << myvector[i];

return 0;

}

* 1. 10 20 0 100 0
  2. 10 20 0 100
  3. 10 20 0
  4. 10 20

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

int main ()

{

int first[] = {5, 10, 15, 20, 25};

int second[] = {50, 40, 30, 20, 10};

vector<int> v(10);

vector<int> :: iterator it;

sort (first, first + 5);

sort (second, second + 5);

it = set\_union (first, first + 5, second, second + 5, v.begin());

cout << int(it - v.begin());

return 0;

}

* 1. 6
  2. 7
  3. 8
  4. 9

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

int main ()

{

vector<int> myvector (4);

fill (myvector.begin(), myvector.begin() + 2, 3);

fill (myvector.begin() + 1, myvector.end() - 1, 4);

for (vector<int> :: iterator it = myvector.begin(); it != myvector.end(); ++it)

cout << ' ' << \*it;

return 0;

}

* 1. 3 4
  2. 3 4 4
  3. cả a & b
  4. 3 4 4 0

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

int main ()

{

vector<int> myvector;

for (int i = 1; i < 6; ++i)

myvector.push\_back(i);

reverse(myvector.begin(), myvector.end());

for (vector<int> :: iterator it = myvector.begin(); it != myvector.end(); ++it)

cout << ' ' << \*it;

return 0;

}

* 1. 1 2 3 4 5
  2. 5 4 3 2 1
  3. 0 1 2 3 4
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

int main ()

{

int myints[] = {10, 20, 30, 30, 20, 10, 10, 20};

int mycount = count (myints, myints + 8, 10);

cout << "10 appears " << mycount << " times.\n";

vector<int> myvector (myints, myints+8);

mycount = count (myvector.begin(), myvector.end(), 20);

cout << "20 appears " << mycount << " times.\n";

return 0;

}

* 1. 3 3
  2. 3 1
  3. 8
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau:

#include <iostream>

#include <algorithm>

using namespace std;

int main ()

{

int myints[] = {10, 20, 30, 30, 20, 10, 10, 20};

int\* pbegin = myints;

int\* pend = myints + sizeof(myints) / sizeof(int);

pend = remove (pbegin, pend, 20);

for (int\* p = pbegin; p != pend; ++p)

cout << ' ' << \*p;

return 0;

}

* 1. 10, 20, 30, 30, 20, 10, 10, 20
  2. 10, 30, 30, 10, 10
  3. 10, 20, 20, 10, 10, 10, 20
  4. Tất cả đều sai

1. Cho biết kết cả của chương trình sau trong file output:

#include <stdio.h>

int main ()

{

FILE \* p;

char buffer[] = { 'x' , 'y' , 'z' };

p = fopen ( "myfile.txt" , "wb" );

fwrite (buffer , 1 , sizeof(buffer) , p );

fclose (p);

return 0;

}

* 1. xyz
  2. zyx
  3. yxz
  4. Tất cả đều sai

1. Cho biết kết cả của chương trình sau trong file output:

#include <stdio.h>

int main ()

{

char buffer[BUFSIZ];

FILE \*p1, \*p2;

p1 = fopen ("myfile.txt", "w");

p2 = fopen ("myfile2.txt", "a");

setbuf ( p1 , buffer );

fputs ("Buffered stream", p1);

fflush (p1);

setbuf ( p2 , NULL );

fputs ("Unbuffered stream", p2);

fclose (p1);

fclose (p2);

return 0;

}

* 1. Buffered stream
  2. Unbuffered stream
  3. Lỗi
  4. Cả a và b đều đúng

1. Cho biết kết quả của đoạn chương trình sau

#include <iostream>

using namespace std;

int main ()

{

char first, second;

cout << "Enter a word: ";

first = cin.get();

cin.sync();

second = cin.get();

cout << first << endl;

cout << second << endl;

return 0;

}

* 1. first
  2. second
  3. Trả về hai kí tự hoặc số đầu tiên lúc nhập vào
  4. Tất cả đều sai

1. Cho biết kết quả của chương trình sau:

#include<iostream>

#include <fstream>

using namespace std;

int main ()

{

ofstream outfile ("test.txt");

for (int n = 0; n < 100; n++)

{

outfile << n;

outfile.flush();

}

cout << "Done";

outfile.close();

return 0;

}

* 1. Done
  2. Lỗi biên dịch
  3. Lỗi trong quá trình thực thi
  4. Tất cả đều sai

# Chương : Giới thiệu về OOP

## Tổng quát về class, object và tính chất của OOP

1. Một lớp có thể chứa gì
   1. dữ liệu
   2. phương thức
   3. cả a và b
   4. Tất đều sai
2. Về tổng quát, có thể xem đối tượng là
   1. một thành viên của lớp
   2. một sự kết hợp của các lớp
   3. một thuộc tính của lớp
   4. một thể hiện của lớp
3. Một thực thể thể hiện một số các thuộc tính và hành vi gọi là  
   a) object  
   b) class  
   c) struct  
   d) Tất cả đều sai
4. Mỗi đối tượng thuộc cùng 1 lớp
   1. Có một bản copy tất cả các hàm và các thuộc tính của lớp
   2. Có một bản copy tất cả các thuộc tính của lớp
   3. Có một bản copy tất cả các hàm của lớp
   4. Chia sẻ con trỏ trỏ tới các thuộc tính và các hàm của lớp
5. Từ khóa nào không thể xuất hiện trong khi định nghĩa lớp
   1. friend
   2. static
   3. template
   4. virtual
6. Cách nào là chính xác để khai báo một đối tượng của một lớp
   1. Classname Objectname;
   2. Class Classname Objectname;
   3. Class Classname Object Objectname;
   4. Classname Object Objectname;
7. Làm sao để truy cập trường dữ liệu của đối tượng
   1. object@datamember
   2. object\*datamember
   3. object->datamember
   4. object.datamember
8. Về tổng quát trong OOP, cho biểu thức x.y có nghĩa là:
   1. Truy cập thành phần x của đối tượng y
   2. Truy cập thành phần y của đối tượng x
   3. Truy cập thành phần y của đối tượng được trỏ tới bởi x
   4. Tất cả đều sai
9. Điều gì không thể truyền vào một hàm
   1. Array
   2. Reference variable
   3. Object
   4. Header File (.h)
10. Thành phần nào là tùy chọn (không bắt buộc phải có) của một lớp
    1. Constructor
    2. Destructor
    3. Thuộc tính
    4. Phương thức
    5. Cả a, b
    6. Cả a, b, d
    7. Cả a, b, c và d
11. Trong C++, định danh nào sau đây KHÔNG thể dùng để đặt tên cho class
    1. \_ABC
    2. Protected
    3. 1st
    4. Friend
12. Ký tự nào cho phép dùng trong tên lớp
    1. $
    2. @
    3. \_
    4. . (dấu chấm)
13. Trong C++, một lớp không có tên
    1. Không được phép
    2. Không thể có constructor
    3. Không thể có destructor
    4. Tất cả đều sai.
14. Trong C++, câu nào sau đây SAI khi nói về tổ chức file của class:
    1. Mỗi class nên được tổ chức thành 2 file header file (.h) và source file (.cpp)
    2. Trong file .h của một lớp thường include các thư viện và file .h của các lớp có sử dụng trong lớp đó.
    3. Không thể bỏ phần khai báo một class trong source file (.cpp)
    4. Có thể bỏ phần cài đặt phương thức của lớp vào header file (.h)
15. Loại nào sau đây thường KHÔNG NÊN chứa trong các header file (.h)
    1. typedef
    2. Tên lớp và các thuộc tính của lớp
    3. Khai báo khuôn mẫu hàm
    4. Cài đặt hàm
16. Quá trình một lớp được kế thừa từ nhiều lớp được gọi là:
    1. multiple inheritance
    2. multilevel inheritance
    3. default inheritance
    4. multiplex inheritance
17. Khái niệm nào trong OOP có nghĩa là chỉ cung cấp các thông tin cần thiết ra bên ngoài
    1. Đóng gói
    2. Trừu tượng hóa
    3. Che dấu dữ liệu
    4. Ràng buộc dữ liệu
18. Câu nào sau đây đúng khi nói về class và struct
    1. class có thể có hàm thành viên trong khi struct không thể
    2. dữ liệu của class mặc định truy cập public trong khi struct là private
    3. có thể khai báo con trỏ tới class còn struct thì không thể
    4. dữ liệu của class mặc định truy cập private còn struct là public.
19. Cách thông thường để truy cập một trường dữ liệu dạng private của một lớp
    1. Kế thừa lớp đó sau đó truy cập thông qua lớp dẫn xuất
    2. Khai báo một hàm bạn hoăc lớp bạn để lấy quyền truy cập
    3. Truy cập thông qua cặp hàm get/set
    4. Tất cả các cách trên đều đúng.
20. Một lớp có thể có nhiều phương thức có cùng tên nhưng khác danh sách tham số, điều này gọi là:
    1. Encapsulation
    2. Method Overloading
    3. Method Overriding
    4. Tất cả đều sai
21. Trong OOP, một chương trình được chạy dựa trên các :
    1. procedure
    2. method
    3. class
    4. object
22. Khái niệm OOP nào sau đây thể hiện việc gom dữ liệu và phương thức lại với nhau:
    1. Abstraction
    2. Encapsulation
    3. Inheritance
    4. Polymorphism
23. Việc trình biên dịch kiểm tra kiểu thực sự của đối tượng mà con trỏ trỏ tới thay vì kiểm tra kiểm con trỏ thể hiện tính chất gì trong OOP:
    1. Abstraction
    2. Encapsulation
    3. Inheritance
    4. Polymorphism
24. Đặc trưng nào trong OOP cho phép sử dụng lại code
    1. Polymorphism
    2. Inheritance
    3. Encapsulation
    4. Data hiding
25. Đặc trưng của lập trình hướng đối tượng là gì:
    1. Data Encapsulation, Inheritance & Exception handling
    2. Data Encapsulation, Inheritance & Polymorphism
    3. Data Encapsulation, Inheritance, Polymorphism & Exception handling
    4. Data Encapsulation, Inheritance, Polymorphism & Friend
    5. Data Encapsulation, Inheritance, Polymorphism & Overloading
26. Đâu không phải một đặc trưng của OOP
    1. Encapsulation
    2. Abstraction
    3. Polymorphism
    4. Exception
27. Đặc trưng nào sau đây không phải của OOP
    1. Dữ liệu được che dấu và không thể truy cập với các hàm bên ngoài
    2. Dữ liệu và phương thức có thể được thêm mới dễ dàng
    3. Đối tượng có thể tương tác với nhau thông qua hàm
    4. Nhấn mạnh tính thủ tục hơn là dữ liệu
28. Mối quan hệ “has-a” thể hiện qua việc
    1. Một lớp chứa ít nhất một thuộc tính là một đối tượng của lớp khác
    2. Một lớp kế thừa từ một lớp khác
    3. Một lớp có chứa ít nhất một hàm ảo
    4. Một lớp có chứa ít nhất một hàm static.
29. \_\_\_\_ là một mối quan hệ trong OOP
    1. Polymorphism
    2. Inheritance
    3. Overloading
    4. Tất cả đều sai
30. Tính đóng gói (Encapsulation) trong OOP cho phép:
    1. Sử dụng lại và sửa đổi code đã có trước đó
    2. Viết các đoạn code thực hiện các công việc khác nhau với cùng một tên phương thức duy nhất
    3. Che dấu và bảo vệ dữ liệu khỏi việc truy cập không mong muốn
    4. Cả a và b
    5. Cả a, b và c
31. Tính kế thừa (Inheritance) trong OOP cho phép:
    1. Sử dụng lại và sửa đổi code đã có trước đó
    2. Viết các đoạn code thực hiện các công việc khác nhau với cùng một tên phương thức duy nhất
    3. Che dấu và bảo vệ dữ liệu khỏi việc truy cập không mong muốn
    4. Cả a và b
    5. Cả a, b và c
32. Tính đa hình (Inheritance) trong OOP cho phép:
    1. Sử dụng lại và sửa đổi code đã có trước đó
    2. Viết các đoạn code thực hiện các công việc khác nhau với cùng một tên phương thức duy nhất
    3. Che dấu và bảo vệ dữ liệu khỏi việc truy cập không mong muốn
    4. Cả a và b
    5. Cả a, b và c
33. Với cơ chế đa hình (Polyphormism) cho phép:
34. Một phương thức có thể có nhiều tên khác nhau
35. Một đối tượng các thể có nhiều tên khác nhau
36. Nhiều phương thức có các thức hoạt động khác nhau có thể có cùng tên
37. Nhiều đối tượng có thể có cùng tên
38. Tất cả đều sai
39. Đối tượng tạo ra nhờ sử dụng toán tử new sẽ được lưu trữ ở bộ nhớ \_\_\_\_\_\_\_\_\_
40. [Cache](javascript:void(0);)
41. [Heap](javascript:void(0);)
42. [Stack](javascript:void(0);)
43. Tất cả đều sai
44. Trong C++, con trỏ nào sau đây được truyền ngầm vào như tham số đầu tiên của bất kỳ hàm non-static của lớp:
    1. ‘self” pointer
    2. std::auto\_ptr pointer
    3. ‘base’ pointer
    4. ‘this’ pointer
    5. cả A và D đều đúng
    6. cả B và D đều đúng
45. Có 3 lớp khai báo như sau:

class Lop1

{

private:

int a,b;

public:

float x,y;

};

class Lop2

{

public:

Lop1 objA;

};

class Lop3

{

private:

Lop1 objB;

};

Các đối tượng objA và objB truy cập được các biến nào của lớp Lop1:

* 1. objA và objB chỉ truy cập được các biến x,y.
  2. objA và objB truy cập được tất cả các biến.
  3. objA truy cập được tất cả các biến. objB truy cập được các biến x,y.
  4. objA và objB chỉ truy cập được các biến a,b.

1. Hãy xem xét đoạn mã dưới đây:

class Lop1

{

private:

int a,b;

};

class Lop2

{

public:

Lop1 obj1;

void nhap()

{

cout<<"\n\t a, b = ";

cin>>obj1.a>>obj1.b;

};

};

void main()

{

Lop2 obj;

obj.nhap();

}

Khi chạy đoạn chương trình trên sẽ xảy ra hiện tượng gì?

* 1. Báo lỗi không truy cập được vào biến a,b của lớp Lop1 do nằm ở phạm vi private.
  2. Hoàn toàn bình thường. Không có lỗi.
  3. Đối tượng obj không gọi được hàm nhập.
  4. Lỗi do không khai báo kế thừa Lop1 của Lop2.

1. Với một lớp Test được viết như sau:

class Test  
{  
 public:

float x,y;

void nhap()

{

float f1,f2;

cout<<”\n\t Nhap so thu nhat : “;cin>>f1;

cout<<”\n\t Nhap so thu hai : “;cin>>f2;

x = f1 ; y = f2;

}

};

Khi tạo ra đối tượng objTest thì dùng lệnh nào sau đây để nhập giá trị vào cho biến x và biến y.

* 1. objTest.nhap( ).
  2. objTest.nhap(objTest.x, objTest.y).
  3. objTest.x.nhap( ) và objTest.y.nhap( );
  4. objTest.nhap(x) và objTest.nhap(y);

1. Kết quả in ra trên màn hình của chương trình dưới đây là gì?

#include <isotream>

using namespace std;

void main() {

class student {

int rno = 10;

} v;

cout << v.rno;

}

* 1. [10](javascript:void(0);)
  2. Giá trị rác
  3. Lỗi Runtime
  4. [Lỗi biên dịch](javascript:void(0);)

1. Giả sử có định nghĩa lớp như sau:

class YourClass {

public:

YourClass(int newInfo, char moreNewInfo);

YourClass();

void doStuff();

private:

int information;

char moreInformation;  
};

Các câu lệnh nào sau đây là hợp lệ?

1. YourClass anObject(42, ’A’);
2. YourClass anotherObject;
3. anObject = YourClass( );
4. anObject = YourClass;
5. Câu a, b, c
6. Câu a, b, d
7. Câu a, b, c và d
8. Câu Trong khi định nghĩa một hàm cho một lớp với thuộc tính x, câu lệnh nào sau đây KHÔNG tương đương với this🡪x:
9. x
10. (\*this).x
11. \*this.x
12. (\*(& (\*this) ) ).x
13. Cho đoạn code sau:

Class1& test(Class1 obj) {

Class 1 \*ptr = new Class1();

return ptr;

}

Giá trị trả về của hàm test là:

1. đối tượng của lớp Class1
2. tham chiếu tới con trỏ ptr
3. tham chiếu tới Class 1
4. đối tượng được trỏ tới bởi con trỏ ptr.
5. Chương trình sau đây in ra màn hình kết quả gì

#include <iostream>

using namespace std;

class CDummy {

public:

int isitme(CDummy& param);

};

int CDummy::isitme(CDummy& param)

{

if (&param == this)

return true;

else

return false;

}

int main()

{

CDummy a;

CDummy\* b = &a;

if (b->isitme(a)) {

cout << "execute";

}

else {

cout << "not execute";

}

return 0;

}

1. execute
2. not execute
3. không in ra gì
4. in ra cả a và b
5. Kết quả của đoạn code dưới đây là gì: (Giả sử tất cả phần #include và các câu lệnh trong đoạn code đều chính xác)

|  |  |
| --- | --- |
| class MyClass {  public:  MyClass() {}  MyClass(const MyClass& myClass)  {  cout << "\*";  }  MyClass& operator=(const MyClass& myClass)  {  cout << "#";  }  }; | void main()  {  MyClass var1, var2;  MyClass var3 = var2 = var1;  MyClass var4(var3 = var2 = var1);  } |

1. \*\*##\*
2. ##\*##
3. #\*##\*
4. \*\*#\*\*
5. Kết quả của đoạn code dưới đây là gì: (Giả sử tất cả phần #include và các câu lệnh trong đoạn code đều chính xác)

|  |  |
| --- | --- |
| *class PointerClass {*  *public:*  *int\* val;*  *int count;*  *PointerClass(int\* val = NULL)*  *{ this->val = val; }*  *};* | *int main(int argc, char\*\* argv)*  *{*  *int i = 6;*  *PointerClass \*c1, \*c2;*  *c1 = new PointerClass(&i);*  *c2 = c1;*  *i++;*  *\*(c2->val) += 1;*  *delete c2;*  *cout << \*(c1->val);*  *return 0;*  *}* |

1. 8
2. 6
3. 5
4. Tất cả đều sai
5. Kết quả của đoạn code dưới đây là gì: (Giả sử tất cả phần #include và các câu lệnh trong đoạn code đều chính xác)

|  |  |
| --- | --- |
| *class MyClass {*  *private:*  *int val;*  *public:*  *MyClass(int val = 0)*  *: val(2)*  *{*  *this->val = val;*  *}*  *int GetSet(int value = 0)*  *{*  *val = (value != 0) ? value : val;*  *return val + 1;*  *}*  *};* | *int main()*  *{*  *MyClass ins1, ins2(3);*  *MyClass ins3(ins1.GetSet(ins2.GetSet()));*  *cout << ins1.GetSet() << ins2.GetSet() << ins3.GetSet();*  *return 0;*  *}* |

1. 546
2. 456
3. 654
4. 665
5. Câu lệnh nào sau đây sinh ra lỗi biên dịch (Giả sử tất cả phần #include và các câu lệnh trong đoạn code đều chính xác)

|  |  |
| --- | --- |
| *class B {*  *public:*  *A\* a;*  *B() { a = new A(); }*  *void Method() {}*  *};* | *int main()*  *{*  *B\* inst;*  *inst = new B();*  *//…*  *return 0;*  *}* |

1. (\*inst).a->Method();
2. (\*inst)->a->Method();
3. (\*inst).(\*a).Method();
4. (\*(\*inst).a).Method();
5. (\*(inst->a)).Method();

## CONSTRUCTOR, DESTRUCTOR, MEMORY MANAGEMENT

1. Một constructor không có bất kỳ tham số nào gọi là \_\_\_ constructor
   1. custom
   2. dynamic
   3. static
   4. default
2. Nếu khi default constructor không được định nghĩa, khi đó các đối tượng của lớp sẽ được tạo ra bằng cách nào:
   1. Trình biên dịch sẽ báo lỗi
   2. Lỗi sẽ xảy ra trong quá trình chạy run-time
   3. Trình biên dịch sẽ tự tạo ra default constructor để tạo đối tượng
   4. Tất cả đều sai
3. Câu nào đúng khi nói về constructor
   1. Một lớp có thể có nhiều hơn 1 constructor
   2. Constructor có thể được kế thừa
   3. Constructor không thể được khai báo trong vùng protected của lớp
   4. Constructor có thể có số lượng tham số tùy ý
   5. Constructor có thể trả về kiểu dữ liệu void
   6. 1, 2, 3, 4
   7. 1, 2, 4
   8. 1, 3, 4
   9. 1, 2, 3, 5
   10. 1, 2, 3, 4, 5
4. Câu nào sau đây SAI khi nói về constructor
   1. Constructor có thể được khai báo dạng static
   2. Constructor có thể có tham số có giá trị mặc định
   3. Có thể gọi hàm thành viên trong trong constructor
   4. Không có câu nào sai
5. Giả sử có class TEST. Gọi câu lệnh nào sau đây thì copy construct sẽ được chạy
   1. TEST T2(T1);
   2. TEST T4 = T1;
   3. TEST T2; T2 = T1;
   4. cả a và b
   5. tất cả a, b, c
6. Cái nào không phải là một dạng constructor
   1. Copy constructor
   2. Friend constructor
   3. Default constructor
   4. Parameterized constructor
7. Một lớp có thể có nhiều constructor nếu
   1. các constructor có tên khác nhau
   2. các constructor có danh sách tham số khác nhau
   3. các constructor cùng danh sách tham số, khác kiểu trả về
   4. các constructor có khác kiểu dữ liệu trả về
8. Điều gì xảy ra khi tạo ra một đối tượng của một lớp bằng cách không truyền vào bất kỳ tham số nào mà lớp đó được cài đặt chỉ toàn constructor có tham số, không có default constructor
   1. Lỗi biên dịch
   2. Lỗi bộ tiền xử lý
   3. Lỗi khi chạy (run-time)
   4. Chạy bình thường, không có lỗi
9. Nếu người lập trình không cấp rõ ràng hàm constructor thì điều gì giúp tạo ra một constructor mặc định không tham số cho lớp
   1. Bộ tiền xử lý (Preprocessor)
   2. Trình biên dịch (Compiler)
   3. Trình liên kết (Linker)
   4. Hàm main()
10. Một lớp có thể có bao nhiêu constructor mặc định
    1. Duy nhất một
    2. Hai
    3. Ba
    4. Không giới hạn
11. Một constructor là một dạng \_\_\_\_ đặc biệt
    1. Class
    2. Thuộc tính
    3. Phương thức
    4. Biến
12. Một constructor có thể là hàm ảo
    1. Đúng
    2. Sai
13. Nếu X là tên của 1 lớp, cách chính xác để khai báo một copy construtor của X là
    1. X (X arg)
    2. X (X\* arg)
    3. X (const X\* arg)
    4. X(const X & arg)
14. Copy constructor nhận bao nhiêu tham số:
    1. Không tham số
    2. 1 tham số
    3. 2 tham số
    4. Không giới hạn số lượng tham số
15. Câu nào sau đây đúng về Shallow copy
    1. Shallow copy tạo ra một bản sao chép cho các vùng nhớ cấp phát động
    2. Shallow copy đơn giản sao chép giá trị của các trường thuộc tính
    3. Shallow copy tạo ra bản sao chép các thuộc tính static
    4. Cả B và C
16. Câu nào sau đây đúng về Deep copy
    1. Shallow copy tạo ra một bản sao chép cho các vùng nhớ cấp phát động
    2. Shallow copy đơn giản là việc sao chép giá trị của các trường thuộc tính
    3. Shallow copy tạo ra bản sao chép các thuộc tính static
    4. Cả B và C
17. default copy constructor thực hiện cơ chế:
    1. Deep copy
    2. Shallow copy
    3. Hard copy
    4. Soft copy
18. Nếu copy constructor nhận vào tham số bằng cách truyền theo tham trị (pass by value), thì copy constructor sẽ:
    1. Gọi constructor 1 tham số của lớp
    2. Hoạt động bình thường
    3. Gọi đệ quy chính nó
    4. Gọi default constructor
19. Copy constructor được gọi khi nào
20. Khi gán một đối tượng bằng một đối tượng khác khi tạo ra nó
21. Khi đối tượng được truyền vào hàm theo cách truyền bằng tham trị (pass by value)
22. Khi hàm trả về một đối tượng
23. A, B
24. B, C
25. Tất cả đều đúng
26. Một lớp có thể có bao nhiêu destructor mặc định
    1. Duy nhất một
    2. Hai
    3. Ba
    4. Không giới hạn
27. Một destructor là một dạng \_\_\_\_ đặc biệt
    1. Class
    2. Thuộc tính
    3. Phương thức
    4. Biến
28. Một hàm destructor nhận bao nhiêu tham số:
    1. 0
    2. 1
    3. 2
    4. Số lượng tham số tùy ý
29. Điều nào sau đây SAI khi nói về destructor
    1. Destructor không nhận bất kỳ tham số cũng như không có giá trị trả về
    2. Destructor có thể được kế thừa
    3. Destructor có thể gọi hàm thành viên
    4. Destructor được tự động gọi khi đối tượng bị hủy
30. Câu nào sau đây SAI khi nói về default destructor:
    1. Default destructor tự động được cung cấp bởi trình biên dịch
    2. Default destructor không có tham số giống như default constructor
    3. Default destructor tự động hủy vùng nhớ của trường thuộc tính kiểu int
    4. Default destructor tự động hủy tất cả các vùng nhớ của các thuộc tính đã cấp phát trước đó
31. Mặc định thì trình biên dịch sẽ KHÔNG cung cấp mặc định thứ nào sau đây:
    1. Constructor không tham số
    2. Destructor
    3. Copy Constructor
    4. Copy Destructor
32. Nếu người lập trình không cấp rõ ràng hàm destructor thì điều gì giúp tạo ra một destructor mặc định không tham số
    1. Bộ tiền xử lý (Preprocessor)
    2. Trình biên dịch (Compiler)
    3. Trình liên kết (Linker)
    4. Hàm main()
33. Hàm destructor của một lớp:
    1. nên là hàm ảo virtual
    2. không nên là hàm ảo virtual
    3. luôn luôn phải là hàm virtual
    4. không bao giờ là hàm ảo virtual
34. Câu nào sau đây KHÔNG ĐÚNG khi nói về destructor
    1. Destructor sẽ được gọi khi đối tượng ra khỏi vùng tồn tại (out-of-scope)
    2. Giống như constructor, destructor có thể có tham số
    3. Destructor có thể làm hàm ảo
    4. Destructor có thể khai báo ở vùng private
    5. Destructor có cùng tên lớp tên lớp, bắt đầu bằng dấu “-”
    6. 2, 3, 5
    7. 2, 3 ,4
    8. 2, 4, 5
    9. 3, 4, 5
35. Điểm khác biệt giữa constructor và destructor là
    1. Constructor có thể nhận tham số còn destructor thì không
    2. Constructor có thể được overload còn destructor thì không
    3. Cả A và B đều đúng
    4. Cả A và B đều sai
36. Nếu một lớp dùng cấp phát động cho trường thuộc tính, phương thức nào nên được cài đặt trong class:
    1. overloaded assignment operator.
    2. copy constructor.
    3. Destructor
    4. Cả b và c
    5. Cả a, b và c
37. Cho class sau:

*class CreateDestroy {*

*public: CreateDestroy() { cout << "constructor, "; }*

*~CreateDestroy() { cout << "destructor, "; }*

*};*

Kết quả của chương trình sau đây là:

*int main() {  
 for (int i = 1; i <= 2; i++)  
 CreateDestroy c2;  
 return 0;  
}*

* 1. constructor, destructor, constructor, destructor,
  2. constructor, constructor, destructor, destructor
  3. constructor, constructor
  4. Không in ra gì

1. Giả sự đoạn code sau là đoạn bắt đầu định nghĩa constructor của lớp Circle:

*Circle::Circle( double r, int a, int b ) //L1  
: Point( a, b ) //L2*

Dòng L2

* 1. thể hiện sự kế thừa phương thức constructor của lớp Point
  2. gọi constructor của Point với tham số a, b
  3. gây ra lỗi biên dịch
  4. hoàn toàn không cần thiết vì constructor của lớp Point được gọi tự động

1. Cho định nghĩa các class như sau:

*class A { };*

*class B: protected A { };*

Đều gì xảy ra khi biên dịch:

* 1. Không thể biên dịch do thân của class A chưa được cài đặt
  2. Không thể biên dịch do thân của class B chưa được cài đặt
  3. Không thể biên dịch do class B kế thừa từ A không phải kiểu kế thừa public
  4. Biên dịch thành công

1. TRẢ LỜI NGẮN: Việc sử dụng toán tử new để cấp phát động mà không dùng toán tử delete giải phỏng vùng nhớ đã cấp sau khi sử dụng xong gây ra vấn đề gì?
   1. Memory Leak
   2. Diamon problem
   3. Cả hai câu đều đúng
   4. Cả hai câu đều sai

Phần dưới đây mô tả đề bài cho câu 225, 226

Giả sử có class X như sau:

*class X {*

*private:*

*int\* arr;*

*X()*

*{*

*arr = new int[1000];*

*}*

*};*

*Code trong hàm main() như sau:*

*X\*\* p = new X[100];*

*for(int i = 0; i < 100; i++){ p[i] = new X();}*

*// …. code….*

*delete[] p;*

Có 2 lỗi trong cách khai báo class X và code trong hàm main liên quan đến cấp phát bộ nhớ, tìm lỗi và đưa ra các giải quyết.

1. Lỗi trong khai báo class X

*Lỗi không có destructor để hủy vùng nhớ cấp phát động*

*Sửa lỗi: bổ sung destructor ~ X () { detete [] arr;}*

* 1. *Đúng*
  2. *Sai*

1. Lỗi trong hàm main:

*Lỗi: hủy vùng nhớ không triệt để gây memory leak*

*Sửa lỗi: bổ sung các dòng lệnh sau trước dòng lệnh delete [] p;*

*for (int i = 0; i < 100; i++) { delete p[i]; }*

* 1. *Đúng*
  2. *Sai*

## STATIC, FRIEND

1. Có bao nhiêu các để để truy cập thành phần static của lớp trong C++
   1. 1 cách
   2. 2 cách
   3. 3 cách
   4. tất cả đều sai
2. Các dữ liệu tĩnh của một lớp
   1. Chỉ có thể truy cập nếu tồn tại ít nhất một đối tượng của lớp
   2. Không thể thay đổi bởi đối tượng của lớp
   3. Chỉ có thể thay đổi trong các hàm static của lớp
   4. Có hơn 1 cách truy cập
3. Câu nói sau đây SAI khi nói về thành phần static của lớp:
   1. Hàm static chỉ không thể truy cập các thần phần non-static
   2. Hàm non-static có thể truy cập thuộc tính static của lớp
   3. Hàm non-static không thể gọi hàm static
   4. Con trỏ this không tồn tại trong hàm static
4. Nếu một lớp có chứa biến static, tất cả các đối tượng của lớp đó đều chứa một bản sao (copy) của biến static đó
   1. Đúng
   2. Sai
5. Biến static phải được khai báo trong vùng có nhãn truy cập public của lớp
   1. Đúng
   2. Sai
6. Ta có thể sử dụng con trỏ this trong hàm thành viên static của lớp
   1. Đúng
   2. Sai
7. Biến static trong một lớp được khởi tạo khi
   1. Tất cả các đối tượng thuộc lớp đó được khởi tạo
   2. Đối tượng cuối cùng thuộc lớp đó được khởi tạo
   3. Đối tượng đầu tiên thuộc lớp đó được khởi tạo
   4. Biến static không cần phải khởi tạo
8. Cái nào sau đây không thể khai báo là bạn (friend)
   1. Function
   2. Class
   3. Object
   4. Operator function
9. Hàm bạn và lớp bạn là tính chất bắt buộc mà bất kỳ ngôn ngữ lập trình hướng đối tương nào cũng phải hỗ trợ
   1. Đúng
   2. Sai
10. Trong C++, cho đoạn code sau

|  |  |
| --- | --- |
| class Square;  class Rectangle {  int width, height;  public:  int area ()  {return (width \* height);}  void convert (Square a);  }; | class Square {  friend class Rectangle;  private:  int side;  public:  Square (int a) : side(a) {}  }; |

Ta có thể rút ra kết luận gì:

* 1. Các hàm thành viên của Rectange có thể truy cập thành phần private của Square
  2. Các hàm thành viên của Square có thể truy cập thành phần private của Rectangle
  3. Square và Rectangle có thể thoải mái truy cập thành phần private của nhau
  4. Tất cả đều sai

1. Kết quả của đoạn code dưới đây là gì: (Giả sử tất cả phần #include và các câu lệnh trong đoạn code đều chính xác)

|  |  |
| --- | --- |
| *class MyClass {*  *public:*  *static int i;*  *MyClass() { i++;}*  *MyClass(int) {}*  *~MyClass() { i--;}*  *};*  *int MyClass::i = 0;* | *int main(int argc, char\*\* argv)*  *{*  *MyClass \*var1, \*var2, \*var3, var4, var5;*  *var1 = new MyClass();*  *var2 = new MyClass(2);*  *delete (var2);*  *cout << MyClass::i << endl;*  *delete (var1);*  *return 0;*  *};* |

* 1. 1
  2. 2
  3. 3
  4. 4

1. Dòng nào sau đây nên loại bỏ hoặc chuyển thành comment để chương trình có thể chạy (Giả sử tất cả phần #include và các câu lệnh trong đoạn code đều chính xác)

// CL (comment line) line that can be commented

|  |  |
| --- | --- |
| *class MyClass {*  *static MyClass\* instance;*  *static bool isSet;*  *MyClass(){};*  *public:*  *static MyClass\* GetInstance()*  *{*  *if (!!!(isSet)) {*  *instance = new MyClass();*  *isSet = true;*  *}*  *return instance;*  *}*  *~MyClass()*  *{*  *delete instance;*  *}*  *};* | *bool MyClass::isSet = false; // CL1*  *MyClass\* MyClass::instance = NULL; // CL2*  *static bool MyClass::isSet = false; // CL3*  *static MyClass\* MyClass::instance = NULL; // CL4*  *int main()*  *{*  *MyClass c0; // CL5*  *MyClass \*c1, \*c2; // CL6*  *c1 = MyClass::GetInstance(); // CL7*  *c2 = new MyClass(); // CL8*  *return 0;*  *}* |

* 1. CL5, CL8
  2. CL1, CL2, CL6
  3. CL3, CL4, CL5
  4. CL5, CL6 và CL8
  5. CL3, CL4, CL5, CL8
  6. CL1, CL2, CL3 và CL6

1. Kết quả của đoạn code dưới đây là gì: (Giả sử tất cả phần #include và các câu lệnh trong đoạn code đều chính xác)

|  |  |  |
| --- | --- | --- |
| *class MyClass {*  *public:*  *int val;*  *MyClass() { val = 3.14256; }*  *static int valS;*  *MyClass& f1()*  *{*  *val++;*  *return \*this;*  *}* | *MyClass& f1(int)*  *{*  *valS %= 2;*  *val--;*  *return \*this;*  *}*  *MyClass& f2()*  *{*  *val /= 2;*  *return \*this;*  *}*  *MyClass& f2(int)*  *{*  *val \*= 3;*  *return \*this;*  *}*  *};* | *int MyClass::valS = 5;*  *int main()*  *{*  *MyClass inst;*  *inst.f1(inst.valS).f2(MyClass::valS). f1().f1(inst.val).f2().f1(inst.val);*  *cout << inst.val << inst.valS;*  *return 0;*  *}* |

* 1. 22
  2. 12
  3. 21
  4. 11

1. Kết quả của chương trình dưới đây là gì

|  |  |
| --- | --- |
| *#include<isotream>*  *using namespace std;*  *class abc {*  *public:*  *static int x;*  *int i;*  *abc() {*  *i = ++x;*  *}*  *};* | *int abc::x;*  *main() {*  *abc m, n, p;*    *cout<<m.x<<" "<<m.i<<endl;*  *}* |

* 1. 3 1
  2. 3 3
  3. 1 1
  4. 1 3

1. Hãy xem xét đoạn mã sau có lỗi ở dòng nào.
2. *class Lop1*
3. *{*
4. *private:*
5. *int a,b;*
6. *friend void Nhap( );*
7. *};*
8. *class Lop2*
9. *{*
10. *private:*
11. *float x,y;*
12. *friend void Nhap( );*
13. *};*
14. *void nhap( )*
15. *{*
16. *Lop1 obj1; Lop2 obj2;*
17. *cout<<”\n\t Nhap a =”; cin>>obj1.a;*
18. *cout<<”\n\t Nhap x =”; cin>>obj2.x;*
19. *}*
    1. Không lỗi ở dòng nào
    2. Lỗi tại dòng 16, 17
    3. Lỗi tại dòng 15.
    4. Lỗi tại dòng 13.
    5. Lỗi tại dòng 5 và 11

Phần dưới đây mô tả đề bài cho câu 242, 243

Cho đoạn code sau

*class Example {*

*public:*

*Example(int y = 10) : data(y) { // empty body }*

*int getIncrementData() const { return ++data;}*

*static int getCount() {*

*cout << “Data is “ << data << endl;*

*return count;*

*}*

*private:*

*int data;*

*static int count;*

*}; // end class example*

Trong đoạn code trên tồn tại 2 lỗi, chỉ ra lỗi và giải thích ngắn gọn cách sửa lỗi

1. Lỗi số 1

*Error: Phương thức getIncrementedData được khai báo const nhưng thay đổi dữ liệu*

*Correction: Bỏ keyword const trong khai báo getIncrementedData*

* 1. *Đúng*
  2. *Sai*

1. Lỗi số 2

Error: Phương thức getCount được khai báo static nên không truy cập được các thành phần non-static của lớp.  
Correction: Bỏ dòng “ return count;”

* 1. Đúng
  2. Sai

# Chương Kế thừa

1. Khi dẫn xuất một lớp cơ sở theo kế thừa private, các thành viên public của lớp cơ sở trở thành các thành viên \_\_\_\_\_\_\_ của lớp dẫn xuất, các thành viên protected của lớp cơ sở trở thành các thành viên \_\_\_\_\_\_\_\_ của lớp dẫn xuất
2. private private
3. public protected
4. protected protected
5. Chọn phát biểu đúng nhất về quan hệ một -một của hai lớp đối tượng:
6. Một đối tượng thuộc lớp này quan hệ với nhiều đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ duy nhất với một đối tượng thuộc lớp này.
7. Một đối tượng thuộc lớp này quan hệ với một đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ duy nhất với một đối tượng thuộc lớp này.
8. Một đối tượng thuộc lớp này quan hệ với nhiều đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ với nhiều đối tượng thuộc lớp này.
9. Lớp đối tượng này là trường hợp đặc biệt của lớp đối tượng kia và lớp đối tượng kia là trường hợp tổng quát hóa của lớp đối tượng này.
10. Chọn phát biểu đúng nhất về quan hệ một - nhiều giữa hai lớp đối tượng
11. Một đối tượng thuộc lớp này quan hệ với nhiều đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ duy nhất với một đối tượng thuộc lớp này.
12. Một đối tượng thuộc lớp này quan hệ với một đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ duy nhất với một đối tượng thuộc lớp này.
13. Một đối tượng thuộc lớp này quan hệ với nhiều đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ với nhiều đối tượng thuộc lớp này.
14. Lớp đối tượng này là trường hợp đặc biệt của lớp đối tượng kia và lớp đối tượng kia là trường hợp tổng quát hóa của lớp đối tượng này.
15. Chọn phát biểu đúng nhất về quan hệ nhiều - nhiều giữa hai lớp đối tượng
16. Một đối tượng thuộc lớp này quan hệ với nhiều đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ duy nhất với một đối tượng thuộc lớp này.
17. Một đối tượng thuộc lớp này quan hệ với một đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ duy nhất với một đối tượng thuộc lớp này.
18. Một đối tượng thuộc lớp này quan hệ với nhiều đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ với nhiều đối tượng thuộc lớp này.
19. Lớp đối tượng này là trường hợp đặc biệt của lớp đối tượng kia và lớp đối tượng kia là trường hợp tổng quát của lớp đối tượng này.
20. Hai lớp đối tượng được gọi là có quan hệ đặc biệt hóa - tổng quát hóa với nhau khi lớp đối tượng này là trường hợp \_\_\_\_\_\_\_\_ của lớp đối tượng kia và lớp đối tượng kia là trường hợp \_\_\_\_\_\_\_\_ của lớp đối tượng này.
21. Tổng quát - Đặc biệt
22. Đặc biệt - Tổng quát
23. Cả hai câu đều đúng
24. Cả hai câu đều sai
25. Hai lớp đối tượng VO (vợ) và CHONG (chồng) có quan hệ hôn nhân với nhau. Quan hệ giữa hai lớp đối tượng này thuộc loại:
26. Một - một
27. Một - nhiều
28. Nhiều -nhiều
29. Đặc biệt hóa - tổng quát hóa
30. Hai lớp đối tượng XE và BANHXE có quan hệ thuộc loại:
31. Một - một
32. Một - nhiều
33. Nhiều -nhiều
34. Đặc biệt hóa - tổng quát hóa
35. Hai lớp đối tượng CHA và CON có quan hệ thuộc loại:
36. Một - một
37. Một - nhiều
38. Nhiều -nhiều
39. Đặc biệt hóa - tổng quát hóa
40. Hai lớp đối tượng HINHTRON và HINHELIP có quan hệ thuộc loại:
41. Một - một
42. Một - nhiều
43. Nhiều -nhiều
44. Đặc biệt hóa - tổng quát hóa
45. Trong quan hệ đặc biệt hóa- tổng quát hóa giữa lớp đối tượng HINHTHOI và HINHBINHHANH thì :
46. HINHTHOI là trường hợp đặc biệt của HINHBINHHANH
47. HINHBINHHANH là trường hợp đặc biệt của HINHTHOI
48. Cả hai câu đều đúng
49. Cả hai câu đều sai
50. Kế thừa dùng để biểu diễn mối quan hệ \_\_\_\_\_\_\_\_ giữa các lớp.
51. Một - một
52. Một - nhiều
53. Nhiều -nhiều
54. Đặc biệt hóa - tổng quát hóa
55. Hai lớp đối tượng CONGTY và TONGGIAMDOC có quan hệ thuộc loại:
56. Một - một
57. Một - nhiều
58. Nhiều -nhiều
59. Đặc biệt hóa - tổng quát hóa
60. Hai lớp đối tượng TAMGIAC và DAGIAC có quan hệ thuộc loại:
61. Một - một
62. Một - nhiều
63. Nhiều -nhiều
64. Đặc biệt hóa - tổng quát hóa
65. Chọn câu đúng nhất trong các câu sau đây
66. Kế thừa cho phép xây dựng lớp mới từ các lớp đã có
67. Kế thừa cho phép tổ chức các lớp chia sẽ mã chương trình chung
68. Trong C++, nhờ kế thừa, ta có cơ chế chuyển kiểu tự động
69. Cả 3 câu đều đúng
70. Chọn câu đúng:
71. Kế thừa cho phép định nghĩa lớp mới từ các lớp đã có
72. Lớp đã có gọi là lớp con hay lớp dẫn xuất
73. Lớp mới gọi là lớp cha hay lớp cơ sở
74. Cả 3 câu đều đúng
75. Sự kế thừa là một mức cao hơn của \_\_\_\_\_\_\_\_ , cung cấp cơ chế gom chung các lớp có liên quan với nhau thành một mức \_\_\_\_\_\_\_\_ đặc trưng cho toàn bộ các lớp nói trên.
76. Khái quát hóa, trừu tượng hóa
77. Trừu tượng hóa, khái quát hóa
78. Lớp cơ sở, lớp dẫn xuất
79. Lớp dẫn xuất, lớp cơ sở
80. Chọn câu đúng:
81. Nhiều lớp có thể dẫn xuất từ một lớp cơ sở
82. Một lớp có thể là dẫn xuất của nhiều lớp cơ sở
83. Cả hai câu đều đúng
84. Cả hai câu đều sai
85. Một lớp \_\_\_\_\_\_\_\_ có thể là lớp \_\_\_\_\_\_\_\_ cho các lớp \_\_\_\_\_\_\_\_ khác.
86. Cơ sở - dẫn xuất - cơ sở
87. Dẫn xuât - cơ sở - dẫn xuất
88. Cơ sở - dẫn xuất - dẫn xuất
89. Dẫn xuất - cơ sở - cơ sở
90. class <\_\_\_\_\_\_\_\_> : <\_\_\_\_\_\_\_\_> <\_\_\_\_\_\_\_\_>

{

};

1. Tên lớp cơ sở - Loại kế thừa - Tên lớp dẫn xuất
2. Tên lớp cơ sở - Loại kế thừa - Tên lớp cơ sở
3. Tên lớp dẫn xuất - Loại kế thừa - Tên lớp cơ sở
4. Tên lớp trừu tượng - Loại kế thừa - Tên lớp dẫn xuất
5. Lớp kế thừa không thể truy xuất được các thành phần có tầm vực là \_\_\_\_\_\_\_\_ trong lớp cơ sở
6. Public
7. Protected
8. Private
9. Cả 3 câu đều đúng
10. Có bao nhiêu loại kế thừa trong C++:
11. 1
12. 2
13. 3
14. 4
15. Các loại kế thừa trong C++
16. Private
17. Public
18. Protected
19. Các câu đều đúng

Cho đoạn chương trình sau ():

class A

{

private:

int m\_iPrivate;

protected:

int m\_iProtected;

public:

int m\_iPublic;

};

Xét class B : public A {};

Khi đó trạng thái của các thuộc tính trong lớp C sẽ là: \_\_\_\_\_\_\_\_ không thể truy xuất

\_\_\_\_\_\_\_\_ có tầm vực protected.

\_\_\_\_\_\_\_\_ có tầm vực public.

1. m\_iPrivate, m\_iProtected, m\_iPublic
2. m\_iProtected, m\_iPublic, m\_iPrivate
3. m\_iPrivate, m\_iPublic, m\_ iProtected
4. m\_iPrivate, m\_iProtected, m\_iProtected

Cho đoạn chương trình sau ():

class A

{

private:

int m\_iPrivate;

protected:

int m\_iProtected;

public:

int m\_iPublic;

};

Xét class C : protected A {};

Khi đó, trạng thái của các thuộc tính trong lớp C sẽ là:

\_\_\_\_\_\_\_\_ không thể truy xuất

\_\_\_\_\_\_\_\_ có tầm vực protected.

\_\_\_\_\_\_\_\_ có tầm vực protected.

1. m\_iPrivate, m\_iProtected, m\_iPublic
2. m\_iProtected, m\_iPublic, m\_iPrivate
3. m\_iPrivate, m\_iPublic, m\_iProtected
4. Câu a, c đúng
5. Xem xét đoạn mã sau:

class A

{

private:

int a,b;

public:

int c,d;

protected:

int e,f:

};

class B: public class A{};

Hỏi lớp B sử dụng được các biến thành viên nào của lớp A

1. a, b
2. a, b, c, d
3. c, d
4. c, d, e, f

Xem xét đoạn mã sau:

class A

{

private:

int a,b;

public:

int c,d;

protected:

int e,f:

};

class B: private class A{};

Hỏi lớp B sử dụng được các biến thành viên nào của lớp A

1. a, b
2. a, b, c, d
3. c, d
4. c, d, e, f
5. Xem xét đoạn mã sau:

class A

{

private:

int a,b;

public:

int c,d;

protected:

int e,f:

};

class B: private class A{};

class C: public class B{};

Hỏi lớp C sử dụng được các biến thành viên nào của lớp A

1. a, b
2. a, b, c, d
3. c, d, e, f
4. Tất cả câu trên đều sai
5. class A

{

private:

int a,b;

public:

int c,d;

protected:

int e,f:

};

class B: public class A{};

class C: private class B{};

Hỏi lớp C sử dụng được các biến thành viên nào của lớp A

1. a, b
2. a, b, c, d
3. c, d, e, f
4. Tất cả câu trên đều sai
5. class A

{

private:

int a,b;

public:

int c,d;

protected:

int e,f:

};

class B: protected class A{};

class C: public class B{};

Hỏi lớp C sử dụng được các biến thành viên nào của lớp A

1. a, b
2. a, b, c, d
3. c, d, e, f
4. Tất cả câu trên đều sai
5. Định nghĩa hàm dựng nào sau đây là đúng nhất:
6. Hàm có trùng tên với tên lớp, được gọi ngay sau khi khai báo đối tượng, không thể kế thừa.
7. Hàm có trùng tên với tên lớp, được gọi ngay sau khi khai báo đối tượng, cóthể kế thừa.
8. Hàm có trùng tên với tên lớp, được gọi ngay sau khi khai báo đối tượng hoặc con trỏ đối tượng, không thể kế thừa.
9. Hàm có trùng tên với tên lớp, được gọi ngay sau khi khai báo đối tượng hoặc con trỏ đối tượng, có thể kế thừa.
10. Khi một đối tượng thuộc lớp kế thừa được tạo lập, hàm dựng của lớp \_\_\_\_\_\_\_\_ được gọi thực hiện trước, sau đó mới đến hàm dựng của lớp \_\_\_\_\_\_\_\_.
11. Cơ sở, dẫn xuất
12. Dẫn xuất, cơ sở
13. Dẫn xuất, dẫn xuất
14. Tất cả đều đúng
15. Khi một đối tượng thuộc lớp kế thừa bị hủy, hàm hủy của lớp \_\_\_\_\_\_\_\_ được gọi thực hiện trước, sau đó mới đến hàm hủy của lớp \_\_\_\_\_\_\_\_.
16. Dẫn xuất, cơ sở
17. Cơ sở, Dẫn xuất
18. Trừu tượng, Khái quát
19. Tất cả đều đúng
20. Con trỏ trỏ dến đối tượng thuộc lớp \_\_\_\_\_\_\_\_ thì có thể trỏ đến đối tượng thuộc lớp \_\_\_\_\_\_\_\_.
21. Dẫn xuất, cơ sở
22. Cơ sở, Dẫn xuất
23. Bạn, cơ sở
24. Tất cả đều đúng
25. Chọn câu đúng:
26. Con trỏ trỏ dến đối tượng thuộc lớp dẫn xuất thì có thể trỏ đến các đối tượng thuộc lớp cơ sở.
27. Con trỏ trỏ đến đối tượng thuộc lớp cơ sở thì không thể trỏ đến các đối tượng thuộc lớp dẫn xuất.
28. Đa kế thừa cho phép một lớp có thể là dẫn xuất của nhiều lớp cơ sở.
29. Cả 3 câu đều đúng.
30. Việc kế thừa giữa 2 lớp chỉ nên được thực hiện khi và chỉ khi giữa chúng có quan hệ \_\_\_\_\_\_\_\_, có nghĩa là lớp này là một trường hợp đặc biệt của lớp kia.
31. Is -a
32. Has -a
33. Một - một
34. Một - nhiều
35. Hai lớp được gọi là có quan hệ \_\_\_\_\_\_\_\_ khi thành phần thuộc tính của lớp này có chứa đối tượng của lớp kia.
36. Is- a
37. Has -a
38. Một - một
39. Một - nhiều
40. Quan hệ \_\_\_\_\_\_\_\_ còn được gọi là quan hệ “bao hàm” hoặc quan hệ “bộ phận – toàn thể”.
41. Is- a
42. Has -a
43. Một - một
44. Đặc biệt hóa - Tổng quát hóa
45. Quan hệ \_\_\_\_\_\_\_\_ không thể được sử dụng để thay thế cho quan hệ \_\_\_\_\_\_\_\_ .
46. Is -a, has- a
47. Has- a, is -a
48. Ca hai câu đều đúng
49. Cả hai câu đều sai
50. class A

{

};

class B: public class A{};

class C: private class B{};

void main()

{

B \*pb;

C c;

A a;

}

Câu lệnh nào sau đây là đúng:

1. pb = & c;
2. pb= & a;
3. C = &a;
4. Cả 3 câu đều đúng
5. class A

{

};

class B: public class A{};

class C: private class B{};

void main()

{

C \*pc;

B b;

A a;

}

Câu lệnh nào sau đây là đúng:

1. pc = & b;
2. pc= & a;
3. Cả 2 câu đều đúng
4. Cả 2 câu đều sai

class A

{

};

class B: public class A{};

class C: private class B{};

void main()

{

C c;

B b;

A \* pa;

}

Câu lệnh nào sau đây là đúng:

1. a= & b;
2. a= & c;
3. Cả 2 câu đều đúng
4. Cả 2 câu đều sai
5. .Toán tử :: được gọi là
6. Toán tử truy vấn dữ liệu
7. Toán tử hai chấm
8. Toán tử phân giải miền xác định
9. Toán tử gán
10. Toán tử :: được dùng để
11. Truy vấn dữ liệu
12. Gán dữ liệu cho một biến
13. Để truy xuất đến thành phần của lớp cơ sở từ lớp kế thừa
14. Không làm gì cả
15. class A

{

public : int a;

};

void main()

{

A a;

a.a =5;

}

Toán tử . được dùng để:

1. Gán giá trị cho biến
2. Truy xuất đến thành phần của lớp cơ sở từ lớp kế thừa
3. Không làm gì cả
4. Xác định thnh phần của lớp
5. Chọn câu đúng

class A {};

class B : public A{};

1. Chương trình báo lỗi
2. Chương trình báo lỗi do phần thân lớp cơ sở chưa được định nghĩa
3. Chương trình chạy bình thường
4. Cả 3 câu đều sai
5. Từ khóa friend không xuất hiện trong
6. Phần private của một lớp
7. Phần public của một lớp.
8. Cả hai câu đều đúng
9. Cả hai câu đều sai
10. Một hàm friend của lớp C không thể truy cập
11. Các thành viên dữ liệu của lớp dẫn xuất của C
12. Các thành viên dữ liệu và các hàm thành viên public của C
13. Các thành viên dữ liệu và các hàm thành viên private của C
14. Các thành viên dữ liệu và các hàm thành viên protected của C
15. Kế thừa ảo dùng để giải quyết vấn đề gì:
16. Vấn đề hình thoi trong đa kế thừa
17. Tránh mất mát bữ liệu bộ nhớ
18. Cả hai câu đều đúng
19. Cả hai câu đều sai
20. Con trỏ lớp cơ sở có thể giữ địa chỉ của
21. đối tượng lớp cơ sở cũng như đối tượng lớp dẫn xuất
22. chỉ các đối tượng lớp dẫn xuất
23. chỉ các đối tượng lớp cơ sở
24. tất cả các phương án đều sai
25. Khai báo nào sau đây là hợp lệ cho lớp dẫn xuất d với lớp cơ sở b1 và b2
26. class d : public b1, b2
27. class d : b1, b2
28. class d : public b1, public b2
29. Cả 3 câu trên đều đúng
30. Chọn câu đúng:
    1. Các lớp cơ sở thì cụ thể hơn các lớp dẫn xuất
    2. C++ cung cấp 3 kiểu kế thừa: private, protected và public
    3. Một lớp dẫn xuất không thể đóng vai trò là một lớp cơ sở cho một lớp khác
    4. Hàm friend của lớp cơ sở không thể truy cập các thành viên protected
31. Câu nào sai:
    1. Các hàm thành viên lớp dẫn xuất có thể truy cập đến các thành phần public và protected của lớp cơ sở
    2. Các hàm của lớp cơ sở không thể được override trong lớp dẫn xuất
    3. Các hàm có thể được override trong nhiều cấp của cây kế thừa
    4. Khi dẫn xuất từ một lớp cơ sở public, các thành viên public của lớp cơ sở trở thành các thành viên public của lớp dẫn xuất
32. Chọn tất cả các câu đúng:
    1. Khi dẫn xuất từ một lớp cơ sở public, các thành viên protected của lớp cơ sở trở thành các thành viên của protected của lớp dẫn xuất
    2. Khi dẫn xuất từ một lớp cơ sở protected, các thành viên protected và public của lớp cơ sở trở thành các thành viên private của lớp dẫn xuất
    3. Khi dẫn xuất từ một lớp cơ sở private, các thành viên private của lớp cơ sở trở thành các thành viên private của lớp dẫn xuất
    4. Khi dẫn xuất từ một lớp cơ sở public, các thành viên public của lớp cơ sở trở thành các thành viên protected của lớp dẫn xuất
33. Chọn tất cả các câu đúng:
    1. Constructor của lớp cơ sở được kế thừa
    2. Các constructor của lớp dẫn xuất phải gọi các constructor của lớp cơ sở
    3. Destructor của lớp dẫn xuất được gọi trước destructor của lớp cơ sở
    4. Khi thừa kế từ 2 lớp cơ sở, chỉ có constructor của một lớp cơ sở được gọi
34. Nếu lớp Alpha thừa kế từ lớp Beta, lớp Alpha được gọi là \_\_\_\_\_\_, lớp Beta được gọi là \_\_\_\_\_\_\_\_\_
    1. lớp cơ sở lớp dẫn xuất
    2. lớp dẫn xuất lớp cơ sở
    3. lớp trừu tượng lớp cơ sở
    4. lớp dẫn xuất lớp trừu tượng
35. C++ cung cấp \_\_\_\_\_\_\_ cho phép một lớp dẫn xuất kế thừa từ nhiều lớp cơ sở
    1. đa hình
    2. đa kế thừa
    3. đa nhiệm
36. Một đối tượng của lớp \_\_\_\_\_\_\_ có thể được xem như một đối tượng của lớp \_\_\_\_\_\_ tương ứng
    1. dẫn xuất cơ sở
    2. cơ sở dẫn xuất
    3. cơ sở trừu tượng
    4. trừu tượng dẫn xuất
37. Khi dẫn xuất một lớp cơ sở theo kế thừa public, các thành viên public của lớp cơ sở trở thành các thành viên \_\_\_\_\_\_\_ của lớp dẫn xuất, các thành viên protected của lớp cơ sở trở thành các thành viên \_\_\_\_\_\_\_\_ của lớp dẫn xuất
    1. protected private
    2. public protected
    3. protected protected
38. Khi dẫn xuất một lớp cơ sở theo kế thừa protected, các thành viên public của lớp cơ sở trở thành các thành viên \_\_\_\_\_\_\_ của lớp dẫn xuất, các thành viên protected của lớp cơ sở trở thành các thành viên \_\_\_\_\_\_\_\_ của lớp dẫn xuất
    1. protected private
    2. public protected
    3. protected protected
39. Một lớp dẫn xuất thường được gọi là lớp con vì nó đóng vai trò như một tập con của lớp cơ sở. Ví dụ: một lớp dẫn xuất nhìn chung nhỏ hơn lớp cơ sở.
    1. Đúng
    2. Sai
40. Một đối tượng của lớp dẫn xuất cũng là một đối tượng của lớp cơ sở
    1. Đúng
    2. Sai
41. Thừa kế là phương pháp để
    1. Chuyển tham số sang các đối tượng của lớp.
    2. Thêm các đặc trưng vào các lớp đã tồn tại mà không cần viết lại chúng.
    3. Tăng cường tính đóng gói và che giấu thông tin.
    4. Tổ chức dữ liệu.
42. Sự khác nhau giữa private và protected?
    1. Lớp dẫn xuất không thể truy xuất các thành phần private, nhưng có thể truy xuất các thành phần protected.
    2. Thành phần protected của lớp cơ sở có thể được truy xuất ở lớp chứa lớp cơ sở như là lớp bạn, còn thành phần private thì không.
    3. Các thành phần private và protected đều có thể truy xuất được từ lớp dẫn xuất.
    4. Các thành phần protected không thể truy xuất được từ lớp dẫn xuất, còn các thành phần private thì được.
43. Xét đoạn chương trình sau

class A {

public:

int x;

A() {x = 0;}

};

class B : public A {};

class C : public A {};

class D : public C, public B {};

int main () {

D obj;

}

* 1. Lỗi thiếu dấu chấm phẩy.
  2. Hai lớp không thể được dẫn xuất từ một lớp cơ sở.
  3. Lớp không được định nghĩa rỗng.
  4. Chương trình không bị lỗi

1. Trong chương trình sau:

#include<iostream.h>

class base {

private:

int x;

public:

void setx(int n) {x = n;}

void showx() {cout << x;}

};

class inherited: public base{

int y;

public:

void sety(int n) {y = n;}

void showy() {cout << y;}

};

int main() {

inherited obj; obj.setx(10);

obj.sety(23); obj.x = 12;

obj.showx(); obj.showy();

return 0;

}

Đoạn chương trình trên bị lỗi bởi mệnh đề nào?

* 1. obj.setx(10);
  2. obj.x = 12;
  3. obj.showy();
  4. Không có lỗi

1. Trong chương trình sau:

#include<iostream.h>

class A {public:

A() {cout << "Constructing A ";}

~A() {cout << "Destructing A ";}

};

class B: public A {public:

B() {cout << "Constructing B ";}

~B() {cout << "Destructing B ";}

};

int main() {

B obj;

return 0;

}

Đoạn chương trình in ra màn hình cái gì?

* 1. Constructing A Destructing A Constructing B Destructing B
  2. Constructing A Constructing B Destructing A Destructing B
  3. Constructing A Constructing B Destructing B Destructing A
  4. Constructing B Constructing A Destructing A Destructing B

1. Trong kế thừa, mối quan hệ giữa lớp dẫn xuất và lớp cơ sở
   1. Là mối quan hệ “is-a”
   2. Là mối quan hệ “has-a”
   3. Là mối quan hệ kết hợp các thuộc tính
   4. Không có mối quan hệ nào
2. Cho đoạn chương trình sau

class A {

private:

int x,y;

};

class B: public A {

public:

int z;

};

void main()

{

B b;

b.x = 1; // Lệnh L1

b.y = 2; // Lệnh L2

b.z = 3; // Lệnh L3

}

* 1. Lệnh L1, L2 sai, lệnh L3 đúng
  2. Các lệnh L1, L2 và L3 đều sai
  3. Lệnh L1 sai, lệnh L2, L3 đúng
  4. Các lệnh L1, L2 và L3 đều đúng

1. Khi thực thi đoạn chương trình sau kết quả sẽ là

class Base{

public:

int xVal;

Base(int x=5){

xVal =x;

cout<<"xVal = "<<xVal<<endl;

}

};

class Derive: public Base

{

public: Derive(){xVal = 10;}

void Print(){

cout<<"xVal = "<<xVal<<endl;

}

};

void main(){

Derive d;

d.Print();

}

* 1. Màn hình xuất hiện: xVal = 5 xVal = 10
  2. Màn hình xuất hiện: xVal = 50
  3. Màn hình xuất hiện: xVal = 10
  4. Chương trình bị lỗi

1. Khi thực thi đoạn chương trình sau kết quả sẽ là

class Base{

public:

~Base(){ cout<<"Base class"<<endl; }

};

class Derive:Base

{

public: ~Derive(){ cout<<"Derive class"<<endl; }

};

void main(){

Base b;

Derive d;

}

* 1. Derive class Base class Base class
  2. Derive class Base class
  3. Base class Derive class
  4. Base class Derive class Base class

1. Khi thực thi đoạn chương trình sau kết quả sẽ là

class BaseA{

protected:int A;

public:

BaseA(){ A = 5; }

void Print(){ cout<<"A = "<<A<<endl; }

};

class BaseB {

protected:int B;

public:

BaseB(){ B = 10; }

void Print(){ cout<<"B = "<<B<<endl; }};

class Derive:public BaseA,public BaseB {};

void main()

{

Derive d;

d.Print();

}

* 1. Chương trình báo lỗi
  2. Màn hình xuất hiện: A = 5
  3. Màn hình xuất hiện: B = 10
  4. Màn hình xuất hiện: A = 5 B = 10

1. Thế nào là overriding
   1. Lớp dẫn xuất định nghĩa một hàm cùng tên nhưng khác kiểu trả về với một hàm ở lớp cơ sở
   2. Lớp dẫn xuất định nghĩa một hàm cùng tên, cùng kiểu trả về nhưng khác các đối số với một hàm ở lớp cơ sở.
   3. Lớp dẫn xuất định nghĩa một hàm hoàn toàn giống lớp cơ sở
   4. Lớp dẫn xuất định nghĩa một hàm cùng tên, cùng các đối số nhưng khác kiểu trả về với một hàm ở lớp cơ sở
2. Quá trình xây dựng lớp mới từ một lớp ban đầu gọi là
   1. Đa hình
   2. Cấu trúc
   3. Kế thừa
   4. Phân tầng
3. Xem xét đoạn mã sau:

class F{

protected:

int a,b;

private:

float c, d;

}

class C: public F{

}

C sử dụng được các biến thành viên nào của F

* 1. a, b, c, d
  2. c, d
  3. a, b
  4. không sử dụng được biến thành viên nào

1. Khi dùng từ khóa protected cho một thành phần trong một lớp
   1. Thành phần này chỉ có thể được truy cập bởi chính lớp đó
   2. Thành phần này có thể được truy cập ở bất cứ đâu
   3. Thành phần này chỉ có thể được truy cập bởi chính lớp đó và lớp dẫn xuất của nó
   4. Thành phần này không thể được kế thừa
2. Cho đoạn mã sau đây

class A

{

public:

int a, b;

void nhap( )

{

cout << “a = ”; cin>>a;

cout << “b = ” ; cin>>b;

tinhtoan( );

}

void tinhtoan( )

{

cout << a +b;

}

};

class B: public A

{

void tinhtoan( )

{

cout << a\*b;

}

};

Nếu tạo ra đối tượng objB thuộc lớp B, khi gọi hàm nhap( ) và nhập a=13 và b=2 thì kết quả

* 1. 15
  2. 26
  3. Báo lỗi biên dịch
  4. Báo lỗi thực thi

1. Cho đoạn mã sau đây:

class A

{

public:

int a,b,c,d;

void nhap()

{

cout<<"\n\t a =";cin>>a;

}

};

class B:A

{

public:

A \*pA;

void nhap()

{

cout<<"\n\t b =";cin>>b;

}

};

class C:A

{

public:

B \*pB;

void nhap()

{

cout<<"\n\t c =";cin>>c;

}

};

class D:A

{

public:

C \*pC;

void nhap()

{

cout<<"\n\t d =";cin>>d;

}

};

Nếu tạo ra một đối tượng con trỏ \*objX nào đó thuộc lớp D, để gọi hàm nhap( ) của lớp C thì dùng lệnh nào sau đây:

* 1. objX->pC->nhap();
  2. objX->nhap();
  3. objX->pC->pB->nhap();
  4. objX->pC->pB->pA->nhap();

1. Quyền truy cập đối với các thành phần trong lớp dẫn xuất khi nó kế thừa private từ lớp cơ sở được xác định thế nào?
   1. Giữ lại tất cả các quyền truy cập ban đầu trong lớp cơ sở
   2. Protected chuyển thành private, public chuyển thành private
   3. Protected chuyển thành private, public chuyển thành protected
   4. Tất cả đều sai
2. Trong C++, Ta có thể override quyền truy cập đối với
   1. Các thành phần public
   2. Các thành phần public và protected
   3. Bất cứ thành phần cụ thể nào
   4. Không thể thực hiện override quyền truy cập
3. Cho đoạn chương trình sau, chọn câu đúng:

class A {

private:

int x;

public:

int y;

};

class B: public A {

public:

int z;

};

class C: private A {

protected:

int t;

};

void main()

{

B c;

C b;

b.x = 0; // Lệnh L1

c.x = 1; // Lệnh L2

c.y = b.z; // Lệnh L3

}

* 1. Lệnh L2 và L3 đúng, lệnh L1 sai
  2. Lệnh L2 đúng, lệnh L1 và L3 sai,
  3. Lệnh L1 và L3 đúng, lệnh L2 sai
  4. Cả ba lệnh đều sai

1. Các thành phần nào trong một lớp chỉ được dùng trong chuỗi phân cấp lớp
   1. Các thành phần có quyền truy cập là public
   2. Các thành phần có quyền truy cập là private
   3. Các thành phần có quyền truy cập là protected
   4. Các hàm thành viên.
2. Điều gì xảy ra nếu lớp cơ sở và lớp dẫn xuất cùng chứa một hàm có cùng khai báo
   1. Trình biên dịch sẽ báo lỗi
   2. Chỉ hàm của lớp cơ sở sẽ được gọi
   3. Chỉ hàm của lớp dẫn xuất sẽ được gọi
   4. Đối tượng của lớp cơ sở sẽ gọi hàm được định nghĩa ở lớp cơ sở, đối tượng của lớp dẫn xuất sẽ gọi hàm được định nghĩa ở lớp dẫn xuất
3. Cho đoạn mã sau đây

class A

{

public:

int a, b;

void nhap( )

{

cout << “a = ”; cin>>a;

cout << “b = ” ; cin>>b;

tinhtoan( );

}

void tinhtoan( )

{

cout << “Tong 2 so a, b:” << a +b;

}

};

class B: public A

{

void tinhtoan( )

{

cout << “Tich 2 so la” << a\*b;

}

};

Nếu tạo ra đối tượng objB thuộc lớp B, khi gọi hàm nhap( ) thì hàm tinhtoan của lớp nào sẽ được gọi

* 1. Lớp A
  2. Lớp B
  3. Của cả hai lớp
  4. Không hàm tinhtoan nào được gọi vì trùng tên.

1. Cho đoạn mã sau, hãy chọn câu đúng:

class A {

private:

int x;

public:

int y;

};

class B: public A {

public:

int z;

};

class C: private A {

protected:

int t;

};

void main()

{

B c;

C b;

b.x = 0; // Lệnh L1

c.x = 1; // Lệnh L2

c.y = b.z; // Lệnh L3

}

* 1. Lệnh L2 và L3 đúng, lệnh L1 sai
  2. Lệnh L2 đúng, lệnh L1 và L3 sai,
  3. Lệnh L1 và L3 đúng, lệnh L2 sai
  4. Cả 3 lệnh đều sai

1. Xem xét đoạn mã sau:

class A{

protected:

int a,b;

public:

float c, d;

}

public B: public A{

}

B sử dụng được các biến thành viên nào của A

* 1. a, b, c, d
  2. c, d
  3. a, b
  4. không sử dụng được biến thành viên nào

1. Khi khai báo kế thừa một lớp từ khóa nào sau đây là không hợp lệ
   1. Public
   2. Private
   3. Protected
   4. Friend
2. Các thành phần của lớp cơ sở có quyền truy cập là private thì được truy cập bởi:
   1. Các phương thức của lớp cơ sở và các lớp dẫn xuất
   2. Các phương thức của lớp cơ sở và các hàm bạn
   3. Các phương thức của lớp cơ sở, các lớp dẫn xuất và các hàm bạn bất kỳ
   4. Ở bất kì đâu
3. Nếu một lớp được định nghĩa để làm lớp cơ sở, hầu hết các thành phần dữ liệu của nó sẽ có kiểu là:
   1. Private
   2. Public
   3. Protected
   4. Polymorphic
4. Xem xét đoạn mã sau:

class A{

protected: int A;

public: A (){ A = 100; }

void Print(){ cout<<"A = "<<A<<endl; }

};

class B {

protected: int B;

public:B(){ B = 200; }

void Print(){ cout<<"B = "<<B<<endl; }};

class C:public A, public B {};

void main()

{

C c;

c.Print();

}

Kết quả của chương trình là gì?

* 1. Chương trình báo lỗi
  2. Màn hình xuất hiện: A = 100
  3. Màn hình xuất hiện: B = 200
  4. Màn hình xuất hiện: A = 100 B = 200

1. Để một thành phần dữ liệu được sử dụng bên ngoài lớp cơ sở và các dẫn xuất của nó, ta phải định nghĩa thành phần này trong phần nào của lớp
   1. common
   2. public
   3. exposed
   4. unrestricted
2. Loại kế thừa nào được sử dụng trong khai báo dưới đây

class A: public X, public Y{}

* 1. Kế thừa nhiều cấp (Multilevel Inheritance)
  2. Đa kế thừa (Multiple Inheritance)
  3. Kế thừa lai (Hybrid Inheritance)
  4. Kế thừa phân cấp (Hierarchical Inheritance)

1. Cơ chế tạo một lớp dẫn xuất từ một lớp dẫn xuất khác được gọi là
   1. Đa hình (Polymorphism)
   2. Đơn kế thừa (Single Inheritance)
   3. Đa kế thừa (Multiple Inheritance)
   4. Kế thừa nhiều cấp (Multilevel Inheritance)
2. Class C kế thừa class B, class B kế thừa class A, đều là kế thừa public. Vậy các phương thức của class C có thể truy cập
   1. Chỉ thành phần dữ liệu public và protected trong C và B
   2. Chỉ thành phần dữ liệu public và protected trong C
   3. thành phần dữ liệu private trong A và B
   4. thành phần dữ liệu protected trong A và B

Chương : Đa hình

1. \_\_\_\_\_\_\_\_ là hàm thành viên được khai báo trong lớp cha và được tái định nghĩa lại ở lớp con.
2. Hàm ảo
3. Hàm static
4. Hàm bạn
5. Hàm thành viên hằng
6. Đa hình runtime chỉ đạt được khi một hàm ảo được truy cập thông qua con trỏ lớp cơ sở.
7. Đúng
8. Sai
9. Ta có thể tạo đối tượng cho một lớp trừu tượng.
10. Đúng
11. Sai
12. Khi một hàm ảo được tái định nghĩa lại ở lớp dẫn xuất, được gọi là :
13. Overloading
14. Overriding
15. Rewriting
16. Tất cả đều đúng
17. Nếu một lớp là dẫn xuất của một lớp trừu tượng, khi đó:
18. Lớp dẫn xuất phải định nghĩa tất cả các hàm thuần ảo.
19. Lớp dẫn xuất cũng sẽ trở thành lớp trừu tượng nếu nó không định nghĩa một hàm thuần ảo nào đó ở lớp cha.
20. Không thể tạo đối tượng cho lớp dẫn xuất nếu nó không định nghĩa một hàm thuần ảo nào đó ở lớp cha.
21. Tât cả đều đúng
22. Một hàm ảo mà không có phần định nghĩa ở lớp cơ sở được gọi là:
23. Pure virtual function
24. Pure static function
25. Pure Const function
26. Friend function
27. Chọn câu sai khi nói về hàm lớp trừu tượng:
28. Ta không thể tạo đối tượng cho lớp trừu tượng..
29. Ta không thể tạo con trỏ cho lớp trừu tượng.
30. Lớp trừu tượng chứa ít nhất một hàm thuần ảo.
31. Cần phải cài đặt lại các hàm thuần ảo ở lớp dẫn xuất.
32. Ta xây dựng một hàm thành phần là phương thức ảo bằng cách thêm từ khóa \_\_\_\_\_\_\_\_ vào trước khai báo hàm.
33. static
34. virtual
35. abstract
36. const
37. Phương thức ảo chỉ hoạt động thông qua con trỏ.
38. Đúng
39. Sai
40. Một hàm là phương thức ảo.
41. Khai báo với từ khóa virtual
42. Phương thức tương ứng ở lớp cơ sở đã là phương thức ảo
43. Cả hai câu đều đúng
44. Cả hai câu đều sai
45. Khi gọi một thao tác, khả năng chọn đúng phiên bản tùy theo đối tượng để thực hiện thông qua con trỏ đến lớp cơ sở được gọi là \_\_\_\_\_\_\_\_
46. Tính đa dạng
47. Tính đa hình
48. Tính phổ biến
49. Cả 3 câu trên đều đúng
50. Phương thức ảo có thể là các hàm thành viên tĩnh.
51. Đúng
52. Sai
53. Cần thiết phải có từ khóa virtual khi định nghĩa một phương thức ảo trong lớp dẫn xuất.
54. Đúng
55. sai
56. Một phương thức ảo có thể được khai báo là friend trong một lớp khác nhưng các hàm friend của lớp thì không thể là phương thức ảo.
57. Đúng
58. Sai
59. Lớp cơ sở trừu tượng là lớp không có đối tượng nào thuộc chính nó.
60. Đúng
61. Sai
62. Trong một cây kế thừa đơn cấp, lớp trừu tượng được sử dụng ở đâu?
63. Lớp cha
64. Lớp con
65. Cả hai câu trên đều đúng
66. Cả hai câu đều sai
67. Chọn câu đúng nhất về hàm thuần ảo:
68. Là hàm chỉ có khai báo không có định nghĩa
69. Là hàm chỉ có định nghĩa không có khai báo
70. Là hàm không có kiểu trả về
71. Câu A, C đúng, B sai.
72. Lớp con của các lớp cơ sở trừu tượng không thể là lớp cơ sở trừu tượng.
73. Đúng
74. Sai
75. chúng ta có thể truy xuất lớp kế thừa thông qua con trỏ của lớp cơ sở.
76. Đúng
77. Sai

class Animal

{

public:

void Talk()

{

cout << "Don't know how to talk!" << endl;

}

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

void main()

{

Cat obj;

Animal \*p = &obj;

p->Talk();

}

Kết quả của chương trình:

1. Meo meo!
2. Don't know how to talk!
3. Cả hai câu đều đúng
4. Cả hai câu đều sai
5. class Animal

{

public:

void Talk()

{

cout << "Don't know how to talk!" << endl;

}

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

void main()

{

Animal obj;

Cat \*p = &obj;

p->Talk();

}

Kết quả của chương trình:

1. Meo meo!
2. Don't know how to talk!
3. Cả hai câu đều đúng
4. Cả hai câu đều sai
5. class Animal

{

public:

virtual void Talk()

{

cout << "Don't know how to talk!" << endl;

}

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

void main()

{

Cat obj;

Animal \*p = &obj;

p->Talk();

}

Kết quả của chương trình:

1. Meo meo!
2. Don't know how to talk!
3. Cả hai câu đều đúng
4. Cả hai câu đều sai
5. class Animal

{

public:

virtual void Talk() =0

{

cout << "Don't know how to talk!" << endl;

}

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

void main()

{

Cat obj;

Animal \*p = &obj;

p->Talk();

}

Kết quả của chương trình:

1. Meo meo!
2. Don't know how to talk!
3. Cả hai câu đều đúng
4. Cả hai câu đều sai
5. class Animal

{

public:

virtual void Talk() =0;

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

void main()

{

Animal obj;

Cat \*p = &obj;

p->Talk();

}

Kết quả của chương trình:

1. Meo meo!
2. Don't know how to talk!
3. Cả hai câu đều đúng
4. Cả hai câu đều sai
5. class Animal

{

public:

virtual void Talk() =0;

};

class Cat: public Animal

{

};

void main()

{

Cat obj;

Animal \*p = &obj;

p->Talk();

}

Kết quả của chương trình:

1. Meo meo!
2. Don't know how to talk!
3. Cả hai câu đều đúng
4. Cả hai câu đều sai
5. class Animal

{

public:

virtual void Talk() =0;

};

void main()

{

Animal \*m = new Animal;

}

1. Đúng
2. Sai
3. class Animal

{

public:

virtual void Talk() = 0;

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

class Dog: public Animal

{

public:

void Talk()

{

cout << "Gau gau!" << endl;

}

};

void main()

{

Animal \* a = new Animal;

Animal \*b=new Dog;

Animal \*c = new Cat;

}

1. Đúng
2. Sai
3. class Animal

{

public:

virtual void Talk() = 0;

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

class Dog: public Animal

{

public:

void Talk()

{

cout << "Gau gau!" << endl;

}

};

void main()

{

Animal \*\* a;

a[0]= new Animal;

a[1]= new Cat;

}

1. Đúng
2. Sai
3. class Animal

{

public:

virtual void Talk() = 0;

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

class Dog: public Animal

{

public:

void Talk()

{

cout << "Gau gau!" << endl;

}

};

void main()

{

Animal \* a = new Cat;

a.Talk();

}

Kết quả của chương trình:

1. Meo meo!
2. Don't know how to talk!
3. Cả hai câu đều đúng
4. Cả hai câu đều sai
5. class Animal

{

public:

virtual Animal()

{

};

virtual void Talk() =0

{

cout << "Don't know how to talk!" << endl;

}

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

void main()

{

Cat obj;

Animal \*p = &obj;

p->Talk();

}

Kết quả của chương trình:

1. Meo meo!
2. Don't know how to talk!
3. Cả hai câu đều đúng
4. Cả hai câu đều sai
5. Liên kết tĩnh:
6. Lời gọi phương thức được quyết định ngay khi biên dịch.
7. Nếu có lỗi sẽ phát hiện ngay tại bước biên dịch.
8. Ưu điểm về tốc độ
9. Cả 3 câu đều đúng
10. Liên kết động:
11. Lời gọi phương thức được quyết định khi thực hiện (runtime)
12. Phiên bản của phương thức phù hợp với đối tượng được gọi
13. Cả 2 câu đều đúng
14. Cả 2 câu đêu sai
15. Các loại đa hình trong lập trình
16. Đa hình tĩnh (overloading)
17. Đa hình động (virtual)
18. Cả hai câu đều đúng
19. Cả hai câu đều sai
20. Đa hình tĩnh là hình thức hai hay nhiều phương thức có cùng tên nhưng được thực thi khác nhau, tùy vào cài đặt.
21. Đúng
22. Sai
23. Up casting là khả năng nhìn nhận đối tượng thuộc lớp \_\_\_\_\_\_\_\_\_\_như là một đối tượng thuộc lớp \_\_\_\_\_\_.
24. Dẫn xuất , cơ sở
25. Cơ sở , dẫn xuất
26. Trừu tượng, cụ thể
27. Tất cả đều sai
28. Down casting là khả năng nhìn nhận một đối tượng thuộc lớp \_\_\_\_\_\_\_\_\_\_ như một đối tượng thuộc lớp \_\_\_\_\_\_\_\_\_\_.
29. Dẫn xuất , cơ sở
30. Cơ sở , dẫn xuất
31. Trừu tượng, cụ thể
32. Tất cả đều sai
33. Dùng phương thức ảo chậm hơn phương thức thông thường vì khi thực hiện mới được xác định cụ thể
34. Đúng
35. Sai
36. Phương thức khởi tạo không được là phương thức ảo nhưng phương thức hủy bỏ có thể là phương thức ảo.
37. Đúng
38. Sai
39. Hàm hủy không được khai báo là ảo.
40. Đúng
41. Sai
42. Đa hình là hiện tượng các đối tượng thuộc các lớp khác nhau có khả năng hiểu cùng một thông điệp theo các cách giốngnhau.
    1. Đúng
    2. Sai
43. class Animal

{

public:

virtual void Talk() = 0;

};

class Cat: public Animal

{

public:

void Talk()

{

cout << "Meo meo!" << endl;

}

};

class Dog: public Cat

{

public:

void Talk()

{

cout << "Gau gau!" << endl;

}

};

void main()

{

Dog d;

Cat \*c= &d;

c->Talk();

}

Kết quả của chương trình:

1. Meo meo!
2. Don't know how to talk!
3. Gau gau!
4. Cả hai câu đều sai
5. Liên kết tĩnh là quy trình liên kết một lời gọi hàm với một định nghĩa hàm ngay tại thời điểm biên dịch.
6. Đúng
7. Sai
8. Liên kết tĩnh được gọi :
9. Static function call binding
10. Compile -time binding
11. Early binding
12. Tất cả câu trên
13. Liên kết động là quy trình liên kết một lời gọi hàm với một định nghĩa hàm tại thời gian chạy.
14. Đúng
15. Sai
16. Liên kết động còn được gọi:
17. Dynamic binding
18. Runtime binding
19. Late binding
20. Tất cả câu trên đều đúng
21. Một phương thức được khai báo là hàm ảo tại lớp cơ sở sẽ được tự động trở thành hàm ảo tại mọi lớp dẫn xuất trực tiếp hoặc gián tiếp.
22. Đúng
23. Sai
24. Mỗi khi tạo một lớp để được dùng làm lớp cơ sở, ta nên khai báo destructor là hàm ảo.
25. Đúng
26. Sai
27. Liên kết tĩnh là quy trình liên kết một lời gọi hàm với một định nghĩa hàm ngay tại thời điểm \_\_\_\_\_\_.
28. Biên dịch (compile time)
29. Chạy chương trình (runtime)
30. Cả hai câu đều đúng
31. Cả hai câu đều sai
32. Liên kết động là quy trình liên kết một lời gọi hàm với một định nghĩa hàm tại\_\_\_\_\_\_\_\_.
33. Thời gian chạy (runtime)
34. Thời gian biên dịch (compile time)
35. Cả hai câu đều đúng
36. Cả hai đều sai
37. Phương thức \_\_\_\_\_\_\_\_không được là phương thức ảo nhưng phương thức \_\_\_\_\_\_ có thể là phương thức ảo.
38. Khởi tạo, hủy
39. Hủy, khởi tạo
40. Ảo, hủy
41. Tất cả câu trên
42. Chúng ta có thể truy xuất lớp \_\_\_\_\_\_\_\_ thông qua con trỏ của lớp \_\_\_\_\_\_.
43. Dẫn xuất, cơ sở
44. Cơ sở, dẫn xuất
45. Cả hai câu đều đúng
46. Cả hai câu đều sai
47. Một hàm thuần ảo được xác định bằng việc đặt \_\_\_\_\_\_\_ vào cuối khai báo hàm trong định nghĩa lớp
    1. = 0
    2. !=0
    3. 0
48. Nếu một lớp chứa một hoặc nhiều hàm thuần ảo, nó là \_\_\_\_\_\_\_\_\_
    1. lớp dẫn xuất
    2. lớp trừu tượng
    3. lớp cơ sở
49. Lời gọi hàm được thực hiện lúc biên dịch, có liên quan tới liên kết \_\_\_\_\_\_\_\_
    1. động
    2. tĩnh
50. Lời gọi hàm được thực hiện lúc runtime, có liên quan tới liên kết \_\_\_\_\_\_\_\_
    1. động
    2. tĩnh
51. Tất cả các hàm ảo trong lớp cơ sở trừu tượng phải được khai báo là hàm thuần ảo
    1. Đúng
    2. Sai
52. Chọn câu sai:
    1. Hàm ảo được khai báo với từ khoá “virtual” trước tên hàm trong lớp cơ sở
    2. Liên kết động là quá trình một hàm thông qua con trỏ lớp cơ sở lúc runtime quyết định gọi hàm ảo lớp dẫn xuất nào
    3. Các constructor có thể khai báo là virtual
    4. Các lớp trừu tượng có thể được dùng cho kế thừa
53. Chọn câu sai:
    1. Bất kỳ destructor nào trong cây kế thừa cũng có thể khai báo là virtual
    2. Khi một lớp có chứa hàm ảo được biên dịch, một bảng hàm ảo được tạo ra cho lớp đó
    3. Tính đa hình cho phép các đối tượng của các lớp mới được thêm vào hệ thống một cách linh động
    4. Hàm thuần ảo có giá trị khởi đầu khác 0
54. Lớp trừu tượng(abstract class) có bao nhiêu thể hiện(instance)
    1. 1
    2. 5
    3. 4
    4. 0
55. Lớp cơ sở trừu tượng là lớp
    1. Được khai báo trong lớp khác
    2. Tất cả các phương thức của nó là ảo (virtual)
    3. Ít nhất cũng chứa 1 hàm ảo (virtual)
    4. Chứa 1 hàm ảo thuần túy (pure virtual function)
56. Đa hình vào thời điểm RunTime được thực hiện qua
    1. Hàm bạn (friend function)
    2. Hàm ảo (virtual function)
    3. Nạp chồng toán tử (operator overloading)
    4. Nạp chồng hàm (function overloading)
57. Cho biết kết quả biên dịch - thực thi chương trình sau:

class A {

public:

virtual void print() = 0;

};

class B: public A {

};

void main()

{

B \*pb;

B b;

pb = &b;

pb -> print();

}

* 1. Chương trình báo lỗi do khai báo biến con trỏ pb thuộc kiểu lớp trừu tượng
  2. Chương trình báo lỗi do khai báo biến b thuộc kiểu lớp trừu tượng.
  3. Chương trình chạy nhưng không hiển thị gì
  4. Chương trình chạy và hiển thị ký tự 0 trên màn hình.

1. Tính đa hình là
   1. Bố trí việc khai báo lớp và định nghĩa hàm thành viên (phương thức) của lớp trong 1 tệp
   2. Kết hợp và định vị dữ liệu và các hàm xử lý dữ liệu đó trong phạm vi của đối tượng, giống như 1 thể thống nhất
   3. Tính chất của các lớp sinh ra các lớp dẫn xuất và kế thừa các đặc tính(dữ liệu và phương thức) từ các từ các lớp cơ sở ban đầu
   4. Là tính chất cho phép một phương thức có các tác động khác nhau trên nhiều loại đối tượng khác nhau

# Chương : Operator overloading

1. *Đáp án nào sau đây không phải là toán tử ép kiểu của C++*
2. explicit\_cast
3. static\_cast
4. dynamic\_cast
5. reinterpret\_cast
6. *Các toán tử nào sau đây không thể overloading trong C++*
7. +=, ?, :: , >>
8. >>, <<, ?, \*, sizeof()
9. :: , . , .\* , ?:
10. :: , ->, \* , new, delete
11. *Một cách tổng quát, Overloading toán tử một ngôi bằng hàm friend thì yêu cầu mấy đối số*
12. không
13. một
14. hai
15. đáp án khác
16. *Một cách tổng quát, Overlading toán tử hai ngôi sử dụng hàm thành viên (member function) thì nó yêu cầu mấy đối số*
17. không
18. một
19. hai
20. ba
21. *Một cách tổng quát, Overloading toán tử một ngôi sử dụng hàm non-member và non-friend thì nó yêu cầu mấy đối số*
22. không
23. một
24. hai
25. ba
26. *Overlading toán tử hai ngôi sử dụng hàm non-member thì nó yêu cầu mấy đối số*
27. không
28. một
29. hai
30. ba
31. *Trong Overloading toán tử thì các toán tử được overloaded phải có ít nhất mấy toán hạng là kiểu người dùng định nghĩa*
32. không
33. một
34. hai
35. ba
36. *Một hàm toán tử (hàm để nạp chồng toán tử) được tạo ra bởi việc sử dụng keyword nào*
37. iterator
38. allocator
39. constructor
40. operator
41. *bộ toán tử nào sau đây không thể overloading bằng hàm friend hoặc hàm non-member*
42. = , ( ) , [ ] , ->
43. <<, = = , [ ] , >>
44. ?, = , ( ) , ++
45. Đáp án khác
46. *Trong trường hợp overloading toán tử hai ngôi bằng hàm thành viên (member function) thì*
47. Toán hạng bên phải phải là object của class hiện tại
48. Toán hạng bên trái phải là object của class hiện tại
49. Cả hai toán hạng đều phải là object của class hiện tại
50. Tất cả đáp án đều sai
51. *Toán tử nào được nạp chồng mặc định bởi bộ biên dịch*
52. + (toán tử cộng)
53. -(toán tử trừ - hai ngôi)
54. == (toán tử so sánh bằng
55. = (toán tử gán)
56. *Toán tử nào nên được định nghĩa là hàm non-member hơn là member*
57. toán tử so sánh bằng =
58. toán tử <<
59. ++ (tiền tố)
60. ++ (hậu tố)
61. *Bộ biên dịch C++ làm thế nào để phân biệt dạng tiền tố và hậu tố của toán tử*
62. không cho phép overloading bằng hàm member
63. không cho phép overloading bằng non-member
64. dạng hậu tố của toán tử có thêm tham số giả
65. dạng tiền tố của toán tử có thêm tham số giả
66. *Overloading toán tử trong C++ là*
67. làm cho các toán tử làm việc được với các objects mới
68. thêm các định nghĩa mới cho các toán tử đã tồn tại với các objects mới
69. tạo toán tử mới
70. cả a và b
71. *Toán tử nào được sử dụng để cấp phát động*
72. toán tử phạm vi ::
73. ( )
74. toán tử new
75. đáp án khác

Cho đoạn code sau:

class A

{

    int i;

public:

    A(int ii = 0) : i(ii) {}

    void show() {  cout << i << “ ”;  }

};

class B

{

    int x;

public:

    B(int xx) : x(xx) {}

    operator A() const {  return A(x); }

};

void g(A a)

{

    a.show();

}

int main()

{

    B b(10);

    g(b);

    g(20);

    return 0;

}

1. *Dòng g(b); trong hàm main có gây ra lỗi biên dịch hay không*
2. có
3. không
4. không đủ thông tin để khẳng định
5. *dòng operator A() const {  return A(x); } có ý nghĩa gì*
6. được sử dụng để chuyển kiểu từ B sang A
7. định nghĩa thêm 1 toán tử mới A
8. sai cú pháp nên không có ý nghĩa
9. có ý nghĩa khác
10. *dòng code g(20); trong hàm main có gây ra lỗi hay không*
11. có
12. không
13. *Chương trình có biên dịch được hay không*
14. được
15. không
16. chưa đủ để khẳng định
17. *Trong hàm main, operator A() const {  return A(x); } có được gọi thực hiện hay không*
18. có
19. không
20. chưa chắc để khẳng định
21. *operator A() const {  return A(x); } được gọi ở khi dòng nào của hàm main được thực hiện*
22. g(b)
23. g(20)
24. cả a và b
25. đáp án khác
26. *Kết quả trả về là gì:*
27. 10 20
28. 20 20
29. Kết quả khác
30. 10 10

Cho đoạn code sau

class Test2

{

    int y;

};

class Test

{

    int x;

    Test2 t2;

public:

    operator Test2 ()  { return t2; }

    operator int () { return x; }

};

void fun ( int x) { cout << "fun(int) called"; }

void fun ( Test2 t ) { cout << "fun(Test 2) called"; }

int main()

{

    Test t;

    fun(t);

    return 0;

}

1. *Dòng operator int () { return x; } có ý nghĩa gì*
2. dùng để chuyển kiểu từ Test sang int
3. dùng để chuyển kiểu từ int sang Test
4. không có ý nghĩa vì sau cú pháp
5. tạo thêm toán tử tên int
6. *dòng operator Test2 ()  { return t2; } có ý nghĩa gì*
7. dùng để chuyển kiểu từ Test sang Test2
8. dùng để chuyển kiểu từ Test2 sang Test
9. không có ý nghĩa vì sau cú pháp
10. tạo thêm toán tử mới tên Test2
11. *Chương trình sẽ cho output là gì*
12. fun(int) called
13. fun(Test 2) called
14. bị lỗi

Cho mẫu class như sau

class A

{

public:

Point& operator++();

Point operator++(int);

Point& operator--();

Point operator--(int);

};

1. *Nếu thực hiện phép toán a++; //a là object của class A thì dòng nào được thực hiện*
2. Point operator++(int);
3. Point operator++();
4. cả hai đều sai
5. *Nếu thực hiện phép toán –-a; //a là object của class A thì dòng nào được thực hiện*
6. Point operator--(int);
7. Point operator--();
8. cả hai đều sai
9. *toán tử nào có thể được xem xét như toán tử một ngôi*
10. <
11. /
12. –
13. ==

Cho đoạn code:

class A {

private:

int a;

public:

A(int i) :a(i) {}

friend ostream& operator << (ostream& os, const A& a) {

os << a.a << " ";

return os;

}

};

int main() {

A b(12);

cout << b;

operator<<(cout, b);

return 0;

}

1. *Kết quả của chương trình*
2. lỗi biên dịch
3. 12 12
4. 12
5. *Tên gọi khác của hàm toán tử (operator function)*
6. function overloading
7. operator overloading
8. member overloading
9. Tất cả đều sai

Cho đoạn code

class sample

{

public:

int x, y;

sample() {};

sample(int, int);

sample operator + (sample);

};

sample::sample (int a, int b)

{

x = a;

y = b;

}

sample sample::operator+ (sample param)

{

sample temp;

temp.x = x + param.x;

temp.y = y + param.y;

return (temp);

}

int main ()

{

sample a (4,1);

sample b (3,2);

sample c;

c = a + b;

cout << c.x << "," << c.y;

return 0;

}

1. *kết quả của chương trình*
2. 5, 5
3. 7, 3
4. 3, 7
5. tất cả đều sai

Cho đoạn code

class Integer

{

int i;

public:

Integer(int ii) : i(ii) {}

const Integer

operator+(const Integer& rv) const

{

cout << "operator+" << endl;

return Integer(i + rv.i);

}

Integer&

operator+=(const Integer& rv)

{

cout << "operator+=" << endl;

i += rv.i;

return \*this;

}

};

int main()

{

int i = 1, j = 2, k = 3;

k += i + j;

Integer ii(1), jj(2), kk(3);

kk += ii + jj;

}

1. *Chương trình cho kết quả là gì*

a) operator+  
operator+=  
b) operator+=  
operator+  
c) operator+  
operator+  
d) tất cả đều không đúng

Cho đoạn code

class myclass

{

public:

int i;

myclass \*operator->()

{return this;}

};

int main()

{

myclass ob;

ob->i = 10;

cout << ob.i << " " << ob->i;

return 0;

}

1. *Kết quả của chương trình là gi:*
2. 10 10
3. 11 11
4. Error
5. runtime error

cho đoạn code

ostream & operator<<(ostream & i, int n)

{

return i;

}

int main()

{

cout << 5 << endl;

cin.get();

return 0;

}

1. *kết quả khi chạy chương trình là gì:*
2. 5
3. 6
4. error
5. runtime error
6. *kiểu trả về khi xây dựng toán tử chuyển kiểu*
7. void
8. int
9. float
10. không cần kiểu trả về
11. *hàm của toán tử chuyển kiểu sẽ có danh sách đối số có mấy đối số*
12. 0
13. 1
14. 2
15. đáp án khác
16. *Có bao nhiêu loại chuyển kiểu trong chuyển kiểu có thể được định nghĩa bởi người dùng*
17. 1
18. 2
19. 3
20. 4
21. *chọn cú pháp nên được sử dụng cho toán tử chuyển kiểu*
22. operator float()const
23. operator float()
24. operator const
25. Các câu trên đều sai

Cho đoạn code

int main()

{

double a = 21.09399;

float b = 10.20;

int c ;

c = (int) a;

cout << c ;

c = (int) b;

cout << c ;

return 0;

}

1. *kết quả trả về là gì*
2. 2110
3. 1210
4. 21
5. đáp án khác

Cho đoạn code

class test

{

public:

operator string ()

{

return "Converted";

}

};

int main()

{

test t;

string s = t;

cout << s << endl;

return 0;

}

1. *outut của chương trình*
2. converted
3. error
4. run time error
5. Các câu trên đều sai

Cho đoạn code

class Complex

{

private:

double real;

double imag;

public:

Complex(double r = 0.0, double i = 0.0) : real(r), imag(i)

{}

double mag()

{

return getMag();

}

operator double ()

{

return getMag();

}

private:

double getMag()

{

return sqrt(real \* real + imag \* imag);

}

};

int main()

{

Complex com(3.0, 4.0);

cout << com.mag();

cout << com;

return 0

}

1. *kết quả của chương trình là gì*
2. 5 5
3. 4 5
4. 6 6
5. Các câu trên đều sai

Cho đoạn code

class sample

{

public:

sample(int i) : m\_i(i) { }

public:

int operator()(int i = 0) const

{

return m\_i + i;

}

operator int () const

{

return m\_i;

}

private:

int m\_i;

friend int g(const sample&);

};

int f(char c)

{

return c;

}

int main()

{

sample f(2);

cout << f(2);

return 0;

}

1. *output của chương trình là*
2. 3
3. 4
4. 5
5. kết quả khác
6. *toán tử nào sau đây xem xét thực hiện từ phải sang trái*
7. +
8. –
9. ép kiểu (type cast)
10. tất cả đều đúng
11. *toán tử nào trong các toán tử sau có độ ưu tiên cao nhất*
12. hậu tố
13. một ngôi
14. so sánh bằng
15. shift
16. *toán tử ?: được gọi là gì*
17. toán tử điều kiện
18. toán tử quan hệ
19. toán tử ép kiểu
20. tất cả đều sai
21. *thứ tự thực hiện của toán tử +*
22. trái sang phải
23. phải sang trái
24. không có đáp án
25. *toán tử | có trên là gì*
26. sizeof
27. or
28. and
29. modulus
30. *toán tử có độ ưu tiên cao nhất trong C++*
31. []
32. ::
33. dynamic\_cast
34. +
35. *toán tử [] (subscript operator) được sử dụng để truy cập kiểu dữ liệu nào*
36. char
37. số
38. array
39. đáp án khác
40. *khi overloading toán tử [] thì hàm sẽ nhận mấy đối số*
41. 0
42. 1
43. 2
44. 3
45. *ý nào sau đây đúng*
46. toán tử [] có độ ưu tiên cao hơn toán tử gán
47. toán tử [] có độ ưu tiên nhỏ hơn toán tử gán
48. toán tử được dùng với số
49. tất cả đều đúng

cho đoạn code

const int SIZE = 10;

class safe

{

private:

int arr[SIZE];

public:

safe()

{

register int i;

for (i = 0; i < SIZE; i++)

{

arr[i] = i;

}

}

int &operator[](int i)

{

if (i > SIZE)

{

cout << "Index out of bounds" <<endl;

return arr[0];

}

return arr[i];

}

};

int main()

{

safe A;

cout << A[5];

cout << A[12];

return 0;

}

1. *output của chương trình là gì*
2. Index out of bounds
3. 40
4. 50
5. 51

Cho đoạn code

class numbers

{

private:

int m\_nValues[10];

public:

int& operator[] (const int nValue);

};

int& numbers::operator[](const int nValue)

{

return m\_nValues[nValue];

}

int main()

{

numbers N;

N[5] = 4;

cout << N[5];

return 0;

}

1. *kết quả chương trình là gì*
2. 5
3. 4
4. 3
5. 6

cho đoạn code

const int limit = 4;

class safearray

{

private:

int arr[limit];

public:

int& operator [](int n)

{

if (n == limit - 1)

{

int temp;

for (int i = 0; i < limit; i++)

{

if (arr[n + 1] > arr[n])

{

temp = arr[n];

arr[n] = arr[n + 1];

arr[n + 1] = temp;

}

}

}

return arr[n];

}

};

int main()

{

safearray sa1;

for(int j = 0; j < limit; j++)

sa1[j] = j\*10;

for(int j = 0; j < limit; j++)

{

int temp = sa1[j];

cout << "Element " << j << " is " << temp;

}

return 0;

}

1. *output của chương trình là gì*
2. 0102030
3. 1020300
4. 3020100
5. error

Cho đoạn code

class A

{

public:

int x;

A(int n = 0) : x(n) {};

int& operator[](int n)

{

cout << "0" ;

return x;

}

int operator[](int n) const

{

cout << "1" ;

return x;

}

};

void foo(const A& a)

{

int z = a[2];

}

int main()

{

A a(7);

a[3] = 8;

int z = a[2];

foo(a);

return 0;

}

1. *output của chương trình là gì*
2. 110
3. 111
4. 011
5. 001

class sample

{

private:

int\* i;

int j;

public:

sample (int j);

~sample ();

int& operator [] (int n);

};

int& sample::operator [] (int n)

{

return i[n];

}

sample::sample (int j)

{

i = new int [j];

j = j;

}

sample::~sample ()

{

delete [] i;

}

int main ()

{

sample m (5);

m [0] = 25;

m [1] = 20;

m [2] = 15;

m [3] = 10;

m [4] = 5;

for (int n = 0; n < 5; ++ n)

cout << m [n];

return 0;

}

1. *output của chương trình là gì*
2. 252015105
3. 510152025
4. 51015
5. không có đáp án đúng

cho đoạn code

class Distance

{

private:

int feet;

int inches;

public:

Distance()

{

feet = 0;

inches = 0;

}

Distance(int f, int i)

{

feet = f;

inches = i;

}

Distance operator()(int a, int b, int c)

{

Distance D;

D.feet = a + c + 10;

D.inches = b + c + 100 ;

return D

}

void displayDistance()

{

cout << feet << inches << endl;

}

};

int main()

{

Distance D1(11, 10), D2;

cout << "First Distance : ";

D1.displayDistance();

D2 = D1(10, 10, 10);

cout << "Second Distance :";

D2.displayDistance();

return 0;

}

1. *output của chương trình sau*

a) First Distance : 1110  
Second Distance :30120  
b) First Distance : 110  
Second Distance :3020  
c) First Distance : 1115  
Second Distance :30125  
d) đáp án khác

class three\_d

{

int x, y, z;

public:

three\_d() { x = y = z = 0; }

three\_d(int i, int j, int k) { x = i; y = j; z = k; }

three\_d operator()(three\_d obj);

three\_d operator()(int a, int b, int c);

friend ostream &operator<<(ostream &strm, three\_d op);

};

three\_d three\_d::operator()(three\_d obj)

{

three\_d temp;

temp.x = (x + obj.x) / 2;

temp.y = (y + obj.y) / 2;

temp.z = (z + obj.z) / 2;

return temp;

}

three\_d three\_d::operator()(int a, int b, int c)

{

three\_d temp;

temp.x = x + a;

temp.y = y + b;

temp.z = z + c;

return temp;

}

ostream &operator<<(ostream &strm, three\_d op) {

strm << op.x << ", " << op.y << ", " << op.z << endl;

return strm;

}

int main()

{

three\_d objA(1, 2, 3), objB(10, 10, 10), objC;

objC = objA(objB(100, 200, 300));

cout << objC;

return 0;

}

1. *output của chương trình là gì:*
2. 55, 106, 156
3. 55, 106
4. 55, 106, 159
5. không có đáp án

cho đoạn code

class Complex

{

private:

float real;

float imag;

public:

Complex():real(0), imag(0){}

Complex operator ()(float re, float im)

{

real += re;

imag += im;

return \*this;

}

Complex operator() (float re)

{

real += re;

return \*this;

}

void display()

{

cout << "(" << real << "," << imag << ")" << endl;

}

};

int main()

{

Complex c1, c2;

c2 = c1(3.2, 5.3);

c1(6.5, 2.7);

c2(1.9);

cout << "c2=";c1.display();

cout << "c2=";c2.display();

return 0;

}

1. *output của chương trình*
2. c2=(9.7,8)  
   c2=(5.1,5.3)
3. c2=(9,8)  
   c2=(5,5)
4. c2=(4.7,8)  
   c2=(2.1,5.3)
5. đáp án khác
6. *có bao nhiêu kiểu toán tử tăng/giảm*
7. 1
8. 2
9. 3
10. 4
11. *chọn ý đúng*
12. về cơ bản toán tử tăng ++ sẽ tăng toán hạng lên 1 đơn vị
13. về cơ bản toán tử tăng ++ sẽ tăng toán hạng lên 2 đơn vị
14. về cơ bản toán tử giảm ++ sẽ giảm toán hạng lên 1 đơn vị
15. đáp án khác

cho đoạn code

int main()

{

int a = 21;

int c ;

c = a++;

cout << c;

return 0;

}

1. *output là gì*
2. 21
3. 22
4. 23
5. 20

cho đoạn code

int main()

{

int x = 5, y = 5;

cout << ++x << --y << endl;

return 0;

}

1. *output là gì*
2. 55
3. 64
4. 46
5. 45

cho đoạn code

int main()

{

int x = 5, y = 5, z;

x = ++x; y = --y;

z = x++ + y--;

cout << z;

return 0;

}

1. *output là gì*
2. 10
3. 11
4. 9
5. 12
6. *chọn ý đúng đối với toán tử tăng/giảm mặc định:*
7. ++ tiền tố nhanh hơn ++ hậu tố
8. ++ hậu tố nhanh hơn tiền tố
9. cả hai bằng nhau
10. không có đáp án đúng

Cho đoạn code sau:

class A {

private:

int a;

public:

A() { a = 0; }

A(int i) :a(i) {

}

A(const A& a2) {

a = a2.a;

}

friend ostream& operator << (ostream& os, const A& a) {

os << a.a << " "<<endl;

return os;

}

A operator + (const A& a2) const {

return A(this->a + a2.a);

}

A operator = (const A& a2) {

a = a2.a;

return \*this;

}

};

int main() {

A b(12);

A c = b + 12;

A d;

d = b + 12;

return 0;

}

1. *dòng code “A c = b + 12;” trong hàm main có gây lỗi không*
2. có
3. không
4. *giá trị d.a sau khi thực hiện chương trình là*
5. 12
6. 24
7. kết quả khác
8. bị lỗi
9. *khi thực hiện hàm main thì hàm khởi tạo “A(int i)” được thực hiện mấy lần*
10. 3
11. 4
12. 5
13. 6
14. *những lệnh nào thực hiện sẽ gọi hàm “A(int i)”*
15. A b(12); A c = b + 12; d = b + 12;
16. A b(12);
17. A b(12); d = b + 12;
18. A b(12); A c = b + 12;
19. *khi thực hiện hàm main thì hàm “A(const A& a2)” được thực hiện mấy lần*
20. 0
21. 1
22. 2
23. 3
24. *những lệnh nào thực hiện sẽ gọi hàm “A(const A& a2)”*
25. A c = b + 12; d = b + 12;
26. d = b + 12;
27. A c = b + 12;
28. đáp án khác
29. *khi thực hiện hàm main thì hàm “A operator = (const A& a2)” được thực hiện mấy lần*
30. 0
31. 1
32. 2
33. 3
34. *trong hàm main, khi thực hiện lệnh “d = b + 12;” thì thứ tự các hàm được gọi ra sao*
35. A operator + (const A& a2) const; A(int i); A operator = (const A& a2); A(const A& a2)
36. A operator + (const A& a2) const; A(int i); A operator = (const A& a2)
37. A operator + (const A& a2) const; A operator = (const A& a2); A(const A& a2)
38. đáp án khác
39. *khi thực hiện dòng lệnh nào sẽ gọi hàm “A operator = (const A& a2)”:*
40. A c = b + 12;
41. d = b + 12;
42. d = b + 12; A c = b + 12;
43. đáp án khác

Cho đoạn code sau:

class A {

private:

int a;

public:

A() { a = 0; }

A(int i) :a(i) {

}

A(const A& a2) {

a = a2.a;

}

A operator + (const A& a2) const {

return A(this->a + a2.a);

}

A operator = (const A& a2) {

a = a2.a;

return \*this;

}

int get\_a(){

return a;

}

};

ostream& operator << (ostream& os, const A& a) {

os << a.get\_a() << " " << endl;

return os;

}

int main() {

A b(12);

cout << b;

return 0;

}

1. *output của hàm main khi thực hiện:*
2. 12
3. b
4. lỗi biên dịch
5. đáp án khác
6. *hàm nào có thể làm chương trình trên hoạt động*
7. thay hàm “int get\_a()” thành “int get\_a() const”
8. bỏ hàm A(int i);
9. bỏ toán tử gán A operator = (const A& a2)
10. làm đồng thời cả a,b,c
11. *sau khi đã khắc phục lỗi, ta bổ sung lệnh operator<<(cout, b+12); vào trước return 0;. hỏi chương trình sẽ cho output là gì*
12. 12 24
13. 24
14. 12
15. bị lỗi biên dịch

cho đoạn code:

class A {

private:

int a;

public:

A() { a = 0; }

A(int i) :a(i) { }

A(const A& a2) {

a = a2.a;

}

A operator = (const A& a2) {

a = a2.a;

return \*this;

}

};

int main() {

A b(12);

A c = b;

A d = 13;

return 0;

}

1. *chương trình có chạy được hay không*
2. có
3. không
4. không xác định được
5. *dòng A b(12) thì hàm nào được gọi*
6. A(int i)
7. A operator = (const A& a2)
8. A(const A& a2)
9. đáp án khác
10. *dòng “A c = b;” có thực hiện được không*
11. có
12. không
13. *nếu lệnh “A c = b;” thực hiện được thì những lệnh nào sau đây được gọi để thực hiện lệnh đó*
14. A operator = (const A& a2)
15. A(int i)
16. A(const A& a2)
17. đáp án khác
18. *dòng lệnh “A d = 13;” thực hiện được hay không*
19. có
20. không
21. *nếu lệnh “A d = 13;” thực hiện được thì những lệnh nào sau đây được gọi để thực hiện lệnh đó*
22. A operator = (const A& a2)
23. A(int i)
24. A(const A& a2)
25. đáp án khác
26. *hàm để overloading toán tử -> sẽ phải là:*
27. hàm member
28. hàm non-member
29. hàm friend
30. không có đáp án đúng
31. *để overloading toán tử + thì ta có thể sử dụng:*
32. hàm member
33. hàm friend
34. hàm non-member và non-friend
35. cả 3 cách trên
36. *cách overloading toán tử ++ dạng tiền tố nào sau đây đúng (với ví dụ A là một class):*
37. hàm member có prototype “A operator++()”
38. hàm non-member có prototype “A operator++(A,int)”
39. hàm member có prototype “A operator++(int)”
40. cả 3 đều sai
41. *cách overloading toán tử ++ dạng hậu tố nào sau đây đúng (với ví dụ A là một class):*
42. hàm member có prototype “A operator++()”
43. hàm member có prototype “A operator++(int)”
44. hàm non-member có prototype “A operator++(A,int)”
45. a và c đều đúng
46. *với a,b là các số nguyên thì lệnh sau a = ++b thì sau lệnh này, a và b có giá trị thay đổi ra sao*
47. a có giá tri lớn b một đơn vị
48. b có giá trị lớn hơn a một đơn vị
49. a bằng giá trị với b
50. không khẳng định được
51. *với a,b là các số nguyên thì lệnh sau a = b++ thì sau lệnh này, a và b có giá trị thay đổi ra sao*
52. a có giá tri lớn b một đơn vị
53. b có giá trị lớn hơn a một đơn vị
54. a bằng giá trị với b
55. không khẳng định được
56. *kiểu định nghĩa nào sau đây có thể dùng để overloading cho toán tử - (với ví dụ A là một class)*
57. hàm member có prototype “A operator – ()”
58. hàm member có prototype “A operator – (A)”
59. hàm non-member có prototype “A operator – (A)”
60. tất cả đều a,b,c đều có thể
61. *các toán tử nào trong cách toán tử sau đây có thể định nghĩa cả dạng hàm member và non-member*
62. +
63. []
64. =
65. ->
66. *Nếu ta đã overloading toán tử + cho class A với dạng hàm member thì ta có thể sử dụng toán tử + với dạng nào sau đây*
67. a + b với a,b là object của A
68. a.operator+(b) với a,b là object của A
69. operator+(a,b) với a,b là object của A
70. a,b đều đúng
71. *nếu ta đã overloading toán tử + cho class A với dạng hàm non-member thì ta có thể sử dụng toán tử + với dạng nào sau đây*
72. a + b với a,b là object của A
73. a.operator+(b) với a,b là object của A
74. a,b đều sai
75. cả a,b đều đúng
76. *nếu ta đã overloading toán tử -- hậu tố cho class A với dạng member thì ta có thể sử dụng toán tử -- hậu tố với dạng nào sau đây*
77. a--, với a là object của class A
78. a.operator--() với a là object của class A
79. a.operator--(1) với a là object của class A
80. a,c đều có thể
81. *nếu ta đã overloading toán tử -- hậu tố cho class A với dạng non-member thì ta có thể sử dụng toán tử -- hậu tố với dạng nào sau đây*
82. a--, với a là object của class A
83. operator--(a,2) với a là object của class A
84. a.operator--() với a là object của class A
85. cả a và b đều có thể
86. *dãy toán tử nào sau đây là toán tử một ngôi:*
87. -, ++, --, !
88. +,-,--,++
89. \*,--,!,-
90. ++,--,[],+
91. *dãy toán tử nào sau đây vừa có dạng một ngôi và vừa có dạng 2 ngôi:*
92. +,-,\*,/
93. --,++,/,[]
94. --,+,/,\*
95. cả 3 đều sai
96. *dãy toán tử =, +=, -=, <<=, >>=,%=,|=,/=, thuộc dạng toán tử nào sau đây:*
97. toán tử gán (assignment)
98. toán tử so sánh
99. toán tử logic
100. toán tử access member

Cho yêu cầu hàm main như sau:

int main()

{

STRING a; //yêu cầu 1

STRING b("abc"); // yêu cầu 2

STRING c('a'); // yêu cầu 3

STRING d="def", f='t'; // yêu cầu 4

STRING e(c); // yêu cầu 5

return 0;

}

1. *để đáp ứng 1st requirement thì class STRING cần định nghĩa hàm khởi tạo nào*
2. hàm khởi tạo không đối số
3. hàm khởi tạo 1 đối số có kiểu char\*
4. hàm khởi tạo 1 đối số có kiểu char
5. hàm khởi tạo 1 đối số có kiểu là STRING
6. *nếu STRING đã thỏa yêu cầu 1, hỏi STRING cần thêm hàm khởi tạo nào để yêu cầu 2 được thỏa mãn*
7. hàm tạo 1 đối số kiểu char\*
8. hàm tạo 1 đối kiểu char
9. không cần thêm hàm tạo mới
10. hàm khởi tạo 1 đối số có kiểu là STRING
11. *nếu STRING đã thỏa yêu cầu 1, yêu cầu 2 từ 2 cầu hỏi trước, hỏi STRING cần thêm hàm khởi tạo nào để yêu cầu 3 được thỏa mãn*
12. hàm tạo 1 đối số kiểu char
13. hàm tạo 1 đối kiểu char\*
14. không cần thêm hàm tạo mới
15. hàm khởi tạo 1 đối số có kiểu là STRING
16. *nếu STRING đã thỏa yêu cầu 1, yêu cầu 2,yêu cầu 3 từ 3 câu hỏi trước, hỏi STRING cần thêm hàm khởi tạo nào để yêu cầu 4 được thỏa mãn*
17. hàm tạo 1 đối số kiểu char
18. hàm tạo 1 đối kiểu char\*
19. không cần thêm hàm tạo mới
20. hàm khởi tạo 1 đối số có kiểu là STRING
21. *nếu STRING đã thỏa yêu cầu 1, 2,3,4 từ 4 câu hỏi trước, hỏi STRING cần thêm hàm khởi tạo nào để yêu cầu 5 được thỏa mãn*
22. hàm tạo 1 đối số kiểu char
23. hàm tạo 1 đối kiểu char\*
24. không cần thêm hàm tạo mới
25. hàm khởi tạo 1 đối số có kiểu là STRING

# Đáp án

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | D |  | D |  | B |  | C |
|  | D |  | C |  | B |  | A |
|  | B |  | A |  | C |  | B |
|  | C |  | C |  | B |  | B |
|  | C |  | B |  | A |  | D |
|  | C |  | A |  | A |  | C |
|  | C |  | D |  | C |  | C |
|  | C |  | B |  | B |  | B |
|  | B |  | A |  | C |  | B |
|  | C |  | B |  | C |  | B |
|  | B |  | B |  | D |  | D |
|  | A |  | A |  | B |  | A |
|  | B |  | B |  | A |  | B |
|  | D |  | C |  | D |  | B |
|  | E |  | C |  | B |  | D |
|  | B |  | A |  | C |  | C |
|  | A |  | A |  | B |  | C |
|  | A |  | B |  | B |  | D |
|  | A |  | B |  | A |  | C |
|  | A |  | B |  | B |  | D |
|  | D |  | A |  | D |  | B |
|  | B |  | A |  | B |  | D |
|  | C |  | B |  | D |  | C |
|  | B |  | C |  | C |  | D |
|  | D |  | D |  | C |  | B |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | C |  | A |  | B |  | C |
|  | A |  | B |  | D |  | B |
|  | A |  | C |  | G |  | D |
|  | C |  | B |  | C |  | A |
|  | A |  | D |  | C |  | A |
|  | D |  | B |  | A |  | A |
|  | B |  | A |  | C |  | D |
|  | D |  | D |  | D |  | E |
|  | D |  | A |  | A |  | C |
|  | A |  | C |  | C |  | B |
|  | A |  | D |  | D |  | A |
|  | C |  | B |  | C |  | C |
|  | C |  | A |  | B |  | D |
|  | B |  | B |  | D |  | A |
|  | A |  | A |  | B |  | B, C |
|  | A |  | D |  | D |  | D |
|  | A |  | C |  | B |  | C |
|  | A |  | A |  | B |  | C |
|  | C |  | C |  | D |  | A |
|  | B |  | D |  | D |  | D |
|  | A |  | A |  | A |  | B |
|  | A |  | B |  | B |  | B |
|  | A |  | C |  | C |  | A |
|  | A |  | A |  | A |  | B |
|  | A |  | D |  | B |  | A |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | C |  | A |  | B |  | B |
|  | B |  | B |  | D |  | C |
|  | D |  | D |  | A |  | A |
|  | B |  | C |  | D |  | B |
|  | B |  | B |  | A |  | B |
|  | A |  | B |  | D |  | B |
|  | B |  | B |  | D |  | A |
|  | C |  | C |  | A |  | D |
|  | F |  | C |  | B |  | C |
|  | A |  | B |  | C |  | C |
|  | C |  | A |  | B |  | C |
|  | A |  | B |  | C |  | D |
|  | B |  | E |  | C |  | C |
|  | D |  | C |  | C |  | B |
|  | D |  | A |  | D |  | A |
|  | B |  | A |  | A |  | A |
|  | A |  | A |  | D |  | A |
|  | C |  | B |  | D |  | D |
|  | C |  | A |  | D |  | B |
|  | E |  | B |  | D |  | B |
|  | A |  | A |  | C |  | D |
|  | B |  | C |  | C |  | B, C |
|  | D |  | C |  | D |  | B |
|  | A |  | A |  | A |  | B |
|  | A |  | B |  | A |  | A |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | B |  | b |  | A |  | C |
|  | C |  | b |  | A |  | A |
|  | B |  | a |  | A |  | D |
|  | A |  | d |  | B |  | A |
|  | C |  | b |  | A |  | D |
|  | A |  | c |  | B |  | A |
|  | D |  | a |  | C |  | A |
|  | B |  | b |  | A |  | A |
|  | C |  | b |  | D |  | A |
|  | A |  | d |  | D |  | A |
|  | A |  | d |  | D |  | A |
|  | A |  | A |  | B |  | A |
|  | A |  | A |  | B |  | B |
|  | A |  | B |  | D |  | B |
|  | C |  | B |  | D |  | A |
|  | C |  | D |  | D |  | B |
|  | C |  | A |  | C |  | C |
|  | C |  | B |  | C |  | D |
|  | B |  | B |  | A |  | D |
|  | A |  | A |  | A |  | D |
|  | B |  | C |  | A |  | B |
|  | C |  | B |  | A |  | B |
|  | D |  | B |  | A |  | D |
|  | C |  | B |  | B |  | A |
|  | D |  | A |  | B |  | C |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | B |  | C |  | B |  | A |
|  | B |  | B |  | A |  | A |
|  | B |  | B |  | D |  | A |
|  | C |  | B |  | A |  | C |
|  | B |  | A |  | A |  | A |
|  | D |  | A |  | A |  | B |
|  | A |  | C |  | A |  | A |
|  | B |  | D |  | B |  | D |
|  | D |  | A |  | A |  | A |
|  | B |  | B |  | A |  | D |
|  | C |  | A |  | B |  | C |
|  | D |  | A |  | A |  | B |
|  | C |  | A |  | A |  | D |
|  | B |  | A |  | B |  | A |
|  | A |  | B |  | B |  | D |
|  | B |  | C |  | C |  | A |
|  | A |  | A |  | A |  | D |
|  | A |  | A |  | B |  | D |
|  | A |  | A |  | B |  | A |
|  | A |  | B |  | B |  | D |
|  | A |  | B |  | A |  | A |
|  | A |  | C |  | B |  | A |
|  | C |  | B |  | C |  | A |
|  | A |  | A |  | A |  | A |
|  | B |  | A |  | A |  | C |
|  |  |  |  |  |  |  | D |