Contents

[1.Lộ trình học 2](#_Toc111753923)

[2.fgets 2](#_Toc111753924)

[3.Strlen 3](#_Toc111753925)

[4.Bộ đệm 3](#_Toc111753926)

[5.CentOS/Red Hat Enterprise Linux là gì? 4](#_Toc111753927)

[hàm trả về một con trỏ 4](#_Toc111753928)

[6.Đề thi 5](#_Toc111753929)

[7.Web học hay 9](#_Toc111753930)

[8.Tham Số và Đối Số Khác Nhau Như Thế Nào 10](#_Toc111753931)

[9.Clearing The Input Buffer 10](#_Toc111753932)

[1000.Document refer: 10](#_Toc111753933)

# 1.Lộ trình học

**(0 - 1) Khởi đầu (Chỉ biết các lệnh cơ bản của C/C++ hoặc chưa biết gì về nó)**



**(2 - 4) Quản lý dữ liệu (Thành thạo cấu trúc điều khiển, biết sử dụng mảng, con trỏ)**



**(5 - 6) Hướng cấu trúc (Biết dùng các hàm để đơn giản hóa chương trình và nối các hàm lại với nhau)**



**(7 - 10) Hướng đối tượng (Biết dùng class hoặc struct để tạo các dạng dữ liệu cho riêng mình)**



**(11 - 12) Advanced C/C++ (Thành thạo namespace, template, ép kiểu dữ liệu, preprocesser)**



**(13 - 15) Input / Output (Biết điều khiển input, output, định dạng output, xử lý với file)**



**(16 - 17) C++ Container (Biết dùng vector, list, map, ... để thay thế mảng thông thường và thành thạo điều khiển container)**



**(18 - 19) C/C++ Standard Library (Biết tất cả các thư viện chuẩn và công dụng của từng thư viện)**



**(20+) C++ Expert (Thành thạo tất cả mọi thứ về C/C++, hoàn thiện các kĩ năng cần thiết khi đi làm việc hay vào các cuộc thi Tin học)**

Vì sao bạn lại chọn học C/C++? Bạn có chút kiến thức gì về máy tính không? (trình bày trong vòng 3 dòng trở lên) / Why you choose to learn C/C++? Do you have any knowlege about the computer? (describe in more than 3 lines)

# 2.fgets

The C library function char \*fgets(char \*str, int n, FILE \*stream) reads a line from the specified stream and stores it into the string pointed to by str. It stops when either **(n-1) characters are read**, the **newline character** is read, or the **end-of-file** is reached, whichever comes first.

Cú pháp: fgets(name, count, file)

Name: tên biến cần nhập

Count: số ký tự tối đa nhập vào

Flie ở đây là stdin

# 3.Strlen

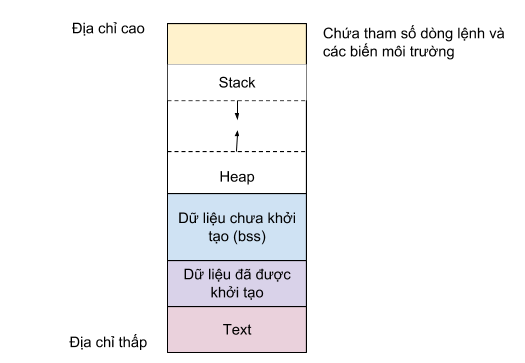
Note that the strlen() function doesn't count the null character \0 while calculating the length.

This-> người ta gọi là con trỏ this. Dùng để thao tác trên các thành viên và phương thức nội tại của một class.

This-> chỉ được sử dụng bên trong class.

Thực ra thì cpp ít khi dùng “this” khi nào mà hàm có tham số hoặc biến cùng tên thì mới dùng để phân biệt thôi

# 4.Bộ đệm

Sơ đồ tổ chức bộ nhớ của một chương trình  
Stack và Heap?Heap là vùng nhớ dùng để cấp phát cho các biến tỉnh hoặc các vùng nhớ  
được cấp phát bằng hàm malloc()

Stack là vùng nhớ dùng để lưu các tham số và các biến cục bộ của hàm.  
Các biến trên heap được cấp phát từ vùng nhớ thấp đến vùng nhớ cao.

Trên stack thì hoàn toàn ngược lại, các biến được cấp phát từ vùng nhớ  
cao đến vùng nhớ thấp.

Stack hoạt động theo nguyên tắc "vào sau ra trước"(Last In First Out -  
LIFO). Các giá trị được đẩy vào stack sau cùng sẽ được lấy ra khỏi stack  
trước tiên.

# 5.CentOS/Red Hat Enterprise Linux là gì?

**Red Hat Enterprise Linux** là một bản phân phối **Linux**thương mại dành cho máy chủ và máy trạm, được phát triển dựa trên Fedora, nhưng có một nền tảng ổn định và được hỗ trợ lâu dài hơn.

Char str1[max]

int d1=sizeof(str1)/sizeof(str1[0]);

**🡪d1=300 kích thước khởi tạo ban đầu cho mảng**

**//===============================================**

**Char \*str1**

int d1=sizeof(str1)/sizeof(str1[0]);

🡪d1=4

# hàm trả về một con trỏ

#include <stdio.h>

//Ở đây chúng ta sẽ viết hàm truyền vào là

**int**\* timSoLonNhat**(int \*x, int \*y)**

{

**if**(\*x>\*y)

   {

**return** x; // trả về địa chỉ con trỏ x trỏ tới

   }

**else**

   {

**return** y; // trả về địa chỉ con trỏ y trỏ tới

   }

}

**int** main**()**

{

**int** \*result;

**int** a = 11;

**int** b = 15;

   result = timSoLonNhat(&a,&b); // truyền vào địa chỉ của a và b

   printf("So lon nhat = %d\r\n",\*result);

**return** 0;

}

Kết quả:

So lon nhat = 15

# 6.Đề thi

Câu 1: Thiết kế biểu đồ lớp và mô tả ảnh bộ nhớ (memory image) khi thực hiện một chương trình Java.  
**Câu 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  
  
**Câu 3: 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  
**Câu 4: 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);  
}  
**Câu 5: 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  
**Câu 6: 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  
**Câu 7: 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  
**Câu 8: 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  
**Câu 9: 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);  
}  
**Câu 10: 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

# 7.Web học hay

Học C

<https://projecteuler.net/archives>

Lưu ý khi học:

Sách thì trên mạng có rất nhiều : bạn có thể search với từ khóa: cấu trúc dữ liệu và giải thuật, tối ưu thuật toán, Data Structure and Algorithms. Bạn có thể ra mấy chổ mua giáo trình Bách Khoa hay Khoa học tự nhiên gì đó, nhiều lắm!  
À còn nữa: bạn phải học con trỏ cho thật bá nhé, đi phỏng vấn người ta toàn hỏi con trỏ không à 

**Nạp chồng toán tử( operator overloading)**

Cú pháp cho nạp chồng toán tử một ngôi:

<kiểu trả về> operator <toán tử cần nạp chồng> (đối số 1)

{

//xử lí

return <biến theo kiểu trả về>;

}

Cú pháp overloading cho toán tử 2 ngôi:

friend <kiểu trả về> operator <toán tử cần nạp chồng> (đối\_số\_1 , đối\_số\_2)

{

//khởi tạo 1 đối tượng để gán giá trị

//các phép gán giá trị

//return đối tượng trả về

}

# 8.Tham Số và Đối Số Khác Nhau Như Thế Nào

Tham Số (Parameter) Được Định Nghĩa Trong Hàm

Đối Số (Argument) Là Giá Trị Truyền Vào Khi Gọi Hàm

# 9.Clearing The Input Buffer

// flushes the standard input

// (clears the input buffer)

while ((getchar()) != '\n');

<https://www.geeksforgeeks.org/clearing-the-input-buffer-in-cc/>

# 1000.Document refer:

-<https://docs.microsoft.com/en-us/cpp/cpp/cpp-language-reference?view=msvc-170>

-<https://en.cppreference.com/w/>

-<https://www.geeksforgeeks.org/c-plus-plus/>

-<https://www.tutorialspoint.com/cplusplus/index.htm>