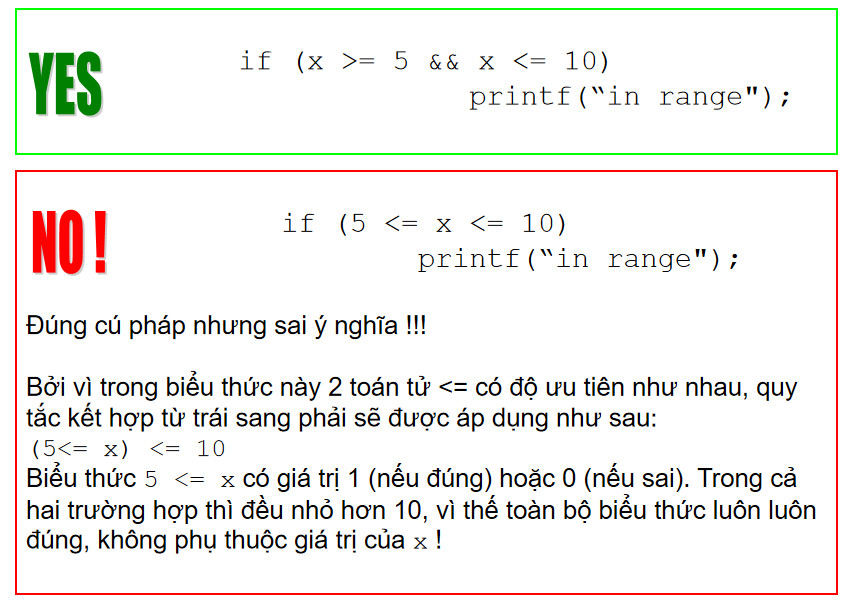
KIỂM TRA KIẾN THỨC LÝ THUYẾT - LẬP TRÌNH CĂN BẢN A

1. T01\_GioiThieu\_PTPhi.ppt

* Kiến thuyết cần nhớ:
* Test:

1. T02\_Bien\_Kieu\_Cautrucrenhanh\_phcuong.ppt

* Kiến thức cần nhớ:
* Biến: vùng nhớ được đặt tên để lưu trữ dữ liệuTrong ngôn ngữ C, biến phải được khai báo trước khi sử dụng. Việc khai báo bao gồm đặt tên cho biến và quy định kiểu dữ liệu của biếnDùng phép toán gán trị để thay đổi giá trị của biến
* Tên có thể gồm chữ cái, chữ số và dấu gạch dưới.Tên phải bắt đầu bằng chữ cái hoặc dấu gạch dưới ( \_ ) .Tên không được trùng với từ khóa (các từ dành riêng của C)
* scanf(chuỗi định dạng, danh sách biến)
* printf(chuỗi định dạng, danh sách tham số)
* notes:



* Test:

1. Trong các tên biến sau, tên nào đặt **sai** quy cách

a. tong b. a15 c. \_var d. 12A1

2. Trong các kiểu dữ liệu sau, kiểu nào có miền giá trị là các **số thực**?

a. int b. char c. float d. long int

3. Nếu công việc trong cấu trúc if-else có từ 2 câu lệnh trở lên ta phải làm gì?

a. Viết lệnh bình thườngb. Viết xuống dòng

c. Dùng cặp ngoặc { } để bao các câu lệnh lại

d. Tất cả đều sai

4. Khi gán a = 5.2 (với a là biến kiểu **int**), giá trị lưu trữ vào a là gì ?

a. 5.2 b. 0 c. 5 d. tất cả đều sai

5. Gán a = (1/2)\*b với a, b là các biến kiểu int và giá trị b đang là 10. Giá trị của a là?a. 5 b. 0 c. 10 d. tất cả đều sai

1. T03\_Cautruclap\_phcuong.ppt

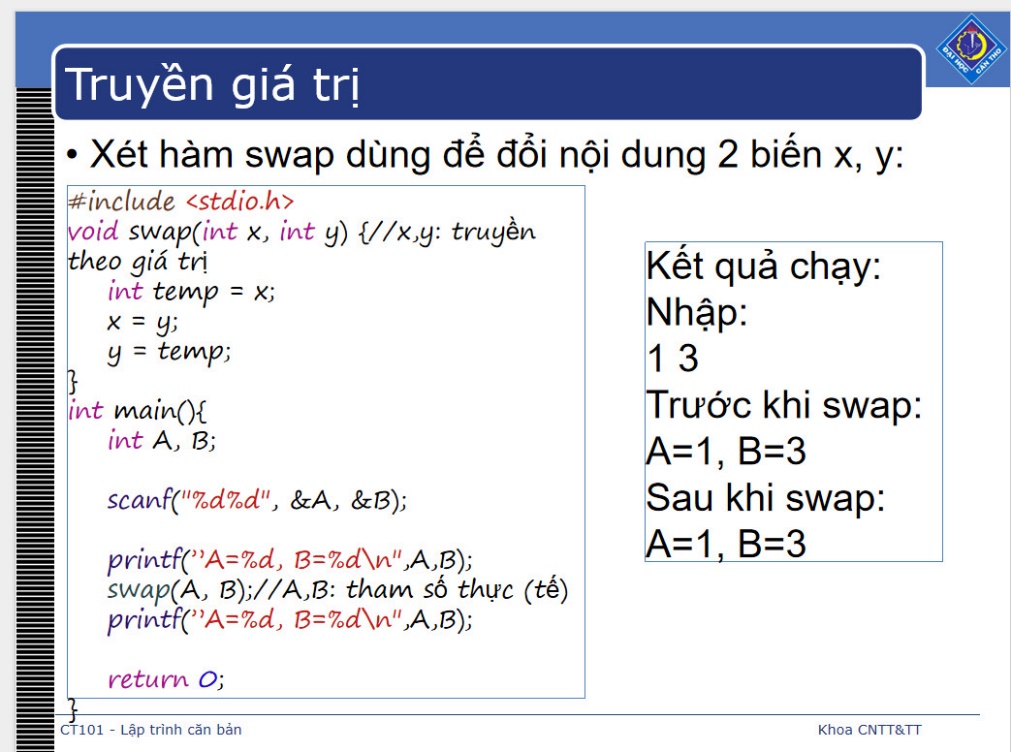
* Kiến thức cần nhớ:
* Chọn theo cách xác định điều kiện lặpKiểm tra điều kiện trước khi lặp: for, whileKiểm tra điều kiện sau khi lặp: doChọn theo số lần lặp:Không xác định trước: whileXác định trước: for
* Test:

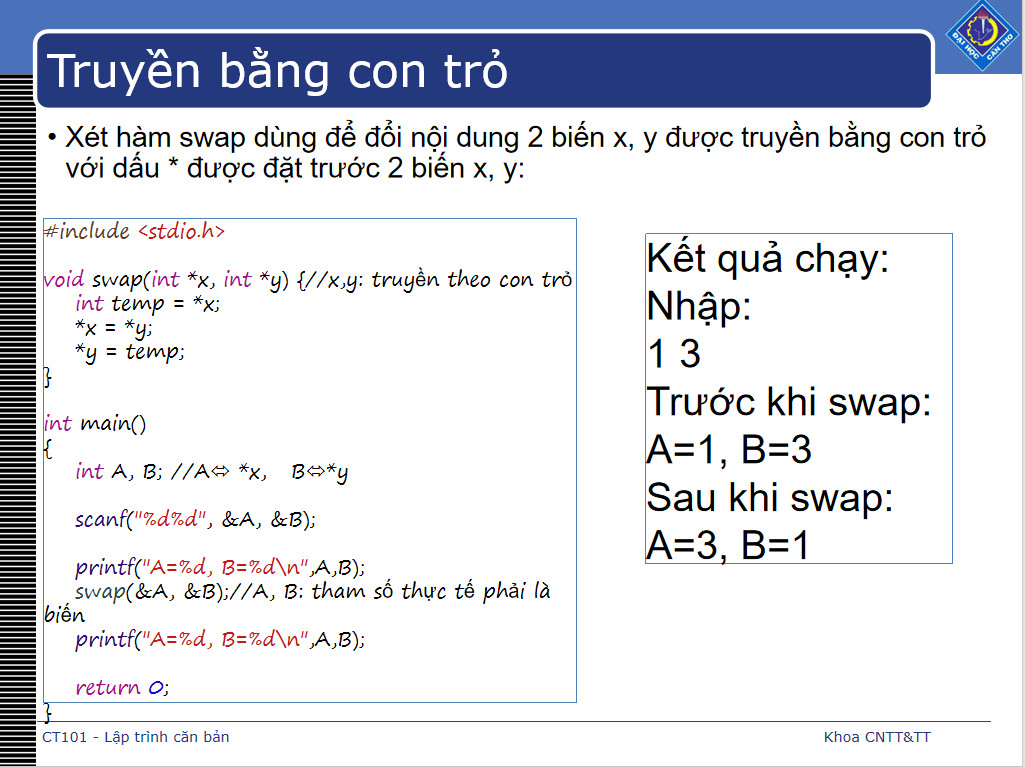
1. Trong C có bao nhiêu cấu trúc lặp ?a. 1 b. 2 c. 3 d. 4
2. Số lần lặp ít nhất của vòng lặp while là bao nhiêu?a. 0 b. 1 c. 2 d. 33.
3. Có thể dùng vòng lặp for để thay thế vòng lặp while (và ngược lại) được không?
4. Không
5. Được
6. Nếu thân vòng lặp có từ 2 câu lệnh trở lên ta phải làm gì?
7. Viết lệnh bình thường
8. Viết xuống dòng
9. Dùng cặp ngoặc { } để bao các câu lệnh lại
10. Tất cả đều sai

5. Các loại vòng lặp có thể dùng lồng nhau được không?a. Được b. Không

1. T04\_ChuongTrinhCon\_PTPhi-V2.ppt

* Kiến thức cần nhớ:
* Đây là hàm chính (main). Hàm main là hàm luôn được gọi trước tiên khi chương trình được thực thi. Từ hàm main này những hàm khác có thể được gọi theo kiến trúc phân cấp hình cây.
* Truyền giá trị:Là phương pháp truyền tham số mà sau đó hàm được truyền có được một phiên bản được lưu trữ riêng biệt giá trị của các tham số đó.Khi truyền giá trị, thì giá trị gốc (được truyền) sẽ không bị thay đổi cho dù hàm được truyền có thay đổi các giá trị này đi nữa.Truyền bằng con trỏ/tham chiếu/tham biến:Là phương pháp truyền tham số mà nó cho phép hàm được truyền tham khảo đến vùng nhớ lưu trữ giá trị gốc của tham số.Nếu ta truyền bằng con trỏ thì giá trị gốc của tham số có thể được thay đổi bởi các mã lệnh bên trong hàm.
* Notes:





* Test:

1. Quiz test

const int LIMIT = 50; // Line 1 LIMIT: biến toàn cục (global variable) int AddEm(int x, int y); // Line 2 int main() { // Line 3 int x = 42, // Line 4 x,y: biến cục bộ (local variable) y = 35; // Line 5 int Sum; // Line 6 Sum = AddEm(x, y); // Line 7 return 0; // Line 8 } // Line 9int AddEm(int x, int y) { // Line 10 int Total; // Line 11 Total: biến cục bộ Total = x + y; // Line 12 if (Total > LIMIT) // Line 13 Total = 0; // Line 14 return (Total); // Line 15} // Line 16

* 1. Phạm vi của biến Sum được khai báo ở dòng 6:a) Dòng 1 đến dòng 16 b) 6 đến 16c) 6 d) 6 đến 7e) 6 đến 9 (đ) f) Không phải các trường hợp trên.
  2. Phạm vi của biến x được khai báo ở dòng 10:a) Dòng 1 đến dòng 16 b) 4 đến 16c) 10 d)10 đến 12e) 10 đến 16 (đ) f) Không phải các trường hợp trên.
  3. Phạm vi của biến LIMIT được khai báo ở dòng 1:a) Dòng 1 đến dòng 16 (đ) b) 1 đến 3c) 1 d) 10 đến 13e) 10 đến 16 f) Không phải các trường hợp trên.
  4. Câu nào sau đây là đúng:a) LIMIT là cục bộ trong hàm main() (s) b) Total là cục bộ đối với hàm AddEm() (đ)c) Sum là cục bộ trong hàm main() (đ) d) LIMIT là toàn cục (đ)e) x là toàn cục (s) f) Tất cả các câu trên đều đúng. (s)g) Tất cả các câu trên đều đúng trừ câu a. (s) h) Chỉ câu b và c là đúng (s)i) Chỉ câu b, c và d là đúng (đ)j) Không có câu nào ở trên là đúng. (s)

1. **Tham số hình thức** được liệt kê trong \_\_\_ của hàm và **tham số thực tế** được liệt kê trong \_\_\_\_ của hàm.a) Lời gọi, cài đặt b) Cài đặt, lời gọi (đ)c) Tiêu đề, thân d) Thân, tiêu đềe) Không phải các câu trên.
2. Một tham số có kiểu đơn giản (ví dụ như int hoặc double) nên được truyền giá trị nếu luồng dữ liệu của tham số đó là:a) Một chiều, đi vào trong hàm (đ) b) Một chiều, đi ra khỏi hàmc) Hai chiều, vào và ra khỏi hàm d) Không phải các câu trên.
3. Câu khẳng định nào sau đây là đúng khi tham số được **truyền theo giá trị**:a) Tham số thật (thực tế) không bao giờ được thay đổi bởi sự thực thi của hàm. (đ) b) Tham số hình thức không bao giờ được(bị) thay đổi bởi sự thực thi của hàm. (s)c) Tham số thật (thực tế) phải là một biến. (s) d) Câu a, b và c đúng. (s)e) Câu a, b và c sai. (s)f) Chỉ có câu a và b là đúng. (s)g) Chỉ có câu a và c là đúng. (s)h) Chỉ có câu b và c là đúng. (s)i) Không có câu nào là đúng. (s)
4. Câu khẳng định nào sau đây là đúng khi tham số được **truyền theo con trỏ**:a) Tham số thật có thể được thay đổi bởi sự thực thi của hàm. (đ) b) Tham số hình thức có thể được thay đổi bởi sự thực thi của hàm. (đ)c) Tham số thật không thể là một biến. (s) d) Câu a, b và c đúng. (s)e) Câu a, b và c sai.(s)f) Chỉ có câu a và b là đúng. (đ)g) Chỉ có câu a và c là đúng. (s)h) Chỉ có câu b và c là đúng. (s)i) Không có câu nào là đúng.(s)
5. Nếu dấu chỉ con trỏ ‘\*’ không được đặt kèm với kiểu của tham số hình thức thì tham số thật tương ứng có thể là :a) Một hằng số. (đ) b) Một tên biến (đ)c) Một biểu thức bất kỳ nào (đ)d) Câu a, b và c đúng. (đ)e) Chỉ có câu a và b là đúng. (s)f) Chỉ có câu a và c là đúng. (s)g) Chỉ có câu b và c là đúng. (s)h) Không có câu nào là đúng. (s)
6. Cho một hàm được định nghĩa như sau:

void func(float \*gamma){ \*gamma = 5.5;}

Câu nào sau đây mô tả chính xác nhất luồng dữ liệu của tham số gamma: a) Một chiều, đi vào hàm. b) Một chiều, đi ra khỏi hàm. (đ)c) Hai chiều, đi vào và ra khỏi hàm.d) Không câu nào đúng.

1. Cho một hàm được định nghĩa như sau:

void demo( int intVal, double \*doubleVal ) { intVal = intVal \* 2; \*doubleVal = (double)intVal + 3.5;}

Đoạn mã lệnh sau đây sẽ in ra những gì?

int myInt = 20;double myDble = 4.8;demo(myInt, &myDble);printf("myInt=%d; myDble=%f", myInt, myDble);

1. myInt=20; myDble=43.5
2. myInt=40; myDble=4.8
3. myInt=20; myDble=4.8
4. myInt=40; myDble=43.5
5. myInt=20; myDble=23.5
6. myInt=40; myDble=23.5
7. Cho một hàm được định nghĩa như sau:

int Power(int\* Base, int\* Exponent ) { int Product = 1; while (\*Exponent >= 1) { Product = Product \* (\*Base); \*Exponent=\*Exponent-1; } return Product;}

Đoạn mã lệnh sau đây sẽ in ra những gì?

int N = 2;int Pow = 3;int Result = Power(&N, &Pow);printf("%d ^ %d=%d", N, Pow, Result);

1. 2 ^ 3 = 8
2. 2 ^ 0 = 8
3. 0 ^ 0 = 0
4. 2 ^ 3 = 1
5. Không có câu nào đúng
6. T05\_DeQuy.ppt

* Kiến thức cần nhớ:
* Nếu sử dụng hàm đệ quy thì thông thường thời gian ta bỏ ra để giải quyết một bài toàn là ngắn hơn so với việc dùng các vòng lặp đơn thuần. Tuy nhiên thời gian chạy là không ngắn hơn, thậm chí là dài hơn nhiều!
* Hàm đệ quy là hàm mà nó gọi chính nó hoặc ở trong một chu trình xoay vòng các lời gọi hàm.Nếu một hàm gọi chính nó, ta gọi là hàm đệ quy tức thì (immediate recursion).
* Test:

1. Hai phần của hàm đệ quy tức thì là:
2. Nếu vấn đề là đơn giản thì giải ra ngay, và (2) nếu vấn đề không thể được giải ra ngay thì chia nó ra thành các vấn đề nhỏ hơn để giải.
3. Chia vấn đề thành các vấn đề nhỏ hơn, và (2) đưa ra lời giải trực tiếp cho các vấn đề khó hơn.
4. Bỏ qua các trường hợp khó, và (2) giải các trường hợp dễ.
5. Giải vấn đề bằng cách yêu cầu nó giải chính nó, và (2) giải các trường hợp đơn giản trong một bước.
6. Làm sao để Sherlock Holmes điều tra một vụ sát nhân?
7. Bắt ngay người quản gia.
8. Tra vấn một nhân chứng, sau đó (2) tra vấn nạn nhân.
9. Loại bỏ một nhân chứng, và (2) loại bỏ một nhân chứng khác.
10. Khi chỉ còn một nghi can thì đó chính là tên sát nhân, (2) Xem xét các chứng cứ để loại trừ một nghi can, sau đó loại trừ các nghi can còn lại.
11. Việc chạy hàm đệ quy tốn rất nhiều bộ nhớ vì:
12. Các hàm đệ quy có xu hướng khai báo rất nhiều biến.
13. Các lời gọi hàm trước đó vẫn còn mở khi một hàm gọi chính nó và thông tin về các lời gọi hàm trước đó vẫn còn được lưu trong stack của bộ nhớ.
14. Có quá nhiều phiên bản của mã lệnh của hàm được tạo ra.
15. Hàm đệ quy cần giá trị cực lớn cho dữ liệu của nó.
16. Giá trị mà hàm mystery sau đây trả về sẽ là gì khi ta gọi nó với tham số có giá trị 4:

int mystery(int number){ if ( number <= 1 ) return 1; else return number \* mystery(number - 1);}

1. 0
2. 1
3. 4
4. 24
5. Hàm sum sau đây có thực sự trả về giá trị là tổng các số từ a đến b hay không, giả sử là a < b:

int sum(int a, int b){ if(a==b) return b; else return a + sum(a + 1, b);}

1. Có
2. Không
3. Hãy dự đoán kết quả của chương trình sau:

#include <stdio.h> int fun(int n) { if (n == 4) return n; else return 2\*fun(n+1); } int main() { printf("%d ", fun(2)); return 0; }

1. 2
2. 4
3. 16 (2\*2\*4)
4. Có lỗi
5. Hàm fun(4,3) sau đây trả về kết quả gì?

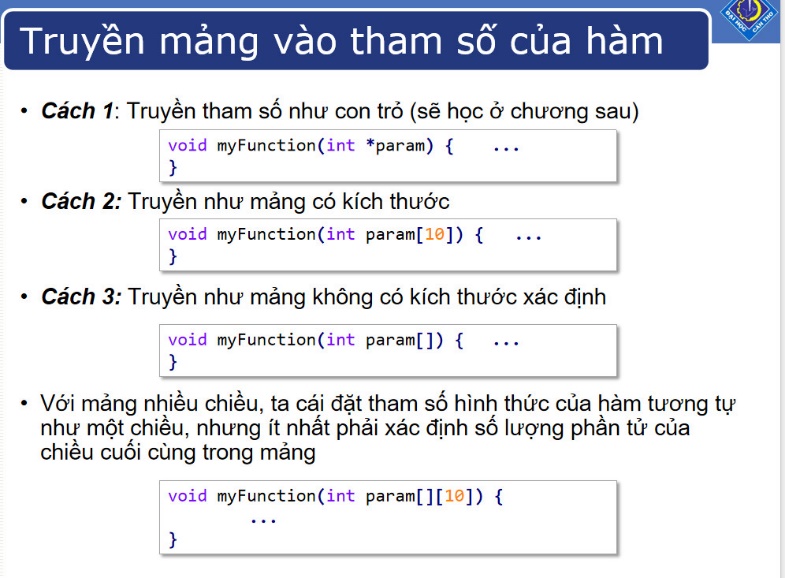
int fun(int x, int y) {  if (x == 0)    return y;  return fun(x - 1,  x + y);}

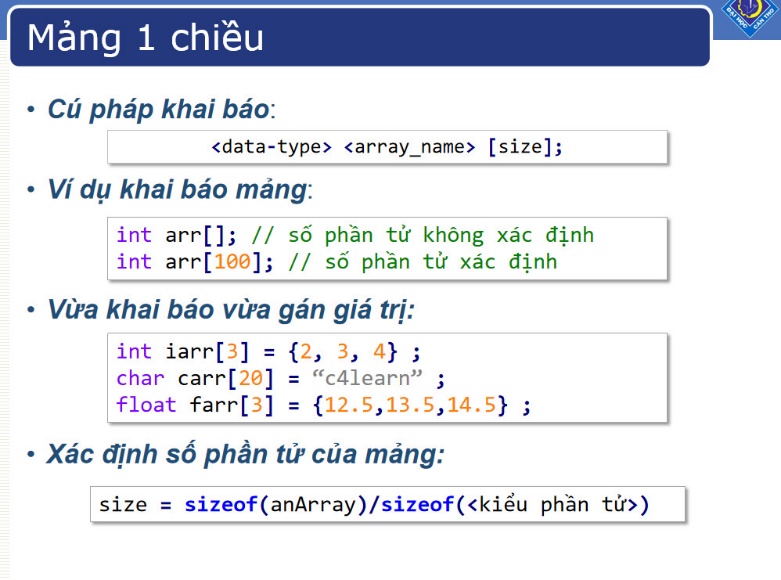
1. 13
2. 12
3. 9
4. 10
5. Hàm fun sau đây muốn tính gì?

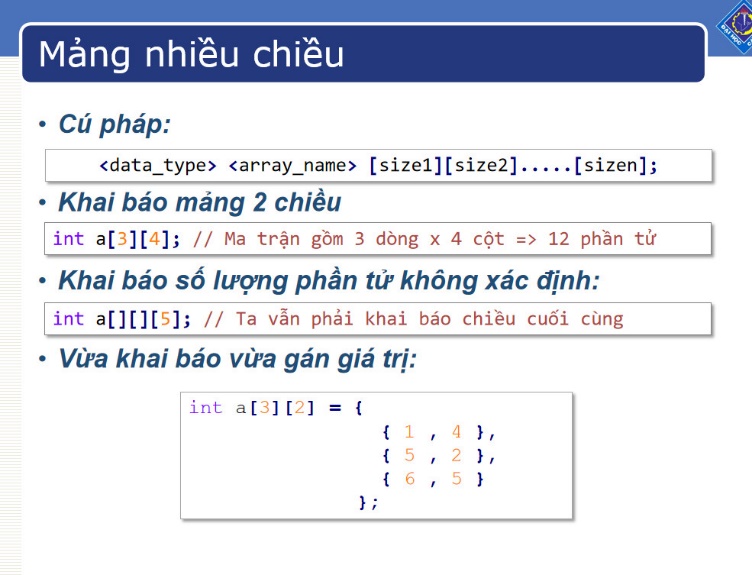
int fun(int x, int y){    if (y == 0)   return 0;    return (x + fun(x, y-1));}

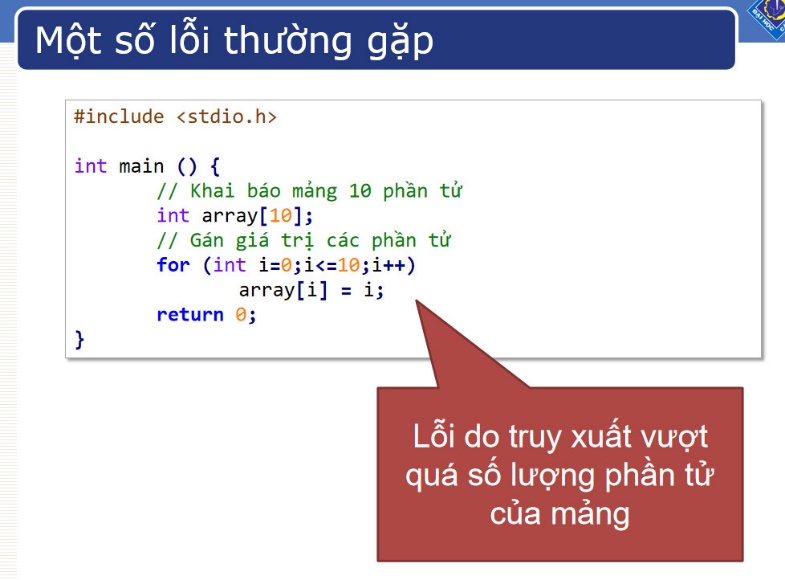
1. x+y
2. x+x\*y
3. x\*y
4. x^y
5. T06\_Mang\_HQThai\_v2\_03022015.ppt

* Kiến thức cần nhớ:
* Mảng là tập hợp các phần tử có cùng một kiểu dữ liệu.Các phần tử trong mảng được lưu trữ nối tiếp nhau trong bộ nhớCác phần tử trong mảng có thể được truy cập thông qua các chỉ số
* Mảng đơn hay mảng một chiều là một mảng tuyến tính.Chỉ có một chỉ số để biểu diễn vị trí phần tử, mỗi phần tử trong mảng một chiều không phải là một mảng khác.
* Mảng nhiều chiều có nhiều hơn một chỉ số để biểu diễn một phần tử trong mảng.Mảng nhiều chiều cũng được gọi là ma trận
* Note:









* Test:

1. Điều gì sẽ xảy ra khi ta gán một phần tử vượt quá chỉ số của mảng? //
2. Phần tử sẽ được gán ở chỉ số 0
3. Trình biên dịch sẽ báo lỗi
4. Chương trình có thể bị treo nếu một số dữ liệu bị ghi đè
5. Kích thước mảng sẽ tự tăng theo
6. Nếu bạn truyền mảng như một tham số của một hàm, bạn đang truyền
7. Giá trị của các phần tử trong mảng
8. Phần tử đầu tiên trong mảng
9. Địa chỉ cơ sở của mảng
10. Địa chỉ của phần tử cuối cùng trong mảng
11. Cho chương trình sau:

#include<stdio.h>int main() { int a[5] = {5, 1, 15, 20, 25}; int i, j, m; i = ++a[1]; j = a[1]++; m = a[i++]; printf("%d, %d, %d", i, j, m); return 0;}

a. 2, 1, 15 b. 1, 2, 5c. 3, 2, 15 d. 2, 3, 20

1. Cho chương trình sau, cho biết nội dung hiển thị:

#include<stdio.h>int main() { int arr[5], i=0; while(i<5) arr[i]=++i; for(i=0; i<5; i++) printf("%d, ", arr[i]); return 0;}

a. 1, 2, 3, 4, 5, b. Giá trị rác, 1, 2, 3, 4,c. 0, 1, 2, 3, 4, d. 2, 3, 4, 5, 6,

1. Cho chương trình sau, cho biết nội dung hiển thị:

#include<stdio.h>int main() { int arr[1]={10}; printf("%d\n", 0[arr]); return 0;}

a. 1 b. 10c. 0 d. 6

1. Cho chương trình sau, cho biết nội dung hiển thị:

#include<stdio.h>int main() { float arr[] = {12.4, 2.3, 4.5, 6.7};

printf("%d\n", sizeof(arr)/sizeof(arr[0])); return 0;}

a. 5 b. 4c. 6 d. 7

1. Nếu mảng bắt đầu bằng địa chỉ 1200 trong bộ nhớ, phần nội dung xuất ra của chương trình này là gì

#include<stdio.h>int main() { int arr[]={2, 3, 4, 1, 6}; printf("%d, %d, %d\n", arr, &arr[0], &arr); return 0;}

a. 1200, 1202, 1204 b. 1200, 1200, 1200c. 1200, 1204, 1208 d. 1200, 1202, 1200

1. Cho chương trình sau, cách cài đặt hàm nào bên dưới là đúng?

#include<stdio.h>int main() { int a[3][4]; fun(a); return 0;}

1. void fun(int p[][4]){ ... }
2. void fun(int \*p[4]){ ... }
3. void fun(int \*p[][4]){ ... }
4. void fun(int \*p[3][4]){ ... }
5. T07\_Chuoi\_HQThai\_V2\_08022015.ppt

* Kiến thức cần nhớ:
* Trong C không có một kiểu chuỗi ký tự riêng mà đó là một mảng ký tự được kết thúc bằng ký tự ‘\0’.Các thao tác trên chuỗi ký tự giống như trên mảngTa cũng có thể khai báo chuỗi ký tự bằng con trỏ, nhưng sẽ đề cập ở những chương sau.
* Test:

1. Với hai chuỗi ký tự **giống nhau**, hàm strcmp() trả lại kết quả gì?
2. -1
3. 1
4. 0
5. Yes
6. Hàm nào trong các hàm sau đây **tìm một ký tự** trong chuỗi và **trả lại vị trí** của kết quả sau cùng tìm kiếm được?
7. strchr()
8. strrchr()
9. strstr()
10. strnset()
11. Cho biết kết quả chương trình sau:

#include<stdio.h>

#include<string.h> int main() { char str1[20] = "Hello", str2[20] = " World"; printf("%s\n", strcpy(str2, strcat(str1, str2))); return 0;}

1. Hello
2. World
3. Hello World
4. WorldHello
5. Cho biết kết quả chương trình sau:

#include<stdio.h>int main() { printf(5+"Good Morning\n"); return 0;}

1. Good Morning
2. Good
3. M
4. Morning
5. Cho biết kết quả chương trình sau:

#include <stdio.h>int main() { static char s[25] = "The cocaine man"; int i=0; char ch; ch = s[++i]; printf("%c", ch); ch = s[i++]; printf("%c", ch); ch = i++[s]; printf("%c", ch); ch = ++(i[s]); printf("%c", ch); return 0;}

1. hhe!
2. de c
3. The c
4. Hhec
5. Cho biết kết quả chương trình sau:

#include<stdio.h>int main() { char str1[] = "Hello"; char str2[10]; char \*t, \*s; s = str1; t = str2; while(\*t++=\*s++); printf("%s\n", str2); return 0;}

1. Hello //(gán \*t=\*s trước, rồi cộng vào giá trị của t và s cho đến khi str1[] gặp “\0”)
2. HelloHello
3. Không in ra gì cả
4. Ello
5. Cho biết kết quả chương trình sau, biết rằng địa chỉ bộ nhớ của chuỗi “Hello1” là 1022

#include<stdio.h>int main() { printf("%u %s\n", &"Hello1", &"Hello2"); return 0;}

1. 1022 Hello2
2. Hello1 1022
3. Hello1 Hello2
4. 1022 1022
5. Xảy ra lỗi
6. T10\_CauTruc\_TNMThai.ppt

* Kiến thức cần nhớ:
* Kiểu cấu trúc là kiểu do người dung định nghĩa.Biến cấu trúc là biến của kiểu cấu trúc.Trong C chuẩn, có thể bỏ từ khoá struct khi khai báo biến (hoặc sử dụng typedef).Nhập các biến kiểu số thực trong cấu trúc phải nhập thông qua một biến trung gian.
* Test:

1. Câu nào sau đây dùng để **truy cập** một biến trong cấu trúc **b**?  
   A. b->var;  
   B. b.var;  
   C. b-var;  
   D. b>var;
2. Câu nào sau đây dùng để **truy cập** một biến trong cấu trúc **\*b**?  
   A. b->var;  
   B. b.var;  
   C. b-var;  
   D. b>var;
3. Câu nào sau đây **đúng**?  
   A. struct {int a;}  
   B. struct a\_struct {int a;}  
   C. struct a\_struct int a;  
   D. struct a\_struct {int a;};
4. Câu nào sau đây dùng để **khai báo biến** kiểu cấu trúc **của cấu trúc foo**?  
   A. struct foo;  
   B. struct foo var;  
   C. foo;  
   D. int foo;
5. T11\_Contro\_TCAN.ppt

* Kiến thức cần nhớ:
* Con trỏ kiểu void: Là con trỏ chưa có kiểu dữ liệu xác định.Có thể cho trỏ tới bất kỳ biến hay vùng nhớ có dữ liệu kiểu gì.Khi truy xuất vùng nhớ đang trỏ tới bởi con trỏ, cần phải ép kiểu một cách tường minh.
* Vùng nhớ stack: cấp phát cho các biến cục bộ,từ địa chỉ cao xuống thấp.
* Vùng nhớ heap: dùng để cấp phát “động” (giới thiệu sau),từ địa chỉ thấp lên cao.Còn một số phân đoạn khác (biến toàn cục,...)Lưu ý: các vùng nhớ này có thể không liên tục (vùng nhớ heap có thể “nằm trên” vùng nhớ stack).
* Chú ý:Vùng nhớ cấp phát động sẽ được cấp phát trong vùng nhớ heap (từ thấp đến cao).Hàm malloc() thường được sử dụng kèm với toán tử sizeof(), dùng để lấy kích thước của một kiểu dữ liệu hoặc biến.Các vùng nhớ cấp phát động không được tự động giải phóng khi chương trình kết thúc  
   phải giải phóng tường minh bằng hàm free() để tránh rò rỉ bộ nhớ.Ứng dụng: tạo các “mảng động”, chuỗi có kích thước “động”,...
* Ứng dụng của con trỏ:

Cấp phát động bộ nhớ:Mảng động: mảng có kích thước có thể thay đổiChuỗi ký tự “động” (phần sau)Truyền tham số cho hàm: cho phép thay đổi giá trị của tham số thực tếCon trỏ trỏ đến cấu trúcTrị trả về của hàm là con trỏ (phụ lục)

* Con trỏ nâng cao:

Con trỏ hàm (function pointer): Một con trỏ có thể trỏ đến một hàm.Ta có thể gọi hàm thông qua con trỏ.Thường được sử dụng trong hàm callback.Con trỏ của con trỏ:Một con trỏ có thể trỏ đến con trỏ khác. int \*\*p; //p là con trỏ có thể trỏ đến các con trỏ int \*pThường được sử dụng để tạo mảng động nhiều chiều, truyền đối số cho hàm bằng con trỏ với đối số là một con trỏ,...

* Con trỏ là 1 biến đặc biệt, dùng để chứa địa chỉ của một biến hay một vùng nhớ.Khai báo con trỏ: <datatype> \*<pointer name>;Hai phép toán cơ bản trên con trỏ là \* và &Con trỏ chứa đ/chỉ vùng nhớ (biến) nào, ta nói con trỏ đang trỏ (tham chiếu) tới vùng nhớ (biến) đó.Ta có thể truy xuất một vùng nhớ thông qua một con trỏ đang trỏ tới vùng nhớ đó.Con trỏ void\* có thể trỏ đến bất kỳ biến kiểu gì.
* Con trỏ có thể được sử dụng như mảng động bằng cách cấp phát vùng nhớ động.Vùng nhớ cấp phát động nằm trong vùng nhớ heap, sẽ không được giải phóng tự động.Mảng (thường) là một hằng con trỏ.Truyền tham số cho hàm bằng con trỏ cho phép hàm thay đổi giá trị tham số thực tế.Con trỏ phải được khởi tạo trước khi sử dụng.Vùng nhớ cấp phát động phải được giải phóng một cách tường minh dùng hàm free().
* Test:

1. Khai báo con trỏ nào sau đây là đúng?
2. int x;
3. int &x;
4. ptr x;
5. int \*x;
6. Lệnh nào sau đây dùng để lấy địa chỉ của biến a?
7. \*a;
8. a;
9. &a;
10. address(a);
11. Lệnh nào sau đây trả về **địa chỉ của biến** đang được **trỏ tới bởi con trỏ a**?
12. a;
13. \*a;
14. &a;
15. address(a);
16. Lệnh nào sau đây trả về **giá trị của biến** mà con trỏ a đang trỏ tới?
17. a;
18. val(a);
19. \*a;
20. &a;
21. Hàm/Lệnh nào dưới đây dùng để **cấp phát vùng nhớ động** trong C?
22. New
23. Malloc
24. Create
25. value
26. Lệnh nào sau đây dùng để **giải phóng một vùng nhớ** được cấp phát động?
27. Free
28. Delete
29. Clear
30. remove
31. **Hàm và con trỏ** trong C có thể được sử dụng **thay thế lẫn nhau**?
32. Đúng
33. Sai
34. Cho p là một con trỏ kiểu int và i là một biến kiểu int. Câu lệnh nào sau đây là đúng?
35. p = 0;
36. p = i;
37. p = p + 1;
38. p = &i;
39. Lệnh int \*x; và int\* x; cho kết quả giống nhau?
40. Đúng
41. Sai
42. **Rò rỉ bộ nhớ** xảy ra khi ta **cấp phát** vùng nhớ động bằng hàm malloc() và sau đó **giải phóng** bằng hàm free().
43. Đúng
44. Sai
45. Con trỏ có thể trỏ đến kiểu dữ liệu nào sau đây?
46. Int
47. Char
48. Structs
49. Tất cả các kiểu trên
50. Now that the pointer is declared, let’s set it to point to something: int a = 5;
51. \*fun\_int = &a;
52. \*fun\_int = \*a;
53. fun\_int = \*a;
54. fun\_int = &a;
55. Now let’s dereference the pointer:
56. \*fun\_int = 12;
57. .fun\_int = 12;
58. &fun\_int = 12;
59. Hãy cho biết kết quả thực thi các chương trình sau:
    1. Chương trình:

#include <stdio.h>void fun(int \*ptr) {    \*ptr = 30;}int main() {   int y = 20;   fun(&y);   printf("%d", y);    return 0;}

1. 20
2. 30
3. Compiler error
4. Runtime error
   1. Chương trình:

#include <stdio.h>void fun(int x) {    x = 30;}int main() {   int y = 20;   fun(y);   printf("%d", y);   return 0;}

1. 20
2. 30
3. Compiler error
4. Runtime error
5. Hãy cho biết kết quả thực thi của chương trình sau:

#include <stdio.h>int main() {   int \*ptr;   int x;    ptr = &x;   \*ptr = 0;   printf(" x = %d\n", x);   printf(" \*ptr = %d\n", \*ptr);    \*ptr += 5;   printf(" x  = %d\n", x);   printf(" \*ptr = %d\n", \*ptr);    (\*ptr)++;   printf(" x = %d\n", x);   printf(" \*ptr = %d\n", \*ptr);   return 0;}

1. x = 0  
   \*ptr = 0  
   x = 0  
   \*ptr = 0  
   x = 0  
   \*ptr = 0
2. x = garbage value  
   \*ptr = 0  
   x = garbage value  
   \*ptr = 5  
   x = garbage value  
   \*ptr = 6
3. x = 0  
   \*ptr = 0  
   x = 5  
   \*ptr = 5  
   x = 6  
   \*ptr = 6
4. x = 0  
   \*ptr = 0  
   x = 5  
   \*ptr = 5  
   x = garbage value  
   \*ptr = garbage value
5. T12\_Dynamic String\_TCAN.ppt

* Kiến thức cần nhớ:
* Phép gán chuỗi (sao chép giá trị): Sao chép giá trị cho chuỗi động:Sao chép cạn (shalow copy)Dùng phép gán = hai con trỏ (chuỗi)Cái được copy là địa chỉ (tham chiếu) của hai con trỏ hai con trỏ trỏ vào cùng một vùng nhớ hai con trỏ sẽ luôn có cùng giá trị (nếu không t/đổi địa chỉ trỏ tới)Sao chép sâu (deep copy)Dùng hàm strcpy() hoặc strdup()Cái được sao chép là dữ liệu hai con trỏ sẽ không còn quan hệ sau khi copy
* Sau khi đã cấp phát bộ nhớ và lưu trữ dữ liệu, giá trị của chuỗi có thể thay đổi ⇒ chiều dài có thể thay đổi.
* Chuỗi động: kích thước có thể thay đổi được.Chuỗi động về bản chất là một mảng động.Khai báo chuỗi động: char \*s;Khởi tạo: char \*s = "..."; sẽ cấp phát vùng nhớ chỉ đọc.Thay đổi chiều dài chuỗi động: cấp phát lại vùng nhớ và sao chép giá trịstrdup() = malloc() + strcpy()Cấp phát vùng nhớ: Dùng hàm malloc() (chưa có dữ liệu)Dùng hàm strdup() (cấp phát + gán giá trị)
* Gán/Sao chép chuỗi:Sao chép cạn: sao chép địa chỉ con trỏSao chép sâu: sao chép dữ liệu của chuỗiMảng các chuỗi động: mảng các con trỏMảng cố định: char \*dsTen[N];Mảng động: char \*\*dsTen;Một số vấn đề cần lưu ý:Phải cấp phát vùng nhớ cho chuỗi trước khi sử dụngPhải lưu ý vấn đề kích thước dữ liệu khi sử dụng hàm scanf() và gets()Phải lưu ý k/thước của chuỗi động trước khi gán giá trịPhải giải phóng bộ nhớ khi không còn s/dụng.
* Notes:

Lưu ý: phải cấp phát bộ nhớ cho ký tự k/thúc chuỗi.

* Test:

1. Ký tự nào sau đây là ký tự **kết thúc chuỗi**?
2. '.'
3. ' '
4. '\0'
5. '\n‘
6. Hàm nào dùng để so sánh hai chuỗi?
7. compare();
8. stringcompare();
9. cmp();
10. strcmp();
11. Khai báo chuỗi nào sau đây là **chưa chính xác**?
12. char \*s1;
13. char \*s1 = "Hi";
14. char s2[] = "Hi";
15. char s2[2] = "Hi"; // (còn ký tự “\0” nữa là 3 ký tự)
16. char s3[] = "Hi\0";
17. char s4[] = {'H', 'i', '\0'};
18. char s5[] = {'H', 'i'}; // (không có ký tự “\0” đánh dấu điểm kết thúc của chuỗi)
19. char s5[3] = {'H', 'i'}; // (không có ký tự “\0” đánh dấu điểm kết thúc của chuỗi)
20. char \*s6 = strdup("Hi");
21. char \*s7 = strcpy("Hi"); // (cần truyền vào 2 đối số)
22. Cho hai câu khai báo sau:char \*p = "Sanjay"; char a[] = "Sanjay"; hãy chọn câu phát biểu **sai**:
23. p là một non-const pointer trỏ tới một non-const string còn **a** là một const **pointer** trỏ tới một non-const string
24. Có thể cho con trỏ p trỏ tới một chuỗi khác, còn a thì không mà chỉ có thể thay đổi giá trị của a.
25. Trong cả hai trường hợp, ký tự ‘\0’ sẽ được thêm vào cuối chuỗi.
26. Chương trình hiển thị kết quả gì?

int main() { char \*str=NULL; strcpy(str,"Hello World"); printf("%s",str); }

1. Hello World
2. Hello World\0
3. NULLHello World
4. Không hiển thị gì hết (hoặc bị lỗi) //(khi khai báo \*str không được cấp phát vùng nhớ nên khi dung strcpy đang cố ghi đè lên một vùng nhớ không được cấp phát trước đó, dẫn đến hành vi không xác định)
5. Hãy cho biết kết quả của chương trình sau:

int main() { char \*x="HELLO"; x+=3; printf("%s",x); ;

1. HELLO
2. ELLO
3. LLO
4. LO
5. Hãy cho biết kết quả của chương trình sau:

void myStrcat(char \*a, char \*b) { int m = strlen(a); int n = strlen(b); int i; for (i = 0; i <= n; i++) a[m+i] = b[i]; }int main() { char \*str1 = "HELLO "; char \*str2 = "World"; myStrcat(str1, str2); printf("%s ", str1);}

1. HELLO world
2. HELLOworld
3. HwEoLrLlOd
4. Chương trình bị lỗi. Nếu chọn câu d) thì giải thích nguyên nhân.

* (không phải là “HELLOWorld” đâu!) vì biến str1 được khai báo là một con trỏ trỏ đến một chuỗi hằng (constant string literal) – và hằng này không thể thay đổi.  
  Khi bạn dùng hàm myStrcat() để thêm nội dung của str2 vào str1, tức là bạn đang cố gắng sửa một vùng nhớ không thể thay đổi, điều này dẫn đến lỗi.

1. Hãy cho biết kết quả của chương trình sau:

void swap(char \*str1, char \*str2) { char \*temp = str1; str1 = str2; str2 = temp; }int main() { char \*str1 = “Hello"; char \*str2 = “World"; swap(str1, str2); printf("str1 is %s, str2"  
" is %s", str1, str2);}

1. str1 is Hello, str2 is World // (chương trình chỉ thực hiện hoán đổi các giá trị con trỏ cục bộ trong hàm swap(), và điều này không ảnh hưởng đến con trỏ mà str1 và str2 trỏ tới trong hàm main()
2. str1 is World, str2 is Hello
3. str1 is World, str2 is World
4. str1 is Hello, str2 is Hello
5. Hãy cho biết kết quả của chương trình sau:

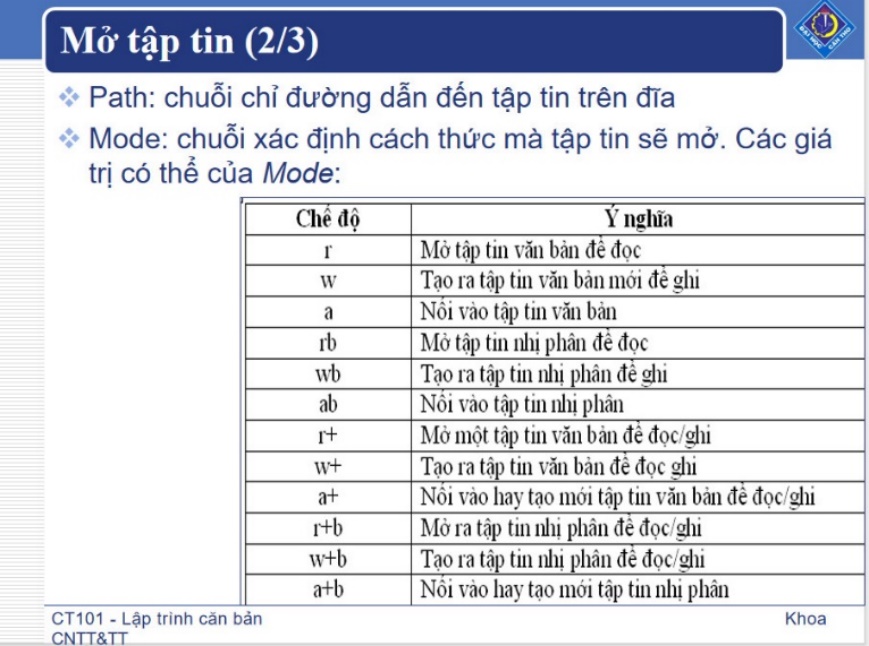
int main() { char str1[] = “Hello World"; char str2[] = {'H', 'e', 'l', 'l', 'o'}; int n1 = sizeof(str1)/sizeof(str1[0]); int n2 = sizeof(str2)/sizeof(str2[0]); printf("n1 = %d, n2 = %d", n1, n2); }

1. n1 = 10, n2 = 9
2. n1 = 10, n2 = 10
3. n1 = 9, n2 = 9
4. n1 = 9, n2 = 10
5. n1=12, n2=5
6. Hãy cho biết kết quả của chương trình sau:

int main() { char \*s= "hello"; char \*p = s; printf("%c\t%c", p[0], s[1]);}

1. Bị lỗi khi thực thi
2. h h
3. h e
4. h l
5. Câu lệnh nào sau đây là **sai**?
6. char str1[5] = "Texas"; // (length = 5 + “\0” = 6)
7. char str2[] = "A character string";
8. char str3[2] = "A";
9. char str4[2] = "TX"; // (length = 2 + “\0” = 3)
10. Cho một biến con trỏ char \*ptr\_ch, **những** câu lệnh nào sau đây là **đúng**?
11. \*ptr\_ch = 'a'; // (gán giá trị ‘a’ vào vùng nhớ mà con trỏ ptr\_ch trỏ tới)
12. ptr\_ch = "A character string"; // (gán địa chỉ của chuỗi cho con trỏ ptr\_ch)
13. ptr\_ch = 'x';
14. \*ptr\_ch = "This is Quiz 2.";
15. T12\_TapTin\_TNMThai.ppt

* Kiến thức cần nhớ:
* Trong C, tất cả hoạt động nhập xuất được thực hiện với các dòng (streams).Thiết bị nhập chuẩn là bàn phím.Thiết bị xuất chuẩn là màn hình.Thiết bị vừa nhập vừa xuất: tập tin
* Stream là chuỗi byte dữ liệu  
  • “Chảy” vào chương trình gọi là stream nhập (input stream).  
  • “Chảy” ra chương trình gọi là stream xuất (output stream)
* Phân loại Stream văn bản (text)  
  • Chỉ chứa các ký tự.  
  • Tổ chức thành từng dòng, mỗi dòng tối đa 255 ký tự, kết thúc bởi ký tự cuối dòng ‘\0’ hoặc ký tự sang dòng mới ‘\n’. Stream nhị phân (binary)  
  • Chứa các byte.  
  • Được đọc và ghi chính xác từng byte.  
  • Xử lý dữ liệu bất kỳ, kể cả dữ liệu văn bản.  
  • Được sử dụng chủ yếu với các tập tin trên đĩa.
* Khái niệm Tập hợp thông tin (dữ liệu) được tổ chức theo một dạng nào đó với một tên xác định. Một dãy byte liên tục (ở góc độ lưu trữ).Được lưu trữ trong các thiết bị lưu trữ ngoài như đĩa mềm, đĩa cứng, USB…  
  • Vẫn tồn tại khi chương trình kết thúc.  
  • Kích thước không hạn chế (tùy vào thiết bị lưu trữ)Cho phép đọc dữ liệu (thiết bị nhập) và ghi dữ liệu (thiết bị xuất).
* Phân loại Theo người sử dụng: quan tâm đến nội dung tập tin nên sẽ phân loại theo phần mở rộng  
  .EXE, .COM, .CPP, .DOC, .PPT, …Theo người lập trình: tự tạo các stream tường minh để kết nối với tập tin xác định nên sẽ phân loại theo cách sử dụng stream trong Ctập tin kiểu văn bản (ứng với stream văn bản) và tập tin kiểu nhị phân (ứng với stream nhị phân).
* Tập tin kiểu văn bản (stream văn bản) Dãy các dòng kế tiếp nhau.Mỗi dòng dài tối đa 255 ký tự và kết thúc bằng ký hiệu cuối dòng (end\_of\_line). Dòng không phải là một chuỗi vì không được kết thúc bởi ký tự ‘\0’.Khi ghi ‘\n’ được chuyển thành cặp ký tự CR (về đầu dòng, mã ASCII 13) và LF (qua dòng, mã ASCII 10). Khi đọc thì cặp CR-LF được chuyển thành ‘\n’.
* Tập tin kiểu nhị phân (stream nhị phân) Dữ liệu được đọc và ghi một cách chính xác, không có sự chuyển đổi nào cả. Ký tự kết thúc chuỗi ‘\0’ và end\_of\_line không có ý nghĩa là cuối chuỗi và cuối dòng mà được xử lý như mọi ký tự khác.
* Biến tập tinĐược dùng để đại diện cho một tập tinCác thao tác lên tập tin sẽ được thực hiện thông qua biến này
* Con trỏ tập tinTại mỗi thời điểm, sẽ có một vị trí của tập tin mà tại đó việc đọc/ghi thông tin sẽ xảy raTa hình dung có 1 con trỏ đang chỉ đến vị trí đóSau khi đọc/ghi xong dữ liệu, con trỏ sẽ chuyển dịch thêm một phần tử về phía cuối tập tin.Sau phần tử dữ liệu cuối cùng của tập tin là dấu kết thúc tập tin EOF
* Tập tin kiểu văn bản:Mỗi dòng dài tối đa 255 ký tự và kết thúc bằng ký hiệu cuối dòng (end\_of\_line). Dòng không phải là một chuỗi.Khi ghi ‘\n’ được chuyển thành cặp ký tự CR và LF.Khi đọc thì cặp CR-LF được chuyển thành ‘\n’.  
  Tập tin kiểu nhị phân:Dữ liệu được đọc và ghi một cách chính xác.Ký tự kết thúc chuỗi ‘\0’ và end\_of\_line không có ý nghĩa.
* Notes:



* Test:

1. Đối tượng nào **đại diện** cho 1 tập tin trong C?  
   A. FILE\*  
   B. fopen  
   C. printf  
   D. fprintf
2. Trước khi đọc hoặc ghi vào tập tin, bạn cần làm gì?  
   A. Gọi fopen  
   B. Tạo tập tin  
   C. Gọi fclose  
   D. Dùng fprintf
3. Làm sao để **ghi 1 chuỗi** vào tập tin văn bản? //  
   A. Mở tập tin và dùng fprintf.  
   B. Mở tập tin và dùng fprintf, kết quả sẽ ghi vào tập tin thay vì hiển thị lên màn hình.  
   C. Mở tập tin and dùng fputc nhiều lần.  
   D. Dùng fread để đọc dữ liệu đưa vào tập tin.
4. Chế độ nào dùng để mở và **ghi tiếp** vào tập tin thay vì tạo tập tin mới?  
   A. a  
   B. r  
   C. w  
   D. W+
5. T15\_ThuatToan\_NVLinh.ppt

* Kiến thức cần nhớ:
* Test:

1. T16\_SearchSort\_NVLinh.ppt

* Kiến thức cần nhớ:
* Test: