- gán: int \*ptr = &n;

- Khai báo con trỏ: int \*ptr;

- con trỏ sau khi khai báo sẽ có giá trị là NULL;

- Biến a = 300; &a là địa chỉ ô nhớ của biến a

- \*ptr bằng giá trị của biến a; ptr bằng giá trị của địa chỉ ô nhớ của biến a;

- Khi thao tác với \*ptr chính là thao tác với biến a;

int a = 300;

int \*ptr;

ptr = &a;

printf("dia chi cua bien a la: %d\n", &a);

printf("gia tri cua con tro ptr la: %d", ptr);

printf("gia tri cua bien ma con cho dang quan ly: %d", \*ptr);

\*ptr = 1000;

- Trong hàm tang2, tham số int \*a nằm ở trên đầu của biến nên nó là địa chỉ của ô nhớ, còn \*a được thao tác +- ở dưới là giá trị nằm trong ô nhớ(biến). Dấu \* ở phần thao tác +- là toán tử giải tham chiếu.

void tang1(int a){

a += 200;

}

void tang2(int \*a){

\*a += 200;

}

int main(){

int a = 300;

tang1(a);

printf("%d\n", a);

tang2(&a);

printf("%d", a);

}

- Tên của mảng cũng được coi là một con trỏ. Con trỏ đó chỉ về ô nhớ của phần tử a[0] trong mảng.

int main(){

int a[3] = {1, 2, 3};

printf("%d\n", a);

printf("%d", &a[0]);

}

- Nếu mảng là int, địa chỉ của ô nhớ đằng sau sẽ hơn 4đv so vs ô nhớ trc.

int main(){

int a[3] = {1, 2, 3};

for(int i = 0; i < 3; i++){

printf("dia chi cua phan tu a[%d] la: %d\n", i, &a[i]);

}

}

Output: dia chi cua phan tu a[0] la: 6487568

dia chi cua phan tu a[1] la: 6487572

dia chi cua phan tu a[2] la: 6487576

- Khi khai báo con trỏ \*b chỉ về mảng a thì b dùng như a.

int main(){

int a[3] = {1, 2, 3};

int \*b = a;

for(int i = 0; i < 3; i++){

printf("%d ", b[i]);

}

}

- Đối với mảng a, có thể thay &a[i] bằng a + i (a + i là một con trỏ trỏ về địa chỉ của phần tử a[i] trong mảng

a[i] bằng \*(a + i) ( \*(a+i) là giá trị của phần tử a[i] trong mảng.

- a là con trỏ trỏ vào phần tử a[0], a là hằng con trỏ(ko thể a++);

int main(){

int a[3];

for(int i = 0; i < 3; i++){

scanf("%d", a + i);

}

for(int i = 0; i < 3; i++){

printf("%d ", \*(a + i));

}

}

- khi tạo con trỏ ptr cho 1 giá trị a[i] trong hàm a, thì khi ptr += 1, con trỏ ptr sẽ trỏ vào phần tử bên cạnh.

#include <stdio.h>

#include <string.h>

int main(){

int a[] = {1, 2, 3, 4, 5, 1};

int \*ptr = &a[2];

ptr++;

printf("%d", \*ptr);

}

output: a[3] = 4.

- con trỏ ptr1 đang trỏ vào a[1].

- int \*ptr2 = ptr1 là cách khai báo con trỏ ptr2 cùng level, cùng trỏ vào a[1] giống ptr1. (ô nhớ của ptr1 khác ptr2)

- mặc dù int \*ptr2 = \*ptr1 nghe hợp lí hơn khi muốn ptr2 trỏ vào a[1] giống ptr1, nhưng khi làm thế thì máy ko compile được. vì dấu \* đầu tiên là để khai báo con trỏ, dấu \* thứ 2 là để giải tham chiếu về a[1].

int main(){

int a[] = {1, 2, 3, 4, 5};

int \*ptr1 = a + 1;

int \*ptr2 = ptr1;

printf("%d %d %d", ptr1, ptr2, \*ptr1);

}

-cách khai báo con trỏ level 2: con trỏ ptr2 trỏ vào con trỏ ptr1.

int main(){

int a[] = {1, 2, 3, 4, 5};

int \*ptr1 = a + 1;

int \*\*ptr2 = &ptr1;

printf("%d %d %d", \*ptr1, \*ptr2, ptr1);

}

- Tóm lại: (\*)

int \*ptr2 = ptr1; tạo trỏ ptr2 giống ptr1 (khác ô nhớ vs ptr1)

int \*ptr2 = \*ptr1; lỗi ko compile được

int \*ptr2 = &ptr1; lỗi vì ptr2 là con trỏ cấp 1 ko lưu được ô nhớ của con trỏ cấp 1 khác

- khai báo con trỏ ptr2, và làm ptr2 giống ptr1 nhưng không cùng 1 dòng code như vd (\*1) ở trên

int main(){

int a[] = {1, 2, 3, 4, 5};

int \*ptr1 = a + 1;

int \*ptr2;

ptr2 = ptr1;

printf("%d %d %d", \*ptr1, \*ptr2, ptr1);

}.

- Cách tạo int ptr[1000000]; (cứ khai báo kiểu bth như thế thì không được)

đây gọi là cấp phát động

int main(){

int \*ptr = (int\*)malloc(1000000 \* sizeof(int)); tạo mảng int có tên ptr có 1000000 phần tử (kiểu như int ptr[1000000])

int n = 5;

for(int i = 0; i < n; i++){

scanf("%d", &ptr[i]);

}

for(int i = 0; i < n; i++){

printf("%d ", ptr[i]);

}

free(ptr); giải phóng bộ nhớ

}