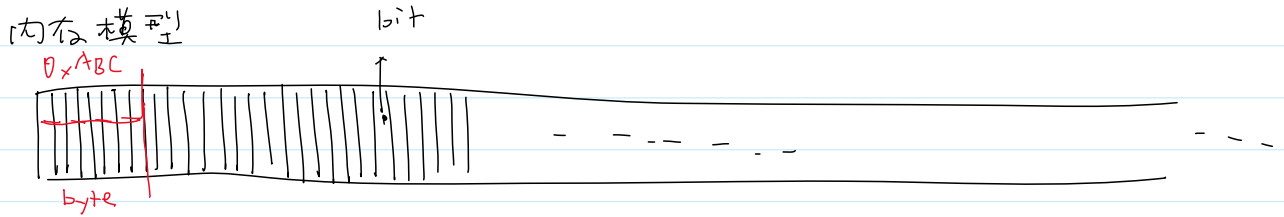


指针基础

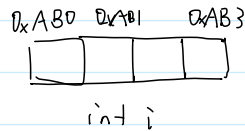
2024年4月27日 9:12



计算机最小的寻址单位, byte

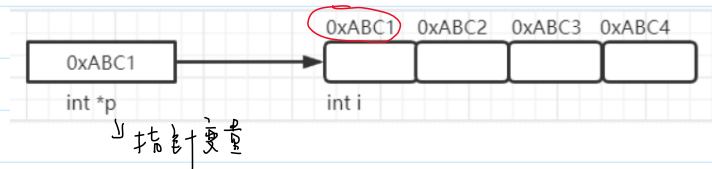
有地址,

变量的地址,
变量首字节的地址。



指针, 地址.

指针变量, 存储地址的变量; 有时候也把指针变量叫作指针.



指向对象的类型. \Rightarrow $\left\{ \begin{array}{l} \text{变量名: } p \\ \text{类型: } \text{int}^* \end{array} \right.$

① 对象所占内存大小.

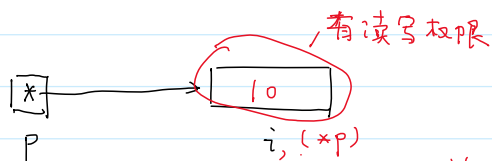
② 解释那片内存空间

$\text{int}^* p, q;$ $\rightarrow \text{int}$
 \downarrow
 int^*

$\text{int } *p, *q;$ $\rightarrow \text{int}^*$
 \downarrow
 int^*

`int a, *p, arr[10];`

`/* 两个基本操作 */
int i = 1;
int* p = &i;`



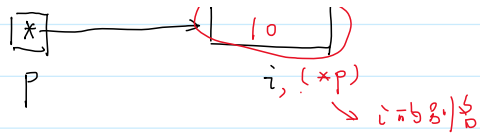
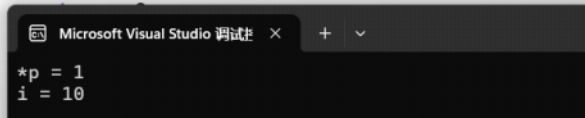
/* 两个基本操作 */

```
int i = 1;
int* p = &i;
```

```
printf("*p = %d\n", *p);
```

```
*p = 10;
```

```
printf("i = %d\n", i);
```



i: 直接访问, 逻辑上访问内存一次.
*p: 间接访问, 逻辑上访问内存两次.

野指针, → 不知道指向什么数据.

```
#include <stdio.h>
```

```
int main(void) {
    int* p;
    printf("p = %p\n", p);
    printf("*p = %d\n", *p);
}
```

```
p = 0x401040
```

int *p;
int *q = 0xABCD; (野指针)

对野指针的引用: 未定义的行为.

↓

Q: 如何给指针变量赋初值?

A. int *p = &i; (✓)

B. int *q = p; (✓)

C. p = NULL; (✓)

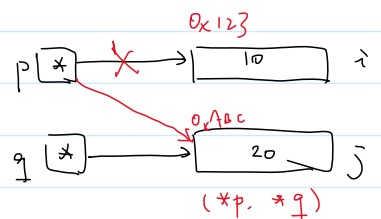
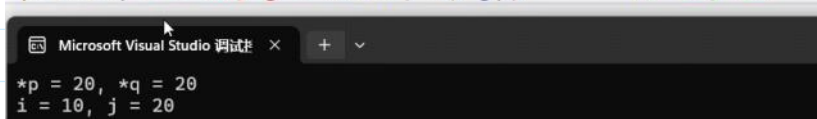
↳ 空指针, 不指向任何对象指针.

野指针: 不知道指向哪块数据.

```
int i = 10, j = 20;
int* p = &i;
int* q = &j;
```

```
p = q;
```

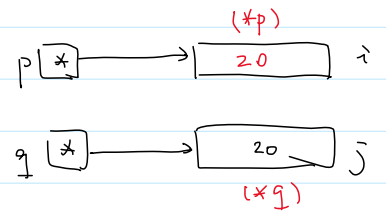
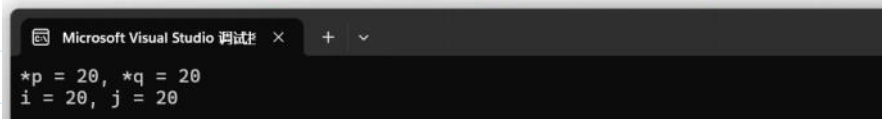
```
printf("*p = %d, *q = %d\n", *p, *q); // 20, 20
printf("i = %d, j = %d\n", i, j); // 10, 20
```



```
int i = 10, j = 20;  
int* p = &i;  
int* q = &j;
```

```
// p = q;  
*p = *q;
```

```
printf("*p = %d, *q = %d\n", *p, *q); // 20, 20  
printf("i = %d, j = %d\n", i, j); // 20, 20
```



指针的应用

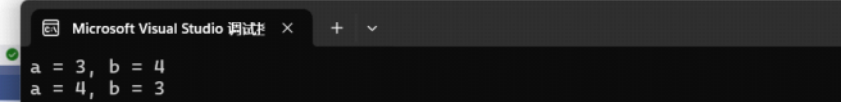
2024年4月27日 10:50

① 作为参数。 → 在被调函数中修改主调函数的值。

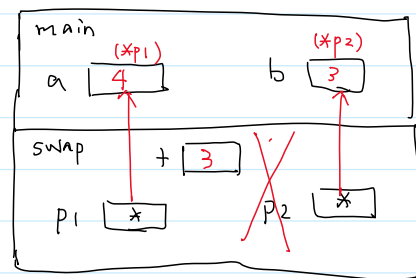
函数调用

```
void swap(int* p1, int* p2) {  
    int t = *p1;  
    *p1 = *p2;  
    *p2 = t;  
}  
  
int main(void) {  
    int a = 3, b = 4;  
  
    printf("a = %d, b = %d\n", a, b);  
    swap(&a, &b);  
    printf("a = %d, b = %d\n", a, b);  
}
```

int *p1 = &a;

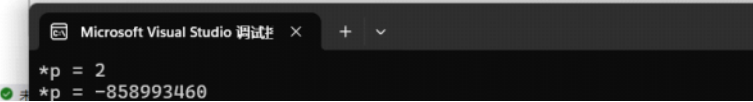


a = 3, b = 4
a = 4, b = 3

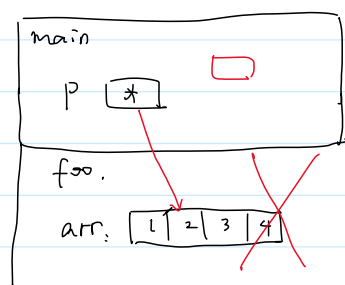


② 指针作为返回值。

```
int* foo(void) {  
    int arr[] = { 1, 2, 3, 4 };  
    return &arr[1];  
}  
  
int main(void) {  
    int* p = foo();  
    printf("*p = %d\n", *p);  
    printf("p = %d\n", p);  
    return 0;  
}
```



*p = 2
p = -858993460



教训：不要返回指向当前栈帧的指针！