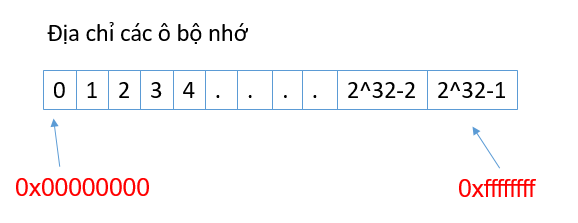
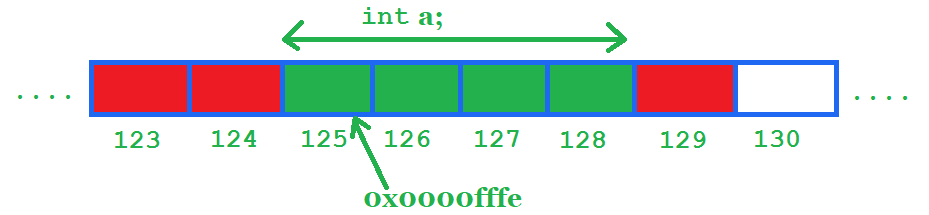
**Khái niệm con trỏ**

Bộ nhớ RAM chứa rất nhiều **ô nhớ**, **mỗi ô nhớ** có **kích thước 1 byte**.

Mỗi ô nhớ có **địa chỉ duy nhất** và địa chỉ này được đánh số từ **0 trở đi**. Nếu CPU 32 bit thì có 2^32 địa chỉ có thể đánh cho các ô nhớ trong RAM.

[](https://gochocit.com/wp-content/uploads/2021/09/dia-chi-don-vi-nho-duoc-danh-dia-chi.png)

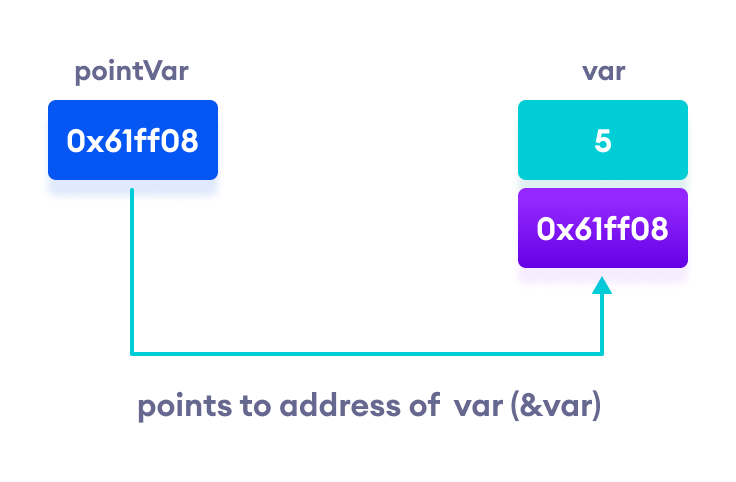
Khi khai báo biến, trình biên dịch dành riêng một vùng nhớ với địa chỉ duy nhất để lưu biến. Trình biên dịch có nhiệm vụ **liên kết** địa chỉ ô nhớ đó với tên biến. Khi gọi tên biến, nó sẽ truy xuất tự động đến ô nhớ đã liên kết với tên biến để lấy dữ liệu. Các bạn phải **luôn phân biệt** giữa **địa chỉ bộ nhớ** và **dữ liệu được lưu trong đó**.

[](https://gochocit.com/wp-content/uploads/2021/09/dia-chi-cua-bien-la-dia-chi-o-nho-dau-tien.png)

Địa chỉ của biến bản chất cũng là **một con số** thường được biểu diễn ở hệ cơ số 16. Ta có thể sử dụng **con trỏ (pointer)**để lưu địa chỉ của các biến.

#### **Con trỏ là gì?**

Trong ngôn ngữ C/C++, **con trỏ (pointer)** là những biến lưu trữ địa chỉ bộ nhớ của những biến khác.

[](https://gochocit.com/wp-content/uploads/2021/09/con-tro-luu-dia-chi-cua-bien.png)

Trong hình trên, biến **var** lưu **giá trị 5** có địa chỉ là **0x61ff08**. Biến **pointVar** là **biến con trỏ**, **lưu địa chỉ** của biến **var**(**trỏ đến vùng nhớ** của biến **var**), tức là nó lưu giá trị **0x61ff08**.

#### **Con trỏ NULL**

Con trỏ **NULL** là con trỏ lưu địa chỉ **0x00000000**. Tức địa chỉ bộ nhớ **0**, có ý nghĩa đặc biệt, cho biết con trỏ không trỏ vào đâu cả.

int \*p2;//con trỏ chưa khởi tạo, vẫn trỏ đến một vùng nhớ nào đó không xác định

int \*p3 = NULL;//con trỏ null không trỏ đến vùng nhớ nào

## **Kích thước của con trỏ**

Ví dụ các khai báo con trỏ sau:

char \*p1;

int \*p2;

float \*p3;

double \*p4;

void \*p5; //Con trỏ đặc biệt trỏ tới mọi kiểu dữ liệu, muốn truy xuất giá trị con trỏ trỏ tới phải ép kiểu

Kích thước của các biến con trỏ **có khác nhau không**? Con trỏ chỉ lưu địa chỉ nên **kích thước của mọi con trỏ là như nhau**. Kích thước này phụ thuộc vào môi trường hệ thống máy tính:

Môi trường **Windows 32 bit**: 4 bytes

Môi trường **Windows 64 bit**: 8 bytes

#### **Con trỏ hàm**

#include <stdio.h>

void tong(int a, int b){

    printf("\ntong %d va %d la %d",a,b,a+b);

}

int main(){

    void (\*ptr2)(int,int)=NULL; //Định nghĩa hàm con trỏ

    ptr2 =&tong;// Khai báo hàm con trỏ

    ptr2(9,7); // tuong duong tong(9,7)

    return 0;

}

Có thể viết ptr2=tong; giá trị của ‘tong’ tương đương địa chỉ của nó, hay nói cách khác tong=&tong

## **Một số lưu ý khi sử dụng con trỏ**

#### **Khi khởi tạo con trỏ NULL**

Chữ **NULL**phải viết hoa, viết thường **null**sẽ bị lỗi.

int \*p1 = NULL;//đúng

int \*p2 = null;//lỗi

#### **Không nên sử dụng con trỏ khi chưa được khởi tạo**

Kết quả tính toán có thể sẽ phát sinh những lỗi không lường trước được nếu chưa khởi tạo con trỏ.

#### **Sử dụng con trỏ đúng cách**

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #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  } |