CS 208

M, 19 Jan 2026

```
char ch = 0xC3;

printf("0x%x\n", ch);
```

Memory

1 byte

1100 0011

① %x expects an int

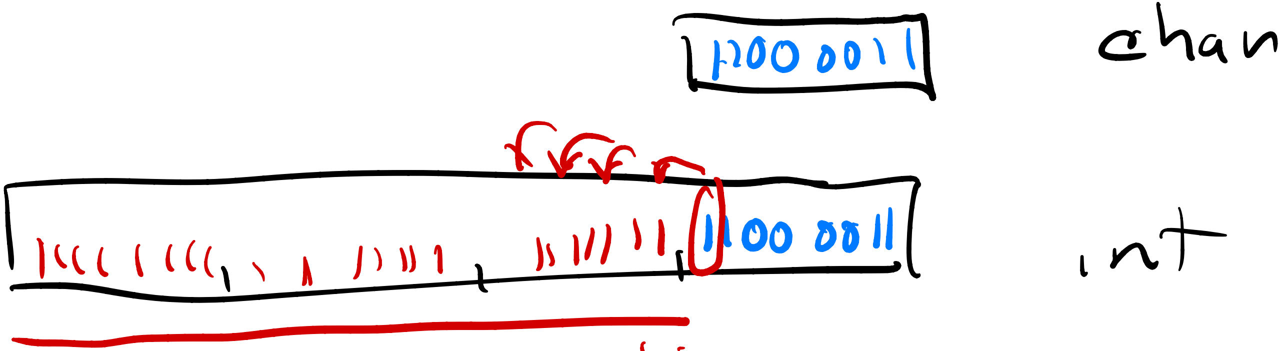
② C does a lot of automatic type conversion

char → int

doesn't discard data (because int is 4 bytes + char is 1, so OK)

```
char ch = 0xC3;
```

compiler wants to convert it to an int



because int is a "signed" type, this conversion does "sign extension"

```
int n = 17;
int *p = &n;
```

"address of n"

"address of an int"

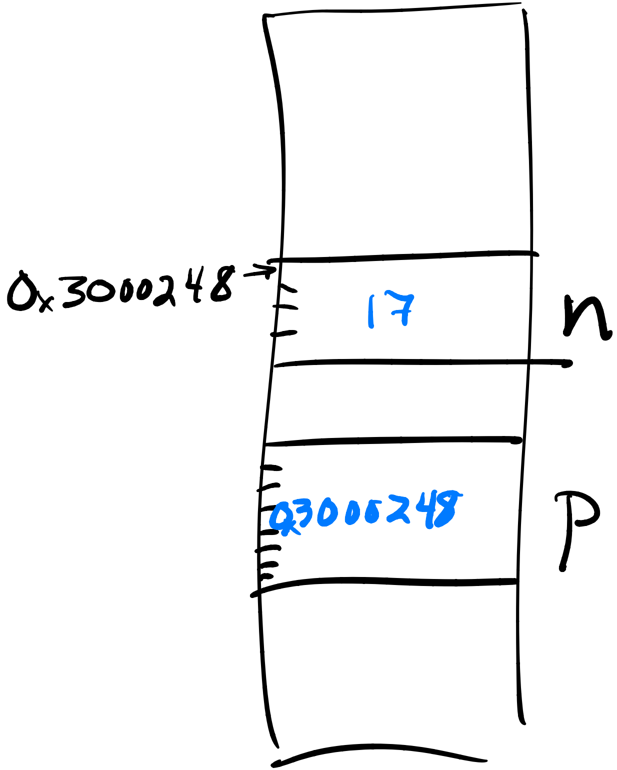
or

"pointer to an int"

or

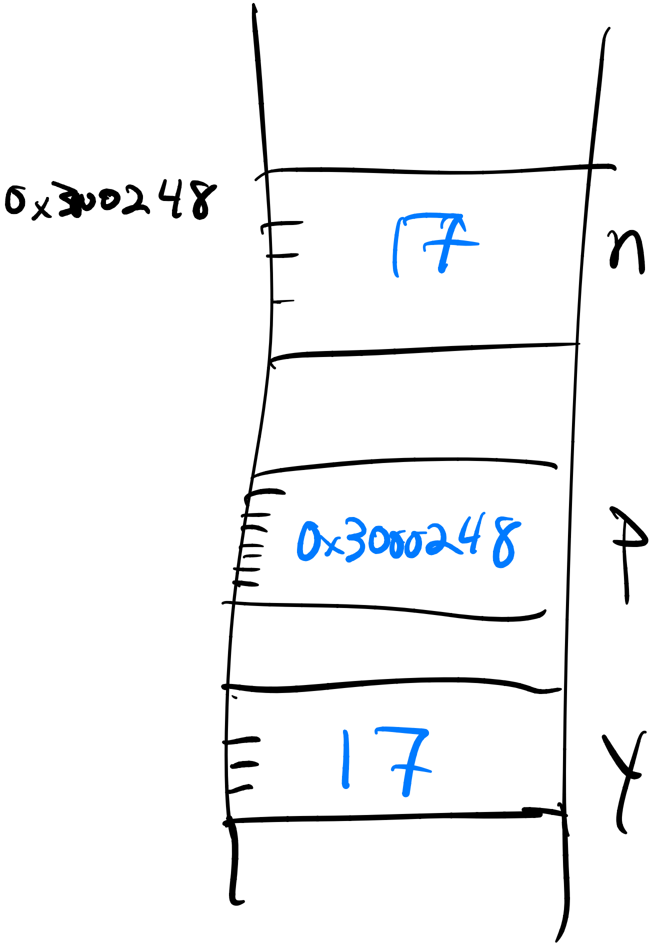
"int pointer"

the type of p

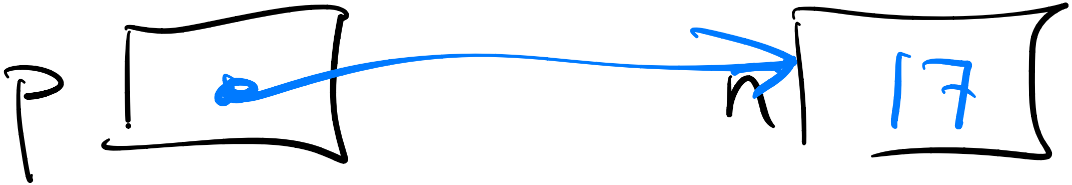


(pointers are 8 bytes long on 64-bit systems)

```
int n = 17;
int *p = &n;
int y = *p;
```

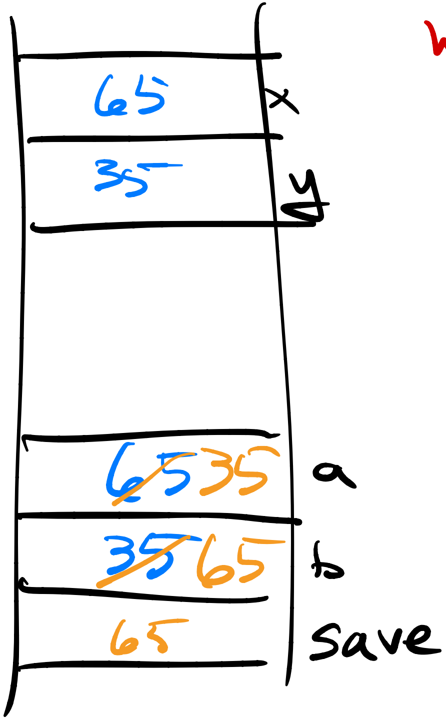


```
int n = 17;
int *p = &n;
```



```
void bad_swap(int a, int b) {
        int save = a;
        a = b;
        b = save;
}
```

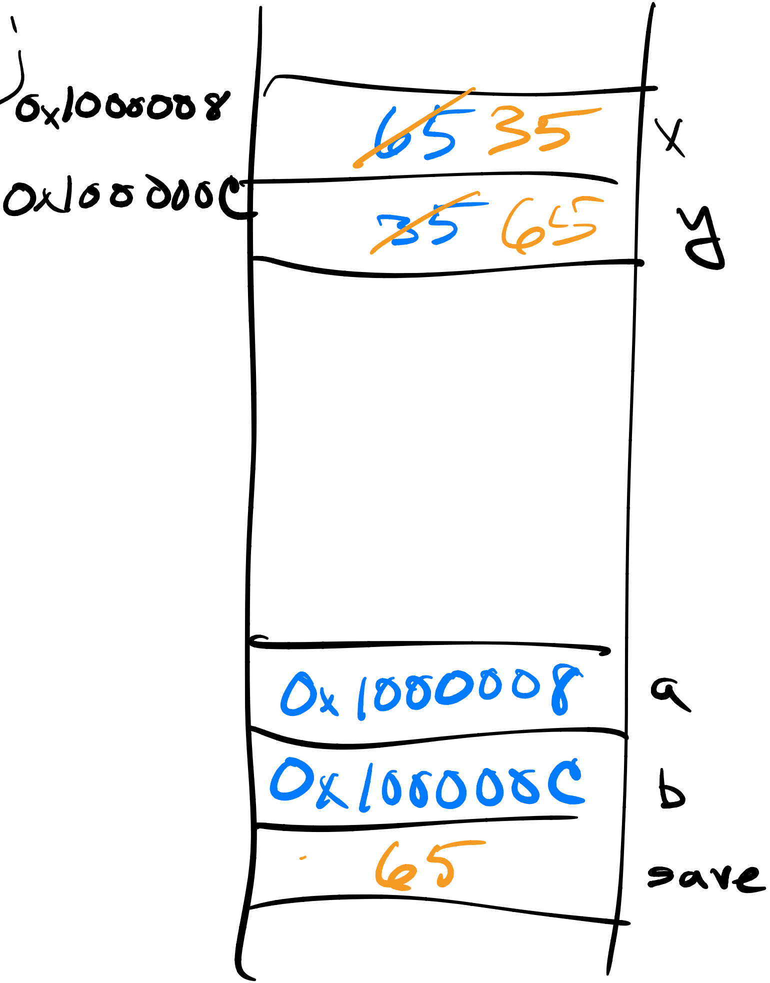
```
int main() {
      int x = 65;
      int y = 35;
   badswap(x, y);
}
```



when you call badswap, x + y get **copied** to a + b

```
swap(&x, &y);
```

```
save = *a;
*a = *b;
*b = save;
```


0x1000008
~~65~~ 35
x
0x100000C
~~35~~ 65
y
0x1000008
a
0x100000C
b
65
save