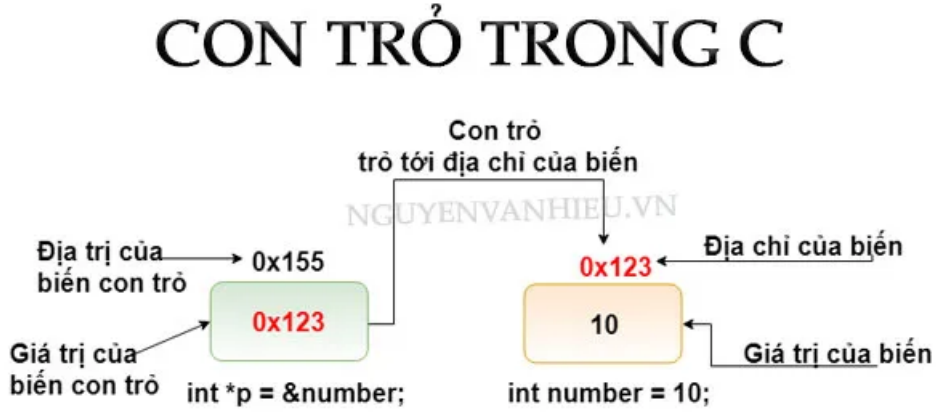
Bài 63. Con trỏ trong C

Bởi **Nguyễn Văn Hiếu** -

Bài số 61 trong 69 bài của khóa học Học C Không Khó

Trong bài học này, Lập trình không khó sẽ hướng dẫn các bạn **cách sử dụng con trỏ** trong ngôn ngữ lập trình C. Bài viết này sẽ giúp các bạn hiểu thế nào là con trỏ, các khái niệm cơ bản liên quan đến con trỏ cũng như cách sử dụng con trỏ trong C. Con trỏ là phần kiến thức khá rộng, do đó bài viết này sẽ hướng dẫn về con trỏ cơ bản; Các bài viết tiếp theo sẽ trình bày chi tiết hơn con trỏ khi làm việc với mảng, cấp phát bộ nhớ và quản lý bộ nhớ,... Mình hi vọng loạt bài học về con trỏ trong C này sẽ giúp các bạn tự tin hơn.



Con trỏ trong C là một loại biến đặc biệt mà giá trị của nó là địa chỉ của 1 biến khác.

22

NỘI DUNG BÀI VIẾT

Địa chỉ của biến trong C

Để hiểu và sử dụng được con trỏ trong C, trước tiên bạn cần hiểu về khái niệm địa chỉ ở trong C. Nếu bạn nào theo dõi khóa học C bá đạo của mình từ đầu thì chắc đã thấy mình nhắc tới khái niệm này rồi. Phần này ta sẽ làm rõ vấn đề này.

```
0
1 int number;
2 printf("\nNhap number = ");
3 scanf("%d", &number);
4 printf("\nnumber = %d", number);
5
```

Bạn hãy nhìn ví dụ trên, tại sao khi dùng hàm `scanf` chúng ta cần truyền vào `&number`, còn hàm `printf` ta lại không có dấu `&` kia? Bởi vì nếu bạn muốn nhập giá trị cho biến, hàm `scanf` cần biết địa chỉ của biến đó ở trong bộ nhớ.

Mỗi biến mà bạn khai báo đều có địa chỉ riêng của nó và giá trị mà nó đang lưu trữ. Để xem được địa chỉ của biến, bạn thêm dấu `&` vào trước tên biến. Xem xét ví dụ dưới đây:

```
0
1  #include <stdio.h>
2  int main()
3  {
4    int number = 5;
5    printf("Gia tri cua number = %d", number);
6
7    // truy xuất địa chỉ bằng cách thêm & trước tên biến
8    printf("\nDia chi cua number = %d", &number);
9    return 0;
10 }
```

```
11
```

Kết quả khi chạy chương trình:

```
0
1 Gia tri cua number = 5
2 Dia chi cua number = 6487580
3
```

Chú ý:

- Bạn có thể sẽ nhận được các địa chỉ khác nhau mỗi khi chạy code trên.
- Để nhận giá trị địa chỉ là `hexa` như ảnh ở đâu bài, bạn thay `%d` bằng `%x` là được.

Con trỏ trong C

Con trỏ là gì? Con trỏ trong C cũng chỉ là là biến, cũng có thể khai báo, khởi tạo và lưu trữ giá trị và có địa chỉ của riêng nó. Nhưng biến con trỏ không lưu giá trị bình thường, nó là biến trỏ tới 1 địa chỉ khác, tức mang giá trị là 1 **địa chỉ**.

Chúng ta cùng thống nhất 1 số khái niệm khi làm việc với con trỏ nhé:

- Giá trị của con trỏ: địa chỉ mà con trỏ trỏ đến.
- Địa chỉ của con trỏ: địa chỉ của bản thân biến con trỏ đó.
- Giá trị của biến nơi con trỏ đang trỏ tới.
- Địa chỉ của biến nơi con trỏ đang trỏ tới = giá trị của con trỏ.

Chính vì con trỏ mang địa chỉ, nó là 1 biến đặc biệt có thêm những quyền năng mà biến bình thường không có. Nhờ việc nó mang địa chỉ, nó có thể trỏ lung tung trong bộ nhớ. Đây là 1 điểm mạnh nếu ta khai thác tốt nhưng nếu quản lý không tốt thì lại là 1 tai hại.

Cách khai báo con trỏ

Con trỏ trong C cũng có thể khai báo giống như biến bình thường, tên biến là một định danh hợp lệ. Cú pháp như sau:

```
<kiểu dữ liệu> * <tên biến>
```

Trong đó:

- Kiểu dữ liệu có thể là: void, int, float, double,...
- Dấu * trước tên biến là ký hiệu báo cho trình biên dịch biết ta đang khai báo con trỏ.

```
int *p_i; // khai báo con trỏ để trỏ tới biến kiểu nguyên
int *p, val; // khai báo con trỏ p kiểu int, biến val (không phải con trỏ) kiểu int
float *p_f; // khai báo con trỏ để trỏ tới biến kiểu thực
char *p_char; // khai báo con trỏ để trỏ tới biến kiểu ký tự
void *p_v; // con trỏ kiểu void (không kiểu)
```

Gán giá trị cho con trỏ

Sau khi khai báo con trỏ, bạn cần khởi tạo giá trị cho nó. Nếu con trỏ được sử dụng mà không được khởi tạo, giá trị của nó sẽ là giá trị rác, điều này sẽ làm chương trình của bạn chạy không đúng, thậm chí là nguy hiểm nếu giá trị rác đó chẳng may lại chính là địa chỉ của 1 biến nào đó bạn đang dùng.

```
int *p, value;
value = 5;
p = &value; // khởi tạo giá trị cho con trỏ p là địa chỉ của value
```

Hoặc bạn cũng có thể khai báo và khởi tạo đồng thời:

```
int value = 5;
int *p = &value; // khai báo con trỏ p và khởi tạo giá trị cho con trỏ là địa chỉ của value
```

22

Lưu ý:

- Con trỏ khi khai báo nên được khởi tạo giá trị ngay.
- Con trỏ kiểu `void` là loại biến con trỏ tổng quát, nó có thể nhận địa chỉ của biến bất kỳ ở bất cứ kiểu dữ liệu nào.

```
0
1  #include <stdio.h>
2  int main()
3  {
4    int number = 5;
5    float *p_int = &number;
6  }
7
8  // Ouput:
9  PS G:\c_cources\day_63> g++ .\Pointer.cpp
10 .\Pointer.cpp: In function 'int main()':
11 .\Pointer.cpp:5:19: error: cannot convert 'int*' to 'float*' in initialization
12     float *p_int = &number;
13                    ^
14
```

- Khởi tạo con trỏ bằng địa chỉ `NULL` nếu chưa cần dùng theo cách sau: `int *p = NULL`. Khi đó con trỏ `NULL` luôn có giá trị `0`.

```
0
1  #include <stdio.h>
2  int main()
3  {
4    void *p_int = NULL;
5    printf("Gia tri cua con tro la %d", p_int);
6  }
7  // Output
8  // Gia tri cua con tro la 0
9
```

22

Bản chất của con trỏ trong C

Bạn sẽ hiểu rõ hơn các quyền năng của con trỏ trong phần này, cũng xem ví dụ dưới đây nào:

```
0
1  #include <stdio.h>
2  int main()
```

```c
{
  // Khai báo + khởi tạo biến value = 10
  int value = 10;

  // Lấy giá trị của biến value
  printf("\nGia tri cua `value` = %d", value);
  // Lấy địa chỉ của biến value
  printf("\nDia tri cua `value` = %d", &value);

  printf("\n-------------------\n");

  /*
  Khai báo + khởi tạo biến con trỏ p
  có giá trị là địa chỉ của biến value
  */
  int *p = &value;

  // Lấy giá trị của con trỏ p
  printf("\nGia tri cua con tro `p` = %d", p);
  // Lấy địa chỉ của con trỏ p
  printf("\nDia tri cua con tro `p` = %d", &p);
  // Lấy giá trị của biến ma con trỏ p đang trỏ tới dùng toán tử *
  printf("\nGia tri cua bien ma con tro `p` dang tro toi = %d", *p);

  printf("\n-------------------\n");

  /*
  Thay đổi giá trị của biến value thông qua con trỏ p
  Giống như hàm scanf() có thể thay đổi giá trị của biến khi nhận vào địa chỉ,
  con trỏ khi có địa chỉ của 1 biến hoàn toàn có thể thay đổi giá trị của
  biến đó theo cách dưới đây:
  */
  // Lấy giá trị của biến value
  printf("\nGia tri cua `value` = %d", value);
  // Thay đổi giá trị của biến value thông qua `p`
  *p = 100;
  // Lấy giá trị của biến value
  printf("\nGia tri cua `value` = %d", value);
  // Lấy giá trị của biến ma con trỏ p đang trỏ tới dùng toán tử *
  printf("\nGia tri cua bien ma con tro `p` dang tro toi = %d", *p);

  printf("\n-------------------\n");

  /*
```

22

```
47    Việc lấy giá trị của biến thông qua con trỏ
48    chỉ là 1 cách khác để lấy được giá trị của biến đó.
49
50    */
51    value = 1000;
52    // Lấy giá trị của biến value
53    printf("\nGia tri cua `value` = %d", value);
54    // Lấy giá trị của biến ma con trỏ p đang trỏ tới dùng toán tử *
55    printf("\nGia tri cua bien ma con tro `p` dang tro toi = %d", *p);
56 }
57
```

Kết quả chạy:

```
0
1  Gia tri cua `value` = 10
2  Dia tri cua `value` = 6487580
3  -------------------
4
5  Gia tri cua con tro `p` = 6487580
6  Dia tri cua con tro `p` = 6487568
7  Gia tri cua bien ma con tro `p` dang tro toi = 10
8  -------------------
9
10 Gia tri cua `value` = 10
11 Gia tri cua `value` = 100
12 Gia tri cua bien ma con tro `p` dang tro toi = 100
13 -------------------
14
15 Gia tri cua `value` = 1000
16 Gia tri cua bien ma con tro `p` dang tro toi = 1000
17
```

Qua ví dụ này, bạn có thể thấy rõ sự đúng đắn của các kết luận sau đây về con trỏ:

22

- Địa chỉ của biến `value` chính là giá trị của con trỏ `p`, đều là `6487580`. Lưu ý mỗi lần chạy thì giá trị địa chỉ này có thể khác nhau nhé.
- Con trỏ có thể lấy giá trị của biến mà nó đang trỏ tới bằng toán tử `*`: `printf("\nDia tri cua con tro p = %d", *p);`
- Con trỏ có thể thay đổi giá trị của biến mà nó đang trỏ tới. Do nó mang địa chỉ của biến, khi đó nó hoàn toàn có quyền thay đổi giá trị của biến đó. Như ở ví dụ trên ta thay đổi giá trị từ 10 lên 100.

Bài học hôm nay chúng ta sẽ chỉ dừng lại ở các kiến thức phía trên, các bài học sau chúng ta sẽ cùng nhau đi tìm hiểu về mối liên hệ giữa con trỏ với mảng và con trỏ với hàm cũng như cách quản lý bộ nhớ khi làm việc với con trỏ trong C.

Các lỗi thường gặp khi làm việc với con trỏ

Giả sử bạn muốn khởi tạo giá trị của con trỏ `p` trỏ tới địa chỉ của biến `value`, khi đó:

```
0
1  int value, *p;
2
3  // Sai! p cần địa chỉ cơ,
4  // value không phải là cái địa chỉ đó.
5  p = value;
6
7  // Sai! *p là giá trị của biến mà con trỏ đang trỏ tới,
8  // &value là địa chỉ.
9  *p = &value;
10
11 // Đúng rồi! p cần 1 địa chỉ,
12 // &value là địa chỉ của biến value.
13 p = &value;
14
15 // Đúng! *p là giá trị của biến mà con trỏ đang trỏ tới, và
16 // c cũng là giá trị (không phải địa chỉ).
17 *p = value;
18
```

Các bạn khi mới học con trỏ sẽ mông lung về dấu `*` ở phần khai báo và khi lấy giá trị của biến mà con trỏ đang trỏ tới:

```c

#include <stdio.h>
int main()
{
    int c = 5;
    // Dấu * ở đây để chúng ta biết chúng ta đang khai báo con trỏ.
    // Không phải lấy giá trị của nó nhé
    int *p = &c;
    // Khai báo trên tương đương
    // int *p;
    // p = &c;
    // Nếu bạn muốn phân biệt 2 thằng này, khi khai báo có thể viết như sau:
    // int* p = &c;

    // Lấy giá chỉ của biến mà con trỏ đang trỏ tới, chính là giá trị của c
    printf("%d", *p); // 5
}

```

Tài liệu tham khảo

Mặc dù mình đã cố gắng trình bày tỉ mỉ, nhưng có thể còn thiếu sót. Dưới đây là 1 số tài liệu bạn nên đọc thêm để hiểu hơn về con trỏ trong C:

1. Tìm hiểu bản chất của con trỏ từ cơ bản tới nâng cao
2. C Pointers (With Example)

Các bài viết trong khóa học

Bài trước: Bài 62. Bài tập chuỗi trong C có lời giải

Bài sau: Bài 64. Mối quan hệ giữa con trỏ và mảng

22

Nguyễn Văn Hiếu

Sáng lập cộng đồng Lập Trình Không Khó với mong muốn giúp đỡ các bạn trẻ trên con đường trở thành những lập trình viên tương lai. Tất cả những gì tôi viết ra đây chỉ đơn giản là sở thích ghi lại các kiến thức mà tôi tích lũy được.