**Bài 3: Pointer**

# **Tổng quan về pointer**

## **Khái niệm**

Pointer là một biến chứa địa chỉ bộ nhớ của một đối tượng khác, đối tượng ở đây có thể biến,hàm, mảng hay một con trỏ khác.

* 1. Cách khai báo

int \*ptr;

char \*ptr\_char;

float \*ptr\_float;

Con trỏ ptr sẽ trỏ tới địa chỉ của một biến:

int x=31;

int \*ptr=&x; // &x ở đây là địa chỉ biến x, dấu \* ở đây là khai báo biến một con trỏ

* Lấy giá trị của con trỏ :

int y=\*ptr; // dấu \* ở đây sẽ chỉ tới giá trị của địa chỉ ptr

Kích thước của con trỏ phụ thuộc vào kiến trúc máy tính và trình biên dịch:MCU sẽ là 4bit còn x64 là 8bit

#include <stdio.h>

int main (){

int \*ptr;

printf(“size of pointer:%lu bytes\n”,sizeof(ptr));

return 0;

}

# **Ứng dụng của con trỏ**

* 1. **Void pointer**

Con trỏ kiểu void dùng để trỏ đến bất kì địa chỉ nào mà không cần biết kiểu dữ liệu của địa chỉ đó. Con trỏ integer sẽ trỏ tới địa chỉ có giá trị kiểu integer, con trỏ double sẽ trỏ tới địa chỉ có giá trị là double

int a =8;

void \*ptr= &a;

printf(“Gia tri:%d\n”,\*(int \*)p);

int a=31;

char c=’A’;

double d=11.3;

int \*ptr=&c;

ở đây sẽ báo lỗi vì con c có kiểu dữ liệu là char nên con trỏ integer không thể xác nhận được nên ở đây t sẽ dùng con trỏ void

void \*ptr=&a;

Con trỏ void cũng có thể dùng tỏ tới 1 mảng lưu trữ nhiều giá trị đa dữ liệu

int a=31;

float b=31.8

char c=’A’;

double d=11.3;

void \*ph[]={&a,&b,&c,&d};

printf(“Gia tri a=: %d”,\*(int\*)ph[0]);

printf(“Gia tri b=:%f”,\*(float\*)ph[1]);

printf(“Gia tri c=:%c”,\*(char\*)ph[2]);

printf(“Gia tri d=:%d”,\*(double\*)ph[3]);

kết quả

### **Pointer to constant (Con trỏ hằng)**

Con trỏ trỏ tới hằng là con trỏ không thể thay đổi giá trị tại địa chỉ mà nó trỏ đến, nhưng mà có thể thay đổi được địa chỉ của nó.

Const int value1= 10;

Const int value2=20;

Const int \*ptr= &value1; // ptr là con trỏ trỏ tới hằng

\*ptr=15;

|  |
| --- |
| #**include** <stdio.h> #**include** <stdlib.h>  **int** **main**() {    **int** value = 5;  **int** **const** \*ptr\_const = &value;   //\*ptr\_const = 7; // wrong  //ptr\_const++; // right    printf("value: %d\n", \*ptr\_const);   value = 9;  printf("value: %d\n", \*ptr\_const);   **return** 0; } |

Chú ý cách code : int const\*ptr\_const

### **Constant pointer (Hằng con trỏ)**

Định nghĩa một con trỏ mà giá trị nó trỏ đến (địa chỉ ) không thể thay đổi được. Tức là khi con trỏ này được khởi tạo thì nó sẽ không thể trỏ tới địa chỉ khác.

**Cú pháp:**

int \*const const\_ptr= &value;// trỏ tới cố định địa chỉ này luôn và không thể gọi nó bằng địa chỉ khác

int \*const PORTA= 0x001010100010; // có thể truy cập địa chỉ của PORTA để thay đổi giá trị của PORTA

**ví dụ : \*PORTA =60;**

#include<stdio.h>

int main()

{

    int value=15;

     int test=15;

     int \*const const\_ptr=&value;

     printf("Value: %d\n",\*const\_ptr);

     \*const\_ptr=20;

     printf("Value: %d\n",\*const\_ptr);

}

**Không thể sử dụng const\_ptr=&test;**

**Kết quả :**

**Value: 15**

**Value: 20**

### **Function pointer**

**Function to pointer** là một biến mà giữ địa chỉ của một hàm nghĩa là nó trỏ tới vùng bộ nhớ chứa mã máy của hàm được định nghĩa trong chương trình.

Cách khai báo:

Cách 1:

Void print(){

Printf(“Hello Word”);

}

int main(){

void (\*ptr)();

ptr= print;// gán địa chỉ hàm print cho con trỏ ptr

ptr(); // gọi con trỏ ptr để thực hiện hàm print

}

Kết quả :

Hello word

Cách 2:

1. #include<stdio.h>
2. #include<assert.h>
3. void tong(int a,int b)
4. {
5. printf("Tong  cua 2 so %d va %d  la: %d\n",a,b,a+b);
6. }
7. void hieu (int a, int b)
8. {
9. printf("Hieu cua 2 so %d va %d la: %d\n",a,b,a-b);
10. }
11. void tich(int a, int b)
12. {
13. printf("Tich cua 2 so %d va %d la: %d\n",a,b,a\*b);
14. }
15. void thuong(int a ,int b)
16. {
17. assert (b!=0);
18. printf("Thuong cua 2 so %d va %d la: %f \n",a,b,(double)a/b);
19. }
20. int main()
21. {
22. void (\*array[4])(int, int ) = {&tong, &hieu, &tich, &thuong};
23. array[0](2,3);
24. array[1](2,3);
25. array[2](2,3);
26. array[3](2,3);
27. }

Kết quả:

Tong cua 2 so 2 va 3 la: 5

Hieu cua 2 so 2 va 3 la: -1

Tich cua 2 so 2 va 3 la: 6

Thuong cua 2 so 2 va 3 la: 0.666667

### **2.5. Pointer to pointer**

Con trỏ tới con trỏ là một kiểu dữ liệu trong ngôn ngữ lập trình cho phép bạn lưu trữ địa chỉ của một con trỏ. Con trỏ trỏ tới con trỏ cung cấp một cấp bậc trò mới, cho phpes bạn thay đổi ía trị của con trỏ gốc. Cấp bậc này có thể hữu ích trong nhiều tình hướng, đặc biệt là khi bạn làm việc với các hàm cần thay đổi giá trị của con trỏ.

Int a =10;// địa chỉ là 0xc1;

Int \*ptr=&a; a là một biến trên RAM có địa chỉ (giả sử địa chỉ 0xc1 và tại địa chỉ này có giá trị là 0xc1)

Int \*ptr=0xc1;

Int \*\*ptp= &ptr;

Prt bản chất là một biến và trên vùng nhớ RAM có địa chỉ giả sử là 0x08 và có giá trị đó là 0xc1A screenshot of a white sheet

AI-generated content may be incorrect.

A diagram of a double pointer

AI-generated content may be incorrect.

#include <stdio.h>

int main() {

    int num = 10;         // Bien so nguyen

    int \*ptr1 = &num;     // Con tro tro den num

    int \*\*ptr2 = &ptr1;   // Con tro tro den ptr1 (pointer to pointer)

    // Hien thi gia tri va dia chi cua num

    printf("Gia tri cua num: %d\n", num);

    printf("Dia chi cua num: %p\n\n", (void\*)&num);

    // Hien thi gia tri va dia chi cua ptr1

    printf("Gia tri cua ptr1 (dia chi cua num): %p\n", (void\*)ptr1);

    printf("Gia tri tro toi boi \*ptr1: %d\n", \*ptr1);

    printf("Dia chi cua ptr1: %p\n\n", (void\*)&ptr1);

    // Hien thi gia tri va dia chi cua ptr2

    printf("Gia tri cua ptr2 (dia chi cua ptr1): %p\n", (void\*)ptr2);

    printf("Gia tri tro toi boi \*\*ptr2: %d\n", \*\*ptr2);

    printf("Dia chi cua ptr2: %p\n", (void\*)&ptr2);

    return 0;

}

Kết quả :

Gia tri cua num: 10

Dia chi cua num: 000000000061FE1C

Gia tri cua ptr1 (dia chi cua num): 000000000061FE1C

Gia tri tro toi boi \*ptr1: 10

Dia chi cua ptr1: 000000000061FE10

Gia tri cua ptr2 (dia chi cua ptr1): 000000000061FE10

Gia tri tro toi boi \*\*ptr2: 10

Dia chi cua ptr2: 000000000061FE08// phần này là do địa chỉ của ptr nên nó là random

### **2.6. NULL pointer**

Con trỏ NULL pointer là một con trỏ không trỏ tới bất kỳ đối tượng hoặc vùng nhớ cụ thể nào có giá trị =0 và địa chỉ= 0xc0

#include <stdio.h>

int main()

{

    int \*ptr=NULL;

    printf("Pointer is %p\n", ptr);

    if (ptr != NULL)

    {

        printf("Pointer is NOT NULL\n");

    }

    else

    {

        printf("Pointer is NULL\n");

    }

}

Sử dụng NULL pointer hữu ích để kiểm tra xem một con trỏ đã được khởi tạo và có trỏ tới đến vùng hợp lệ hay chưa bởi bình thường khi khai báo như int\*ptr; thì ptr sẽ có địa chỉ tự phát nên để tránh lỗi chương trình khi lập trình chúng ta nên khai báo con trỏ Null để tránh derefering hay là bị lỗi chương trình.



A screenshot of a computer program

AI-generated content may be incorrect.

Cái này hay nha đọc lại và code lại