Chào mừng các bạn trở lại với Series Lập Trình Căn Bản C. Hôm nay chúng ta sẽ học về Con Trỏ trong ngôn ngữ C.

Với những bài học trước, chúng ta đã tìm hiểu khá nhiều về ngôn ngữ C, trong đó chúng ta đã học về Biến.

Khi học về biến, mình đã hướng dẫn cho các bạn cách khai báo nó, gán và sử dụng nó.

Ví dụ:

#include "stdio.h"

int main(){

int a = 5;

printf("%d", a);

return 0;

}

Ở đây chúng ta đã khai báo 1 biến a = 5 và mình đã thành công với việc in nó ra.

**Chúng ta thành công khai báo nó và in nó ra, vậy còn bản chất của nó bên trong như thế nào nhỉ?**

Biến mà chúng ta đang thao tác ở đây là về mặt GIÁ TRỊ.

Chúng ta có một biến a, và giá trị của nó là 5. Muốn sử dụng nó thì ta chỉ việc in nó ra là xong.

Khi đi sâu hơn, chúng ta sẽ biết rằng nó quản lí 2 thứ:

* Giá trị
* Địa chỉ của ô nhớ.

Khi chúng ta tạo ra một biến a, thì nó sẽ có một giá trị nào đấy và được cấp phát một vùng nhớ ở đâu đó trong hệ điều hành, và để biết nó ở đâu, chúng ta sẽ sử dụng dấu **&**.

Ví dụ: Tạo một biến a, in ra giá trị và địa chỉ của nó.

|  |  |
| --- | --- |
| Code | Kết quả |
| #include "stdio.h"  int main(){  int a = 5;  printf("gia tri: %d\n", a);  printf("dia chi: %d", &a);  return 0;  } | gia tri: 5  dia chi: 2293324 |

Khi chạy đoạn code trên, giá trị các bạn sẽ giống mình, nhưng địa chỉ có thể khác, vì mỗi máy tính sẽ cấp phát một vùng nhớ riêng.

Giờ chúng ta đã biết cách lấy địa chỉ đó ra, nhưng đợi đã, có gì đó quen quen ở đây.

Các bạn nhớ lại khi chúng ta nhập dữ liệu, có phải nó như này

Scanf(“dac\_ta”, &ten\_bien);

Nhập biến nguyên kiểu a 🡺 scanf(“%d”, &a);

Chữ & đó là chúng ta đang lấy địa chỉ của ô nhớ để gán giá trị nó vào đấy các bạn ạ. Giờ chúng ta sẽ bắt đầu chính thức tìm hiểu con trỏ nhé.

1. Khai báo.

Khi khai báo một con trỏ, bạn sẽ thêm dấu \* ở trước tên biến, việc này tương tự với xuất kết quả.

Ví dụ:

int \*a, \*b; *// con trỏ a, b*

int a, \*b; *// con trỏ b, biến nguyên a*

Và để gán nó cho một biến bình thường, ta sử dụng dấu &.

Ví dụ:

#include "stdio.h"

int main(){

int a = 5;

int \*p;

p = &a;

printf("a = %d\n", a);

printf("\*p = %d", \*p);

return 0;

}

Giờ thay đổi một chút, ở hàm in chúng ta không sử dụng dấu \* thì sao nhỉ?

|  |  |
| --- | --- |
| Code | Kết quả |
| #include "stdio.h"  int main(){  int a = 5;  int \*p;  p = &a;  printf("a = %d\n", a);  printf("p = %d\n", p);  printf("p = %d", \*p);  return 0;  } | a = 5  p = 2293316  \*p = 5 |

Như nhận thấy thì kết quả của biến p bình thường mang địa chỉ của a, và \*p mang giá trị của a.

Lúc này:

* p sẽ tương đương với &a ( địa chỉ).
* \*p tương đương với a ( giá trị).

Các bạn lưu ý là con trỏ cũng có vùng nhớ riêng, để in địa chỉ của con trỏ thì các bạn sử dụng dấu &p.

1. Đặc điểm của con trỏ.

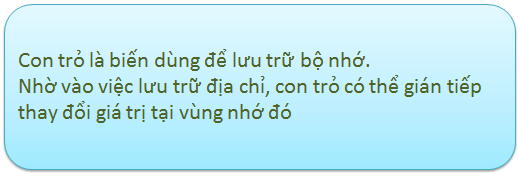
Con trỏ có một cái rất hay, đó là khi nó trỏ tới vùng nhớ nào thì nó có thể thay đổi giá trị ở vùng nhớ đó.

Ví dụ:

|  |  |
| --- | --- |
| Code | Kết quả |
| #include "stdio.h"  int main(){  int a = 5;  int \*p;  p = &a;  printf("a = %d\n", a);    \*p = 10;  printf("a = %d\n", a);  printf("\*p = %d", \*p);  return 0;  } | a = 5  a = 10  \*p = 10 |

Rõ ràng mình không làm gì với a, nhưng giá trị của nó đã thay đổi. Đây là một đặc điểm rất hay của con trỏ.

Và từ đây, chúng ta có thể rút ra một kết luận về con trỏ



1. Ứng dụng con trỏ

**Con trỏ với hàm**

Trước đây, khi thao tác với hàm, bạn muốn thay đổi giá trị của một thứ gì đó thì thường sẽ sử dụng **return**.

Nhưng có một điểm chết ở return là bạn sẽ chỉ thay đổi được giá trị của 1 biến, còn 2 biến trở lên thì không thể thay đổi.

Và khi học được con trỏ, bạn đã có thể làm được điều đó.

Thao tác với hàm bằng con trỏ như sau

Ví dụ 1: Nhập vào 2 biến, tăng giá trị của 2 biến lên 1.

|  |  |
| --- | --- |
| Code | Kết quả |
| #include "stdio.h"  void change(int \*pa, int \*pb){  \*pa = \*pa + 1;  \*pb = \*pb + 1;  }  int main(){  int a = 3, b = 2;  printf("before: a = %d, b = %d\n", a, b);  change(&a,&b);  printf("after: a = %d, b = %d", a, b);  return 0;  } | before: a = 3, b = 2  after: a = 4, b = 3 |

Kết quả đã có sự thay đổi, Các bạn hãy chú ý một chút về chương trình.

* Ở hàm con, thay vì sử dụng các biến bình thường, chúng ta thêm dấu \* trước tên biến.
* Ở hàm Main, chúng ta thêm dấu & trước tên biến.

Ví dụ 1 đã xong, giờ chúng ta qua ví dụ 2 nhé, nhập vào 2 biến, hoán đổi giá trị 2 biến đó lại.

|  |  |
| --- | --- |
| Code | Kết quả |
| #include "stdio.h"  void swap(int \*pa, int \*pb){  int tmp = \*pa;  \*pa = \*pb;  \*pb = tmp;  }  int main(){  int a = 5, b = 100;  printf("before: a = %d, b = %d\n", a, b);  swap(&a,&b);  printf("after: a = %d, b = %d", a, b);  return 0;  } | before: a = 5, b = 100  after: a = 100, b = 5 |

Đã có sự thay đổi, tương tự như vậy thì chúng ta có thể làm nhiều hơn với chương trình rồi.

**Con trỏ với Mảng/ Chuỗi**

1. **Mảng**

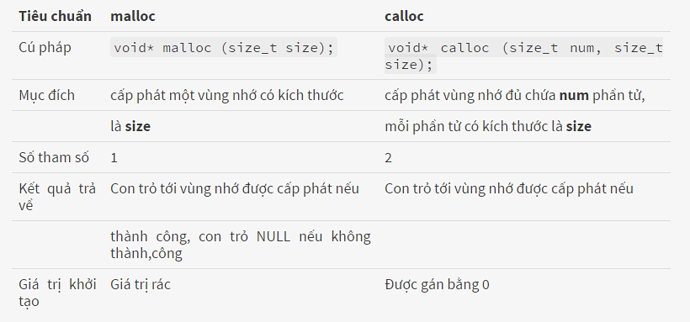
Với mảng / chuỗi, khi tạo một chương trình, chúng ta sẽ phải cung cấp cho nó một độ dài tối đa, và độ dài đó không được thay đổi. Đó người ta gọi là mảng Tĩnh.

Nhưng khi học về con trỏ, chúng ta có thể tạo ra một mảng / chuỗi với kết quả có thể thay đổi, người ta gọi là mảng Động.

Để sử dụng mảng động, chúng ta cần cấp phát bộ nhớ thông qua 3 câu lệnh: malloc, calloc, realloc.

Và chúng ta sẽ giải phóng nó thông qua câu lệnh free().

Ở đây mình sẽ hướng dẫn các bạn về malloc, calloc, còn về realloc các bạn có thể tự tìm hiểu thêm.



Ví dụ:

#include “stdio.h”

#include “stdlib.h”

int main()

{

int n;

printf("Nhap n: ");

scanf("%d", &n);

int \*px = (int \*)malloc(n \* sizeof(int)); // cấp phát bộ nhớ kích thước n

// nhập

for(int i = 0 ; i < n; i++){

printf("Nhap px[%d]: ", i);

scanf("%d", (px + i)); // cách truy xuất mới

}

// xuất

for(int i = 0 ; i < n; i++){

printf("%d ", px[i]);

}

free(px); // thu hồi vùng nhớ

return 0;

}

Như các bạn để ý thì khi chúng ta biết về con trỏ, việc nhập và xuất sẽ có thêm cách truy xuất mới, tương tự scanf ở vòng lặp for đầu tiên. Trong đó px là mảng, i là chỉ số.

Trước khi kết thúc bài học thì mình sẽ giải thích một chút về việc giải phóng mảng px trong đoạn chương trình trên.

Các bạn tưởng tượng rằng bạn là một người nông dân thời phong kiến, để có đất đi cày thì bạn phải đi thuê, sau khi thuê đất rồi thì bạn có thể canh tác, làm đất và trồng bất cứ thứ gì trên đó.

Sau khi trồng xong, bạn sẽ thu hoạch và trả đất đó lại nơi bạn đã thuê.

Ở đây, việc cấp phát vùng nhớ tương tự với việc bạn đi thuê đất.

Việc gán, thêm ký tự,… trong vùng nhớ đó tương tự với việc bạn trồng cây cối trên đó.

Và cuối cùng bạn cần phải trả nó lại nơi đã thuê, đấy chính là việc bạn giải phóng nó.

Và câu chuyện đó cũng chính là phần cuối cho bài giảng về con trỏ của chúng ta, ở phần sau tôi sẽ nói về Struct, và đó cũng là bài giảng cuối cùng của chúng ta.