**I. Các câu hỏi về lập trình cấu trúc, đệ quy.**

1. Thiết kế lưu đồ thuật toán cho mỗi chức năng của một chương trình C nào đó.
2. Bản chất biến trong một chương trình C là gì. Phân biệt biến toàn cục, biến cục bộ và biến môi trường. Cho ví dụ minh họa.

**- Bản chất của Biến**: là vùng nhớ dùng để lưu trữ các giá trị của chương trình. Mỗi biến gắn liền với một kiểu dữ liệu và một định danh duy nhất gọi là tên biến.

- Biến toàn cục: Biến được khai báo bên ngoài phạm vi các hàm, biến này được dùng chung cho các hàm(dùng cho cả chương trình)

- Biến cục bộ: là biến được dùng và khai báo trong 1 hàm và chỉ được dùng trong hàm đã khai báo nó.

- Biến môi trường: là môi trường để chạy một ngôn ngữ lập trình. Ví dụ để chạy được java thì cần có JDK

1. Phân biệt giữa biên dịch và thông dịch; cho ví dụ minh họa. Mô tả quá trình biên dịch một chương trình C; chỉ rõ các file trung gian tạo ra khi biên dịch trong IDE CodeBlock

Biên dịch: chuyển các câu lệnh được gõ bằng 1 ngôn ngữ lập trình(mã nguồn) sang một chương trình tương đương nhưng bằng một ngôn ngữ máy tính mới (chương trình đích). Lần sau muốn chạy lại chương trình, chỉ cần chạy lại chương trình đã được dịch.  
Ví dụ: khi soạn xong 1 chương trình C, sau khi biên dịch bạn sẽ được chương trình dạng mã máy (.exe)

Thông dịch: sau khi soạn thảo một chương trình thì quá trình thông dịch là quá trình xảy ra lúc runtime (khi thực hiện), trình thông dịch sẽ dịch từng lệnh của chương trình và thực thi. Lần sau muốn chạy lại chương trình thì phải thông dịch lại.

Ví dụ: php hoặc asp. Mỗi khi bạn chạy website, trình thông dịch sẽ dịch lại từ đầu mã nguồn và thực thi.

File trung gian tạo ra khi biên dịch trong IDE CodeBlock:

-File.exe:

-File.o

1. Mô tả quá trình cấp phát bộ nhớ trong quá trình thực hiện một chương trình C.

Khi ta viết 1 chương trình hay khai báo 1 biến trong 1 chương trình thì chúng đều có giá trị và địa chỉ của chúng. Khi chương trình khởi chạy, hệ điều hành sẽ phân ra thành 4 vùng nhớ trong RAM. Biến và mã nguồn sẽ được lưu trữ tại các vùng nhớ này. Các vùng nhớ:

Code Segment: Mã nguồn của chúng ta sau khi hoàn tất quá trình Build sẽ chuyển thành các đoạn mã máy 0 và 1. Khi chương trình khởi chạy, các đoạn mã máy này sẽ được nạp vào Code Segment. Cách duy nhất để truy xuất vào vùng nhớ này là sử dụng Con trỏ hàm. Code Segmentcó kích thước cố định

Data Segment: Khi chương trình được khởi chạy, toàn bộ các biến toàn cục và static đều được lưu trữ ở đây, các đoạn chuỗi cố định cũng được lưu trữ trong Data Segment. Data Segment có kích thước cố định.

Stack Segment: Đây là vùng nhớ mà chúng ta cần quan tâm. Khi một hàm được gọi, hàm đó sẽ được đưa vào vùng nhớ Stack, các biến được khai báo trong hàm đó cũng được đưa vào vùng nhớ Stack. Khi hàm kết thúc, toàn bộ các biến trong hàm cùng với bản thân hàm sẽ được tự động giải phóng để các hàm sau sử dụng. Stack Segment có kích thước cố định

Heap Segment: Đây là vùng nhớ khác mà chúng ta cần quan tâm, vùng nhớ này chúng ta phải hoàn toàn kiểm soát nó. Khi chúng ta sử dụng con trỏ và cấp phát động một vùng nhớ cho con trỏ quản lý, vùng nhớ này sẽ nằm trong Heap Segment. Heap Segment có kích thước không cố định

1. Mô tả quá trình thực hiện và tổ chức các thành phần bên trong bộ nhớ của một hàm đệ quy.
2. Bản chất quá trình chuyển điều khiển giữa chương trình chính và chương trình con. Tại sao trong chương trình sử dụng nhiều chương trình con lại làm tăng thời gian thực hiện.
3. Giải thích cách tính miền giá trị của các kiểu dữ liệu. Mô tả cách tổ chức của mảng trong bộ nhớ. Trong C một mảng có tối đa bao nhiêu phần tử.
4. Bản chất biến là gì; địa chỉ ô nhớ là gì; con trỏ là gì; con trỏ hàm là gì. Ý nghĩa sử dụng của con trỏ hàm; cho ví dụ minh họa.

**- Bản chất của Biến**: là vùng nhớ dùng để lưu trữ các giá trị của chương trình. Mỗi biến gắn liền với một kiểu dữ liệu và một định danh duy nhất gọi là tên biến.

*Con trỏ đơn giản là một biến kiểu số nguyên dùng để chứa địa chỉ.*

1. Trong một chương trình C, bản chất biến là gì; địa chỉ ô nhớ là gì. Mô tả quá trình thực hiện của một chương trình.

**II. Các câu hỏi về lập trình hướng đối tượng**

1. Thiết kế biểu đồ lớp và mô tả ảnh bộ nhớ (memory image) khi thực hiện một chương trình Java.
2. **Trình bày về nạp chồng và ghi đè, chỉ rõ ý nghĩa sử dụng.**

* *Nạp chồng – Overloading:*

+ là phương thức có cùng tên, kiểu dữ liệu truyền vào khác nhau

* *Ghi đè – Overriding:*

+ Là phương thức xuất hiện ở lớp cha rồi có thể xuất hiện tiếp ở lớp con

+ Là phương thức có cùng tên và các đối số phải có kiểu dữ liệu giống nhau

1. **Phân biệt giữa các thuộc tính static và non-static về: thời gian tồn tại, ý nghĩa và cách sử dụng, tổ chức bộ nhớ**

* *Biến static:*

+ phụ thuộc vào class, không phụ thuộc đối tượng

+ thời gian tồn tại: từ lúc chương trình bắt đầu đến khi kết thúc

+ Ý nghĩa: Mọi đối tượng đều có thể sử dụng chung biến static này

+ cách sử dung: teeclass.tenbienstatic

+ khi gọi lại biến static thì không phải cấp phát bộ nhớ mới

* *Non-static:*

+ phụ thuộc vào đối tượng

+ ý nghĩa: khi đối tượng truy cập mới được sử dụng

+ khi gọi biến non-static ở những đối tượng khác nhau thì sẽ cấp phát những bộ nhớ khác nhau

1. **Trình bày về tính đa hình, ý nghĩa sử dụng, cho ví dụ minh họa.**

* Đa hình có nghĩa là 1 biến của supertype có thể tham chiếu đến đối tượng subtype
* Ý nghĩa: thường được sử dụng để gọi các phương thức nạp chồng của đối tượng
* Ví dụ:

Class Nguoi{

String hoten;

}

Class SV extends Nguoi{

}

Class NV extends Nguoi{

}

Public static void showHoten(Nguoi ng){

System.out.println(“hoten: ” + ng.hoten);

}

1. **Phân biệt giữa lập trình hướng đối tượng và hướng cấu trúc. Phân biệt giữa lớp và đối tượng; lớp trừu tượng và lớp thông thường.**

Phân biệt giữa lớp và đối tượng; lớp trừu tượng và lớp thông thường.

Lớp định nghĩa 1 thực thể, đối tượng là trường hợp của thực thể ấy

Đối tượng là mô hình thực , trong khi lớp là mô hình khái niệm

Tất cả các đối tượng của cùng 1 lớp có cùng thuộc tính và phương thức

Lớp trừu tượng bắt buộc phải có từ khóa abstract, và chứa các phương thức trừu tượng

Lớp trừu tượng là giống như 1 lớp thông thường nhưng không thể tạo đối tượng từ lớp trừu tượng

Lớp trừu tượng vẫn có thể được kế thừa từ những lớp thông thường , các class kế thừa bắt buộc phải có các phương thức trừu tượng

1. **Mô tả cách cấp phát bộ nhớ của biến cục bộ, tham số, thuộc tính non-static, static. Nên sử dụng các thành phần nào trong chương trình khi muốn tối ưu hiệu năng, khi muốn tối ưu bộ nhớ.**

- Cấp phát biến toàn cục: khai báo biến không nằm trong 1 hàm nào (kể cả hàm main()) có phạm vi trong suốt toàn bộ chương trình

- Tham số là biến được truyền vào hàm khi thực hiện lời gọi hàm từ hàm main()

- Non-static = biến cục bộ: khai báo trong hàm main() or bất kỳ hàm nào đó và chỉ có phạm vi trong hàm, khi thoát khỏi hàm sẽ được giải phóng

- static: là biến dùng chung cho tất cả các đối tượng, tức là khi gọi lại biến này với các đối tượng khác nhau thì vùng nhớ không cần phải cấp phát mới

ví dụ : static int n;

- Muốn tối ưu hiệu năng: sử dụng biến toàn cục, và biến static

- Muốn tối ưu bộ nhớ : sử dụng biến cục bộ, và biến non-static

1. **Trình bày cách cấp phát và tổ chức của một mảng các đối tượng trong bộ nhớ. Cho ví dụ minh họa.**

Ví dụ có class SV{}

SV array = new SV[20];(khởi tạo và khai báo mảng đối tượng SV)

For(int i = 0 ; i < n ; i++)

Array[i] = new SV(); (khởi tạo từng đối tượng của mảng đối tượng SV)

Mảng đối tượng tức là giống như mảng các số nguyên, nhưng mỗi phần tử của mảng đối tượng là 1 đối tượng của class nào đó, và từng đối tượng của mảng đối tượng cần được khởi tạo như khởi tạo 1 đối tượng bình thường

1. **Trình bày về đơn kế thừa, đa kế thừa trong lập trình hướng đối tượng; cho ví dụ minh họa. So sánh giữa lớp trừu tượng (abstract class) và giao diện (interface); cho ví dụ minh họa.**

- đơn kế thừa tức là 1 class chỉ được kế thừa duy nhất từ 1 class

- đa kế thừa tức là 1 class có thể kế thừa nhiều interface

or interface kế thừa interface

so sánh abstract and interface

+ abstract mang tính năng của 1 class bình thường (ngoại trừ việc tạo các đối tượng)

Còn interface chỉ chứa các hằng final static, và các abstract method

+ tính kế thừa: với abstract thì 1 class chỉ có thể kế thừa 1 abstract class

Còn với interface, 1 class kế thừa được nhiều interface

1. **Bản chất và ý nghĩa sử dụng của: lớp trừu tượng, giao diện và tính đa hình.**

- nhiều class kế thừa abstract class thì tất cả các class này có bản chất giống với abstract class

Ví dụ:

abstract class Nguoi{

String hoten;

Abstract void nhapTen();

}

Class NhanVien extends Nguoi{

Void nhapTen(){

System.out.println(“Nhap ten Nhan vien: “);

Hoten = input.nextLine();

}

}

Class Sinhvien extends Nguoi{

Void nhapTen();

System.out.println(“Nhap ten Sinh vien: “);

Hoten = input.nextLine();

}

}

* 1 class kế thừa Interface : tức là thêm và bổ sung đầy đủ các thức hoạt động của nó

**Ví dụ:**

Public interface Sound

{

Public abstract String whatSound();

}

Public abstract class Eat

{

Public abstract String whatEat();

}

Public class Dog extends Eat implements Sound

{

Public String whatSound()

{

Return “gâu gâu”;

}

Public String whatEat()

{

Return “gặm xương”;

}

}

Public class Chicken extends Eat

{

Public String whatEat(){

Return “ăn thóc”;

}

}

**Đa hình:**

- Đa hình có nghĩa là 1 biến của supertype có thể tham chiếu đến đối tượng subtype

- Ý nghĩa: thường được sử dụng để gọi các phương thức nạp chồng của đối tượng

- Ví dụ:

Class Nguoi{

String hoten;

}

Class SV extends Nguoi{

}

Class NV extends Nguoi{

}

Public static void showHoten(Nguoi ng){

System.out.println(“hoten: ” + ng.hoten);

}

1. **Với cùng chức năng và thuật toán, so sánh tốc độ thực hiện của chương trình được viết bằng C và viết bằng Java. Giải thích.**

Lý do:

C: là biên dịch - tức là dịch toàn bộ chương trình sang mã máy rồi mới thực thi. Khi thực hiện lại thì chỉ chạy file thực thi

Java: vừa biên dịch vừa thông dịch: biên dịch chương trình nguồn sang thành bytecode, rồi bytecode mới được môi trường thực thi thực hiện

**III. Các câu hỏi về lập trình phân tán**

1. Mô tả quá trình xử lý (client-server) khi truy xuất và thực hiện các chức năng trong một trang web.
2. Phân biệt sự khác nhau khi gửi dữ liệu từ client lên server theo phương thức POST và GET.

**Phương thức GET:**

Client Gửi Lên: Phương thức GET là phương thức gửi dữ liệu thông qua đường dẫn URL nằm trên thanh địa chỉ của Browser. Server sẽ nhận đường dẫn đó và phân tích trả về kết quả cho bạn. Server sẽ phân tích tất cả những thông tin đằng sau dấu hỏi (?) chính là phần dữ liệu mà Client gửi lên.

Ví dụ: Với URL freetuts.net?id=12 thì Server sẽ nhận được giá trị id = 12

**Phương thức POST:**

Client Gửi Lên: Với phương thức GET thì dữ liệu được thấy trên URL thì phương thức POST thì hoàn toàn ngược lại, POST sẽ gửi dữ liệu qua một cái form HTML và các giá trị sẽ được định nghĩa trong các input gồm các kiểu (textbox, radio, checkbox, password, textarea)và được nhận dang thông qua tên (name) của các input đó.

**So Sánh Giữa POST và GET**

**Giống nhau:**

Đều gửi dữ liệu lên Server

**Khác nhau:**

Phương thức POST bảo mật hơn GET vì dữ liệu được gửi ngầm bằng mắt thường không thể nhìn thấy được.

Phương thức GET dữ liệu được gửi tường minh, chúng ta có thể thấy trên URL nên nó không bảo mật.

Phương thức GET luôn luôn nhanh hơn POST vì dữ liệu gửi đi được Browser giữ lại trong cache, khi thực thi với POST thì Server luôn thực thi lệnh rồi trả về cho Client, còn với GET thì Browser sẽ kiểm tra trong cache có chưa, nếu có thì trả về ngay chứ không cần gửi lên Server.

Khi nào dùng GET, POST.

Khi dữ liệu bạn muốn SEO thì phải sử dụng phương thức GET.

Khi dữ liệu bạn không cần bảo mật thì dùng phương thức GET, ngược lại dữ liệu bảo mật thì dùng phương thức POST.

Ví dụ khi đăng nhập, Comment, đăng tin dùng phương thức POST. Còn khi lấy tin ra thì dùng phương thức GET…

1. Phân tích hoạt động khác nhau khi chọn một nút nhấn kiểu reset và một nút nhấn kiểu submit.

- Nút Submit là một trường hợp đặc biệt của button, cũng như nút Reset. Nút này đưa thông tin hiện tại từ các trường của form tới địa chỉ URL được chỉ ra trong thuộc tính ACTION của thẻ form sử dụng cách thức METHOD chỉ ra trong thẻ FORM.

Giống như đối tượng button và reset, đối tượng submit có sẵn thuộc tính name và value, cách thức click() và thẻ sự kiện onClick.

- Sử dụng đối tượng reset, bạn có thể tác động ngược lại để click vào nút Reset. Cũng giống đối tượng button, đối tượng reset có hai thuộc tính là name và value, và một cách thức click(), một thẻ sự kiện onClick.

Hầu hết những người lập trình khong sử dụng thẻ sự kiện onClick của nút reset để kiểm tra giá trị của nút này, đối tượng reset thường dùng để xoá form.

1. Phân tích các điểm khác biệt giữa lập trình hệ cô lập và hệ phân tán