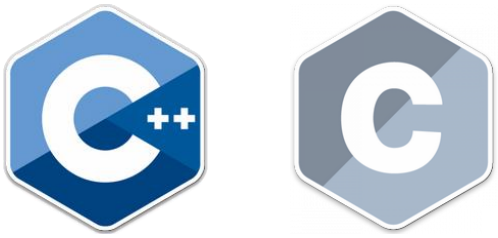
CON TRỎ

Phần 1 - Dẫn nhập



[2017 – 04 – 01]

Biên soạn bởi: Nguyễn Trung Thành
https://www.facebook.com/abcxyztcit

“ Học từ cái đáy của thế giới đi lên là cách duy nhất bạn trở thành master. ”

Nguyễn Trung Thành

Mục lục

A. Giới thiệu

1. Điều kiện để học được tài liệu này

- Có kiến thức nền tảng vững chắc với ngôn ngữ C (hoặc C++).
- Hiểu biết cơ bản về máy tính, bộ nhớ RAM.

2. Dẫn nhập

Từ trước đến nay khi khai báo 1 biến, thông thường chúng ta sẽ quan tâm đến GIÁ TRỊ của biến đó.

Ví dụ:

C/C++

```
int main()
{
	int x = 2, y = 5;
	int tong = x + y;
	printf("Tong = %d", tong); // cout << tong
	return 0;
}
```

Trong đoạn code trên, ta cần tính tổng của x và y. Như vậy cái mà ta quan tâm là GIÁ TRỊ của biến x, biến y và biến tổng.

Ngoài ra ta không cần quan tâm cái gì khác. Thông thường khi học các ngôn ngữ lập trình ta đều nghĩ như vậy.

Tuy nhiên trong ngôn ngữ C/C++, nó chuyên sâu hơn. Bạn nên nhớ 1 điều quan trọng:

Một biến có 2 tính chất:

- Giá trị.
- Địa chỉ của biến đó trong bộ nhớ.

Wowwwwwwwwwwww. Từ xưa đến nay ta 99.99% quan tâm tính chất đầu tiên mà bỏ sót tính chất thứ 2: ĐỊA CHỈ của biến.

Bạn thử nhớ lại lệnh scanf xem.

C

```c
int main()
{
	int x = 0;
	scanf("%d", &x);
	printf("x = %d", x);
	return 0;
}
```

Toán tử & có tác dụng lấy địa chỉ của biến. Vì vậy mà &x sẽ trả về địa chỉ của biến x.

➔ Lệnh scanf nhận vào ĐỊA CHỈ của biến x.

Bây giờ ta xem thử địa chỉ của biến có giá trị là bao nhiêu ?

C	C++

`#include <stdio.h>`
 	
`int main()`
`{`
` int x = 0;`
` printf("Gia tri cua x la %d \n", x);`
` printf("Dia chi cua x la %d \n", &x);`
` return 0;`
`}`
	Không nên thử nghiệm với C++

Chạy thử chương trình:

```
Gia tri cua x la 0
Dia chi cua x la 5241008
```

Ghi chú: kết quả chạy trên máy tính của bạn sẽ khác với minh họa ở trên.

Một lưu ý nho nhỏ là: khi sử dụng %d tức là ta in ra ở hệ thập phân (hệ 10). Khi sử dụng %p tức là ta in ra theo dạng địa chỉ ở hệ hex (hệ 16).

Thêm một chút code....

C

```c
#include <stdio.h>

int main()
{
	int x = 0;
	printf("Gia tri cua x la %d \n", x);
	printf("Dia chi cua x la %d \n", &x);
	printf("Dia chi (hex) cua x la %p \n", &x); // hoặc thay thế %p bằng %x
	return 0;
}
```

C++

```cpp
#include <iostream>
using namespace std;

int main()
{
	int x = 0;
	cout << "Gia tri cua x la " << x << endl;
	cout << "Dia chi cua x la " <<  (int)&x << endl;
	cout << "Dia chi (hex) cua x la " <<  &x << endl;
	return 0;
}
```

Chạy thử chương trình:

```
Gia tri cua x la 0
Dia chi cua x la 4193296
Dia chi (hex) cua x la 003FFC10
```

Ghi chú: kết quả chạy trên máy tính của bạn sẽ khác với minh họa ở trên.

Ghi chú thêm: 4193296 ở hệ thập phân tương đương 3FFC10 ở hệ thập lục phân (hex).

Khi sử dụng địa chỉ, hệ điều hành sẽ thường sử dụng hệ hex hơn là sử dụng hệ thập phân.

Tuy nhiên nhằm đơn giản hóa vấn đề, toàn bộ bài giảng của mình sẽ được minh họa bằng hệ thập phân.

B. Giới thiệu con trỏ

Nối tiếp với 2 ví dụ dẫn nhập ở trên, bạn hãy chạy thử đoạn code sau:

C

```c
#include <stdio.h>

int main()
{
	int x = 0;
	int *p = &x;

	printf("Gia tri cua x la %d \n", x);
	printf("Dia chi cua x la %d \n", &x);
	printf("Gia tri cua p la %d \n", p);
	return 0;
}
```

C++

```cpp
#include <iostream>
using namespace std;

int main()
{
	int x = 0;
	int *p = &x;

	cout << "Gia tri cua x la " << x << endl;
	cout << "Dia chi cua x la " << (int)&x << endl;
	cout << "Gia tri cua p la " << (int)p << endl;
	return 0;
}
```

Chạy thử chương trình:

```
Gia tri cua x la 0
Dia chi cua x la 5241008
Gia tri cua p la 5241008
```

Bạn có thấy điều gì không ???  p là biến có kiểu dữ liệu int*.

Ta gán p = &x, điều này có nghĩa là gì ?

Ta đang có nhu cầu dùng biến p để LƯU ĐỊA CHỈ của biến x.

Vì vậy nên: giá trị của p cũng chính là địa chỉ của biến x.

Ta nói rằng: p là con trỏ. Đó là lý do con trỏ ra đời: con trỏ giúp lưu trữ địa chỉ của biến, mở rộng ra là nó lưu trữ địa chỉ trên bộ nhớ RAM.

Một câu hỏi to đùng được đặt ra: con trỏ lưu địa chỉ để làm gì ?

Đây là 1 trong những câu trả lời đơn giản nhất:

C/C++

```
int main()
{
	int x = 0;
	int *p = &x;

	printf("x = %d \n", x); // cout << x << endl;

	*p = 21;
	printf("x = %d \n", x); // cout << x << endl;

	return 0;
}
```

Chạy thử chương trình:

```
x = 0
x = 21
```

Rõ ràng ta đâu có câu lệnh gán x = 21 nào đâu, nhưng tại sao vẫn in ra x = 21 thế ?

Câu lệnh *px = 21 hoàn toàn tương đương lệnh gán x = 21.

Bạn có biết khi chạy đoạn code trên bạn đã thực hiện 1 điều cực kì quyền năng:

Thông qua con trỏ p, bạn đã gián tiếp thay đổi giá trị của x.

Có được con trỏ p, bạn có thể gián tiếp điều khiển biến x.

- ***Không cần biết chủ nhà là ai***, chỉ cần biết địa chỉ của căn nhà là bạn tùy ý quyết định số phận căn nhà, biết được địa chỉ nhà thì bạn có thể đập phá nó, xây thêm 1 tầng nữa, lấy hết tất cả đồ đạc của căn nhà luôn cũng được.
- Hack game
    - Các chỉ số máu HP, năng lượng mana, sức mạnh, tiền của nhân vật đều được lưu trữ bởi các biến trong bộ nhớ.
    - Biết được địa chỉ của các biến đó → gián tiếp thay đổi giá trị thông qua con trỏ → có thể hack được game.

GHI NHỚ

Con trỏ là biến dùng để lưu trữ địa chỉ bộ nhớ.

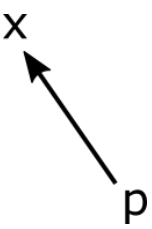
Nhờ vào việc lưu trữ địa chỉ, con trỏ có thể gián tiếp thay đổi giá trị tại vùng nhớ đó (gián tiếp thay đổi giá trị của biến).

C. Các lưu ý khi sử dụng con trỏ

1. Trỏ đúng kiểu dữ liệu

Ôn lại kiến thức một tí:

```
int x = 0;
int *p = &x;
```



Khi ta cho p lưu địa chỉ của x thì ta nói rằng: p trỏ đến x.

Vì p trỏ đến x nên p có thể gán giá trị gián tiếp cho x (bạn đã thấy lúc nãy).

Bạn hãy để ý trong đoạn code trên:

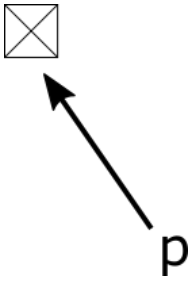
- Biến x có kiểu int.
- Biến p có kiểu int*.

→ Kiểu dữ liệu phải tương ứng mới nhau thì mới được. Con trỏ có kiểu int* không thể lưu địa chỉ của biến kiểu float hay kiểu char…

```
int    x = 3;
float  y = 8;
int   *p;

p = &x; // ok

p = &y; // không được, ERROR

p = (int*)&y; // ép kiểu, ok
```

2. Các dạng con trỏ đặc biệt

Ta có một dạng con trỏ đặc biệt: **con trỏ NULL**. Con trỏ NULL là con trỏ mà không trỏ đến cái gì cả.



Một thao tác tốt khi lập trình đó là khi mới khởi tạo giá trị cho con trỏ, bạn nên gán = NULL cho an toàn.

Ví dụ minh họa:

C/C++

```
int main()
{
	int x = 0;
	int *p = NULL;

	p = &x;

	printf("x = %d \n", x); // cout << x << endl;

	*p = 21;
	printf("x = %d \n", x); // cout << x << endl;

	return 0;
}
```

Trong ngôn ngữ C++ 11 trở lên, ta có thể sử dụng nullptr thay vì NULL.

Ta cũng có một dạng con trỏ đặc biệt khác, đó là **con trỏ void***.

Ở ví dụ lúc trước bạn thấy con trỏ int* chỉ trỏ được vào biến kiểu int.

Tuy nhiên con trỏ void* là ngoại lệ, nó có thể trỏ vào biến kiểu int, float hoặc bất kì kiểu dữ liệu nào khác đều được.

3. Tránh nhầm lẫn khi khai báo con trỏ

Rất nhiều bạn nhầm lẫn nhe ☺.

Ta có thể khai báo 1 con trỏ bằng nhiều cách, ví dụ như…

```
int *p;
int* p;
int * p;
```

Cả 3 dòng trên đều cùng 1 ý nghĩa: khai báo con trỏ p.

Tuy nhiên, bạn **hãy cẩn thận**, xem đây. Hãy dự đoán ý nghĩa của các đoạn code sau:

```
int *a, *b;
```

Đáp án: khai báo 2 con trỏ a và b. Cả 2 con trỏ này đều có kiểu int*.

```
int *a, b;
```

Đáp án: khai báo con trỏ a có kiểu int* và khai báo b có kiểu int.

```
int* a, b;
```

Đáp án: khai báo con trỏ a có kiểu int* và khai báo b có kiểu int.

→ Coi chừng bị nhầm lẫn.

Vì vậy 1 thao tác tốt là bạn nên khai báo con trỏ bằng cách để dấu sao nằm cạnh tên biến.

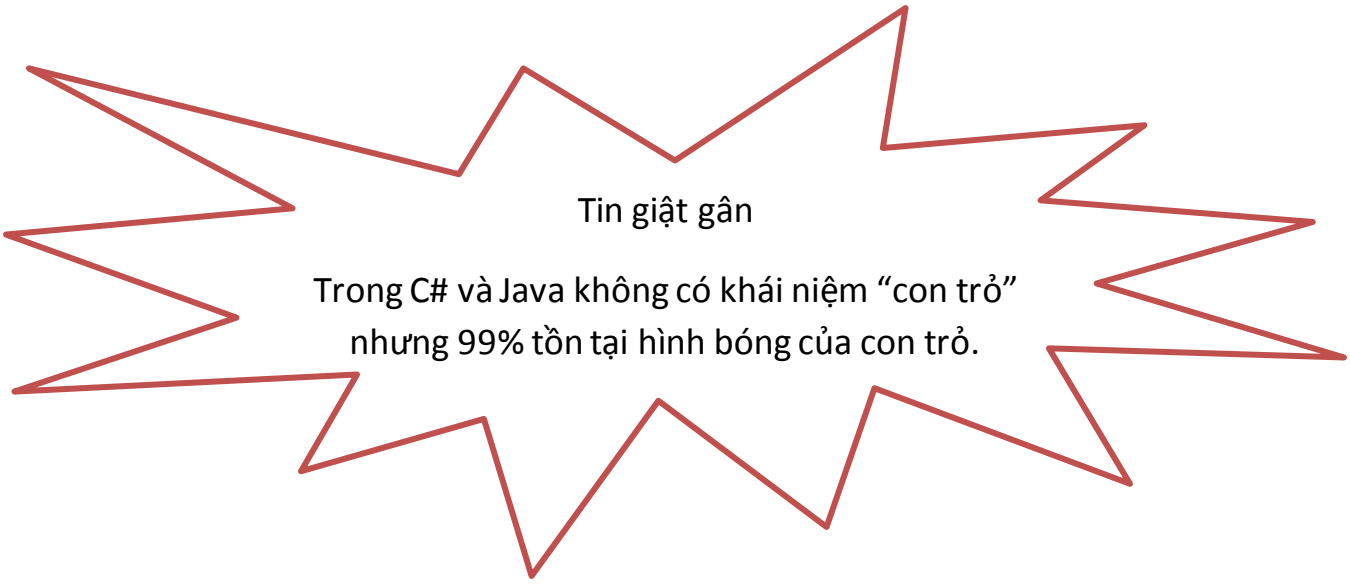
`int *p` thay vì `int* p`

D. Vì sao học con trỏ

Khi nhắc đến con trỏ, người ta thường nghĩ đến ngôn ngữ C/C++. Đó là vì con trỏ là 1 trong những công cụ quyền lực của C/C++. Các ngôn ngữ lập trình khác như C#, Java, Python thì không có con trỏ.

Như vậy bạn học con trỏ bạn chỉ áp dụng được trong ngôn ngữ C và C++. Sang ngôn ngữ lập trình khác thì bạn sẽ vứt đi không quan tâm con trỏ nữa ?

Bạn đã sai rồi.



Con trỏ quá quyền lực nên nó là con dao 2 lưỡi, nó có thể gây nguy hiểm cho chương trình. Vì vậy các ngôn ngữ lập trình bậc cao như C#, Java ngăn chặn điều này, **thay thế khái niệm "con trỏ" bằng khái niệm "tham chiếu"**.

"Tham chiếu" có thể xem là con trỏ nhưng nó bị giới hạn quyền lực đi.

Ví dụ, cho con trỏ p trỏ vào 1 biến số nguyên. Trong C/C++ ta nói rằng "p là con trỏ và p trỏ đến 1 số nguyên", trong C# hoặc Java ta nói rằng "p là tham chiếu đến 1 số nguyên".

Vì vậy nếu bạn học tốt con trỏ trong C/C++ thì sang C# Java bạn học hiểu nhanh, hiểu tường tận, tránh bị sai các lỗi lặt vặt. Nếu bạn nào không có nền tảng C/C++ mà học ngay C#/Java thì có thể bị chết ngay lập tức vì không hiểu tham chiếu là gì, vì sao code bị sai.

E. Sang bài giảng tiếp theo, bạn sẽ học gì ?

Sang phần 2 tiếp theo, bạn sẽ bắt đầu đi sâu hơn về con trỏ.

Học ở phần 1 này, bạn đã biết con trỏ lưu địa chỉ bộ nhớ trên máy tính. Vì vậy sang phần 2, bạn sẽ bắt đầu học sâu hơn về địa chỉ bộ nhớ RAM, từ đó bạn sẽ có nền tảng vững chắc để học tốt hơn.

F. Bài tập

Để dành sang phần 2 làm bài tập luôn cho sướng ☺.

G. Tổng kết

Hi vọng tài liệu này sẽ giúp ích cho bạn. Cảm ơn bạn đã xem.

Tác giả: Nguyễn Trung Thành
Facebook: https://www.facebook.com/abcxyztcit