



计算机导论与程序设计——第10篇

# 指针与动态存储管理

*Computer Introduction and Programming*

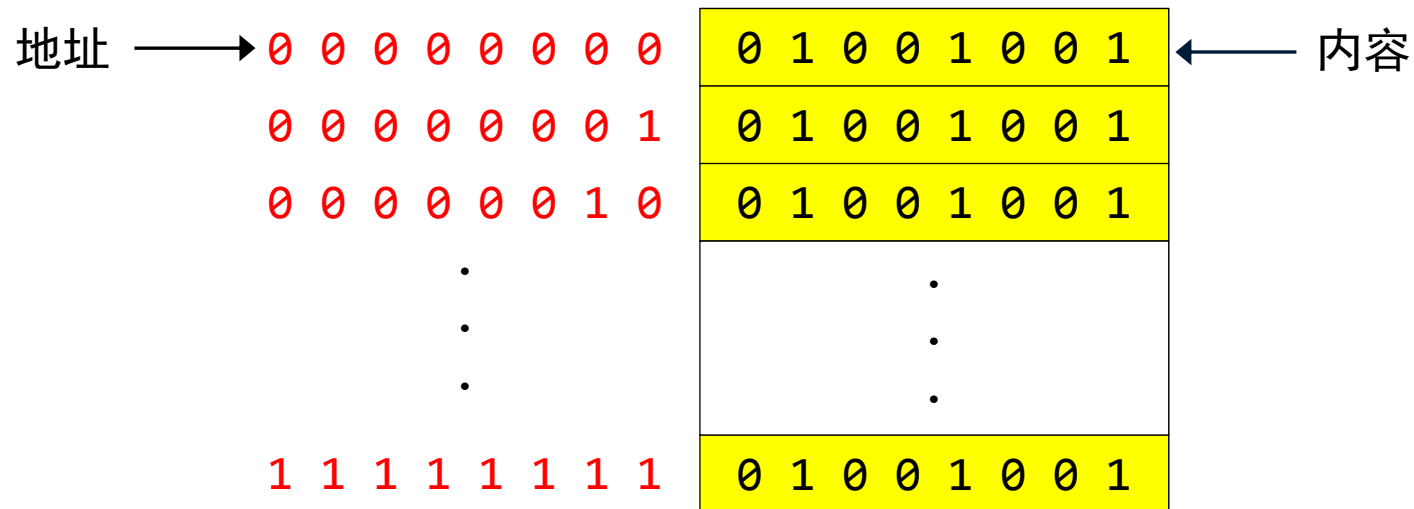
# 学习目标



- 掌握指针的含义以及在内存中的表示
- 掌握如何定义和使用指针引用变量
- 掌握指针与一维数组的关系以及指针运算的意义及其使用方法
- 熟悉指针作为函数参数的含义与使用方法
- 了解动态内存管理的概念
- 熟悉常用内存管理的函数及使用方法

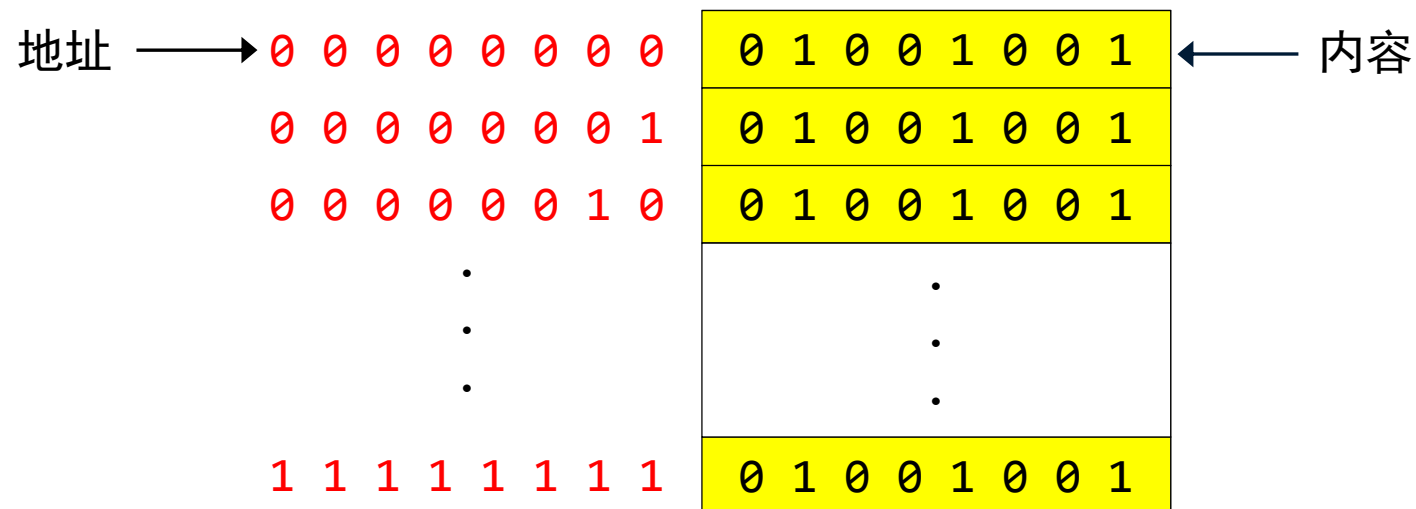
什么是地址？

# (回顾)主存储器(内存)



- 存储信息的基本单位是位(bit)
- 每8位二进制数合在一起称为一个字节(Byte)
- 存储器的一个存储单元一般存放一个字节的信  
息
- 存储容量指存储器所能存储的全部二进制信息量。单位有KB(千字节)、MB(兆字节)、GB(吉字节)、TB(太字节)。

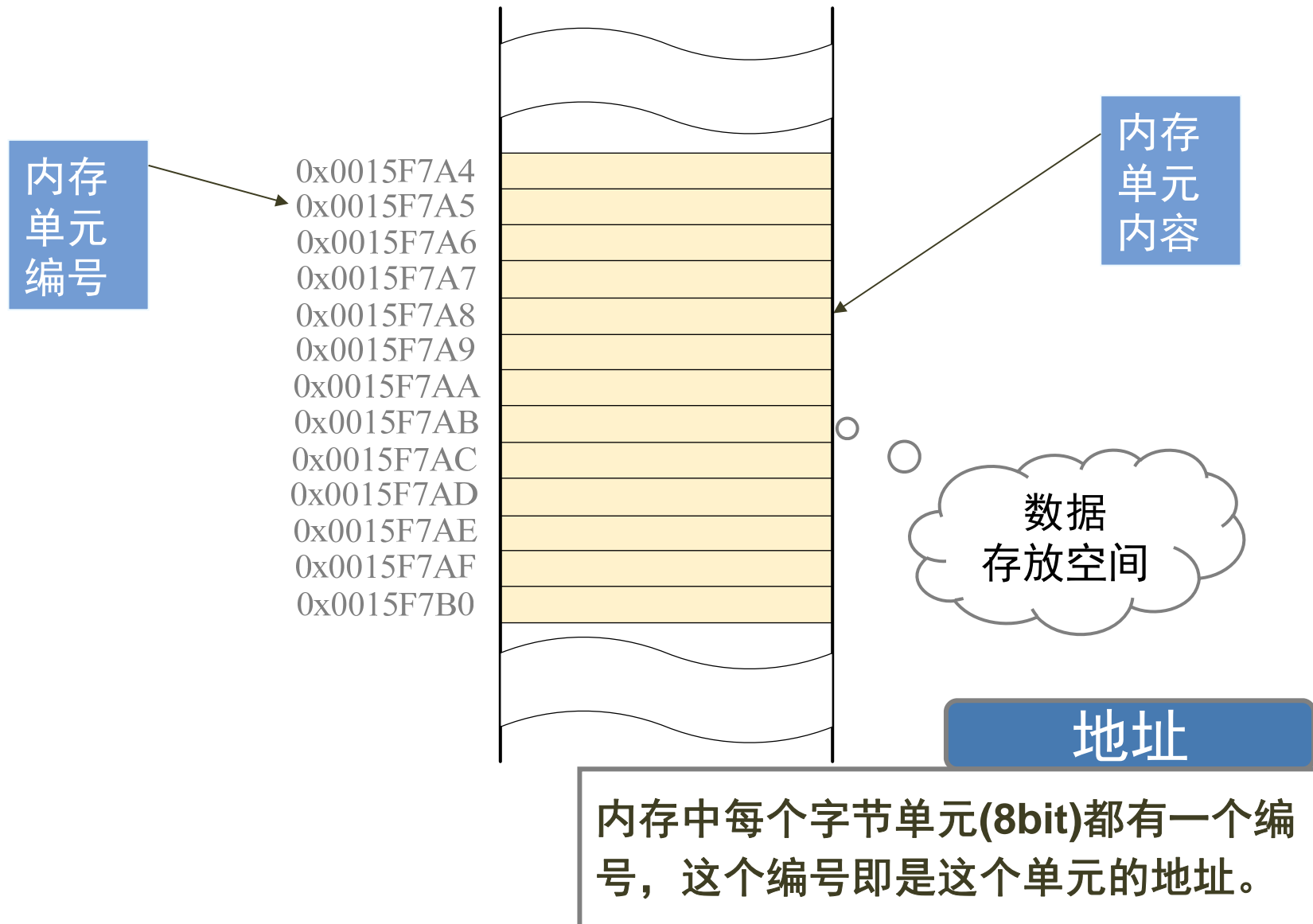
# (回顾)主存储器(内存)



- 每个存储单元都有唯一的编号，称为地址
- 存储单元的地址和该地址内存放的内容是两个完全不同的概念

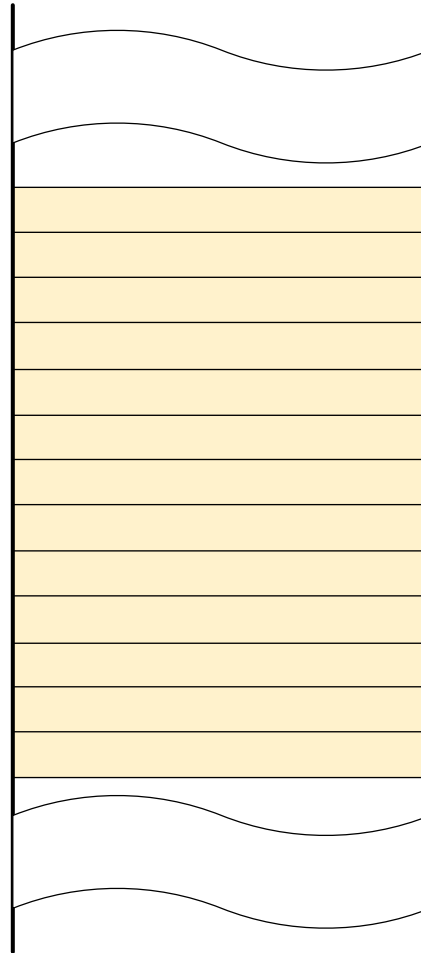
变量是如何在内存中存储的？

# 数据的存放与位置



# 数据的存放与位置

0x0015F7A4  
0x0015F7A5  
0x0015F7A6  
0x0015F7A7  
0x0015F7A8  
0x0015F7A9  
0x0015F7AA  
0x0015F7AB  
0x0015F7AC  
0x0015F7AD  
0x0015F7AE  
0x0015F7AF  
0x0015F7B0

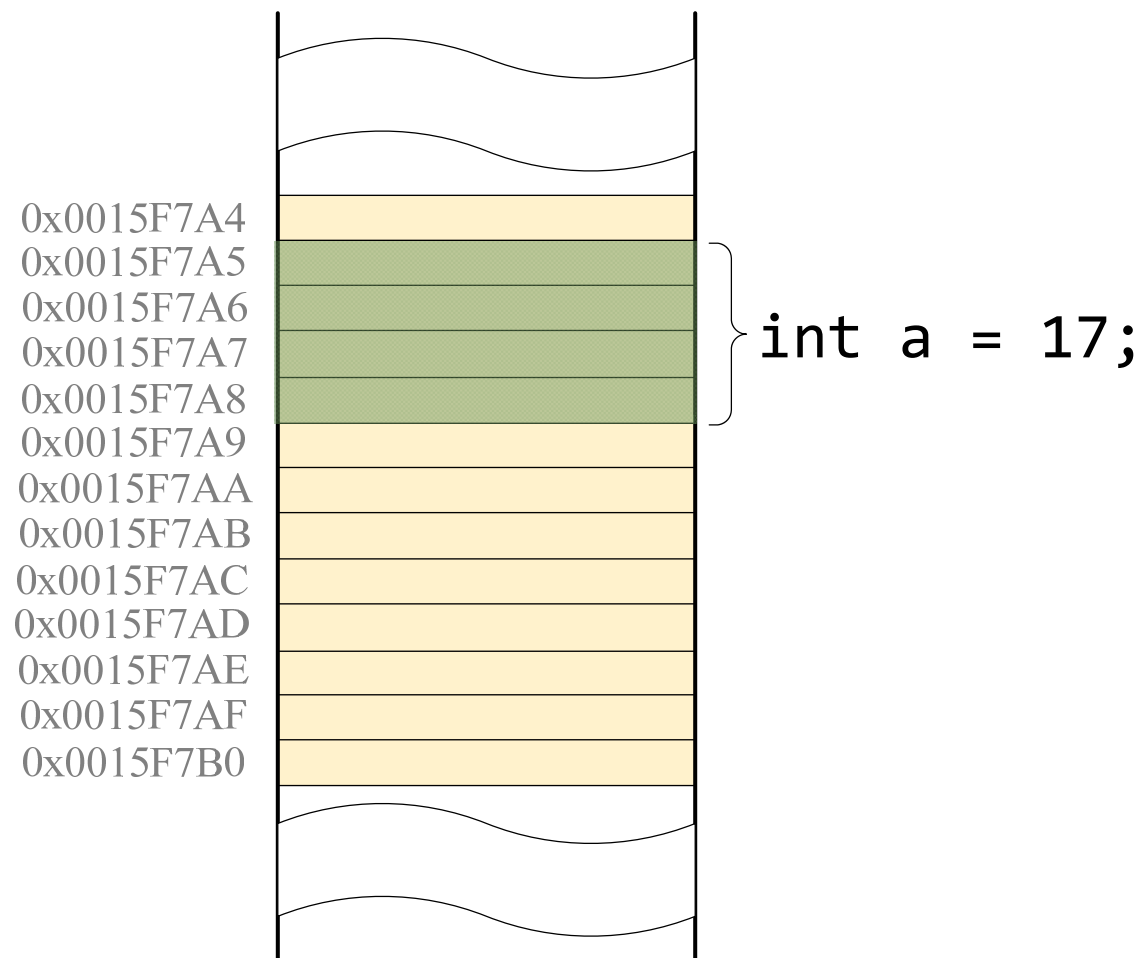


```
int a = 17;  
int b = -17;
```

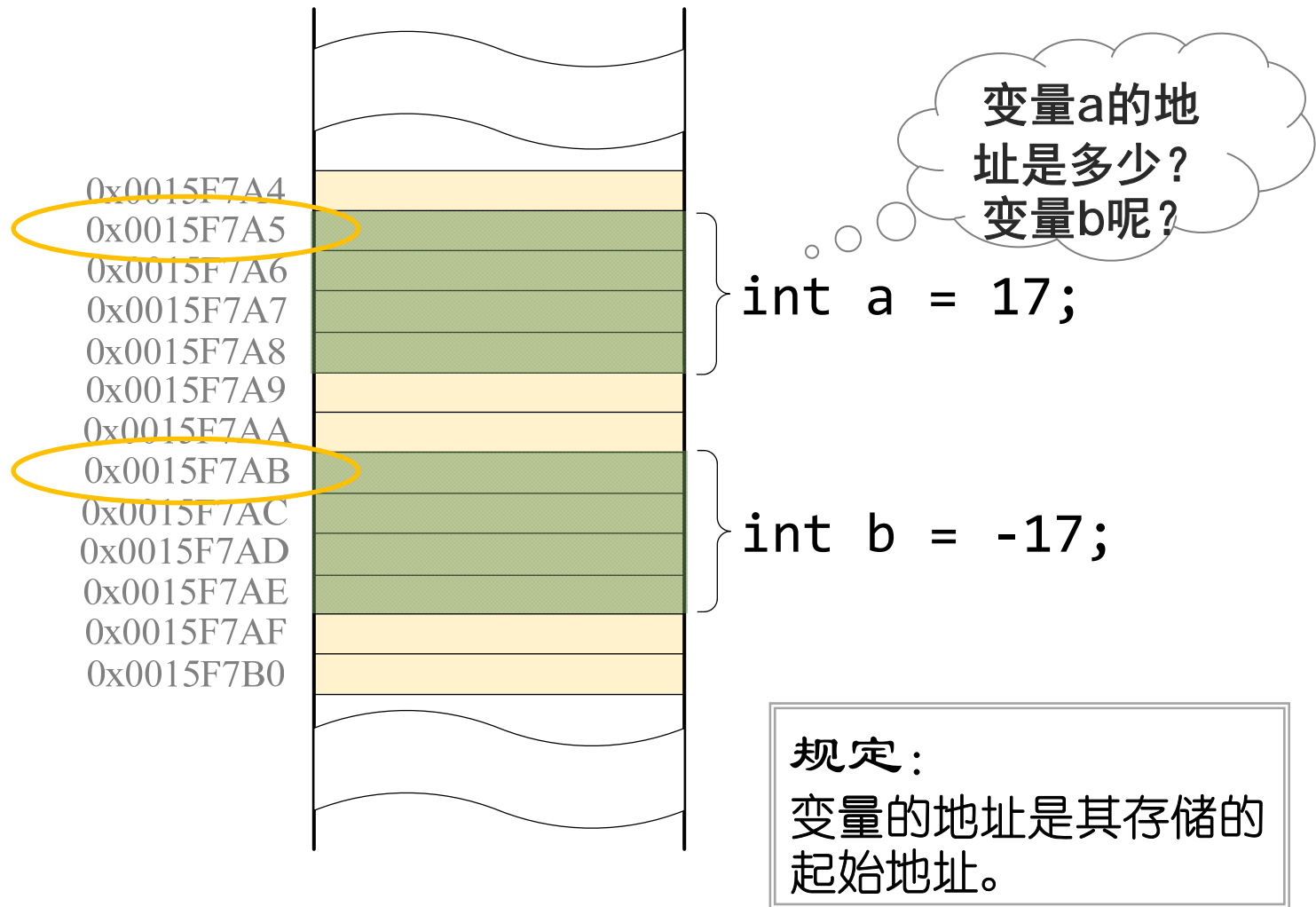
变量的类型决定了存储空间的大小，而存储的位置由编译器分配。



# 数据的存放与位置



# 数据的存放与位置



# (回顾)数值格式的表示

---

- 整数的表示范围
  - ① 计算机通常采用固定的二进制位数(码长)来表示数，因此，数的表示范围是有限的；
  - ② 为了区分正数和负数，约定最高位(即最左边的那一位)为0时表示正数，为1时表示负数，即符号数字化。

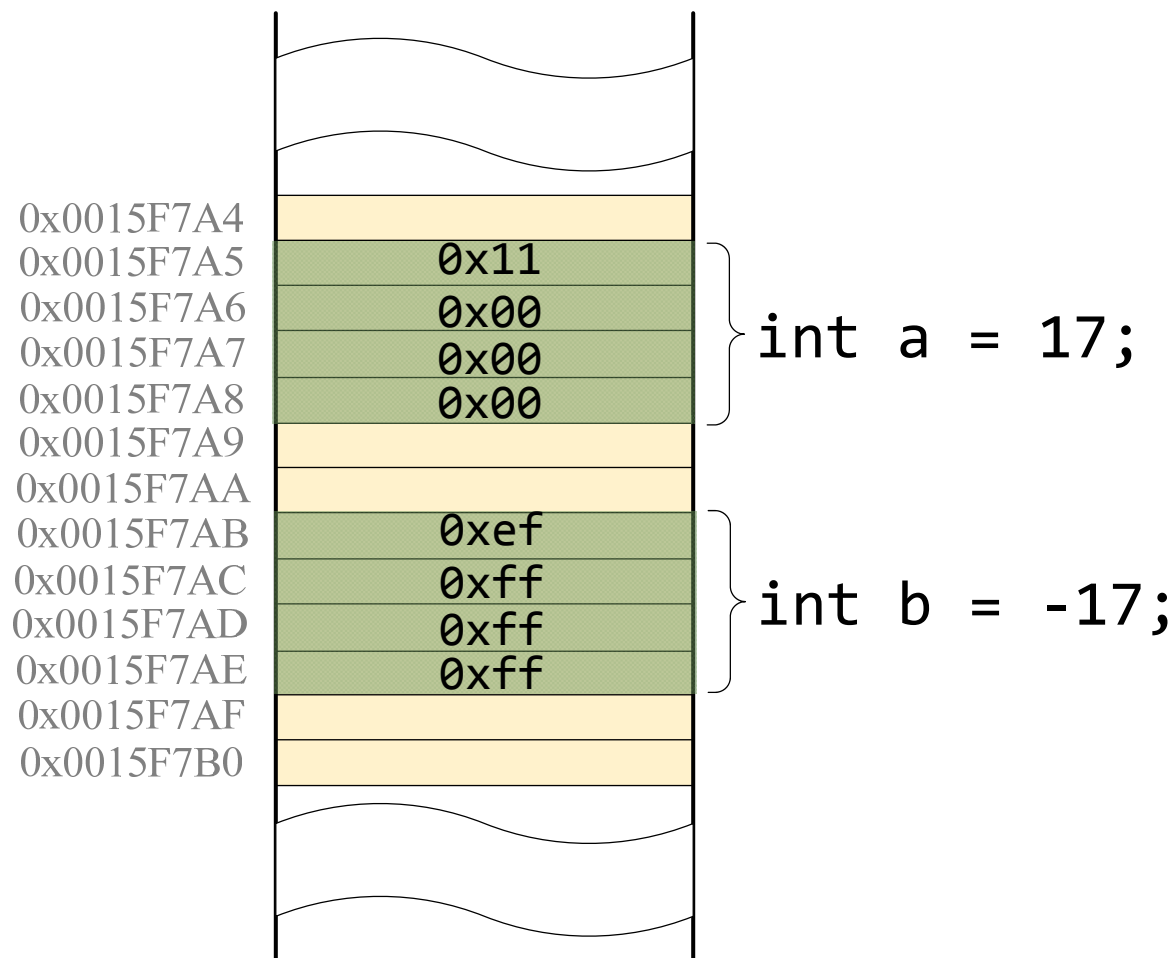
# (回顾)数值格式的表示

---

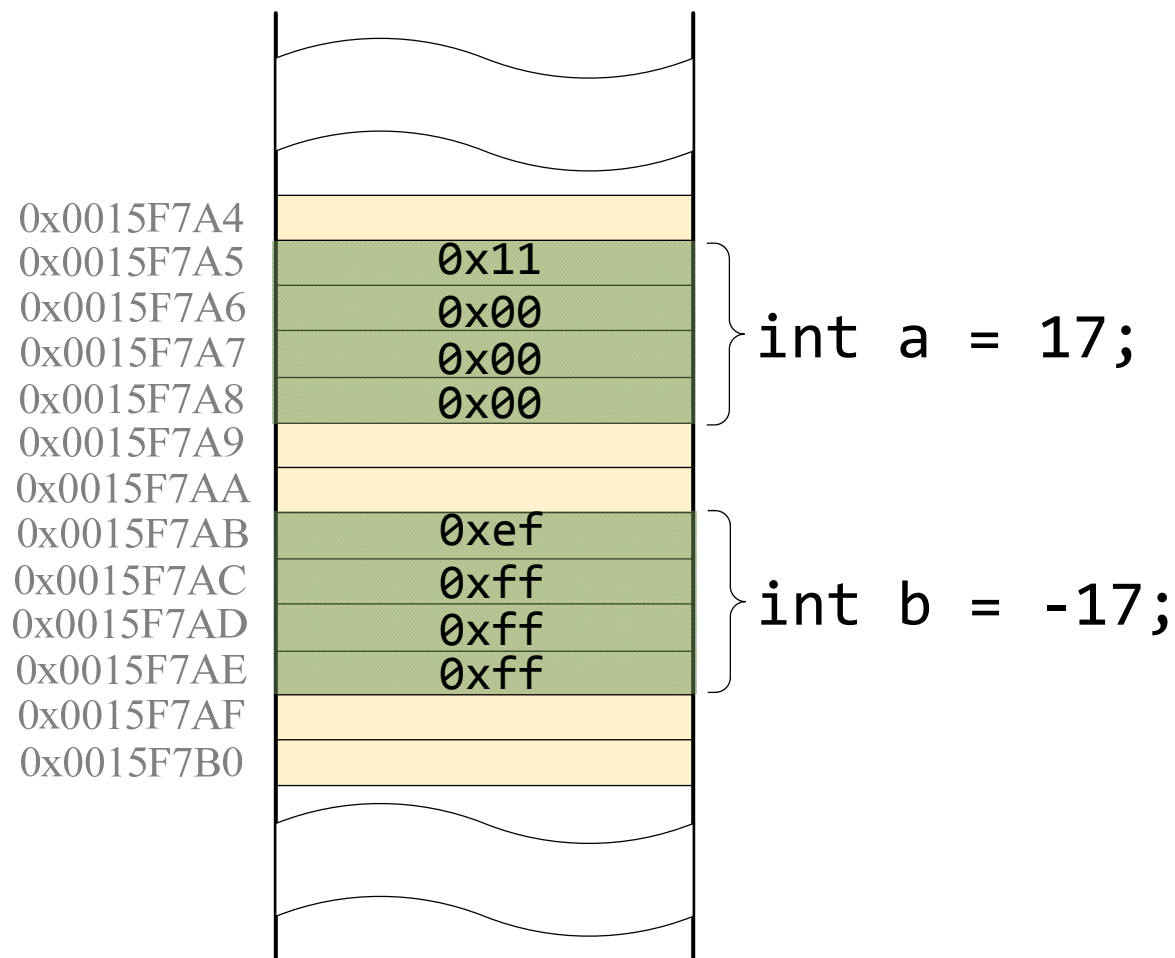
- 相关术语

- ① 真值：实际的二进制数据；
- ② 原码：把真值的符号数字化处理；
- ③ 反码：一个负数的原码保留符号位不变，其余按位取反就成为该数的反码；
- ④ 补码：一个负数的反码末位加1就成为该数的补码；
- ⑤ 一个正数的原码、反码和补码形式是一样的。

# 数据的存放与位置

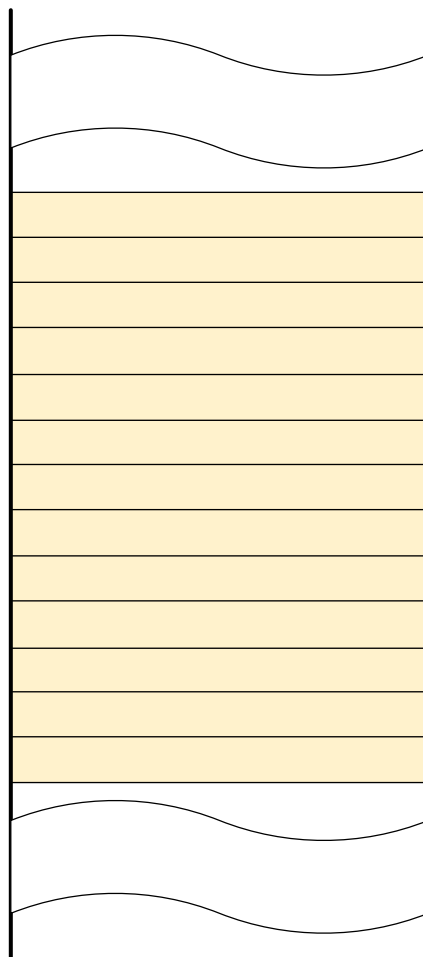


# 数据的存放与位置



# 数据的存放与位置

0x0015F7A4  
0x0015F7A5  
0x0015F7A6  
0x0015F7A7  
0x0015F7A8  
0x0015F7A9  
0x0015F7AA  
0x0015F7AB  
0x0015F7AC  
0x0015F7AD  
0x0015F7AE  
0x0015F7AF  
0x0015F7B0



```
char c1 = 'A';  
char c2 = 'a';
```

变量的类型决定了存储空间的大小，而存储的位置由编译器分配。

# (回顾)字符的表示

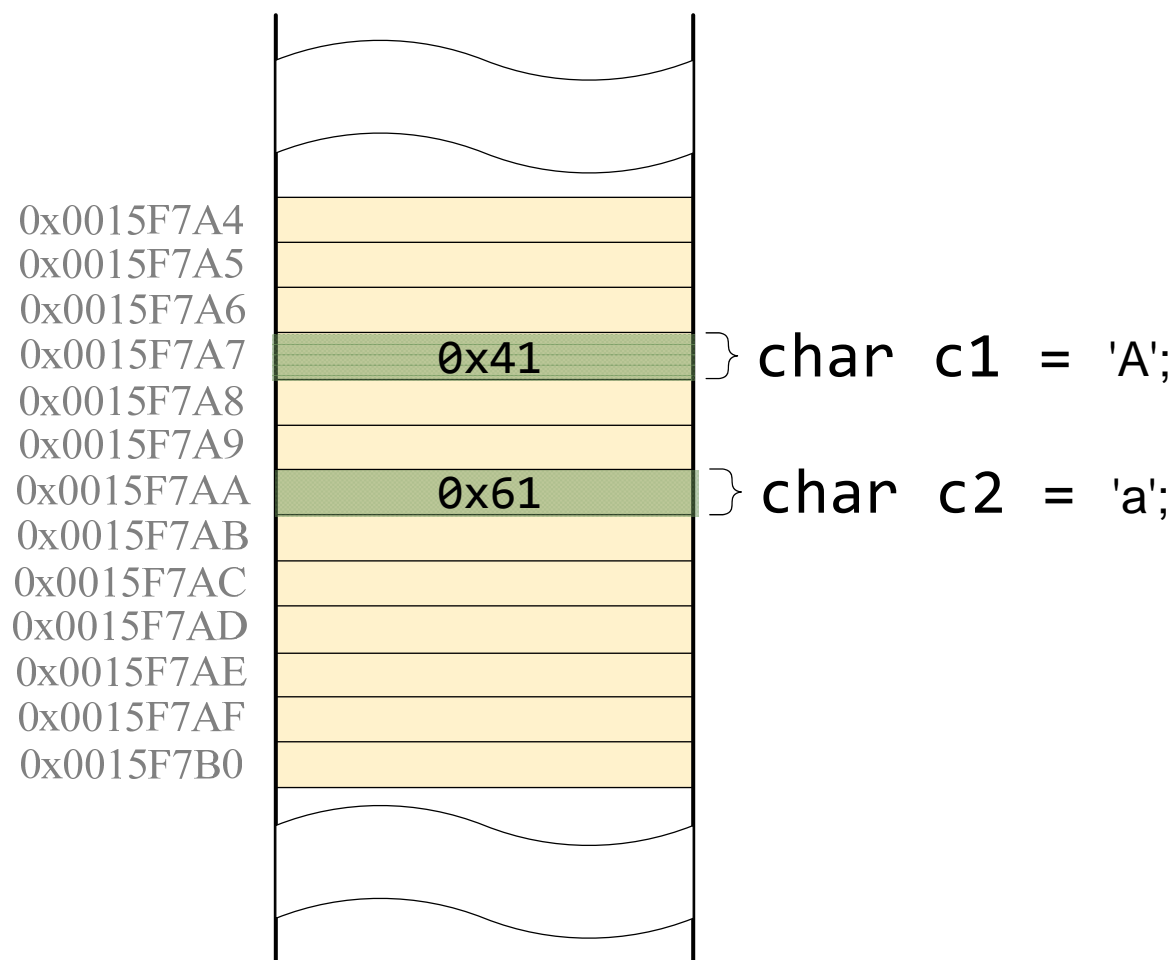
---

- 数字、字母和符号统称为字符；
- 字符必须按特定规则用二进制编码表示，才能被计算机识别和处理；
- 计算机中最常用的字符编码是ASCII码，即American Standard Code for Information Interchange(美国信息交换标准代码)。
- ASCII码采用7位二进制编码，可以表示 $2^7$ 即128个字符。



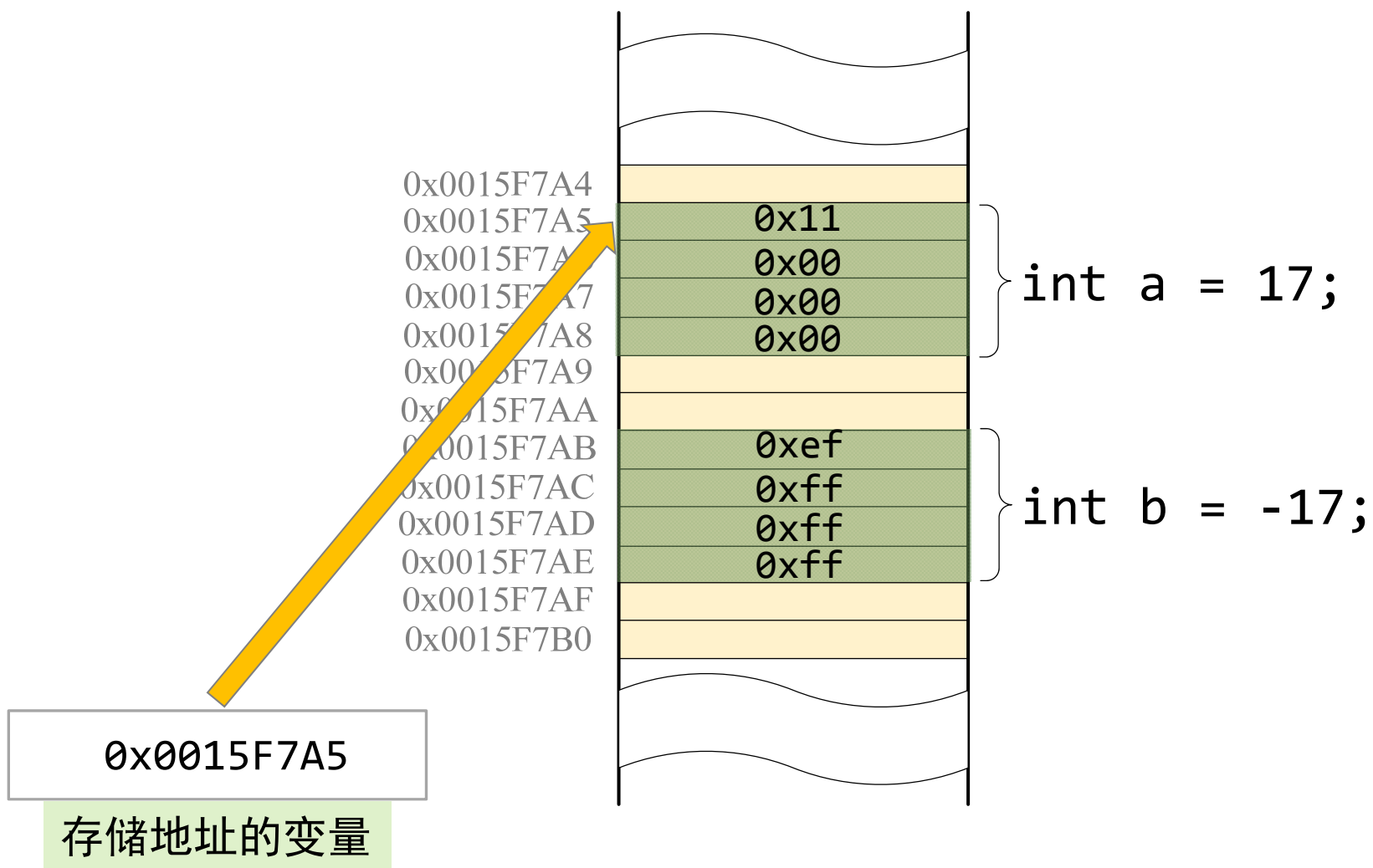
高位 低位	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	EXT	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	—	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

# 数据的存放与位置

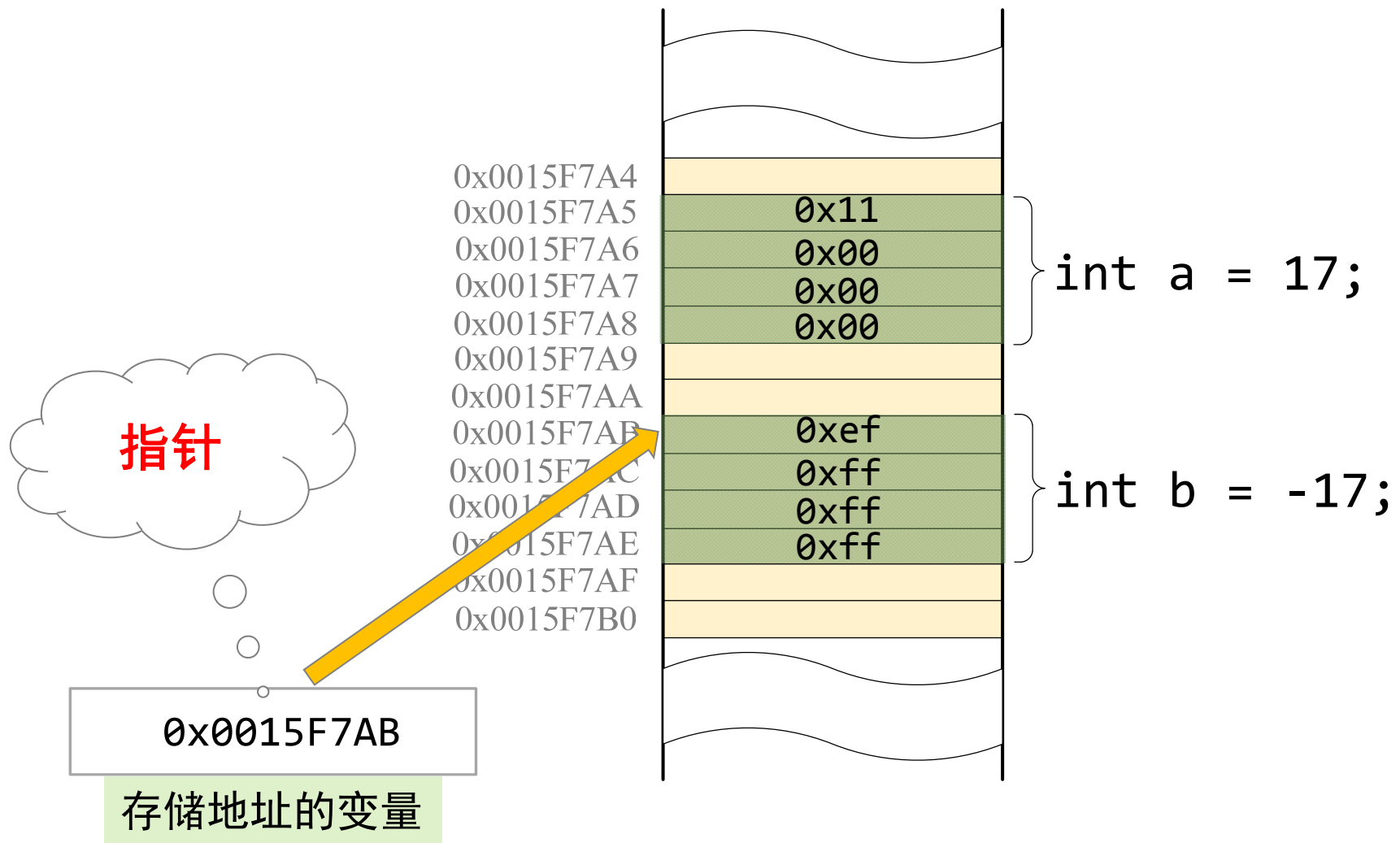


如果知道一个变量的地址，而且知道变量的类型，那么也可以获得该变量的值。

# 数据的存放与位置



# 数据的存放与位置



# 指针的含义



## 概念

指针就是用来存放地址的变量。和普通变量相比：

- 它的值是地址；
- 一个指针存放了某个变量的地址，通常就说这个指针指向了这个变量。
- 它的类型是它指向单元的数据的类型。

# 指针变量的定义

## 定义形式

类型说明符 \*变量名;

指针的类型是其指向  
单元数据的类型，不  
一定是int类型

int \*p1;

p1 → int

float \*p2;

p2 → float

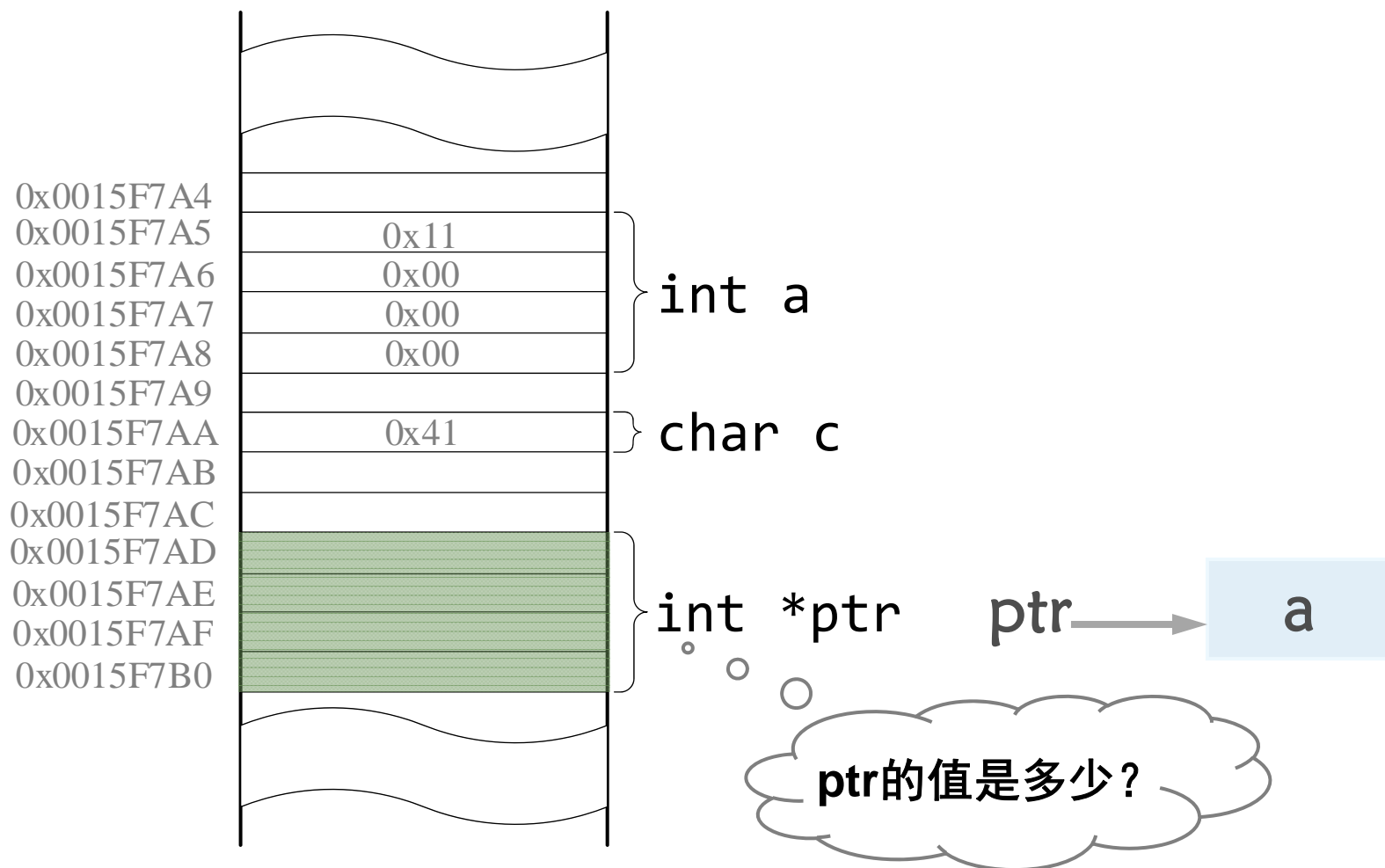
char \*p3;

p3 → char

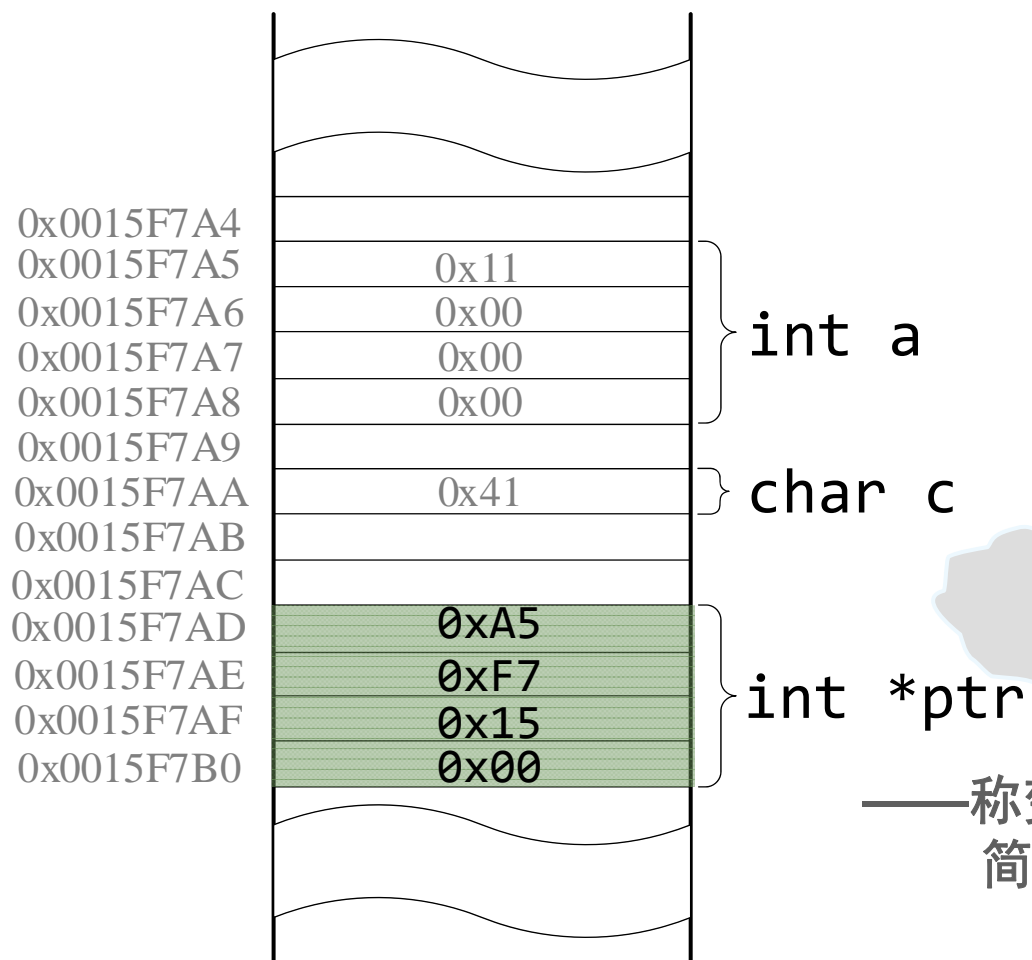
指针变量的存储空间是多大呢？



# 指针的含义

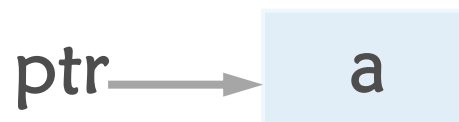


# 指针的含义

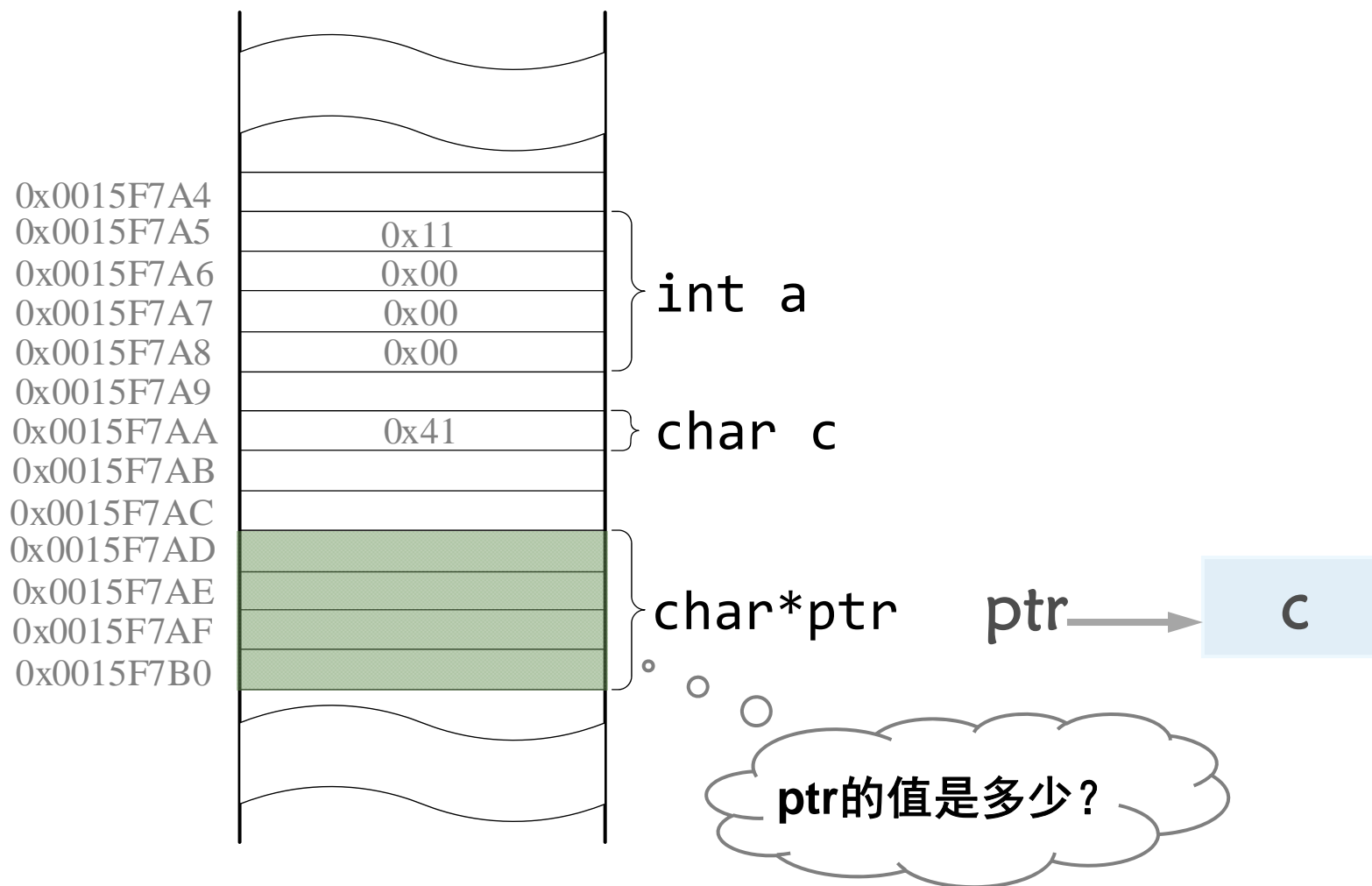


变量ptr的值是变量a的地址

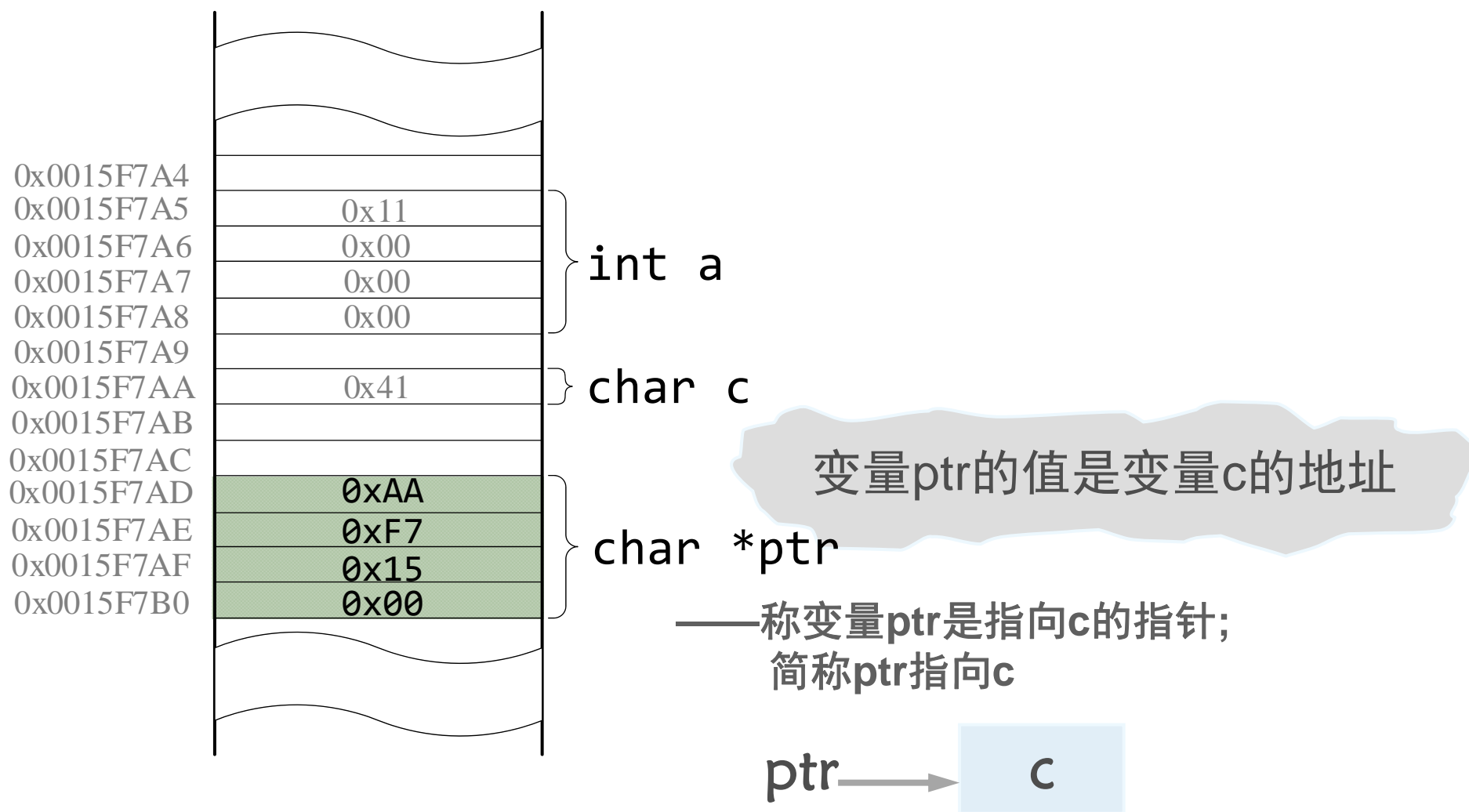
——称变量ptr是指向a的指针;  
简称ptr指向a



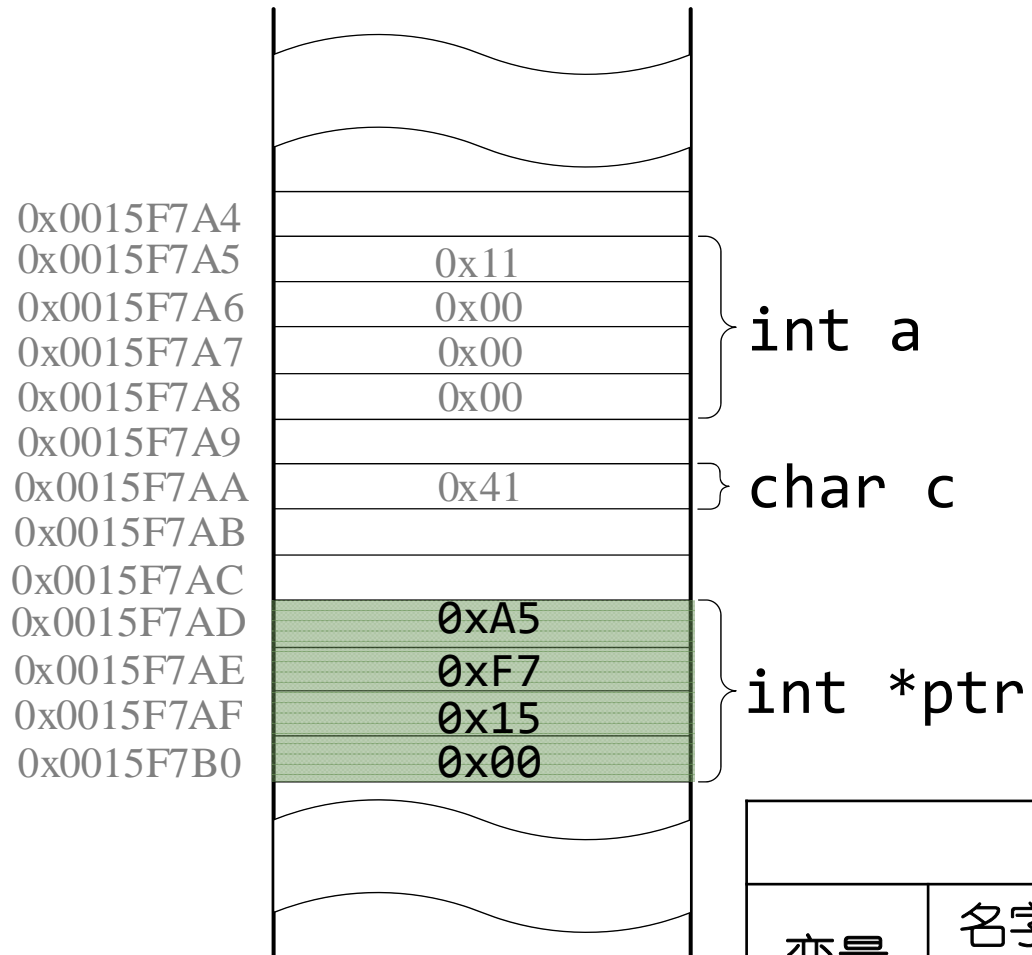
# 指针的含义



# 指针的含义



# 指针的含义



指针的类型是  
什么?

		普通变量	指针变量
变量的三要素	名字	标识符	标识符
	内容	数值	地址
	地址	内存单元编号	内存单元编号

## 与指针的有关运算符

运算符	名称	含义
&	取地址运算符	取普通变量的地址
*	指针运算符	引用指针变量指向的变量

【例10.1】通过指针变量访问整型变量。

```
1 #include<stdio.h>
2
3 int main() {
4     int a = 10;
5     int *ptr;
6
7     ptr = &a;
8     printf("a=%d\n", *ptr);
9
10    *ptr = 9;
11    printf("a=%d\n", a);
12 }
```

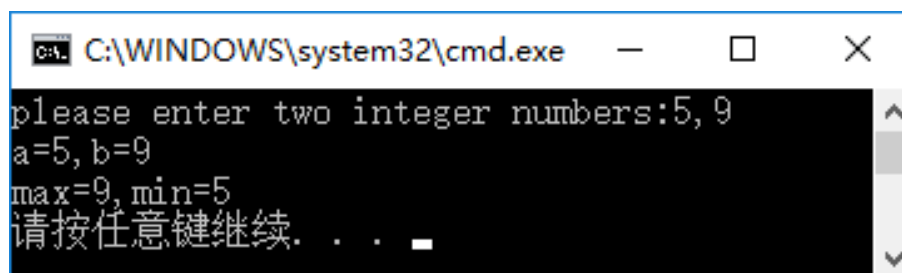
注意

指针变量要先赋值再使用

【例10.2】输入a和b两个整数，按先大后小的顺序输出a和b。

解题思路：不交换整型变量的值，而是交换两个指针变量的值（即a和b的地址）。

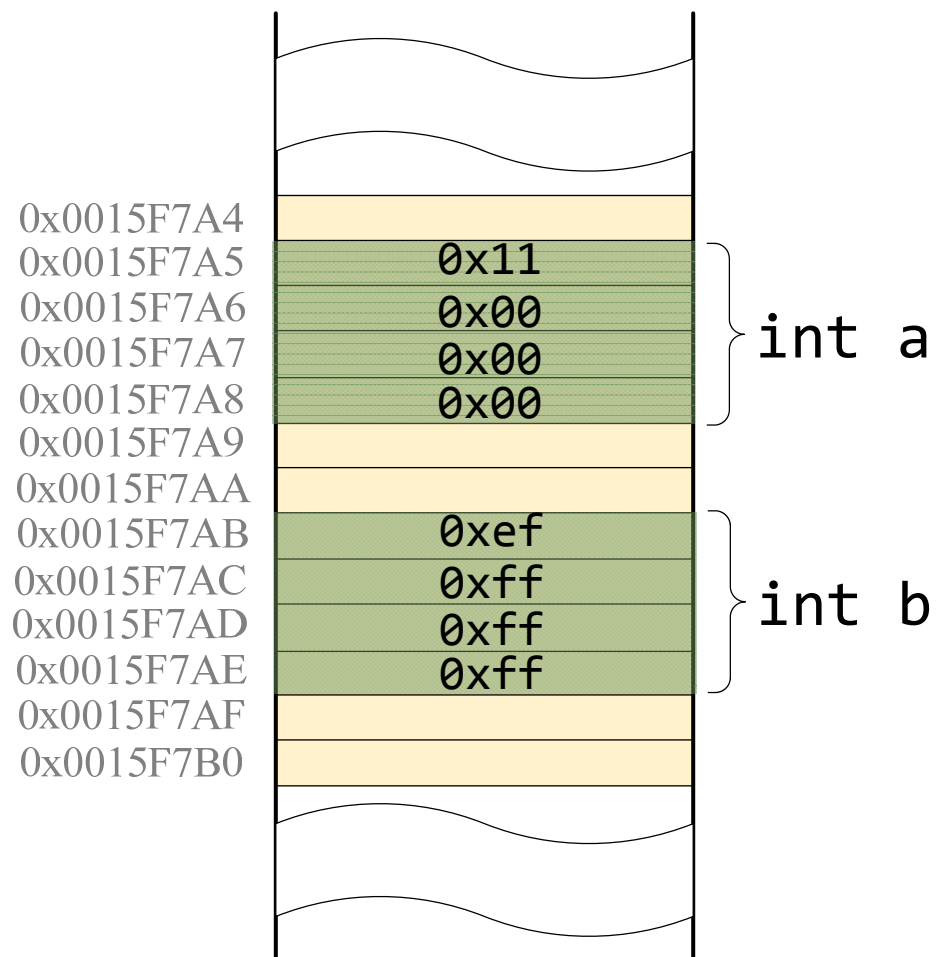
```
1 #include <stdio.h>
2 int main() {
3     int *p1,*p2,*p,a,b;
4     printf("please enter two integer numbers:");
5     scanf("%d,%d",&a,&b); // 输入两个整数
6     p1=&a; // 使p1指向变量a
7     p2=&b; // 使p2指向变量b
8     if (a<b) { // 如果a<b,使p1与p2的值互换
9         p=p1;
10        p1=p2;
11        p2=p;
12    }
13    printf("a=%d,b=%d\n",a,b); // 输出a,b
14    printf("max=%d,min=%d\n",*p1,*p2); // 输出p1和p2所指向的变量的值
15    return 0;
16 }
```



```
C:\WINDOWS\system32\cmd.exe
please enter two integer numbers:5,9
a=5,b=9
max=9,min=5
请按任意键继续. . .
```



【例10.3】利用指针实现两个整型变量值交换的函数，在主函数中调用该函数



【例10.3】利用指针实现两个整型变量值交换的函数，在主函数中调用该函数

```
1 #include <stdio.h>
2
3 void swap(int *pa, int *pb) {
4     int tmp;
5
6     tmp = *pa;
7     *pa = *pb;
8     *pb = tmp;
9 }
10
11 int main() {
12     int a, b;
13     int *p1, *p2;
14
15     scanf("%d %d", &a, &b);
16     printf("before swapping a = %d, b = %d\n", a, b);
17     p1 = &a;
18     p2 = &b;
19     swap(p1, p2);
20     printf("after swapping a = %d, b = %d\n", a, b);
21 }
```