Type Casting

Converting an expression of a given data type into another
Data type

Type Casting: Implicit

```
int i = 3;
float p;
p = i;
printf("%f", p);
```

Type Casting: Implicit

```
int i = 3;
float p;
p = i;
printf("%f", p);
```

Type Casting: Implicit

```
int i = 3;
float p;
p = i;
printf("%f", p);
```

Type Casting: Implicit

```
int i = 3;
float p;
p = i;
printf("%f", p);
```


```
3.000000
```

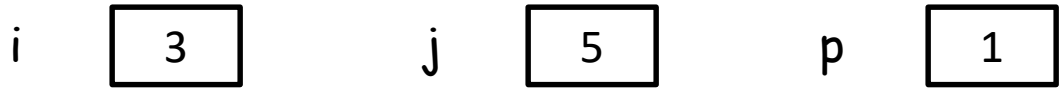
Type Casting: Implicit

```
int i = 3;
int p;
p = i;
printf("%d", p);
```


```
3
```

Type Casting: Explicit

```
int i = 3, j = 5;
float p;
p =  j / i;
printf("%f", p);
```




```
1.000000
```

Type Casting: Explicit

(data type) expression

```
int i = 3, j = 5;
float p;
p = (float) j / i;
printf("%f", p);
```

i	3	j	5	p	1.666667

1.666667

Type Casting: Explicit

(data type) expression

```
int i = 3, j = 5;
float p;
p = j / (float) i;
printf("%f", p);
```

i	3	j	5	p	1.666667

1.666667

Type Casting: Explicit

(data type) expression

```
int i = 3, j = 5;
float p;
p = (float) j / (float) i;
printf("%f", p);
```

i	3	j	5	p	1.666667

```
1.666667
```

Type Casting: Explicit

(data type) expression

```
int i = 3, j = 5;
float p;
p = (float) (j / i);
printf("%f", p);
```

i	3	j	5	p	1.666667

```
1.666667
```

Logical Expression

Produces 0(false) /1(true)

Relational & Equality Operator

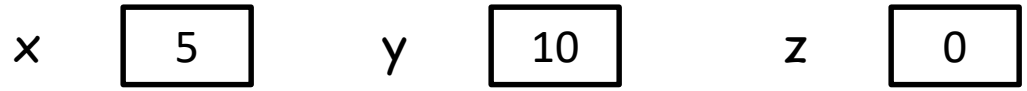
C equality or relational operators	Example of C condition	Meaning of C condition
>	x > y	x is greater than y
<	x < y	x is less than y
>=	x >= y	x is greater equal than y
<=	x <= y	x is less equal than y
==	x == y	x is equal to y
!=	x != y	x is not equal to y

Logical Expression

```
int x = 5, y = 10;
int z = x > y;
printf("%d", z);
```

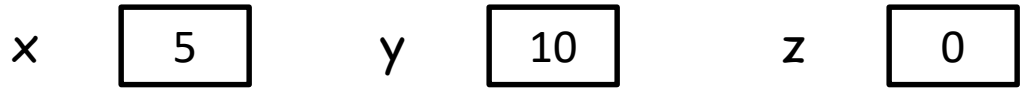
Logical Expression

```
int x = 5, y = 10;
int z = x > y;
printf("%d", z);
```



Logical Expression

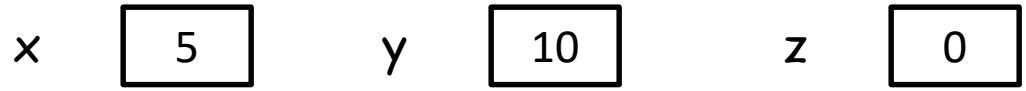
```
int x = 5, y = 10;
int z = x > y;
printf("%d", z);
```




```
0
```

Logical Expression

```
int x = 5, y = 10;
int z = x == y
printf("%d", z);
```



Logical Operator

Operator	Name
!	Logical not
&&	Logical and
||	Logical or

Logical Operator

p	!p
0	1
1(Non zero)	0

Truth table of logical not

p	q	P && q
0	0	0
0	1(Non zero)	0
1(Non zero)	0	0
1(Non zero)	1(Non zero)	1

Truth table of logical and

Logical Operator

p	q	P || q
0	0	0
0	1(Non zero)	1
1(Non zero)	0	1
1(Non zero)	1(Non zero)	1

Truth table of logical or

Logical Operator

```
int x = 5, y = 10;
int z1 = x == y;
int z2 = x >= y;
int z = z1 && z2;
printf("%d", z);
```

x 5 y 10

Logical Operator

```
int x = 5, y = 10;
int z1 = x == y;
int z2 = x >= y;
int z = z1 && z2;
printf("%d", z);
```

x	5	y	10	z1	0

Logical Operator

```
int x = 5, y = 10;
int z1 = x == y;
int z2 = x >= y;
int z = z1 && z2;
printf("%d", z);
```

x	5	y	10	z1	0	z2	0

Logical Operator

```
int x = 5, y = 10;
int z1 = x == y;
int z2 = x >= y;
int z = z1 && z2;
printf("%d", z);
```

x	5	y	10	z1	0	z2	0	z	0

Logical Operator

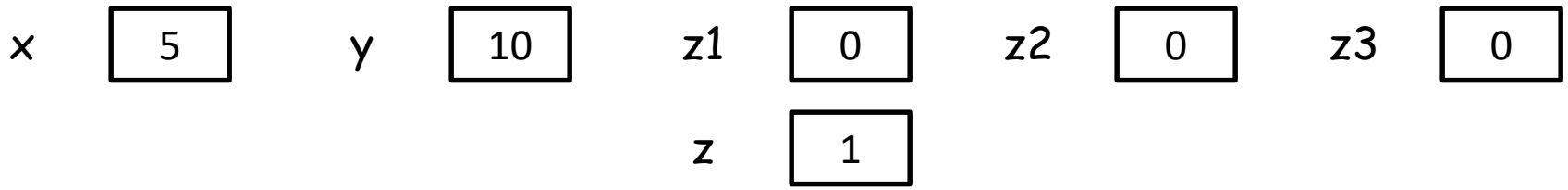
```
int x = 5, y = 10;
int z1 = x == y;
int z2 = x >= y;
int z = z1 && z2;
printf("%d", z);
```

x	5	y	10	z1	0	z2	0	z	0

```
0
```

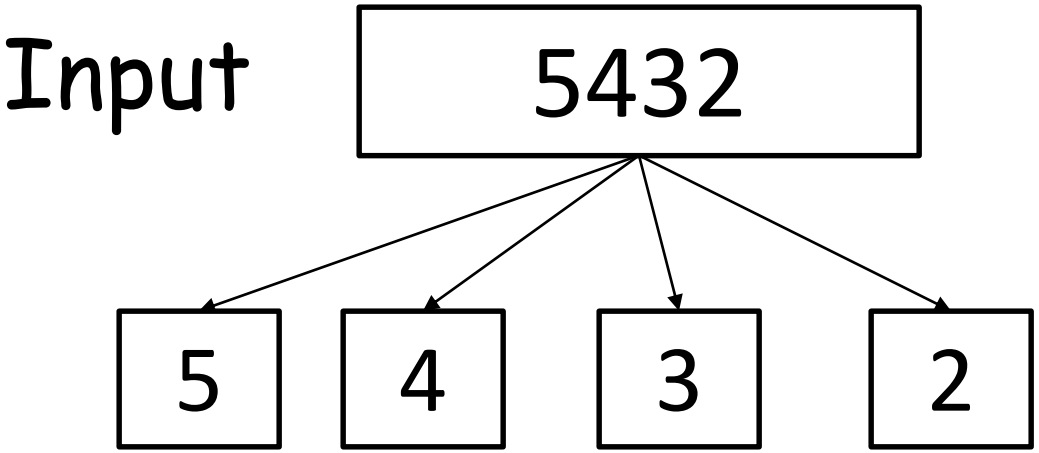
Logical Operator

```
int x = 5, y = 10;
int z1 = x == y;
int z2 = x >= y;
int z3 = z1 && z2;
int z = !z3
printf("%d", z);
```




```
1
```

Separate digits



Separate digits

```
int num;
scanf("%d", &num);
```

Task1:

Input a number

-> 5432

Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
```

Task2:

10) 5432 (543

5430

2

Separate digits

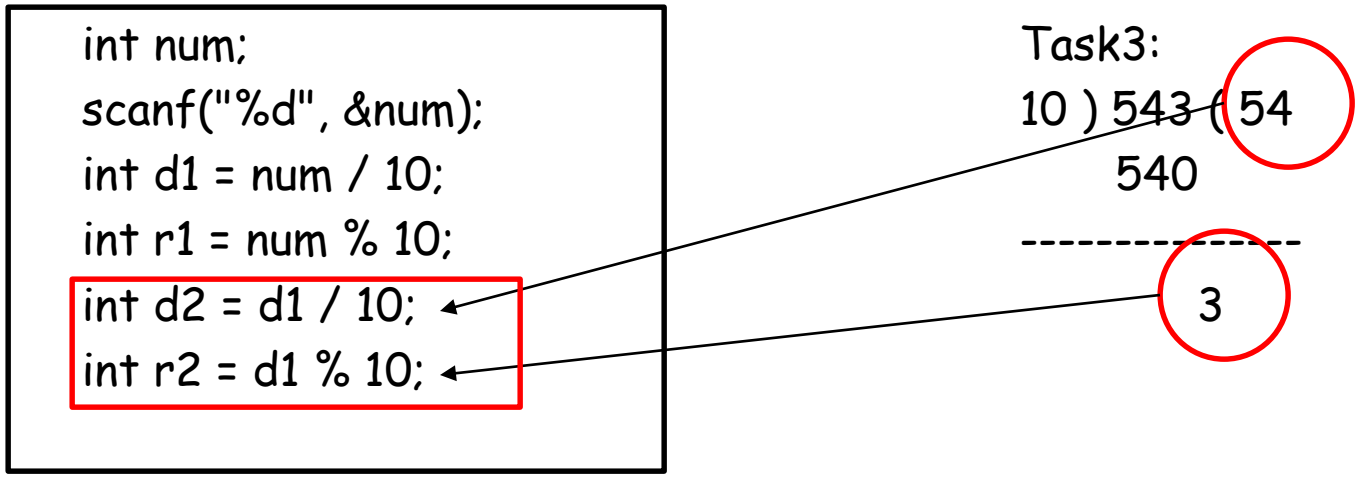
```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
```

Task3:

10) 543 (54

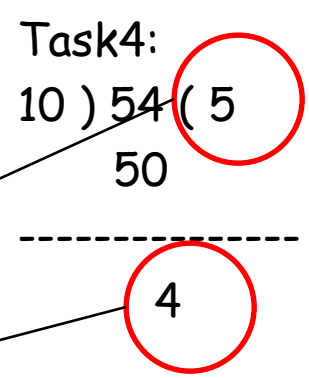
540

3



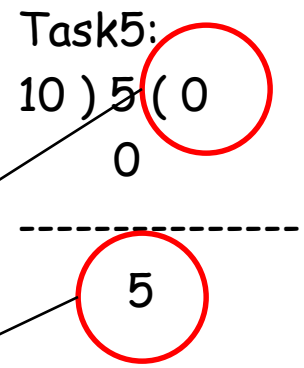
Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
```



Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```



Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
printf("%d %d\n", d1, r1);
int d2 = d1 / 10;
int r2 = d1 % 10;
printf("%d %d\n", d2, r2);
int d3 = d2 / 10;
int r3 = d2 % 10;
printf("%d %d\n", d3, r3);
int d4 = d3 / 10;
int r4 = d3 % 10;
printf("%d %d\n", d4, r4);
printf("%d %d %d %d", r1, r2, r3, r4);
```

Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```

num 

Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```

num | 5432

Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```

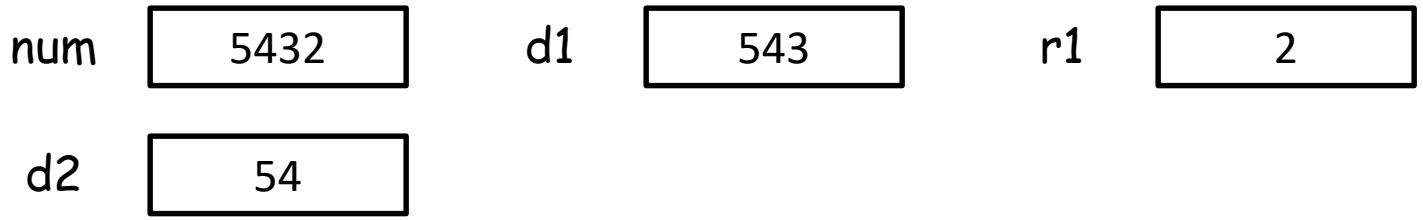
num | 5432 | d1 | 543

Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```

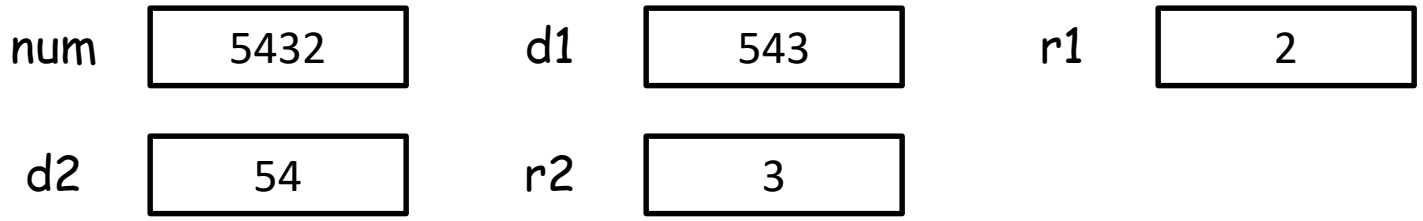
Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```



Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```



Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```

num	5432	d1	543	r1	2
d2	54	r2	3	d3	5

Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```

num	5432	d1	543	r1	2
d2	54	r2	3	d3	5
r3	4				

Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```

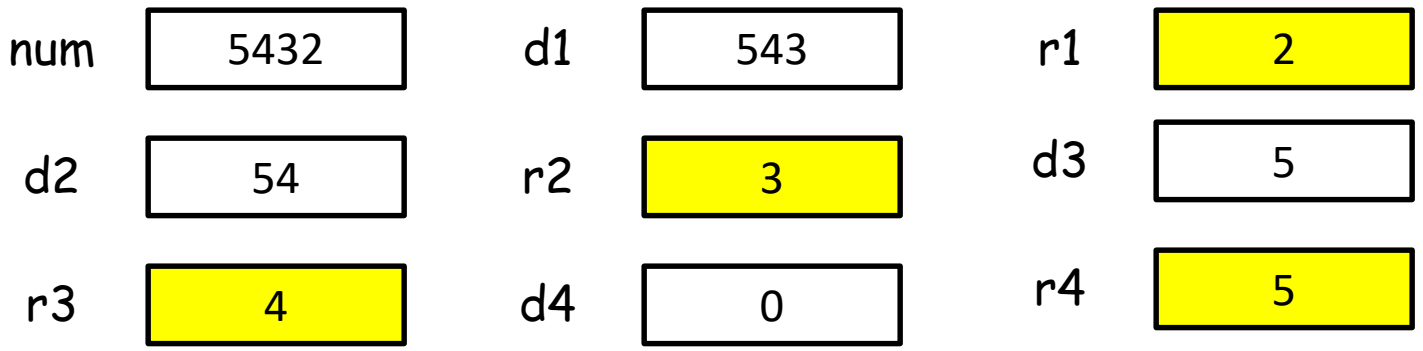
num	5432	d1	543	r1	2
d2	54	r2	3	d3	5
r3	4	d4	0		

Separate digits

```
int num;
scanf("%d", &num);
int d1 = num / 10;
int r1 = num % 10;
int d2 = d1 / 10;
int r2 = d1 % 10;
int d3 = d2 / 10;
int r3 = d2 % 10;
int d4 = d3 / 10;
int r4 = d3 % 10;
```

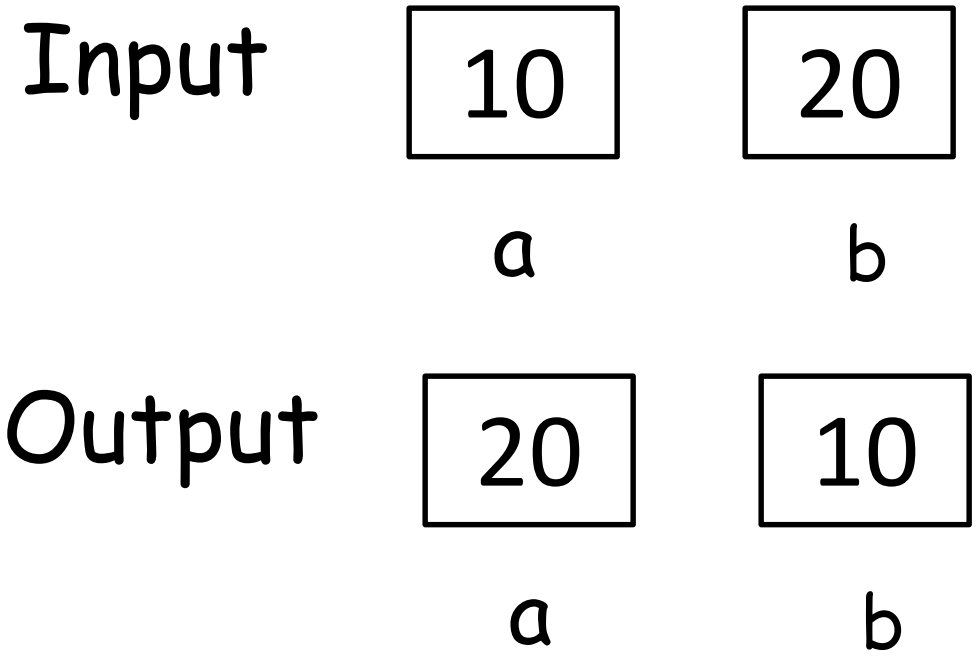
num	5432	d1	543	r1	2
d2	54	r2	3	d3	5
r3	4	d4	0	r4	5

Separate digits



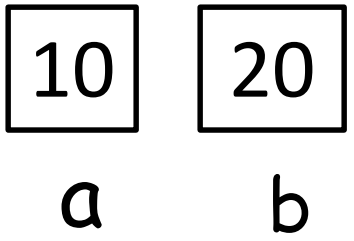

```
printf("%d %d %d %d", r1, r2, r3, r4)
```

Swap 2 variables



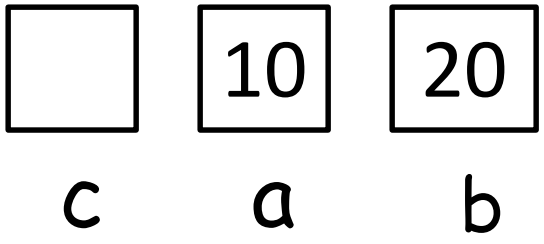
Swap 2 variables

```
int a, b;
scanf("%d %d ", &a, &b);
```



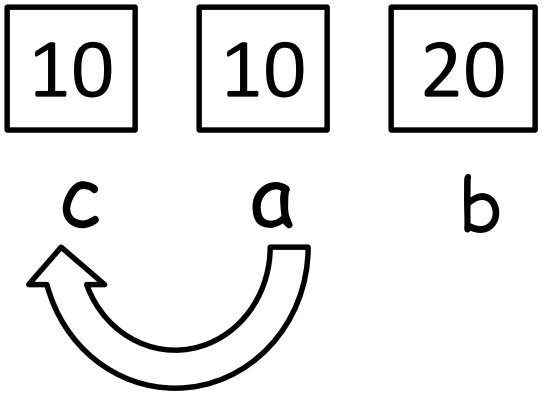
Swap 2 variables

```
int a, b;
scanf("%d %d ", &a, &b);
int c;
```



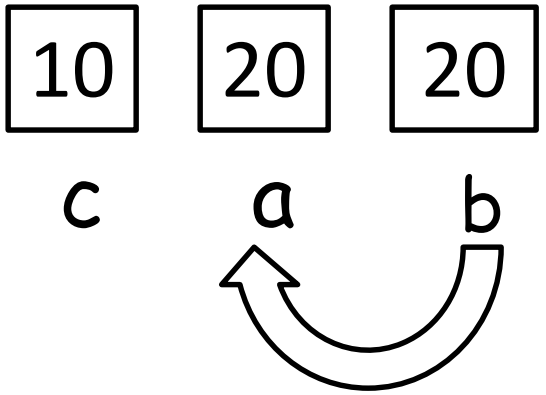
Swap 2 variables

```
int a, b;
scanf("%d %d ", &a, &b);
int c;
c = a;
```



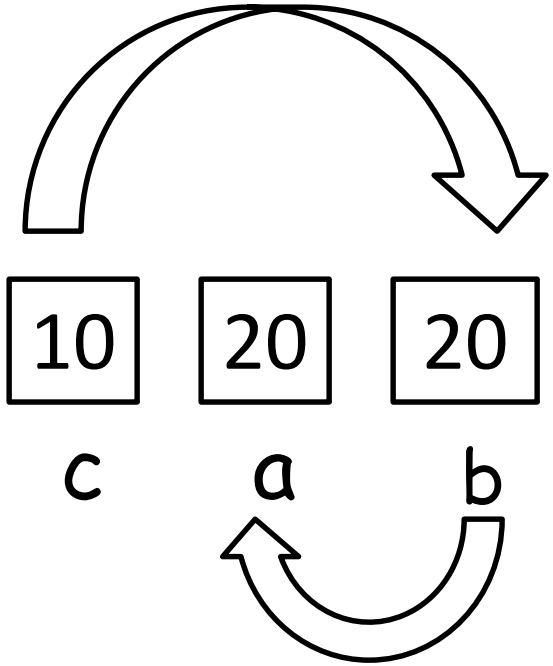
Swap 2 variables

```
int a, b;
scanf("%d %d ", &a, &b);
int c;
c = a;
a = b;
```

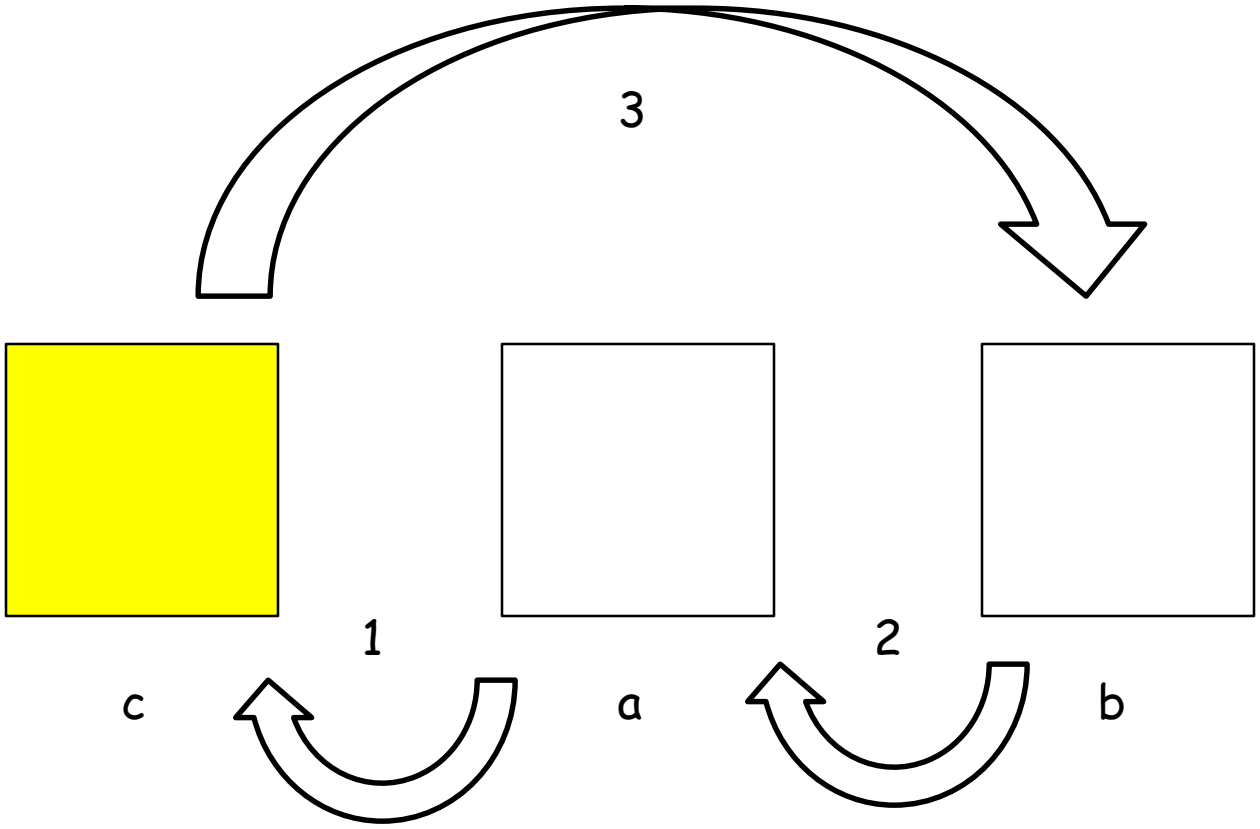


Swap 2 variables

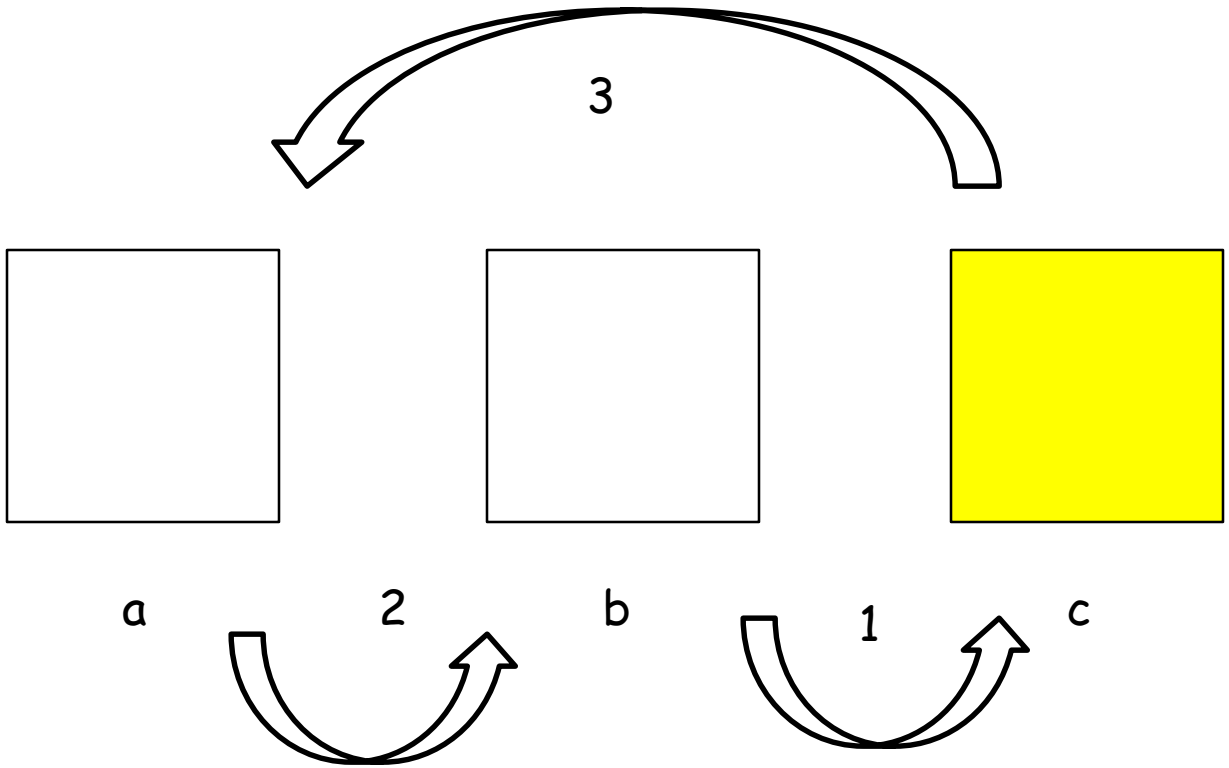
```
int a, b;
scanf("%d %d ", &a, &b);
int c;
c = a;
a = b;
b = c;
```



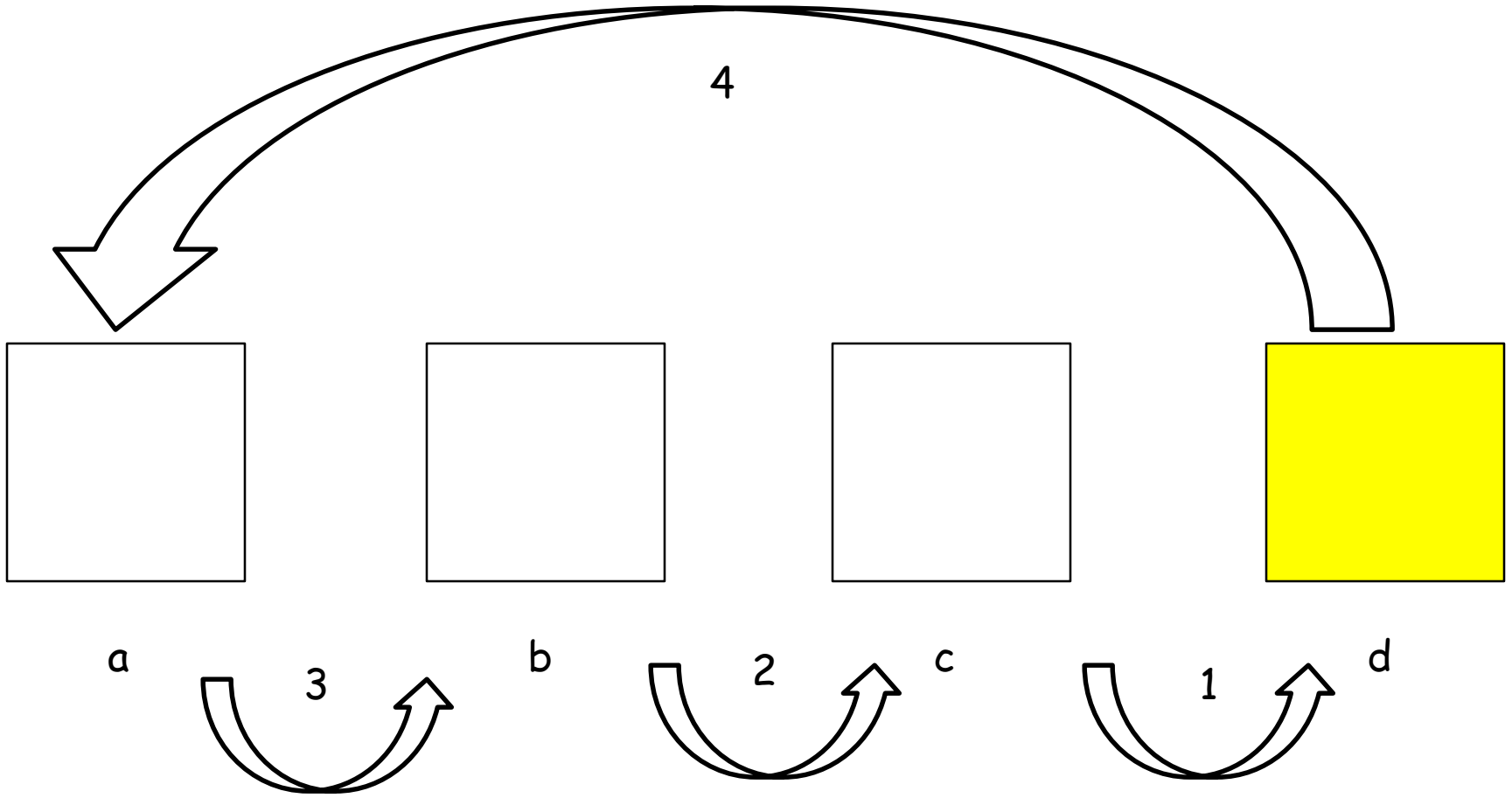
Swap 2 variables: left cyclic shift



Swap 2 variables: right cyclic shift



Swap 3 variables: right cyclic shift



Statements: Review

Return Statements:

```
return 0
```

Expression Statements:

```
b = 3;
a = a + b;
b++ ;
b > 0;
b > 0 && b < 10;
```