

Con trỏ & Địa chỉ



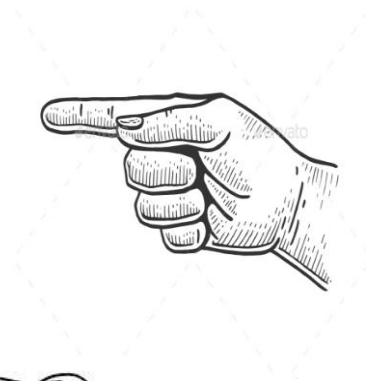
- Giải thích con trỏ là gì và nó được sử dụng ở đâu
- Gán giá trị cho con trỏ
- Biến con trỏ và toán tử con trỏ
- Số học & So sánh con trỏ
- Con trỏ và mảng

Phần 1

CON TRỎ LÀ GÌ?



Con trỏ



Con trỏ là gì?

- Con trỏ là một biến, chứa địa chỉ vùng nhớ của biến khác
- Nếu một biến chứa địa chỉ của biến khác thì biến đầu tiên biến được cho là trỏ đến biến thứ hai
- Một con trỏ cung cấp một phương pháp gián tiếp để truy cập giá trị của một mục dữ liệu
- Con trỏ có thể trỏ tới các biến thuộc các kiểu dữ liệu cơ bản khác như int, char hoặc double hoặc dữ liệu tổng hợp như mảng hoặc cấu trúc

Con trỏ dùng để làm gì?

Một số tình huống có thể sử dụng con trỏ là:

- Để trả về nhiều giá trị từ một hàm
- Để truyền mảng và chuỗi thuận tiện hơn từ một chức năng khác
- Để thao tác mảng dễ dàng bằng cách di chuyển con trỏ tới chúng thay vì tự di chuyển các mảng
- Để phân bổ bộ nhớ và truy cập nó (Phân bổ bộ nhớ trực tiếp)

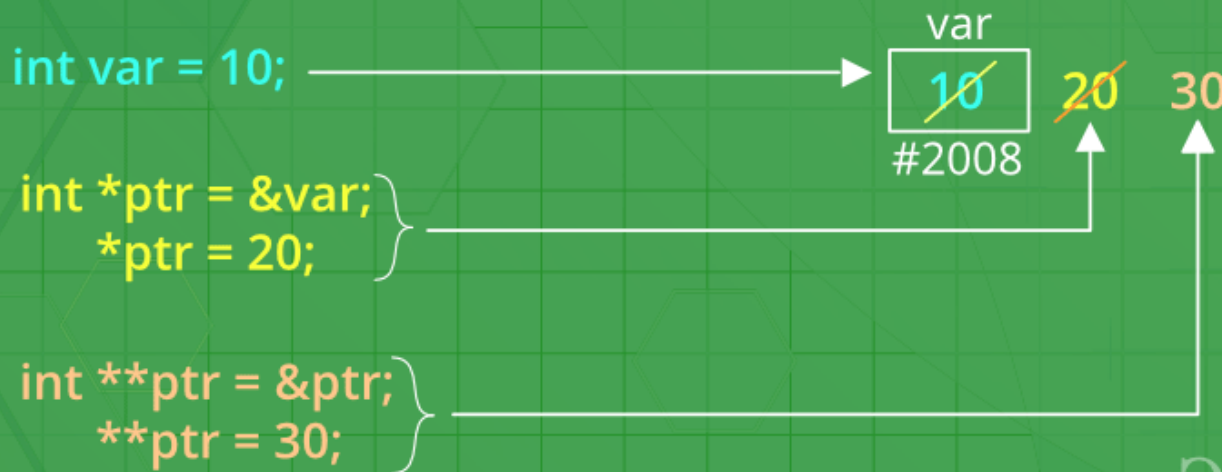
- Đây là cách chúng ta có thể khai báo con trỏ:

```
int* p;
```

- Chúng ta đã khai báo một con trỏ p kiểu int. Bạn cũng có thể khai báo con trỏ theo những cách sau:

```
int *p1;  
int * p2;
```

How pointer works in C



Phần 2

Gán giá trị cho con trỏ

- Các giá trị có thể được gán cho con trỏ thông qua nhà điều hành.

`ptr_var = &var;`

- Ở đây địa chỉ của var được lưu trong biến ptr_var
- Cũng có thể gán giá trị cho con trỏ thông qua một biến con trỏ trỏ đến một mục dữ liệu có cùng kiểu dữ liệu

`ptr_var = &var;`

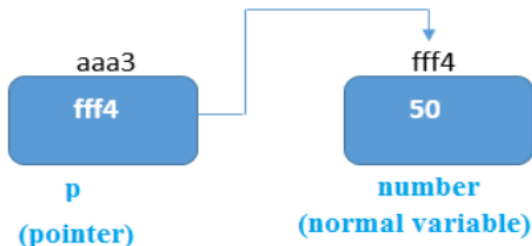
`ptr_var2 = ptr_var;`

- Các biến có thể được gán giá trị thông qua con trỏ cũng vậy

*** ptr_var = 10;**

- Khai báo trên sẽ gán 10 cho biến var nếu ptr_var trỏ tới var

- Một ví dụ về việc sử dụng con trỏ để in địa chỉ và giá trị được đưa ra dưới đây:



- Trong hình trên, biến con trỏ lưu trữ địa chỉ của biến số, nghĩa là: **fff4**. Giá trị của biến số là **50**. Nhưng địa chỉ của biến con trỏ là **aaa3**.
- Nhờ sự giúp đỡ của* (**toán tử gián tiếp**), chúng ta có thể in giá trị của con trỏ biến đối **P**.

Ví dụ về con trỏ - 1

- Một ví dụ về việc sử dụng con trỏ để in địa chỉ và giá trị được đưa ra dưới đây:

bao gồm <stdio.h>

int chủ yếu()

{

int số = 50;

int *P;

// lưu trữ địa chỉ của biến số p = &số;

// P chứa địa chỉ của số do đó in P đưa ra địa chỉ của số. printf("Địa chỉ của biến p là %x \n", P);

//* được sử dụng để hủy đăng ký một con trỏ do đó nếu in

*P, // giá trị được lưu trữ tại địa chỉ chứa bởi P. printf("Giá trị của biến p là %d \n", *P); trở lại 0;

}

Địa chỉ của biến số là fff4 Địa chỉ của biến p là fff4 Giá trị của biến p là 50

Phần 3

BIẾN CON TRỞ VÀ HOẠT ĐỘNG CON TRỞ

- Một khai báo con trỏ bao gồm một kiểu cơ sở và một tên biến đứng trước một ký tự*

✓ **Cú pháp khai báo chung là:**

tên loại;

✓ **Ví dụ:**

int *var2;

- Có 2 toán tử đặc biệt được sử dụng với con trỏ:

&Và*

- Toán tử & là toán tử một ngôi và nó trả về địa chỉ bộ nhớ của toán hạng

var2 = &var1;

- Toán tử thứ hai* là sự bổ sung của &. Nó là toán tử một ngôi và trả về giá trị chứa trong vị trí bộ nhớ được trỏ bởi giá trị của biến con trỏ

nhiet độ = *var2;

/* Kết quả của chương trình này có thể khác nhau trong các lần chạy khác nhau. Lưu ý rằng chương trình in địa chỉ của một biến và một biến có thể được gán địa chỉ khác trong các lần chạy khác nhau. */

```
# bao gồm <stdio.h>
```

```
int chủ yếu() {
```

```
    int x;
```

```
    // In địa chỉ của x
```

```
    printf("%p", &x);
```

```
    trở lại 0;
```

```
}
```

- Để truy cập địa chỉ của một biến vào một con trỏ, hãy sử dụng toán tử một ngôi **&** (**ký hiệu**) trả về địa chỉ của biến đó. Ví dụ **&x** cung cấp địa chỉ của biến **x**.

Một nhà điều hành nữa là **đơn nhất** *(Dấu hoa thị) được sử dụng cho hai mục đích: Để khai báo một con trỏ Biến đổi. Khi một biến con trỏ được khai báo trong C/C++, phải có dấu * trước tên của nó:

// Chương trình C minh họa việc khai báo các biến con trỏ.

bao gồm <stdio.h>

int chủ yếu() {

 int x = 10;

 // 1) Vì có * trong khai báo, ptr trở thành một biến con trỏ (một biến //
 lưu trữ địa chỉ của một biến khác)

 // 2) Vì có int trước * nên ptr là con trỏ tới một biến kiểu số nguyên int

 *ptr;

 // Toán tử & trước x được sử dụng để lấy địa chỉ của x. Địa chỉ của x được gán cho ptr. ptr
 = &x;

 trở lại 0;

}

Để truy cập giá trị được lưu trong địa chỉ, chúng ta sử dụng toán tử một ngôi (*) trả về giá trị của biến nằm tại địa chỉ được chỉ định bởi toán hạng của nó. Điều này cũng được gọi là **Hội thảo**.

// Chương trình C minh họa việc sử dụng * cho con trỏ trong C

bao gồm <stdio.h>

int chủ yếu() {

int Var = 10; // Một biến số nguyên bình thường

int *ptr = &Var; // Một biến con trỏ chứa địa chỉ của var.

// Dòng này in giá trị tại địa chỉ được lưu trong ptr. Giá trị được lưu trữ là giá trị của biến "var"

printf("Giá trị của Var = %d\n", *ptr);

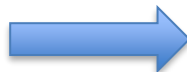
// Đầu ra của dòng này có thể khác nhau trong các lần chạy khác nhau ngay cả trên cùng một máy.

printf("Địa chỉ của Var = %p\n", ptr);

// Sử dụng ptr như một vế trái của bài tập

* ptr = 20; // Giá trị tại địa chỉ bây giờ là 20
printf("Sau khi thực hiện *ptr = 20, *ptr là %d\n", *ptr);
trở lại 0;

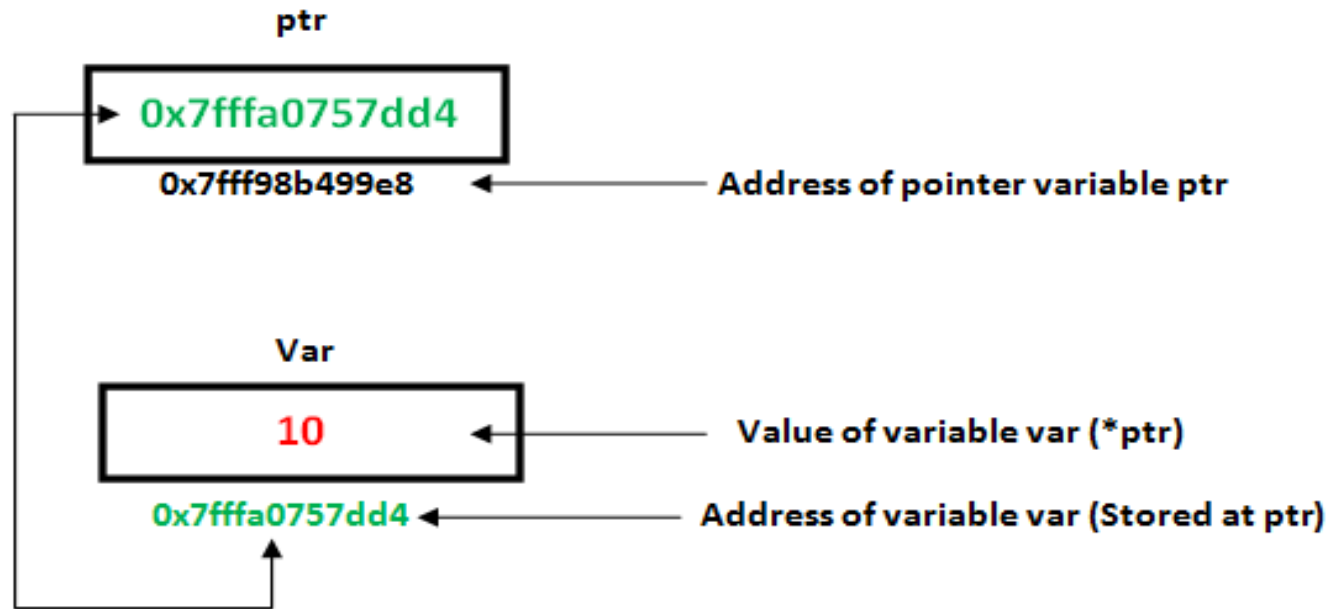
}



Giá trị của Var = 10

Địa chỉ của Var = 0x7ffa057dd4 Sau
khi thực hiện *ptr = 20, *ptr là 20

Dưới đây là hình ảnh minh họa của chương trình trên:



phần 4

TOÁN HỌC CON TRỎ VÀ SO SÁNH

- Phép cộng và phép trừ là những phép toán duy nhất có thể được thực hiện trên con trỏ

```
int var, *ptr_var;  
ptr_var = &var;  
var = 500;  
ptr_var++ ;
```

- Chúng ta hãy giả sử rằng **var** được lưu trữ tại địa chỉ **1000**
- Sau đó **ptr_var** có giá trị **1000** được lưu trữ trong đó. Vì số nguyên là 2 byte, sau biểu thức "**ptr_var++;**" **ptr_var** sẽ có giá trị như **1002** và không **1001**

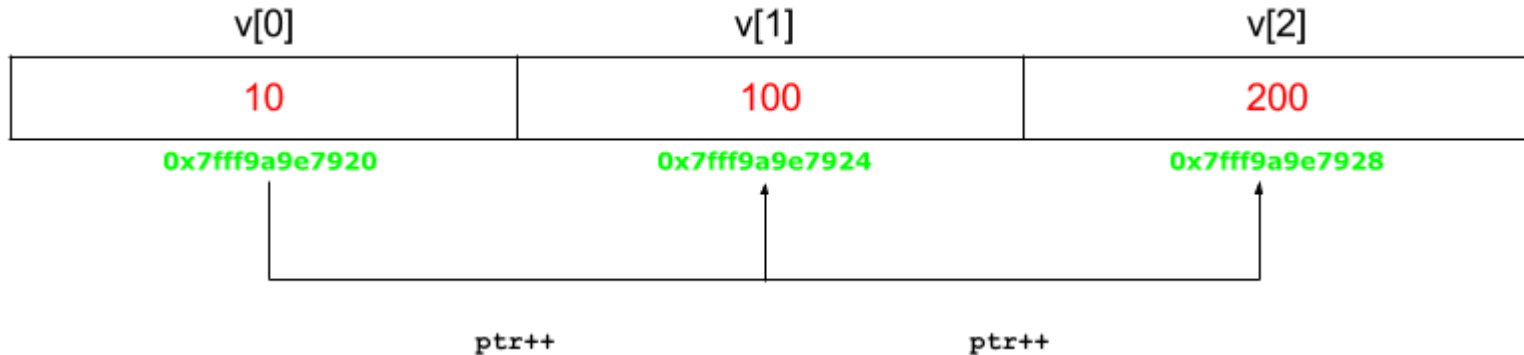
| | |
|--|--|
| <code>++ptr_var</code> or <code>ptr_var++</code> | points to next integer after var |
| <code>--ptr_var</code> or <code>ptr_var--</code> | points to integer previous to var |
| <code>ptr_var + i</code> | points to the <i>i</i> th integer after var |
| <code>ptr_var - i</code> | points to the <i>i</i> th integer before var |
| <code>++*ptr_var</code> or <code>(*ptr_var)++</code> | will increment var by 1 |
| <code>*ptr_var++</code> | will fetch the value of the next integer after var |

- Mỗi lần con trỏ tăng lên, nó sẽ trỏ đến vị trí bộ nhớ của con trỏ tiếp theo.

phần tử thuộc loại cơ sở của nó

- Mỗi lần giảm nó sẽ trỏ đến vị trí của phần tử trước đó
- Tất cả các con trỏ khác sẽ tăng hoặc giảm tùy theo độ dài của dữ liệu loại họ đang trỏ đến

Ví dụ về số học tăng dần:



Số học con trỏ - 4

```
// C++ program to illustrate Pointer Arithmetic
// in C/C++
#include <bits/stdc++.h>

// Driver program
int main()
{
    // Declare an array
    int v[3] = {10, 100, 200};

    // Declare pointer variable
    int *ptr;

    // Assign the address of v[0] to ptr
    ptr = v;

    for (int i = 0; i < 3; i++)
    {
        printf("Value of *ptr = %d\n", *ptr);
        printf("Value of ptr = %p\n\n", ptr);

        // Increment pointer ptr by 1
        ptr++;
    }
}
```



```
Output:Value of *ptr = 10
Value of ptr = 0x7ffcae30c710

Value of *ptr = 100
Value of ptr = 0x7ffcae30c714

Value of *ptr = 200
Value of ptr = 0x7ffcae30c718
```

- Hai con trỏ có thể được so sánh trong một biểu thức quan hệ với điều kiện cả hai con trỏ đều trỏ đến các biến cùng loại
- Hãy coi ptr_a và ptr_b là 2 biến con trỏ trỏ đến các phần tử dữ liệu a và b.

Trong trường hợp này, có thể so sánh như sau:

| | |
|--------------------------------|--|
| <code>ptr_a < ptr_b</code> | Returns true provided a is stored before b |
| <code>ptr_a > ptr_b</code> | Returns true provided a is stored after b |
| <code>ptr_a <= ptr_b</code> | Returns true provided a is stored before b or ptr_a and ptr_b point to the same location |
| <code>ptr_a >= ptr_b</code> | Returns true provided a is stored after b or ptr_a and ptr_b point to the same location. |
| <code>ptr_a == ptr_b</code> | Returns true provided both pointers ptr_a and ptr_b points to the same data element. |
| <code>ptr_a != ptr_b</code> | Returns true provided both pointers ptr_a and ptr_b point to different data elements but of the same type. |
| <code>ptr_a == NULL</code> | Returns true if ptr_a is assigned NULL value (zero) |

Phần 5

CON TRỎ VÀ Mảng

- Địa chỉ của một phần tử mảng có thể được biểu diễn theo hai cách:
 - ✓ Bằng cách viết phần tử mảng thực tế đứng trước dấu và (&)
 - ✓ Bằng cách viết một biểu thức trong đó chỉ số dưới được thêm vào tên mảng

Con trỏ và Mảng một chiều-2

// Chương trình C minh họa Tên mảng dưới dạng con trỏ trong C

bao gồm <stdio.h>

int chính()

{

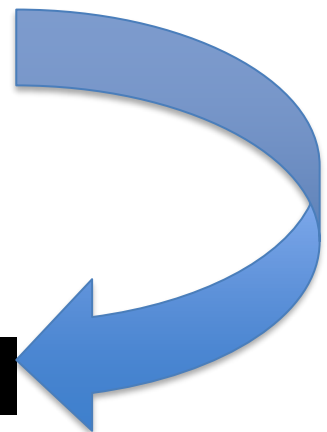
int val[3] = { 5, 10, 15}; // Khai báo một mảng

int *ptr; // Khai báo biến con trỏ // Gán địa chỉ
của val[0] cho ptr

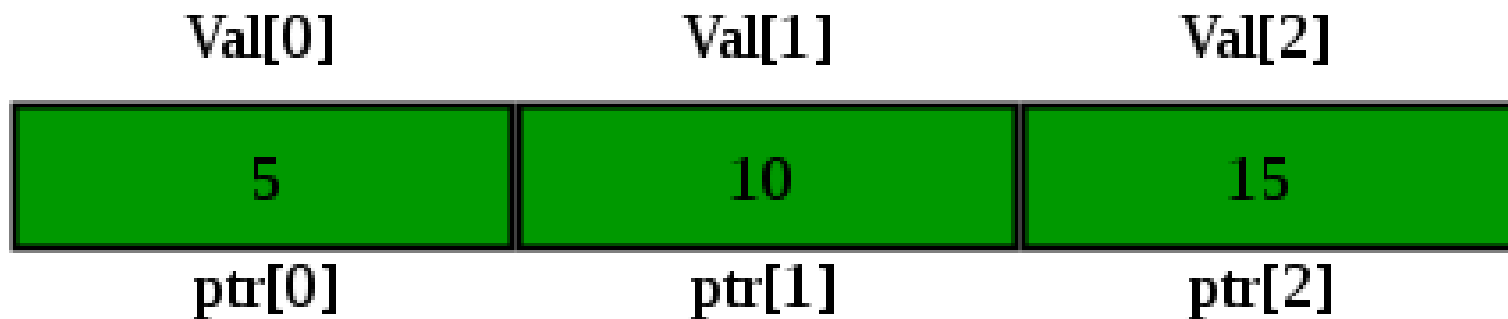
ptr = giá trị; // hoặc dùng: **ptr=&val[0];** (Cả hai đều giống
nhau) printf("Các phần tử của mảng là: "); printf("%d %d
%d", ptr[0], ptr[1], ptr[2]);
trả về 0;

}

Các phần tử của mảng là: 5 10 15



Vị trí các phần tử trong mảng trên trong bộ nhớ ảo được thể hiện bằng hình sau:



Con trỏ và mảng một chiều-4

```
# bao gồm<stdio.h>
```

```
trống rỗng chủ yếu()
```

```
{
```

```
int ary[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; vì (
```

```
int tôi = 0; tôi < 10; tôi ++) {
```

```
// in giá trị của từng phần tử trong mảng
```

```
printf("\ni=%d, ary[i]=%d, *(ary+i)=%d", &i, ary[i], *(ary + i)); printf(
```

```
"&ary[i]= %X, ary+i=%X", &ary[i], ary+i); // %X cho kết quả thập lục
```

```
phân không dấu
```

```
}
```

```
}
```

Đầu ra:

| | | | | |
|-----|-----------|-------------|-------------|-------------|
| i=0 | ary[i]=1 | *(ary+i)=1 | &ary[i]=194 | ary+i = 194 |
| i=1 | ary[i]=2 | *(ary+i)=2 | &ary[i]=196 | ary+i = 196 |
| i=2 | ary[i]=3 | *(ary+i)=3 | &ary[i]=198 | ary+i = 198 |
| i=3 | ary[i]=4 | *(ary+i)=4 | &ary[i]=19A | ary+i = 19A |
| i=4 | ary[i]=5 | *(ary+i)=5 | &ary[i]=19C | ary+i = 19C |
| i=5 | ary[i]=6 | *(ary+i)=6 | &ary[i]=19E | ary+i = 19E |
| i=6 | ary[i]=7 | *(ary+i)=7 | &ary[i]=1A0 | ary+i = 1A0 |
| i=7 | ary[i]=8 | *(ary+i)=8 | &ary[i]=1A2 | ary+i = 1A2 |
| i=8 | ary[i]=9 | *(ary+i)=9 | &ary[i]=1A4 | ary+i = 1A4 |
| i=9 | ary[i]=10 | *(ary+i)=10 | &ary[i]=1A6 | ary+i = 1A6 |

- Mảng hai chiều có thể được định nghĩa là một con trỏ tới một nhóm các mảng một chiều liên kề
- Một khai báo mảng hai chiều có thể được viết là:

`data_type (*ptr_var) [expr 2];`

thay vì

`data_type ptr_var [expr1] [expr 2];`

- Xem xét ký hiệu con trỏ cho mảng số hai chiều. xem xét khai báo sau. Hãy xem ví dụ dưới đây:

```
int nums[2][3] = { {16, 18, 20}, {25, 26, 27} };
```

- Nói chung, **số[i][j]** tương đương với ***(*(số+i)+j)**

| <u>Ký hiệu con trỏ</u> | <u>Ký hiệu mảng</u> | <u>Giá trị</u> |
|--------------------------|---------------------|----------------|
| * (* số) | số[0][0] | 16 |
| * (* số + 1) | số[0][1] | 18 |
| * (* số + 2) | số[0][2] | 20 |
| * (*(số + 1)) | số[1][0] | 25 |
| * (*(số + 1) + 1) | số[1][1] | 26 |
| * (*(số + 1) + 2) | số[1][2] | 27 |

Cảm ơn

Hỏi đáp

