CảcNet

Bài 1

1. .NET là gì
2. Một hệ điều hành
3. Một nền tảng phát triển đa nền tảng và mã nguồn mở
4. Một ngôn ngữ lập trình
5. Một trình duyệt web
6. Tại sao .NET được coi là nền tảng an toàn về bộ nhớ?
7. Vì nó có thể chạy trên nhiều hệ điều hành khác nhau
8. Vì nó sử dụng GC và các trình biên dịch ngôn ngữ nghiêm ngặt
9. Vì nó không có lỗi bo mật
10. Vì nó không bao giờ sứ dụng bộ nhớ
11. Tại sao C# được coi là ngon ngữ lập trình chính trong .NET?
12. Vì nó không yêu cầu bộ thu gom rác
13. Vì nó hỗ trợ lập trình hướng đối tượng và runtime của nó hỗ trợ các tính năng này
14. Vì nó có cú pháp đơn giản
15. Vì nó được sử dụng trên tất cả các hệ điều hành
16. Tại sao .NET có thể chạy trên nhiều kiến trúc máy tính khác nhau?
17. Vì nó không sứ dụng mã gốc
18. Vì nó không cần biên dịch mã nguồn
19. Vì nó su dụng trình biên dịch JIT cho mỗi kiến truc CPU được hỗ tr
20. Vì nó su dụng mã nguồn mở
21. Lý do nao để NuGet được coi là trình quản lý goi quan trọng trong .NET?
22. Vì nó chứa hàng trăm nghìn gói thực hiện chức năng
23. Vì nó không cần biên dịch
24. Vì nó không sứ dụng bộ nhớ
25. Vì nó không hỗ trợ các hệ điều hành khác
26. Tại sao .NET lại được coi là mot nen tang ma nguồn mở quan trọng?
27. Vì nó hỗ trợ nhiều ngôn ngữ lập trình
28. Vì nó không cần cài đặt trên hệ điều hành
29. Vì nó cho phép cộng đồng đóng góp mã nguồn
30. Vì nó không sử dụng bộ thu gom rác
31. Sự khác biệt chính giữa top-level statements và cách viết truyền thống trong C# là gì?
32. Cách viết truyền thong không cần khai bao phương thức Main
33. Cach viet truyền thong dơn gian hơn top-level statements
34. Top-level statements không cần khai báo không gian tên và lớp
35. Top-level statements chỉ sứ dụng được trong .NET 5 trở về trước
36. Điều gì se xay ra nếu bạn khong bao gồm thuoc tính <ImpLicitUsings>enabLe</ImpLicitUsings> trong tệp dự án (.csproj)?
37. Dự an sẽ không biên dịch được
38. Tất ca các chỉ thị using sẽ bị vô hiệu hóa
39. Trình biên dịch sẽ tự động thêm tất cả các chỉ thị using có sẵn
40. Các chỉ thị using mac định sẽ không được tự động thêm vào
41. Tại sao việc sử dụng các chỉ thị using toàn cục (global using directives) lại hữu ích trong một dự án lớn?
42. Tự động tối ưu hóa mã
43. Giúp quản lý các không gian tên dễ dàng hơn
44. Tăng tốc độ biên dịch
45. Giảm kích thước tệp mã nguồn
46. Lợi ích chính của việc sử dụng Immediate Window trong quá trình debug là gì?
    1. Nhấn F9
    2. Chọn Debug > Add Breakpoint
    3. Nhấn vào vòng tròn màu đo bên trái dòng

mã

* 1. Sử dụng Immediate Window

1. Kiểm tra cú pháp mã nguồn
2. Tạo báo cáo lỗi tự động
3. Thay đổi giá trị các biến mà không cần chạy lại chương trình từ đầu
4. Tăng tốc độ chạy của chương trình
5. Khi nào thì nên sử dụng cau hình Release thay vì Debug trong Visual Studio?
6. Khi bạn đang gỡ lỗi chương trình
7. Khi bạn đang tối ưu hóa và chuẩn bị phát hành ứng dụng
8. Khi bạn đang viết tài liệu cho chương trình
9. Khi bạn đang thêm tính năng mới vào ứng dụng
10. Lệnh nào để tạo dự án console trong .NET 6+ sử dụng top-level statements?
11. dotnet build console
12. dotnet run console
13. dotnet new console
14. dotnet create console
15. Khi debug một ứng dụng Console App trong Visual Studio, làm cách nào để đặt điểm dừng (breakpoint)?
16. Nhấn F9
17. Sứ dụng Immediate Window
18. Chon Debug > Add Breakpoint
19. Nhấn vào vòng tròn màu đó bên trái dòng mã
20. Để thay đổi gia trị cua một biến mà không cần chạy lại chương trình, bạn sử dung công cụ nào trong Visual Studio?
21. Call Stack
22. Watch Window
23. Output Window
24. Immediate Window
25. Làm the nao để bạn sử dụng hệ thong loại của .NET để lập trình hướng đối tượng?
26. Tránh sứ dụng các thư viện
27. Tránh sứ dụng lập trình hướng đối tượng
28. Sử dụng các loại, kế thừa, giao diện và phương pháp áo
29. Chỉ su dụng các loại du liệu nguyên thuy
30. Intermediate Language (IL) trong .NET là:
31. Mã lệnh dễ dàng chính sửa và debug
32. Mã lệnh thực thi trên CPU cụ thể
33. Mã lệnh được chuyển đổi từ mã nguồn.
34. Ngôn ngữ lập trình độc lập với phần cứng
35. Vì sao các thu vien trong .NET được coi là tối ưu hoa hiệu suat trên nhiều hệ điều hành?
36. Vì chung đã được tối uu hoa cho hiệu suất trên nhiều hệ điều hành và kiến trúc chip
37. Vì chúng bao gồm các tính năng bảo mật mạnh mẽ
38. Vì chúng không cần quản lý bộ nhớ
39. Vì chúng được viết bằng ngôn ngữ máy
40. Trình biên dịch JIT (Just-in-Time Compiler) trong .NET được sử dụng để
41. Biên dịch mã IL thành mã máy khi người dùng yêu cầu
42. Quản lý các file manifest cho assembly
43. Tạo các tham chiếu giua các kiểu dữ liệu
44. Kiểm chứng metadata và mã lệnh
45. CLR là viết tat của
46. Common Type System
47. Common Language Runtime
48. Common Language Specification
49. Java Virtual Machine
50. Khi bạn muốn phát hành một ứng dụng Console App, bước nào sau đây là cần thiết?
51. Chạy ứng dụng bằng lệnh dotnet run
52. Sứ dụng lệnh dotnet release
53. Thay đổi từ chế độ Debug sang Release
54. Sứ dụng lệnh dotnet start
55. Khi debug một ứng dụng console trong Visual Studio, làm cách nào để thay đổi giá trị của một biến mà không cần chạy lại chương trình?
56. Sứ dụng lệnh dotnet change variable
57. Sứ dụng Immediate Window
58. Sử dụng lệnh dotnet debug variable
59. Sử dụng Watch Window
60. Bạn sẽ giải quyết vấn đề rò rỉ bộ nhớ trong một ứng dụng .NET như thế nào?
61. Bằng cách quản lý bộ nhớ thủ công
62. Bằng cách sứ dụng mã không được quản lý
63. Bằng cách bó qua vấn đề
64. Bằng cách sử dụng trình thu gom rác (garbage collector)
65. Để tạo một dự án console với cấu trúc mã truyền thống không sử dụng top-level statements, bạn sử dụng tùy chọn nào trong lệnh dotnet new console
    1. -disable-top-level
    2. -use-program-main
    3. -use-traditional
    4. -use-main

Bài2

1. Trong C#, phuong thuc nao duoc su dung de doc mot chuoi ky tu tu ban phim?
2. Console.ReadLine(
3. Console.Read()
4. Console.Input()
5. Console.ReadKey()
6. Tại sao C# phân biệt chữ hoa và chữ thường khi đặt tên biến?
7. Để ngôn ngữ dễ đọc hơn
8. Dể hỗ trợ lập trình viên trong việc định danh biến một cách rõ ràng
9. Để tăng độ phức tạp của ngôn ngữ
10. giảm xung đột tên biến
11. Trong C#, thứ tự đúng của chuyển đổi kiểu ngầm định để chuyển đổi kiểu nhỏ hơn thành kiểu lớn hơn là
12. char -> int -> long -> float -> double
13. char -> int -> float -> long -> double
14. bool -> char -> int -> long -> float -> double
15. bool -> char -> int -> long -> double -> float
16. Khi sử dụng phương thức Console.ReadLine(), dữ liệu nhập vào sẽ được lưu trữ dưới dạng kiểu dữ liệu nào?
17. double
18. string
19. char
20. Int
21. Từ khoa nao được sử dung để khai bao mot phương thức mà không có giá trị trả về trong C#?
22. void
23. static
24. class
25. Return
26. Để khai báo một mảng không có kích thước xác định trước, cú pháp nào sau đây là đúng?
27. int[] arr = new int[];
28. int[] arr = {1, 2, 3, 4, 5};
29. int[] arr = new int();
30. int[] arr = new int[5x];
31. Viết chương trình để kiểm tra số nguyên dương có hai chữ số. Đoạn mã nào sau đây là đúng?

(A)

int num = int.Parse(Console.ReadLine());

while (num < 10 || num > 99){

num = int.Parse(Console.ReadLine());

(B)

int num = int.Parse(Console.ReadLine());

while (num >= 10 && num <= 99){

num = int.Parse(Console.ReadLine());

}

(C)

int num = int.Parse(Console.ReadLine());

while (num < 1 || num > 99){

num = int.Parse(Console.ReadLine());

}

(D)

int num = int.Parse(Console.ReadLine());

while (num < 10 && num > 99){

num = int.Parse(Console.ReadLine());

}

1. Kiểu dữ liệu nào trong C# chứa chuỗi trực tiếp của dữ liệu mà nó chứa?
2. Kiểu class
3. Kiểu giá trị
4. Kiểu tham chiếu
5. Kiểu delegate
6. Kieu du lieu nao la lớp co sở cho tat ca cac kiểu du lieu ma ngưoi sử dụng được phep định nghĩa trong C#?
7. Object
8. Array
9. Class
10. String
11. Điều gì xảy ra khi hai biến tham chiếu cùng trỏ tới một địa chỉ trong C#?
12. Thao tác trên một biến sẽ ảnh hưởng tới biến kia
13. Các biến sẽ không thể thao tác trên dữ liệu
14. Các biến sẽ sao chép dữ liệu lẫn nhau
15. Hai biến sẽ tự động bị xóa
16. Vòng lặp nào sẽ luôn được thực thi ít nhất một lần, ngay cả khi điều kiện là sai?
17. foreach
18. while
19. for
20. do/while
21. Từ khóa nào được sử dụng để nhảy ra khỏi một vòng lặp trong C#?
22. continue
23. exit
24. break
25. Return
26. Tại sao phải sử dụng Convert. Tolnt32() khi đọc đầu vào từ người dùng trong C#?
27. Để chuyển số thực thành chuỗi
28. Để chuyển số nguyên thành chuỗi
29. Để chuyển chuỗi thành số nguyên
30. Để chuyển chuỗi thành số thực
31. Tên biến trong C# không được phép bắt đầu bằng ký tự nào?
32. Ký tự trong báng chữ cái
33. Ký tự gạch dưới
34. Ký tự số
35. Ký tự chữ hoa
36. Toán tử logic nào trả về True nếu yêu cầu cả hai biểu thức đều phải đúng?
37. !
38. ||O
39. &&
40. ^
41. Để khai báo một mảng không có kích thước xác định trước, cú pháp nào sau đây là đúng?
42. int[] arr = new int();
43. int[] arr = {1, 2, 3, 4, 5};
44. int[] arr = new int[];
45. int[] arr = new int[5x];
46. Đầu ra của đoạn mã C# sau sẽ là gì?

double x = 10.25;

Console.Write(Convert.Tolnt32(x));

1. Lỗi
2. 10.25
3. 10.30
4. 10
5. Đầu ra của đoạn mã C# sau sẽ là gì?

bool x = true;

Console.Write(Convert.ToString(x));

1. false
2. True
3. true
4. False
5. Đặc điểm nào sau đây đúng với kiểu dữ liệu tham chiếu trong C#?
6. Chứa địa chỉ của phương thức
7. Chứa giá trị trực tiếp
8. Chứa tham chiếu tới dữ liệu
9. Không thể chứa giá trị nào

Bài3

1. Một trường (field) trong lớp là gì?
2. Một biến được khai báo trực tiếp trong một lớp.
3. Một tệp tin chứa mã nguồn.
4. Một thư viện chứa các lớp và phương thức.
5. Một phương thức để thực hiện các hành động nhất định.
6. Mot constructor la gi?
7. Một biến chứa giá trị mặc định cho đối tượng.
8. Một lớp kế thừa từ lớp khác.
9. Một phương thức đặc biệt được sử dụng để khởi tạo các đối tượng.
10. Một phương thức để hủy đối tượng
11. Lớp trừu tượng là gì?
12. Một lớp chứa các phương thức tĩnh.
13. Một lớp không thể được sử dụng để tạo đối tượng.
14. Một biến toàn cục có thể truy cập từ bất kỳ đâu.
15. Phương thức get trong thuộc tính được sử dụng để làm gì?
16. Trả về giá trị của biến.
17. Khởi tạo đối tượng mới.
18. Thiết lập giá trị cho biến.
19. Gọi một phương thức khác.
20. Điểm khác biệt chính giữa lớp (class) và đối tượng (object) là gì?
21. Lớp là thể hiện của đối tượng.
22. Đối tượng là mẫu để tạo ra lớp.
23. Đối tượng và lớp là giống nhau.
24. Lớp là mẫu để tạo ra đối tượng.
25. Làm the nao để tạo mot đoi tượng cua lớp Car với tên myCar và thiết lập color thành "blue"?
26. Car myCar = Car("blue");
27. Car myCar = new Car(); myCar.color = "blue";
28. Car myCar = "blue";
29. Car myCar; myCar.color = "blue".
30. Dap an nao duoi day la dung de trien khai phuong thuc animalSound cua giao dien IAnimal trong lop Dog?

(A)

class Dog : IAnimal{

void IAnimal.animalSound(){

Console.WriteLine("Woof");

}

(B)

class Dog : IAnimal{

public void animalSound(){

Console.WriteLine("Woof");

}

}

(C)

class Dog : IAnimal{

void animalSound(){

Console.WriteLine("Woof");

}

}

(D)

class Dog : IAnimal{

private void animalSound(){

Console.WriteLine("Woof");

}

}

1. Làm the nao để tạo mot constructor trong lớp Car nhận tham số color và thiết lập giá trị cho trường color?
2. public Car(string c) { this.c = color; }
3. public Car(string color) { this.color = color; }
4. public Car(string color) { Car.color = color;}
5. public Car(string color) { color = color; }
6. Làm thế nào để gọi phương thức honk của đối tượng myCar thuộc lớp Car, kế thừa từ lớp Vehicle?
7. myCar.honk();get
8. Car.honk();
9. myCar->honk();
10. Vehicle.honk(myCar);
11. Lam the nao de khai bao mot lop Dog ke thua tu lop Animal va ghi de phuong thuc makeSound?

(A)

class Dog: Animal {

public override void makeSound() {

Console.WriteLine("Bark");

}

}

(C)

class Dog: Animal {

override void makeSound() {

Console.WriteLine("Bark");

}

}

(D)

class Dog: Animal {

void makeSound() {

Console.WriteLine("Bark");

}

}

1. Lam the nao đe sử dụng tu khoa sealed đe ngan lớp Vehicle không bị kế thừa?
2. sealed class Vehicle { ... }
3. private sealed class Vehicle { ... }
4. sealed class public Vehicle { ... }
5. protected sealed class Vehicle { ... }
6. Trong C#, từ khoa nào được dùng để định nghĩa một thuộc tính tự động?
7. read; write;
8. field; property;
9. get; set;
10. getValue; setValue;
11. Trong C#, khi nào thì phương thức khởi tạo ngầm định được tạo ra?
12. Khi lớp không có phương thức tĩnh
13. Khi lớp không có thuộc tính
14. Khi trong khai báo lớp không có định nghĩa phương thức khởi tạo nào
15. Khi lớp không có phương thức húy
16. Trong C#, phương thức nào của lớp cho phép khởi tạo giá trị của các thành phần dữ liệu của đối tượng?
17. Method
18. Property
19. Constructor
20. Destructor
21. Trong C#, tại sao nên sử dụng phương thức tĩnh?
22. Để giảm thời gian biên dịch
23. Để tiết kiệm bộ nhớ
24. **Để gọi phương thức mà không cần tạo đối tượng của lớp**
25. Để dễ dàng mở rộng lớp
26. Làm the nao để truy cap một trường trong một đối tượng trong C#?
27. Sử dụng cú pháp dấu chấm (.)
28. Sử dụng dấu gạch chéo (/);
29. Sử dụng dấu chấm phẩy (;)
30. Sử dụng dấu hai chấm ( :: )
31. Tại sao cần có bộ điều chỉnh truy cập (access modifiers) đối với các thành viên của lớp?
32. Để hỗ trợ đa kế thừa.
33. Để tăng tốc độ thực thi của chương trình.
34. Để kiểm soat khả nang hiển thị của các thành viên lớp
35. Để giảm kích thước mã nguồn.
36. Giao diện (interface) là gì?
37. Một lớp chứa các phương thức tĩnh.
38. Một biến toàn cục có thể truy cập từ bất kỳ đâu.
39. Một bản hợp đồng chứa các định nghĩa cho một nhóm các chức năng liên quan mà một lớp không trừu tượng hoặc một cấu trúc phải thực hiện.
40. Một tệp tin chứa mã nguồn C#.
41. Trong C#, làm thế nào để đảm bảo một thuộc tính chỉ có thể đọc mà không thể ghi?
42. Chỉ định phương thức set là private
43. Chỉ định phương thức set và bó qua phương thức ngầm
44. Chỉ định phương thức get và bó qua phương thức set
45. Chỉ định phương thức publiclà public
46. Khi sử dụng từ khóa this trong phương thức thành phần của lớp, từ khóa này trỏ đến đối tượng nào?
47. Đối tượng hiện tại của lớp đó
48. Đối tượng cha của lớp đo
49. Đối tượng mới được tạo
50. Đối tượng con của lớp đó
51. Trong lập trình hướng đối tượng, tính chất nào cho phép đối tượng có thể có các hành động khác nhau dựa trên việc gọi cùng một tên phương thức?
52. Tính trừu tượng
53. Tính đa hình
54. Tính đóng gói
55. Tính kế thừa
56. Từ khoa nao được sử dụng để tạo mot đoi tượng trong C#?
57. object
58. class
59. create
60. new
61. Một constructor là gì?
62. Một biến chứa giá trị mặc định cho đối

tượng

1. Một phương thức để hủy đối tượng
2. Một lớp kế thừa từ lớp khác.
3. Một phương thức đặc biệt được sử dụng

để khởi tạo các đối tượng.

1. Tại sao lập trình hướng đối tượng (OOP) dễ bảo trì hơn lập trình thủ tục?
   1. Vì OOP có ít dòng mã hơn.
   2. Vì OOP không sử dụng biến toàn cục
   3. Vì OOP không cần biên dịch mã nguồn
   4. Vì OOP giúp giữ mã DRY và làm cho mã dễ

bảo trì, sửa đổi và gỡ lỗi hơn.

1. Làm the nao để khai bao một lớp Animal là trừu tượng và chứa phương thức trừu tượng **makeSound?**

A)

class Animal{

abstract void makeSound();

(B

abstract class Animal{

public void makeSound();

}

(C)

abstract class Animal

{public abstract void makeSound();

}

(D)

class Animal{

public abstract void makeSound();

1. Làm thế nào để ghi đè phương thức makeSound trong lớp con Bird kế thừa từ **lớp cha Animal?**

(A)

class Bird: Animal {

public override void makeSound() {

Console.WriteLine("Chirp ..... ");

} }

(B

class Bird: Animal {

public void makeSound() {

Console.WriteLine("Chirp ..... ");

} }

(C)

class Bird: Animal {

override void makeSound() {

Console.WriteLine("Chirp ..... ");

} }

(D)

class Bird: Animal {

void makeSound() {

Console.WriteLine("Chirp ..... ");

} }

1. Trong C#, tại sao lại sử dụng thuộc tính thay vì biến thành viên public?
2. Để giảm thời gian biên dịch
3. Để dễ dàng mở rộng lớp
4. Để tiết kiệm bộ nhớ
5. Để kiểm soát dữ liệu khi truy xuất và gán

giá trị

1. Làm the nao đe khai bao mot thuoc tính Name công khai với phương thức get và set trong lớp Person?
   1. private string Name { get; set; }
   2. protected string Name { get; set; }
   3. public string Name { get; set; }
   4. internal string Name { get; set; }
2. Trong C#, tham số nào được sử dụng để truyền gia trị đầu vào cho phương thức và không bị thay đổi trong phương thức đó?
3. Tham số tham chiếu
4. Tham số tham trị
5. Tham số hằng
6. Tham số đầu ra
7. Làm thế nào để tạo một đối tượng từ lớp Car?
8. Car myObj = create Car();
9. myObj = new Car;
10. Car myObj = new Car();
11. Car myObj = Car();
12. Làm thế nao để khai báo và khởi tạo một đối tượng Ford của lớp Car với model "Mustang", color "red", và year 1969?
13. Car Ford = Car("Mustang", "red", 1969);
14. Car Ford = new Car(); Ford.model =

"Mustang"; Ford.color = "red"; Ford.year =1969;

1. Car Ford; Ford.model = "Mustang";

Ford.color = "red"; Ford.year = 1969;

1. Car Ford = new ("Mustang", "red", 1969);
2. Nguyên tắc "Don't Repeat Yourself" (DRY) có nghĩa là gì?
3. Việc sử dụng cùng một biến nhiều lần trong các chương trình.
4. Việc giữ mã nguồn khô ráo, không bị lỗi
5. Việc trích xuất các đoạn mã chung và đặt chúng ở một nơi duy nhất để tái sử dụng.
6. Việc viết lại mã nguồn nhiều lần để dễ bảo trì.

Bài 4

1. Events trong .NET là gì?
2. Là một biến toàn cục
3. Là một dạng hạn chế của delegate để phù hợp với mô hình

publisher/subcriber

1. Là một phương thức
2. Là một dạng mở rộng của delegate để phù hợp với mô hình publisher/subcriber
3. Delegate có thể được sử dụng để làm gì trong C#?
4. Lưu trữ tham chiếu đến một hoặc nhiều phương thức
5. Tạo ra một biến toàn cục
6. Quản lý bộ nhớ
7. Thay thế lớp
8. Generics trong C# cho phép làm gì?
9. Tăng độ phức tạp của mã nguồn
10. Định nghĩa các lớp, giao diện và phương thức với các kiểu dữ liệu được xác định sau tại thời điểm sứ dụng
11. Giảm bộ nhớ sử dụng
12. Tăng tốc độ xử lý
13. Tại sao việc sử dụng delegate giúp mã nguồn trở nên linh hoạt hơn?
14. Delegate cho phép thay đổi hành vi của các phương thức mà không cần

thay đổi mã nguồn của chúng.

1. Delegate làm tăng độ phức tạp của mã nguồn.
2. Delegate giúp giảm bộ nhớ sử dụng
3. Delegate giúp tăng tốc độ xử lý.
4. Trong lập trình sự kiện (events), tại sao cần phải hủy đăng ký sự kiện sau khi sử dụng?
5. Để tăng tính linh hoạt của mã nguồn.
6. Để đám bao an toàn dữ liệu.
7. Để giảm bộ nhớ sử dụng và tránh rò rí bộ nhớ.
8. Để giảm số dòng lệnh
9. Khai báo delegate nào dưới đây trong C# là đúng
10. public delegated void Action();
11. public delegate void Action;
12. public delegate void Action();
13. public void delegated Action();
14. Khai báo delegate nào trong C# dưới đây là đúng
15. public delegate voided Action<in T1, in T2>(T1 arg1, T2 arg2);
16. public delegate void Action<in T1, in T2>(T1 arg1, T2 arg2)
17. public delegate void Action<in T1, in T2>(T1 arg1, T2 arg2);
18. public delegated void Action<in T1, in T2>(T1 arg1, T2 arg2);\
19. Lambda expression có thể chứa nhiều câu lệnh không?
20. Có, nhưng không thể sứ dụng dấu ngoặc nhọn
21. Không, chỉ có thể chứa một câu lệnh duy nhất
22. Có, nhưng phải sứ dụng từ khóa "return"
23. Có, nhưng phai đặt trong dấu ngoac nhọn {}\
24. Khi nào nên sử dụng EventArgs trong sự kiện?
25. Khi sự kiện cần truyền dữ liệu bổ sung
26. Khi cần uỷ nhiệm cho sự kiện
27. Khi sự kiện là phương thức tĩnh
28. Khi sự kiện không cần truyền dữ liệu nào
29. Delegate multicast được tạo ra bằng cách nào?
30. Sử dụng toán tử + hoặc += để thêm các phương thức vào delegate
31. Sử dụng toán tử - để loại bó các phương thức khỏi delegate
32. Sử dụng toán tử \* để nhân các phương thức trong delegat
33. Sứ dụng toán tứ / để chia các phương thức trong delegate
34. Đap an nào dưới đay dùng để khai báo một sự kiện cho delegate sau

public delegate void ClickHandler();

1. public event ClickHandler OnClick;
2. public ClickHandler OnClick;
3. public event ClickHandler OnClick();
4. public event void OnClick;
5. Thứ tự thực hiện các phương thức trong delegate multicast là gì?
6. Theo thứ tự ngau nhiên
7. Theo thứ tự chúng được thêm vào
8. Theo thu tu ngược lại chung được thêm vào
9. Theo thứ tự alphabet
10. Event thường được sử dụng trong ngữ cảnh nào?
11. Trong các ứng dụng GUI và callback
12. Trong quản lý bộ nhớ
13. Trong các lớp tĩnh
14. Xử lý ngoại lệ
15. Từ khóa nào được sử dụng để khai báo một event?
16. class
17. interface
18. event
19. Delegate
20. Anonymous method có thể truy cập vào những gì?
21. Chỉ các biến tĩnh
22. Chỉ các biến toàn cục
23. Các biến cục bộ của phạm vi chứa nó
24. Chỉ các phương thức tĩnh