Pointers

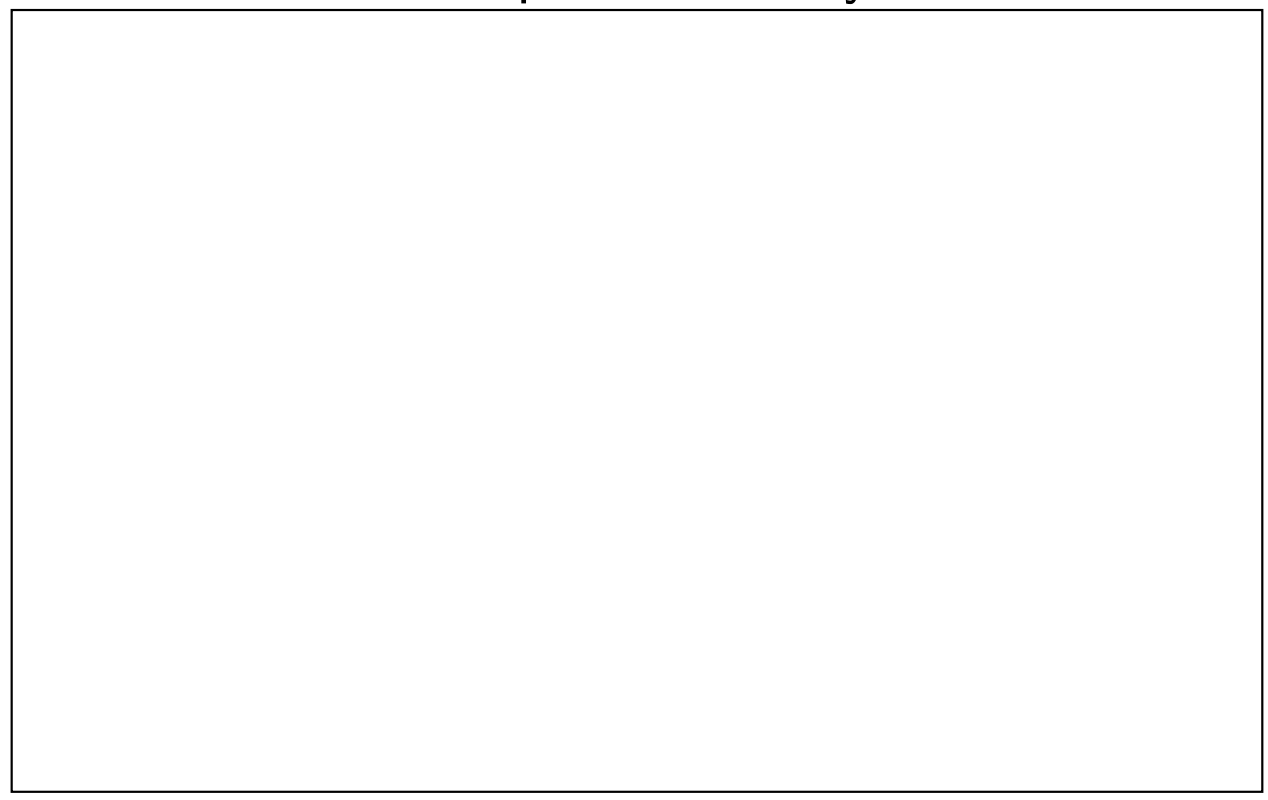
Deepanjan Kesh

```
int a;
```

```
int a;
```

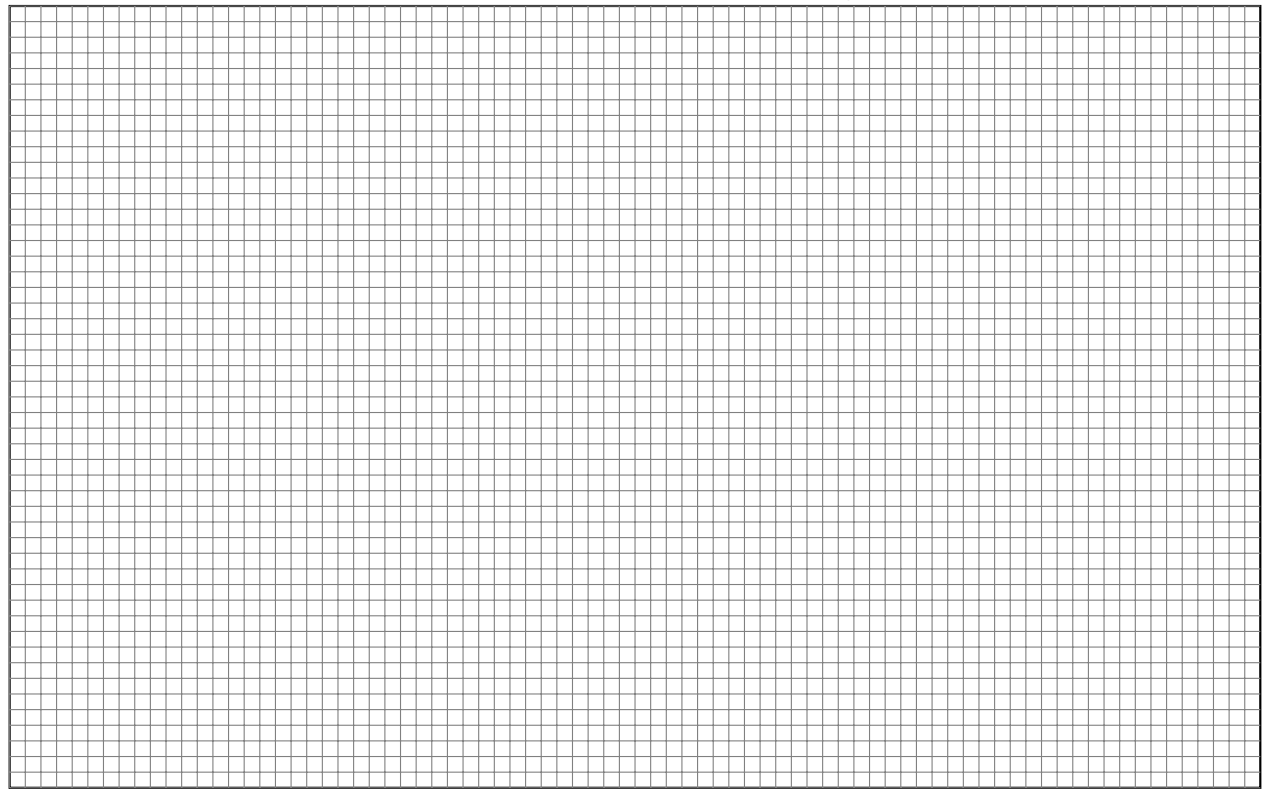
Computer Memory

```
int a;
```



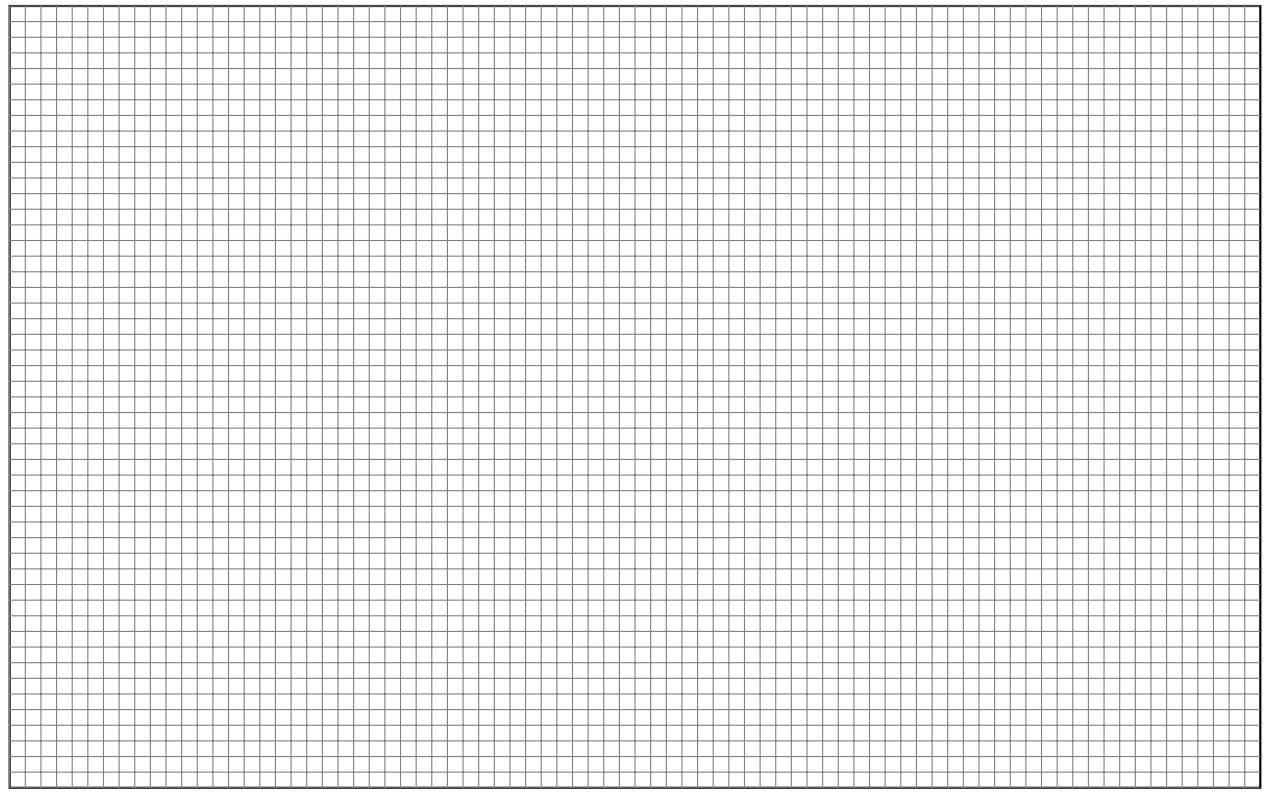
```
int a;
```

Computer Memory



```
int a;
```

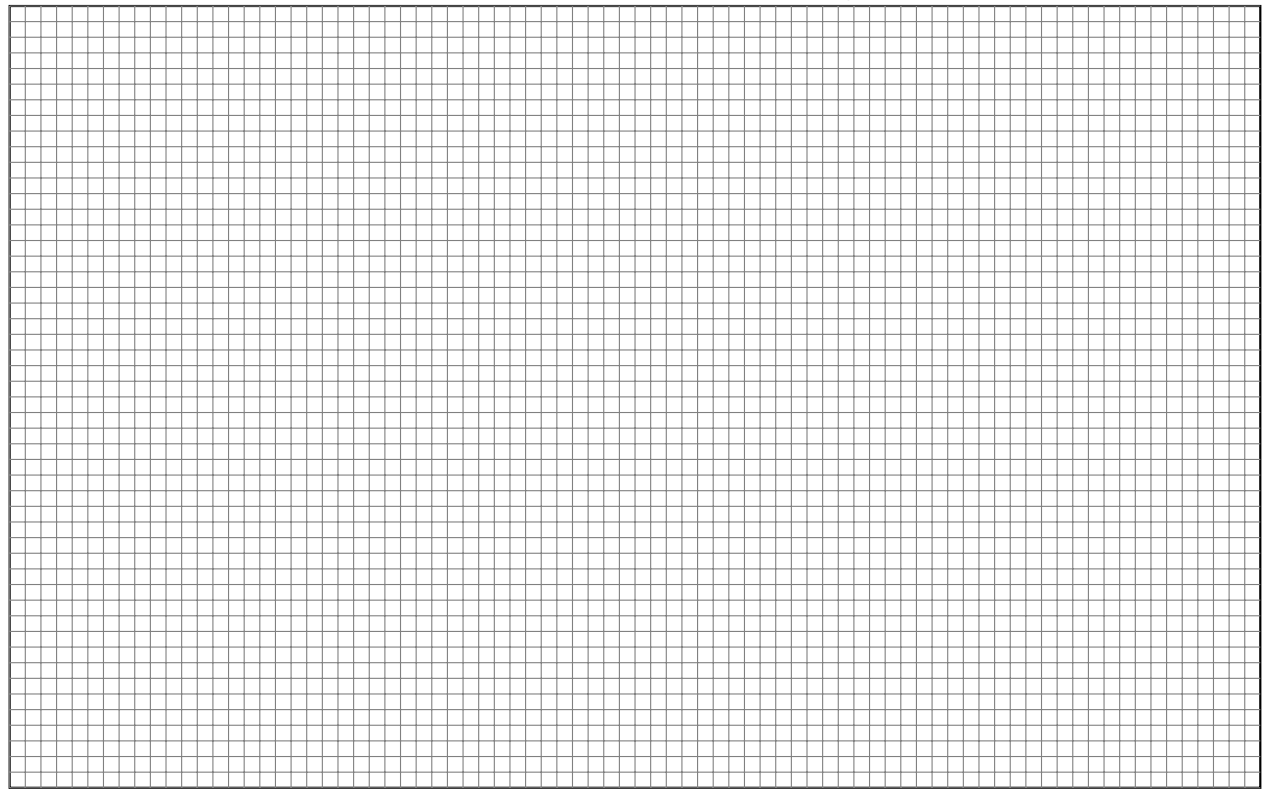
Computer Memory



256 GB

```
int a;
```

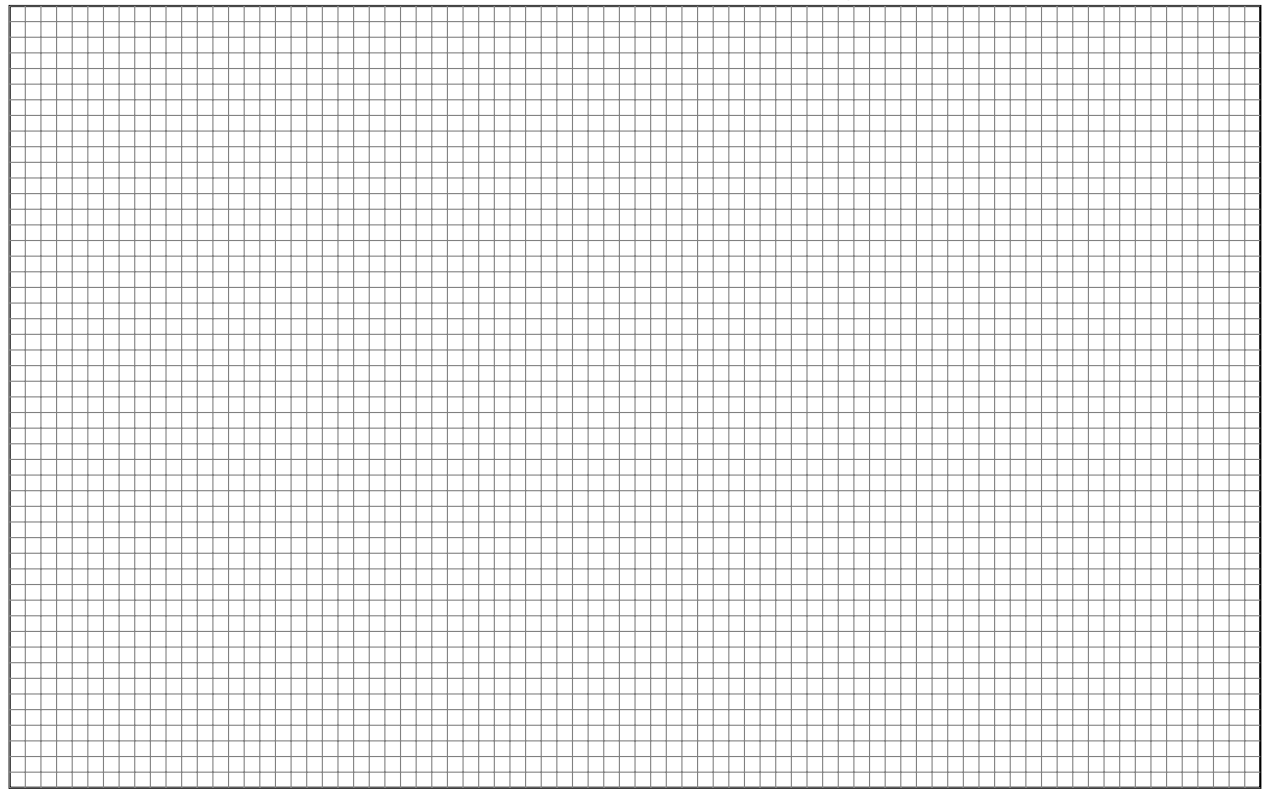
Computer Memory



256 GB = 34359738368 bytes

```
int a;
```

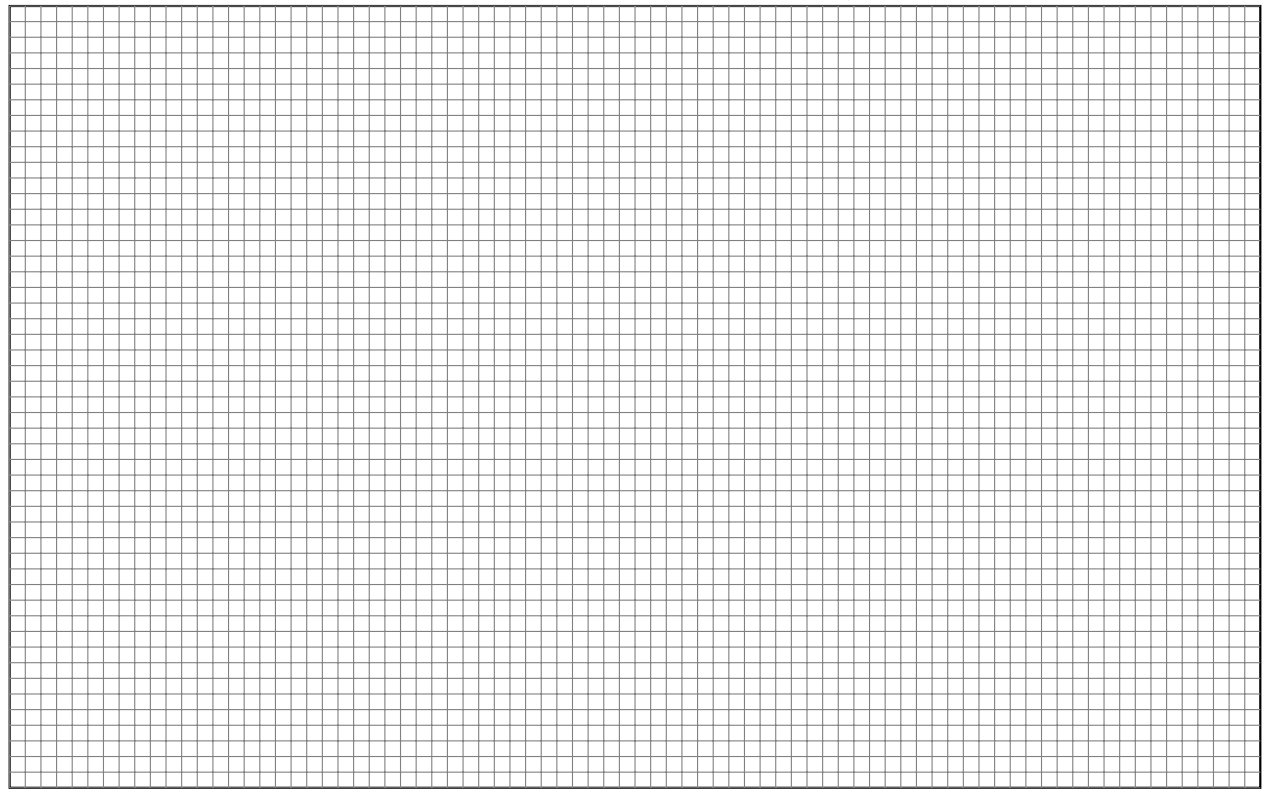
Computer Memory



256 GB = 34359738368 squares

```
int a;
```

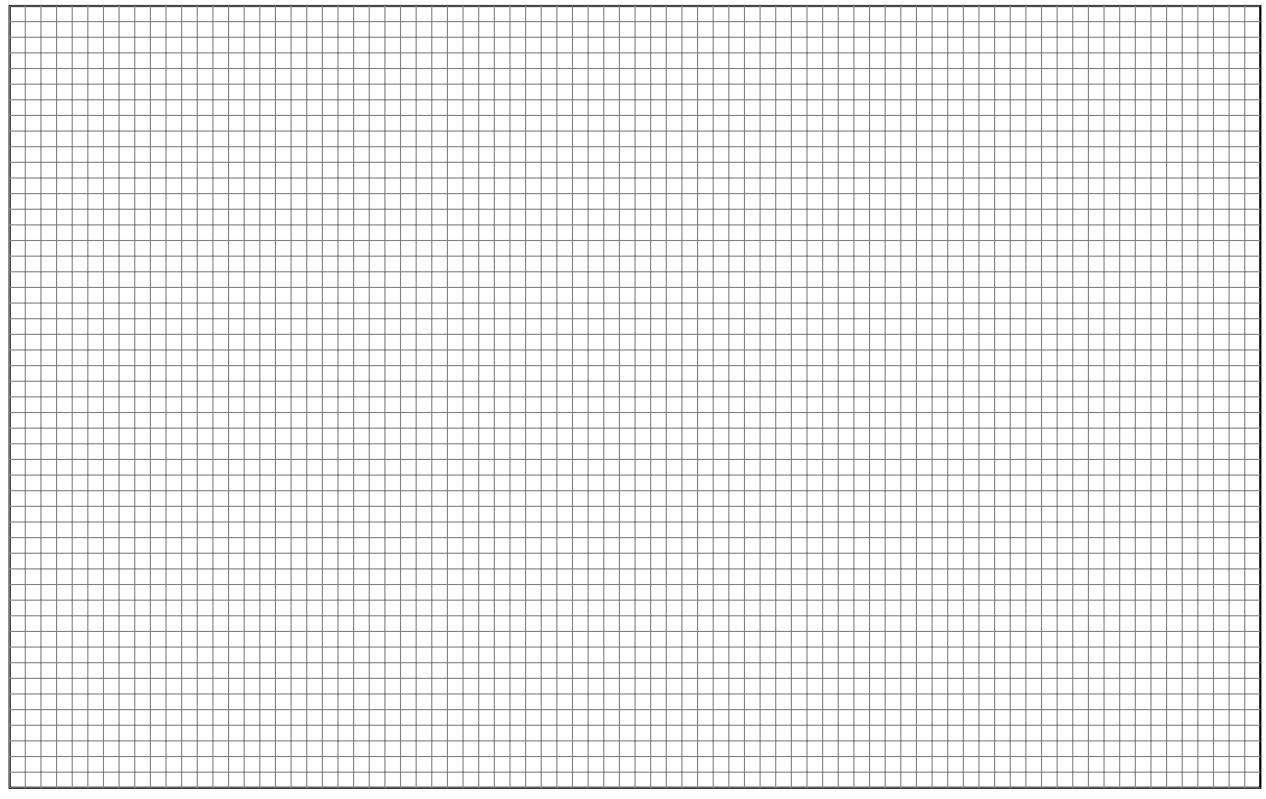
Computer Memory



256 GB = 34359738368 unique squares

```
int a;
```

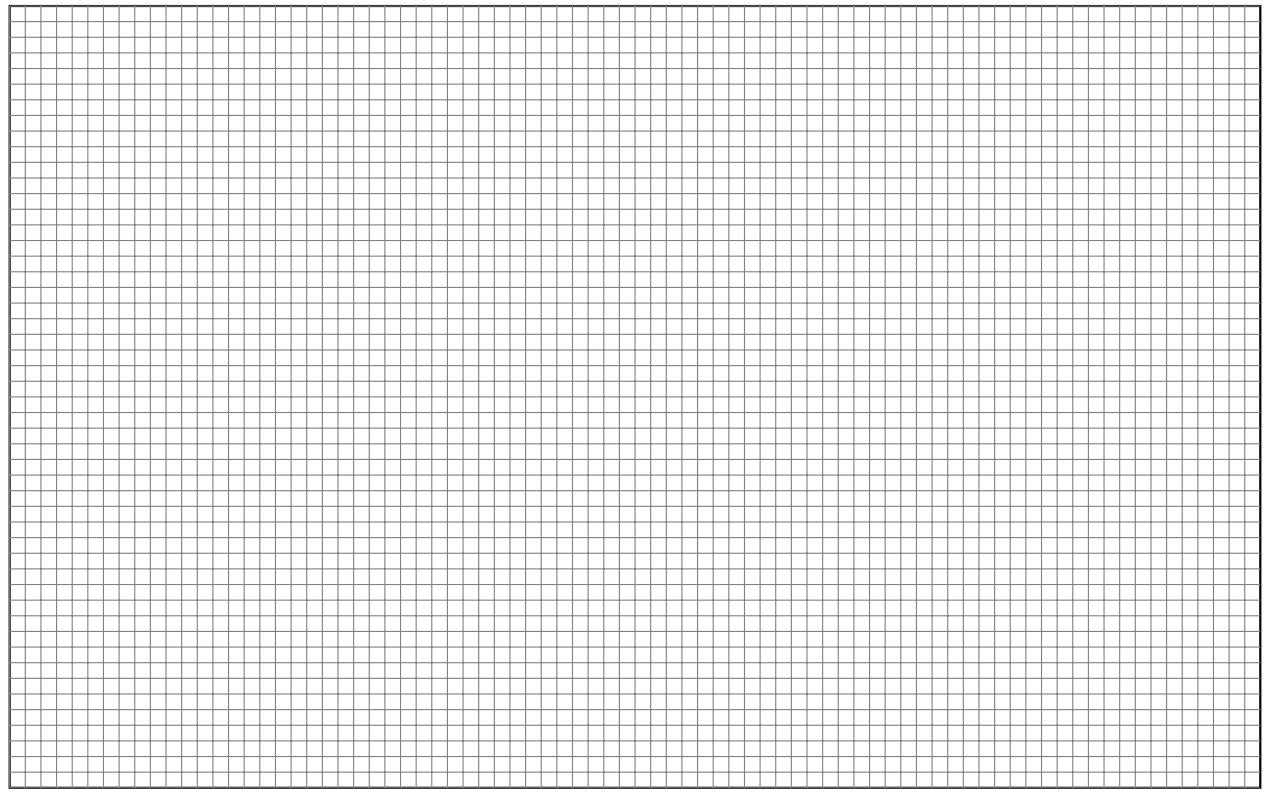
Computer Memory



Assign a unique address to each byte

```
int a;
```

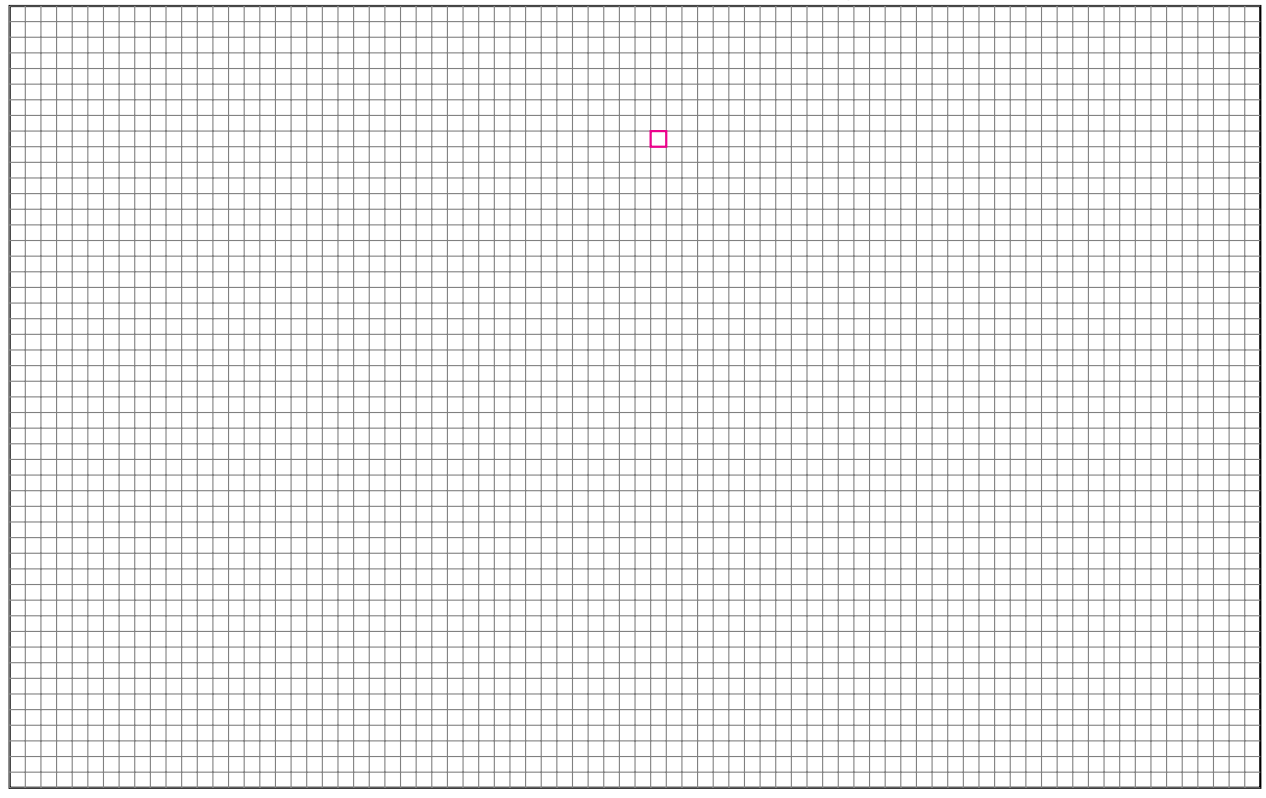
Computer Memory



Assign a unique address to each byte
Numbers from 0 to 34359738367

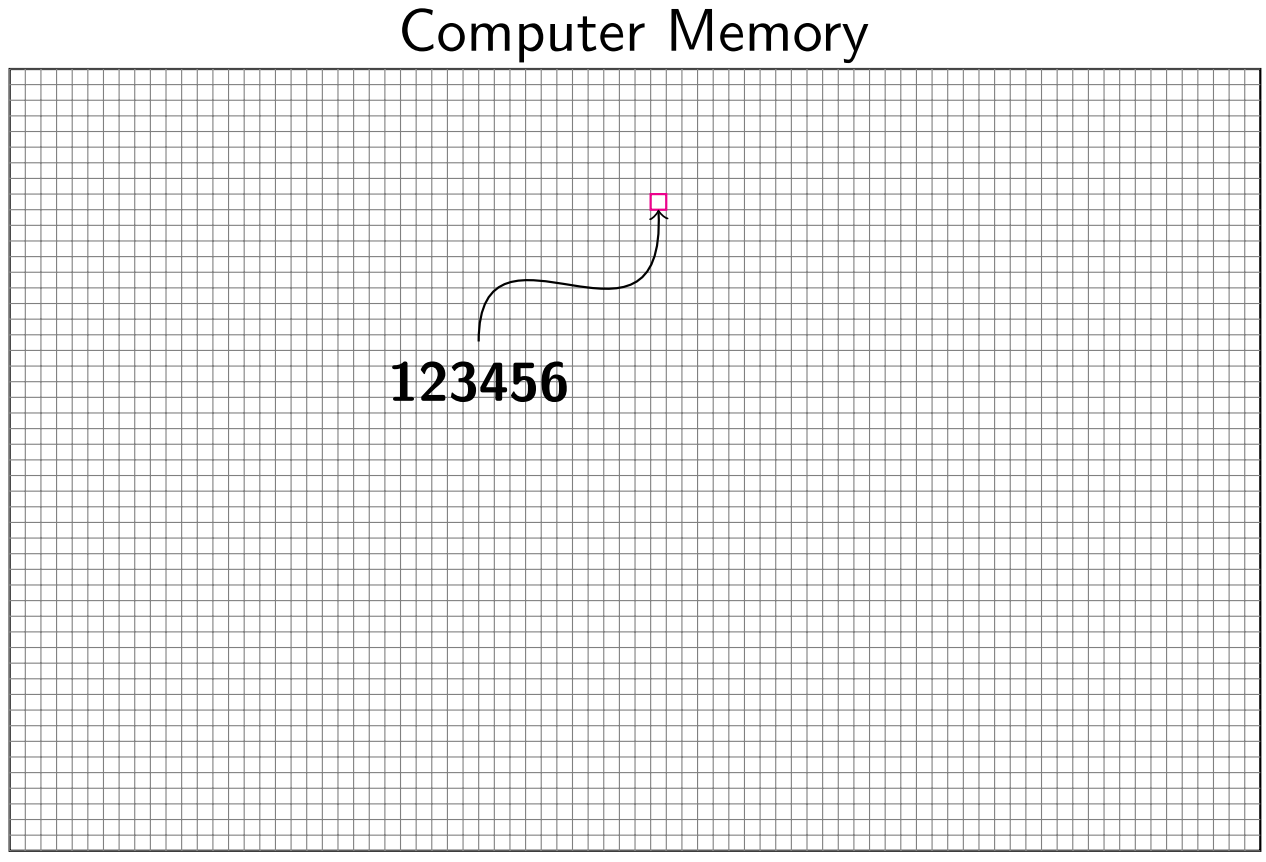
```
int a;
```

Computer Memory



Assign a unique address to each byte
Numbers from 0 to 34359738367

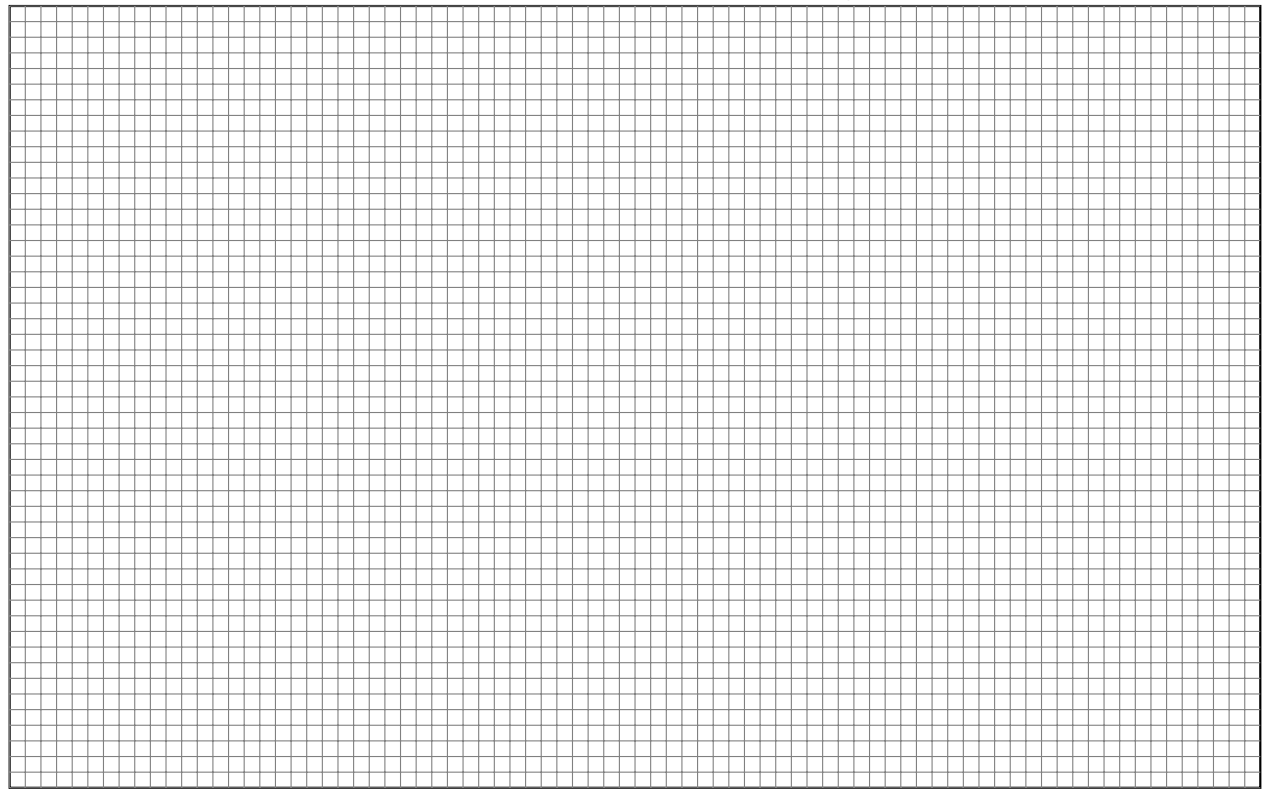
```
int a;
```



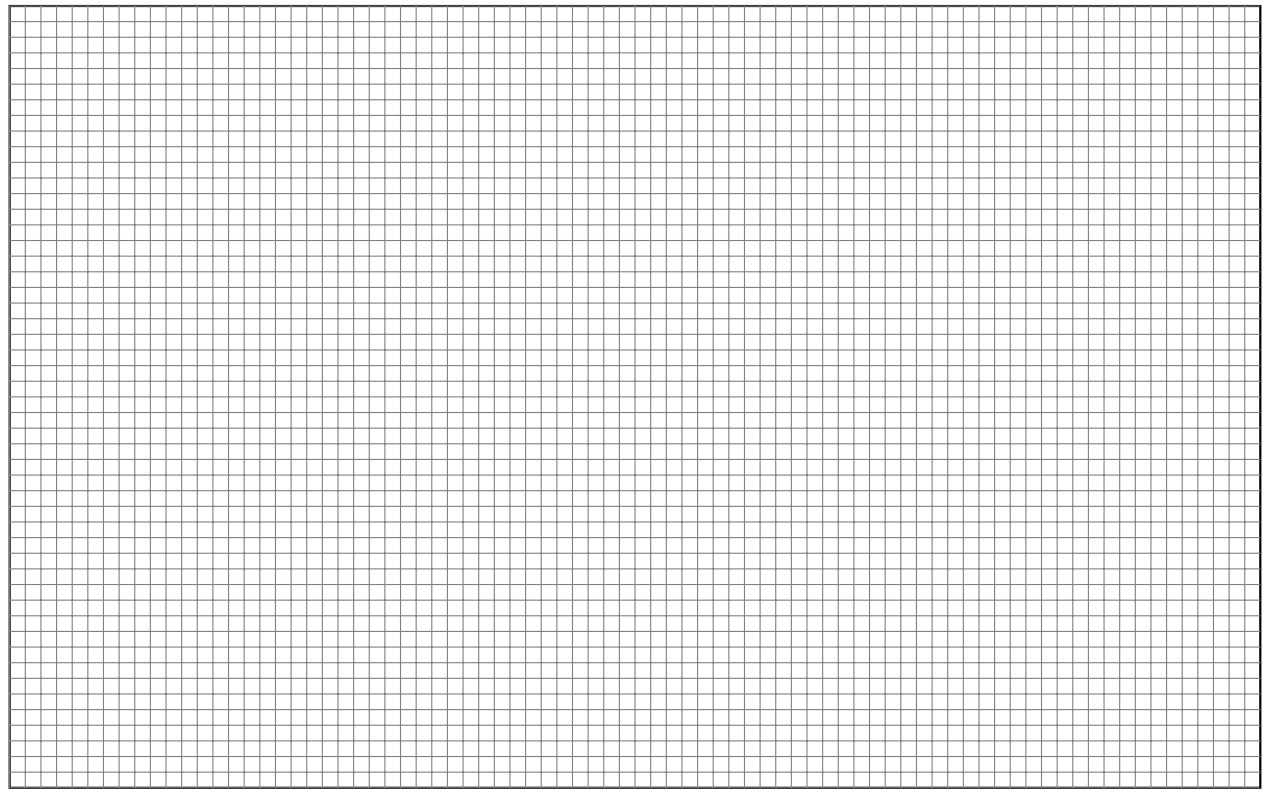
Assign a unique address to each byte
Numbers from 0 to 34359738367

```
int a;
```

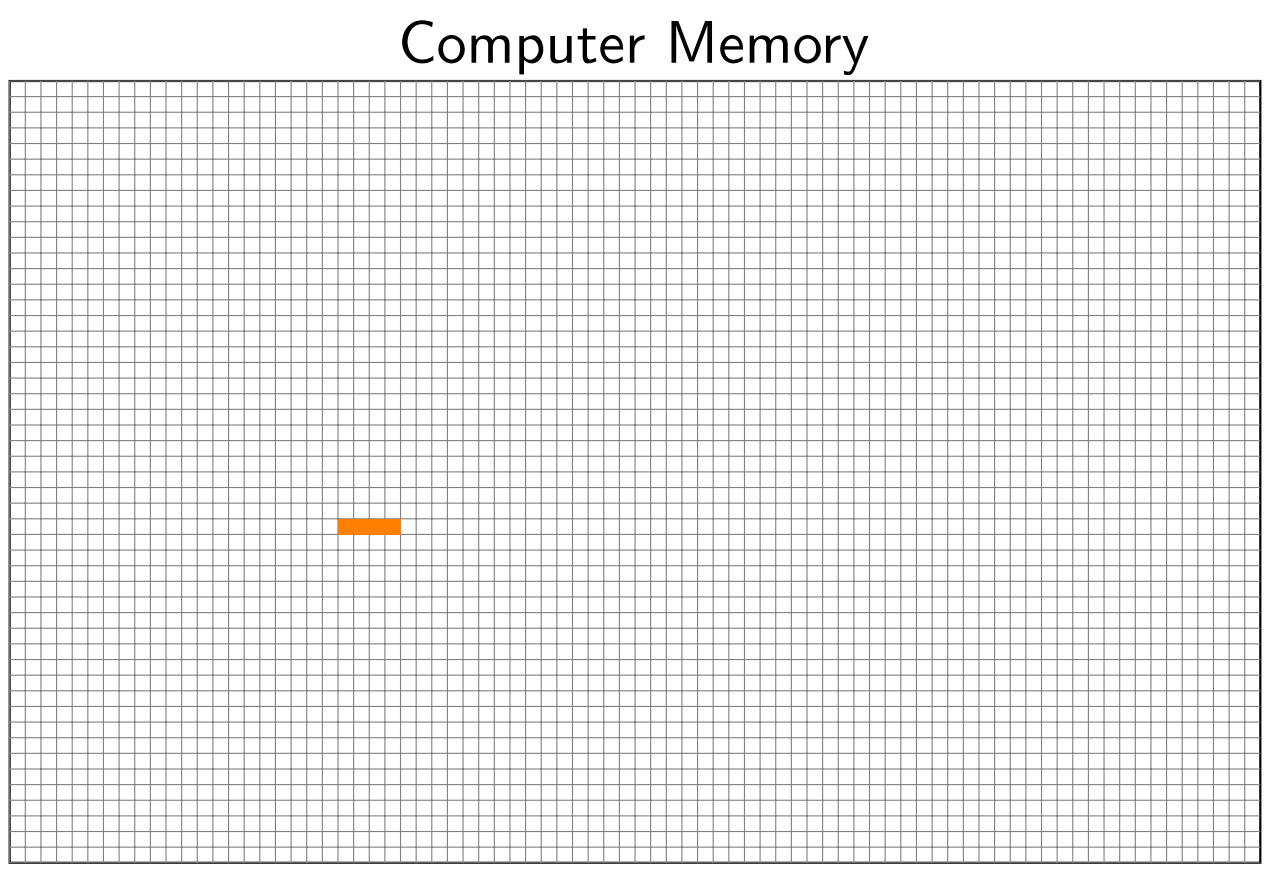
Computer Memory



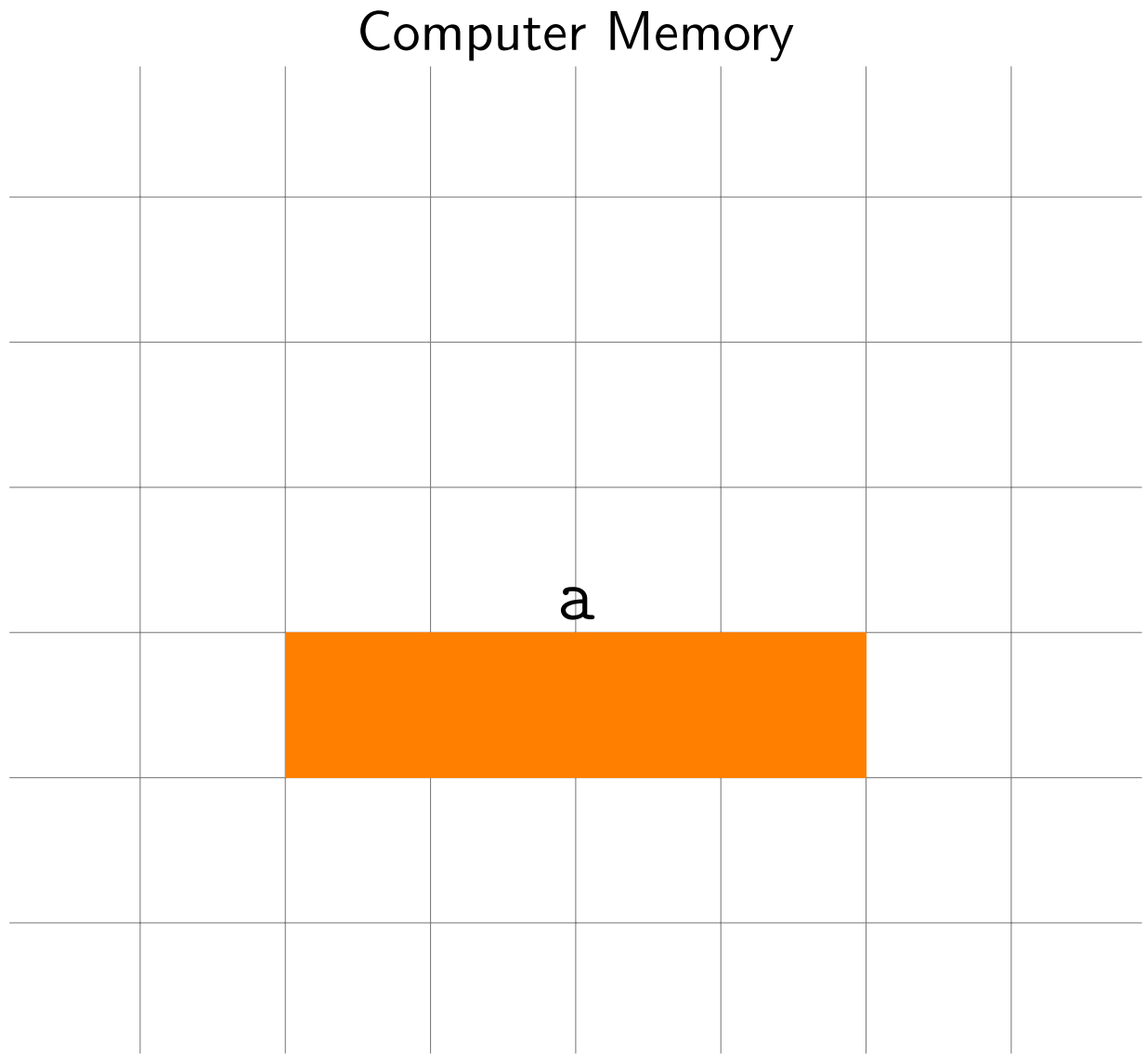
```
int a;
```



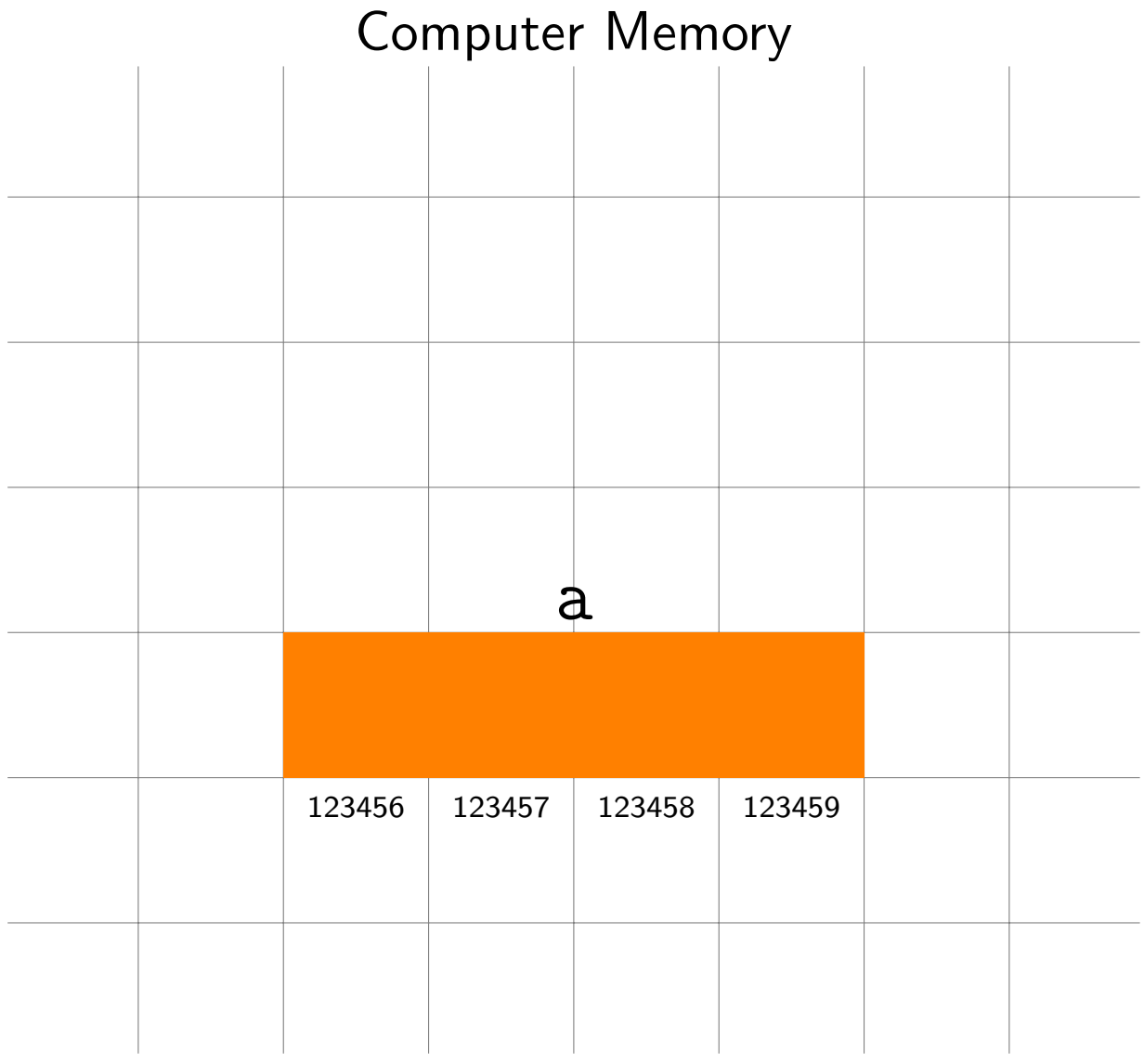
```
int a;
```



```
int a;
```
✓



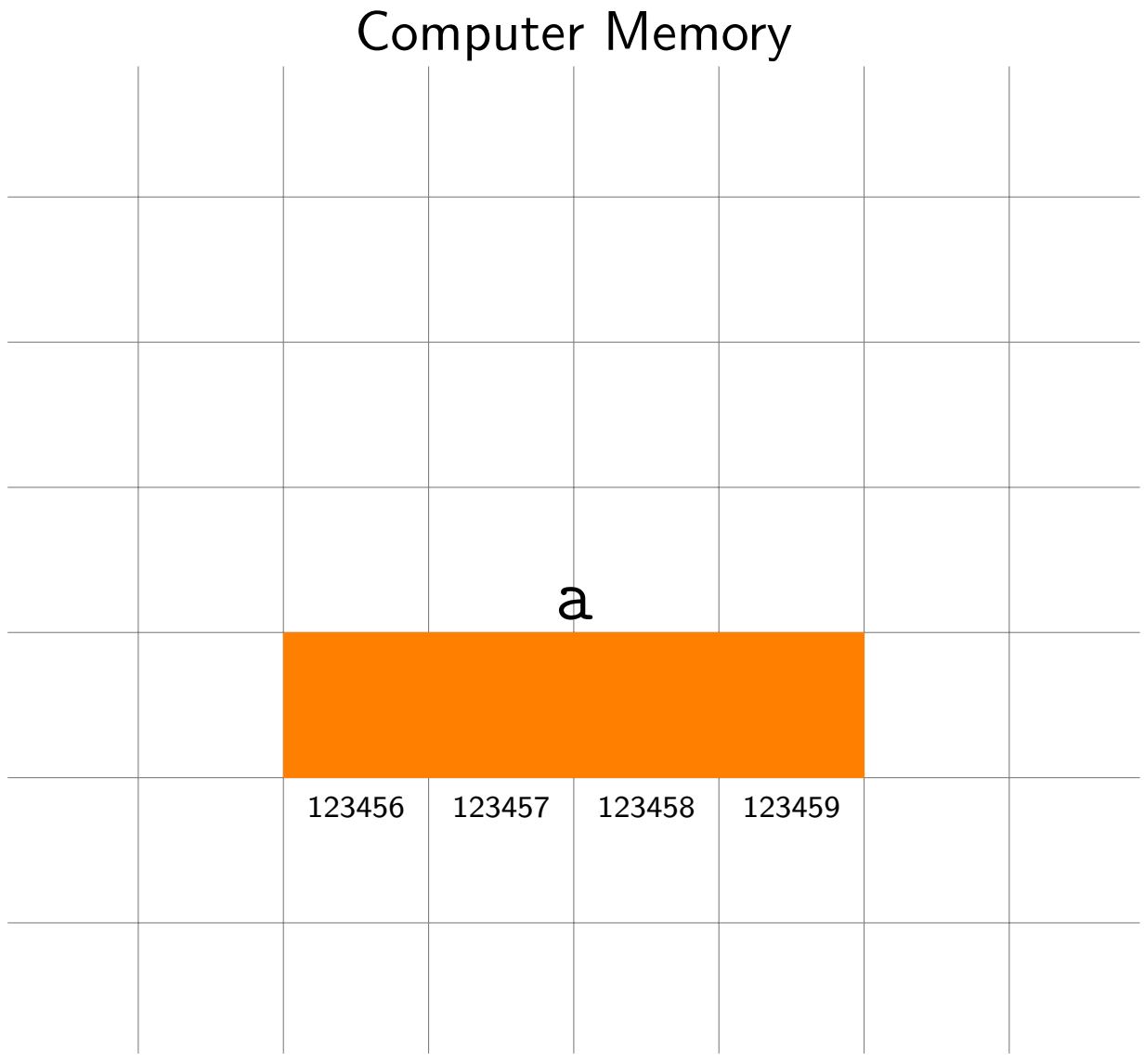
```
int a;
```



```
int a;
```

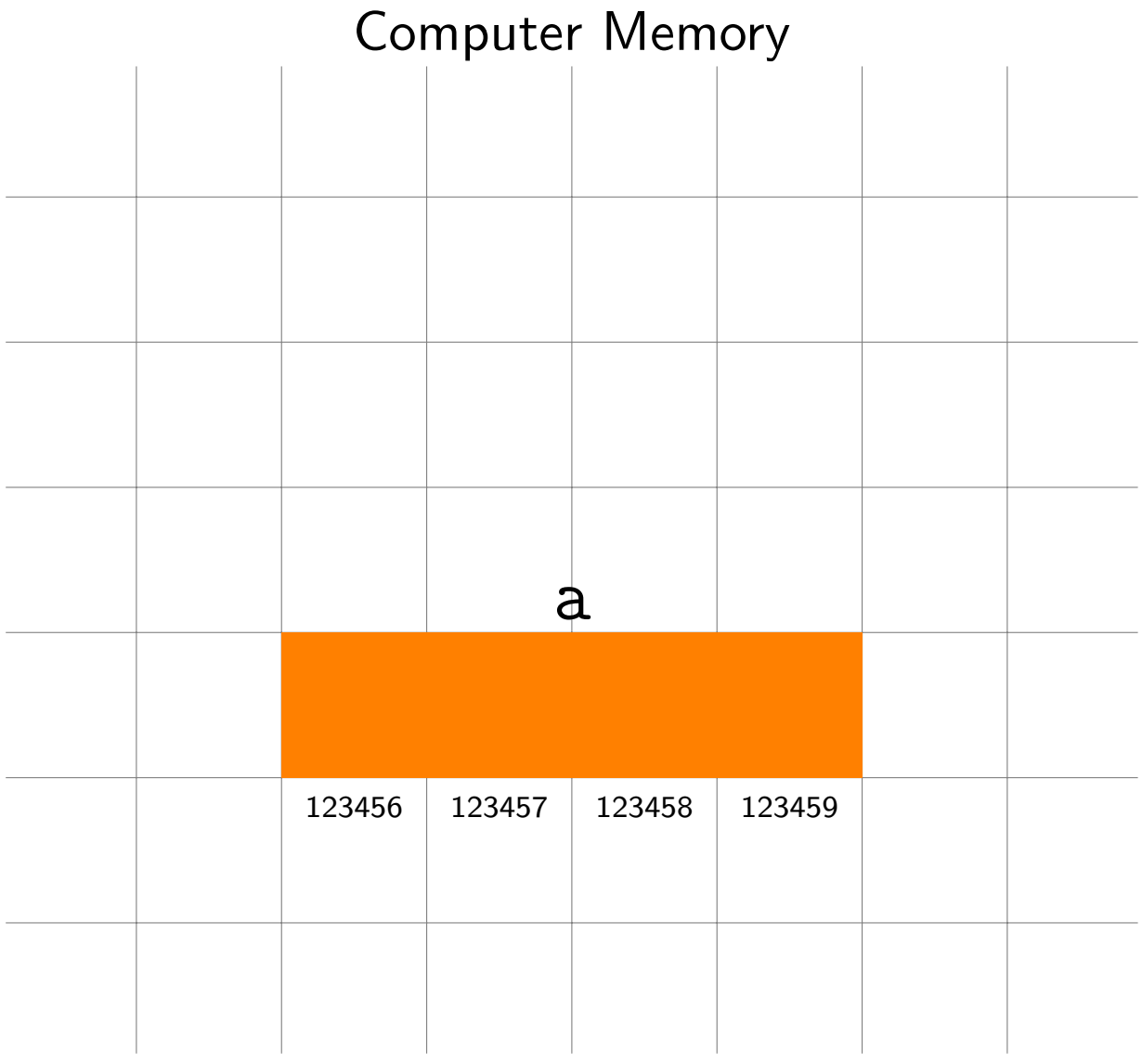
✓

```
a = 5;
```



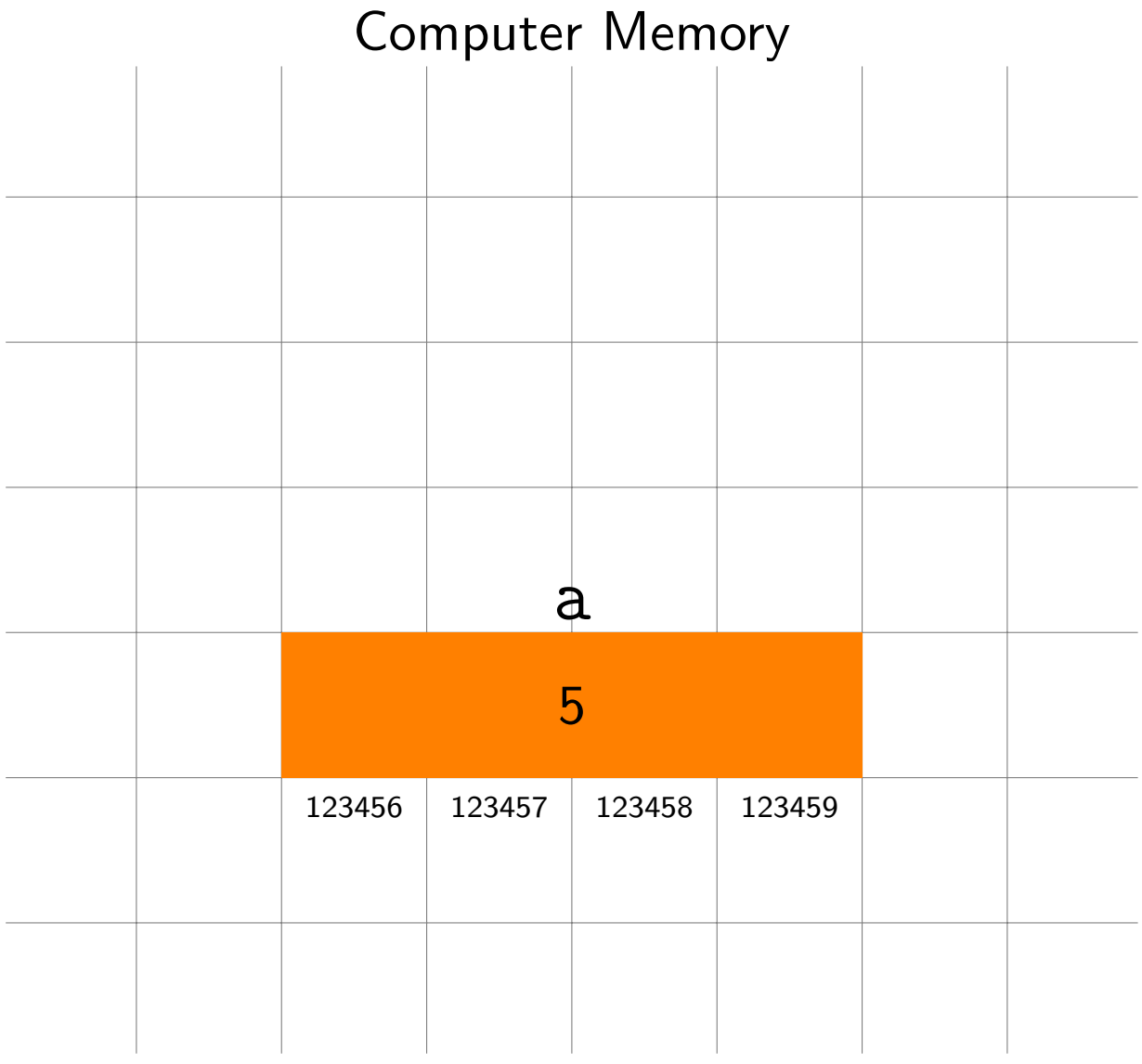
```
int a;
```
✓

```
a = 5;
```
✓



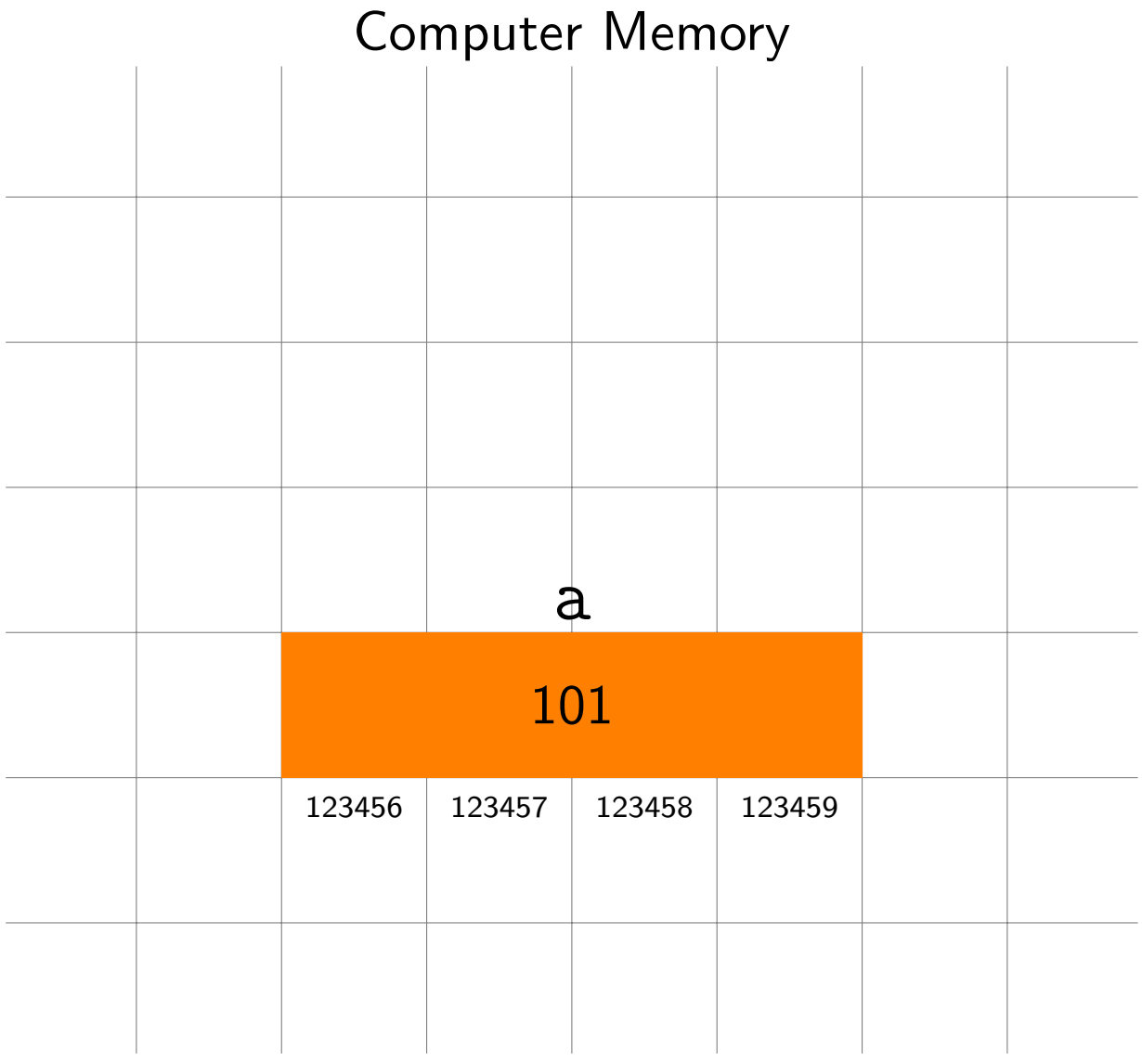
```
int a;
```

```
a = 5;
```



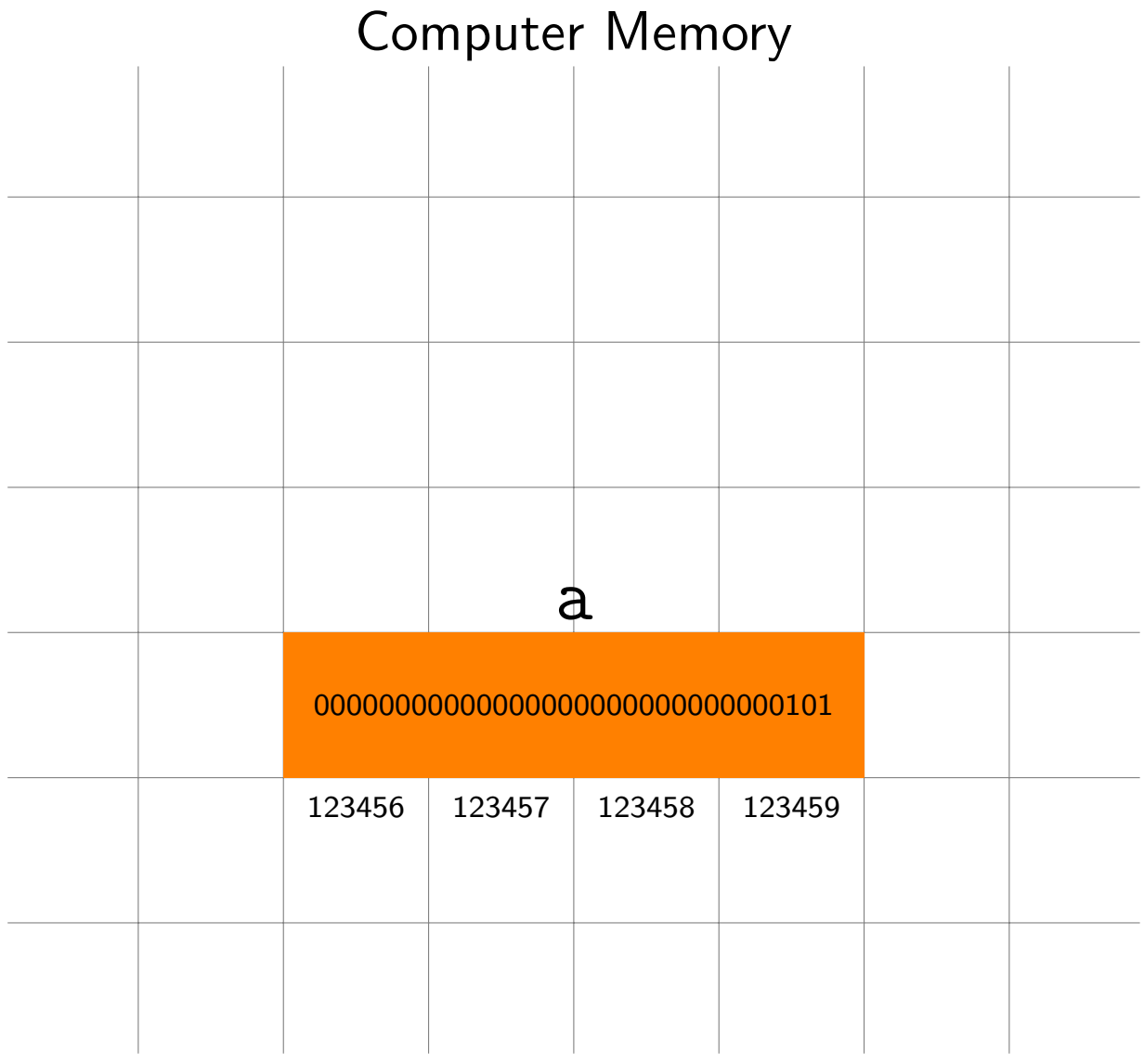
```
int a;
```

```
a = 5;
```



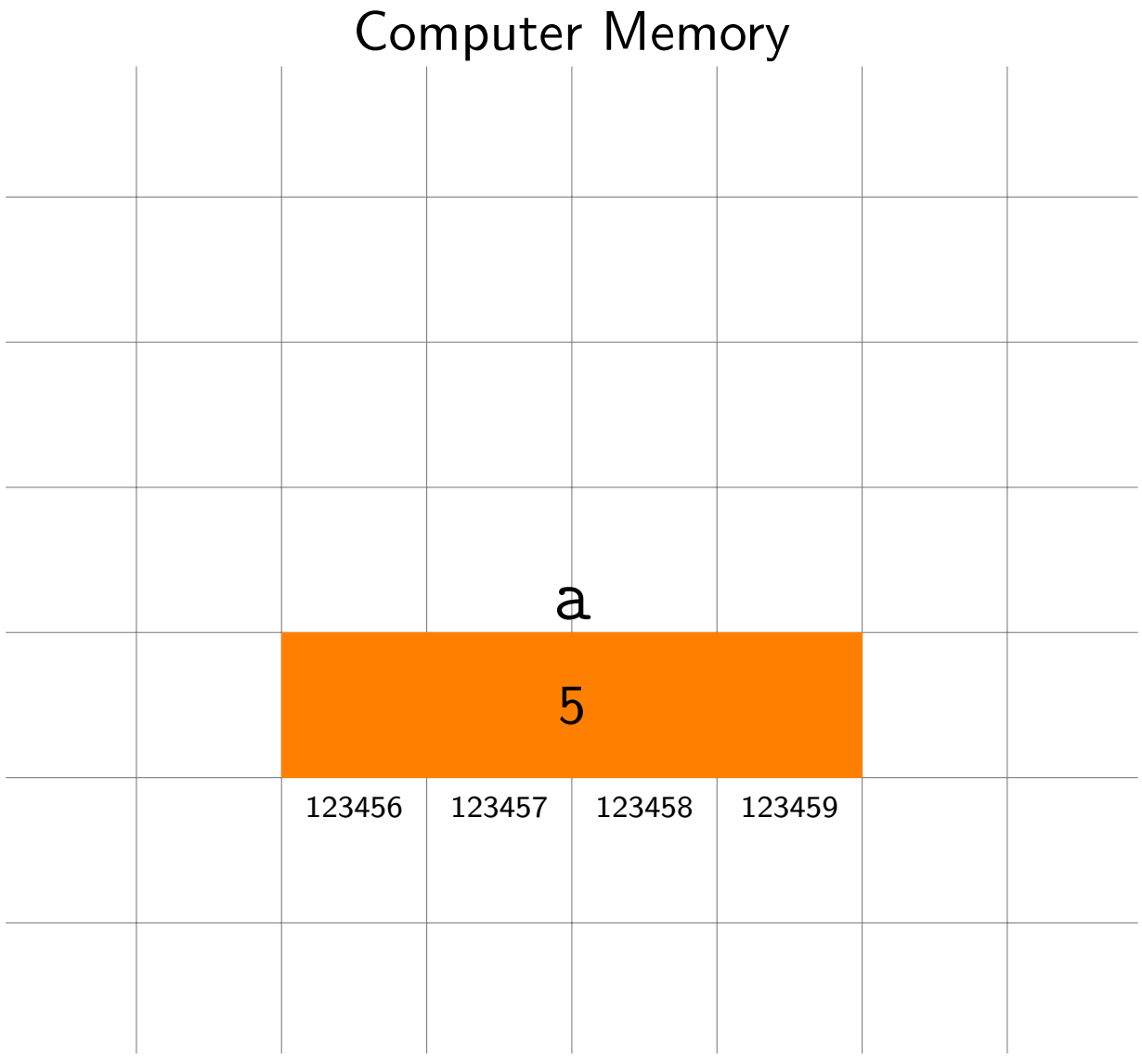
```
int a;
```

```
a = 5;
```

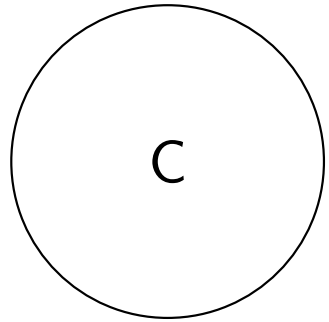

Computer Memory
a
00000000000000000000000000000101
123456
123457
123458
123459

```
int a;
```

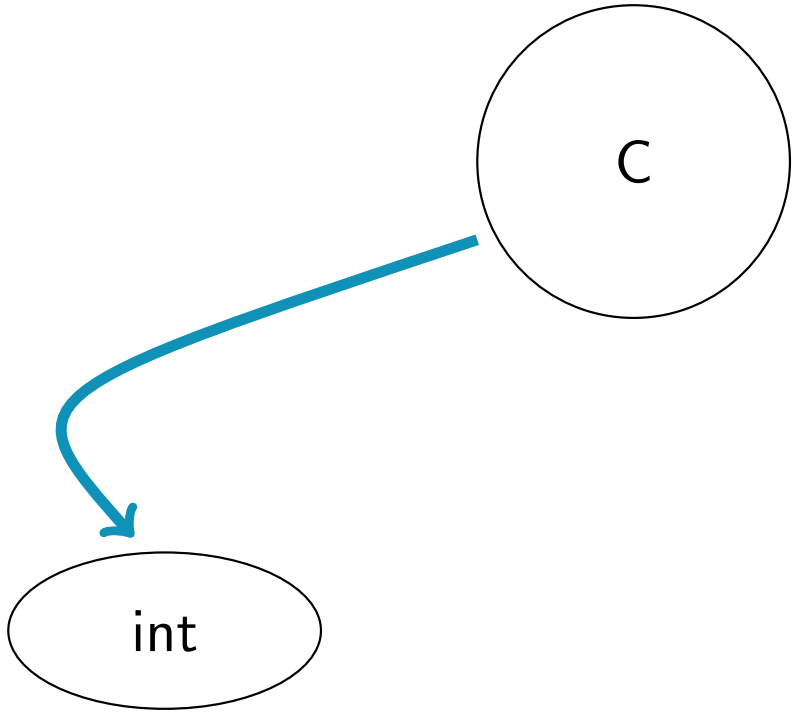
```
a = 5;
```



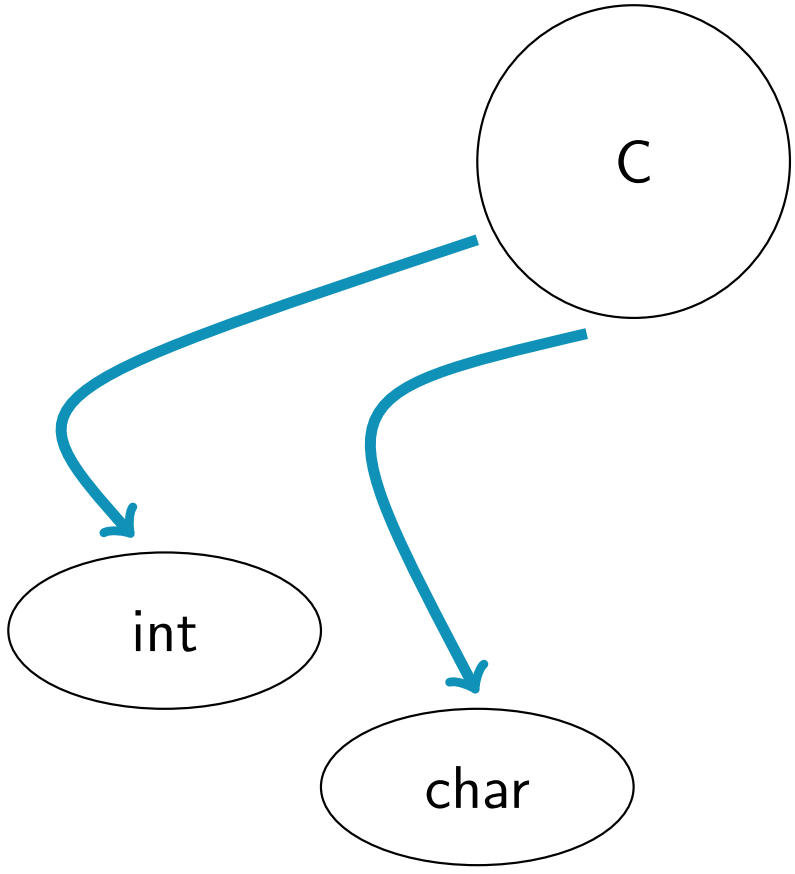
Data Types



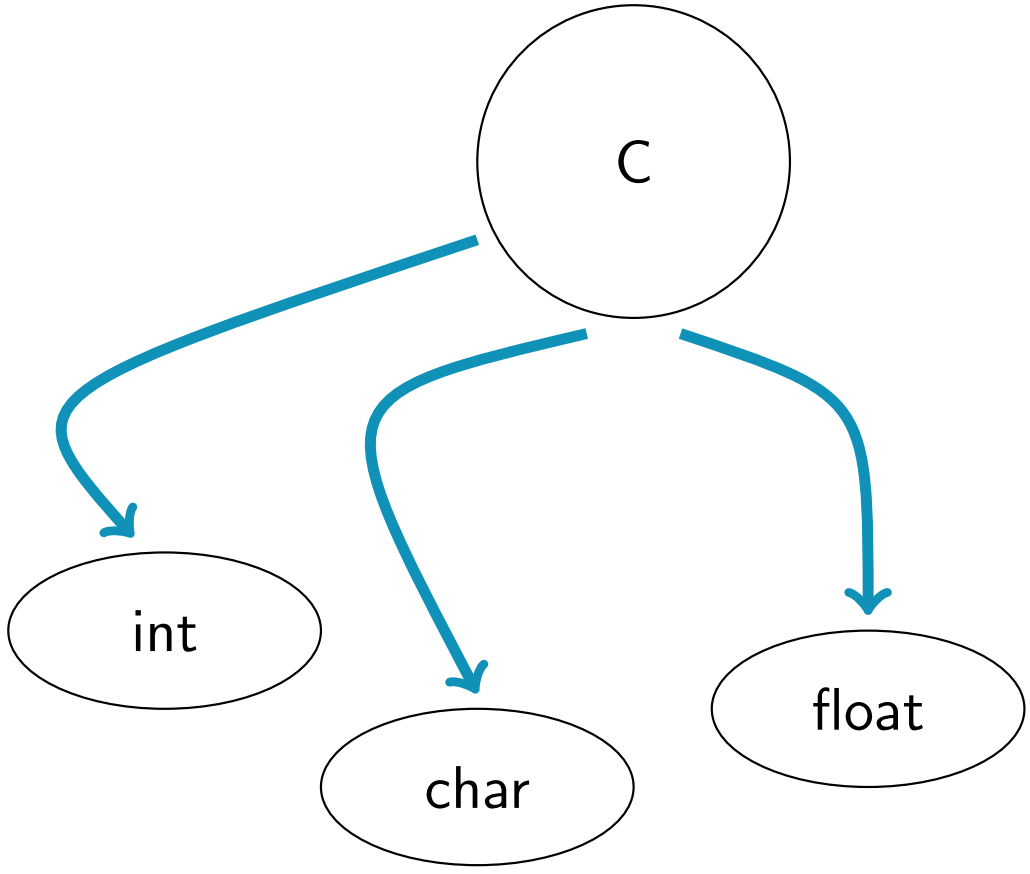
Data Types



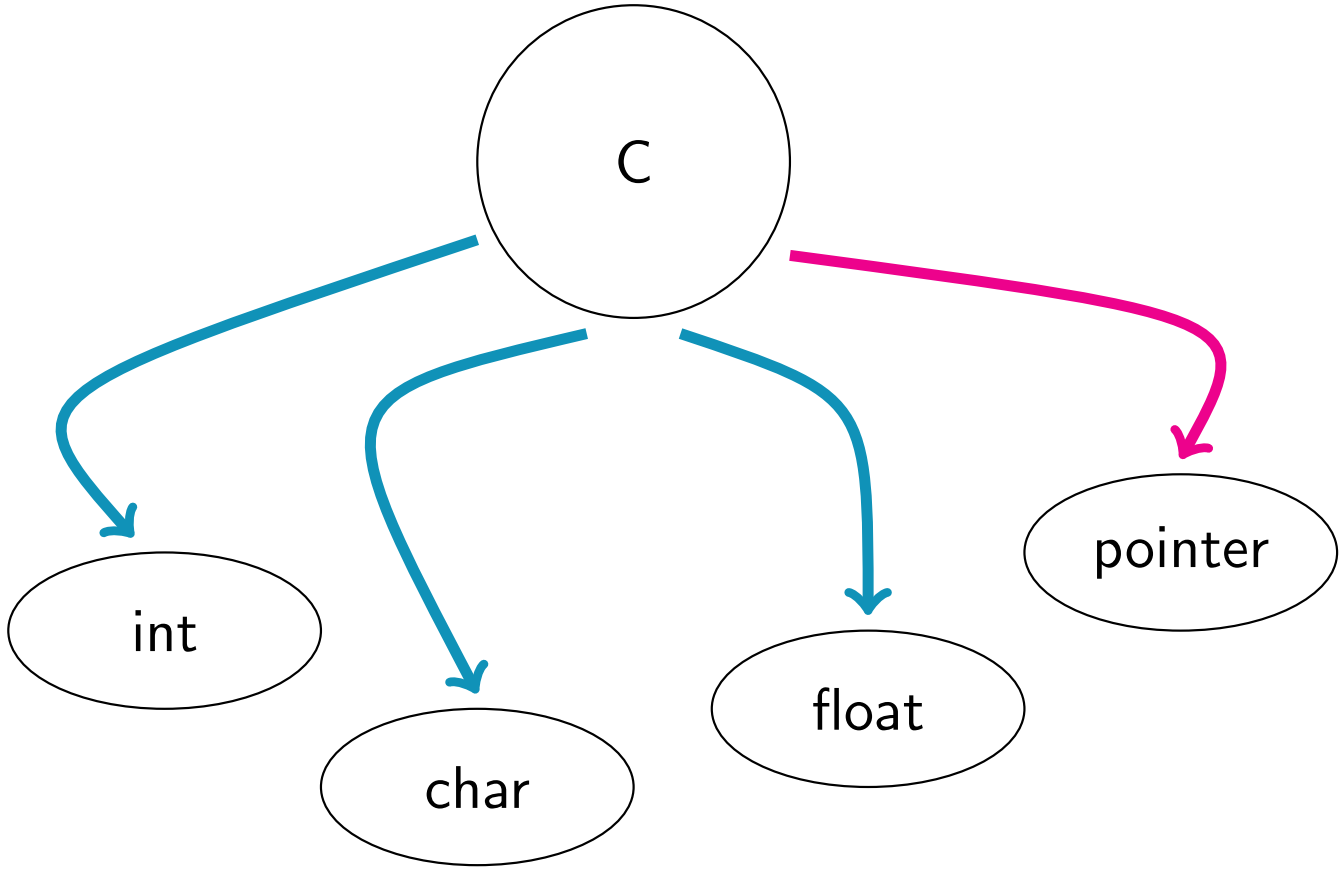
Data Types



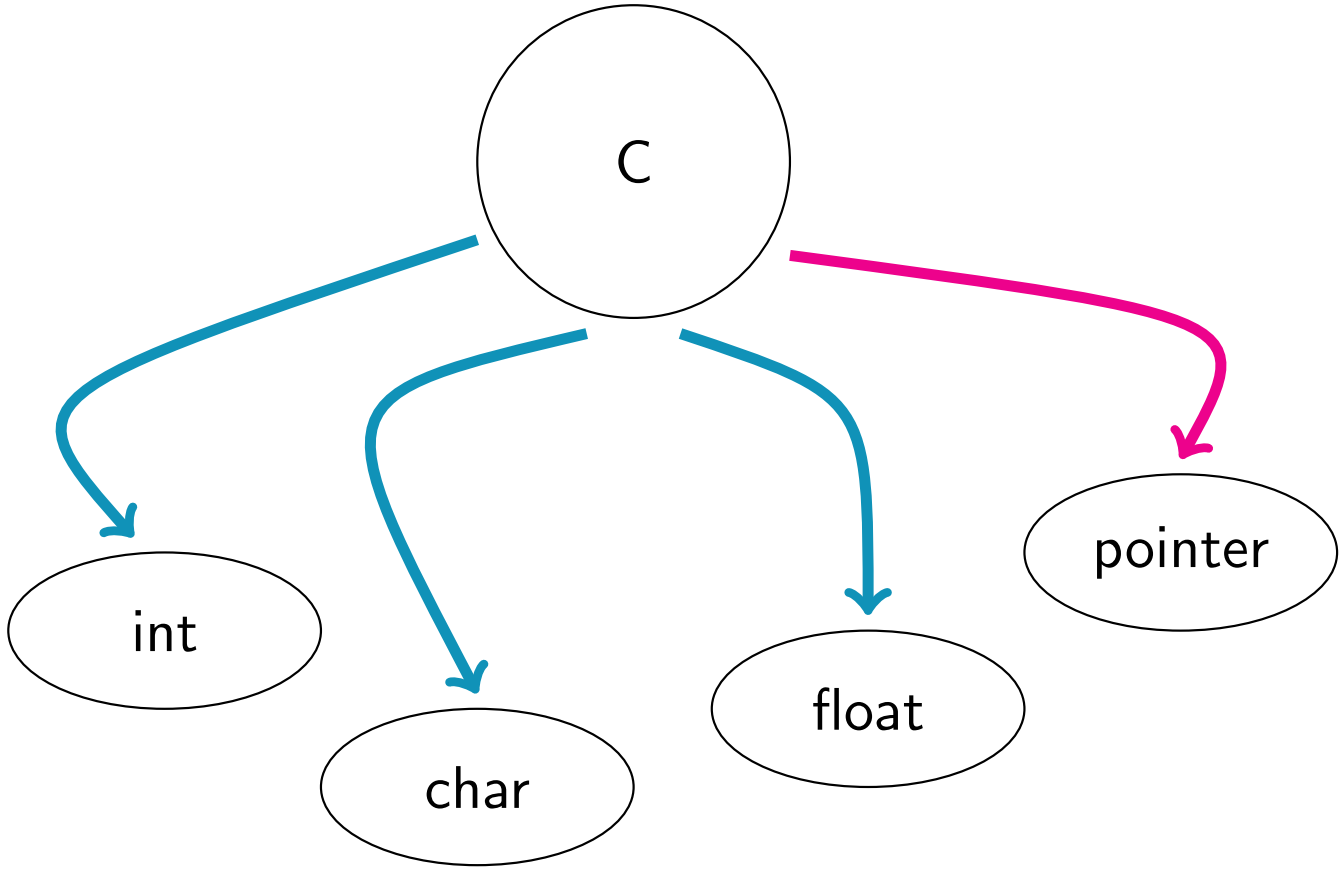
Data Types



Data Types

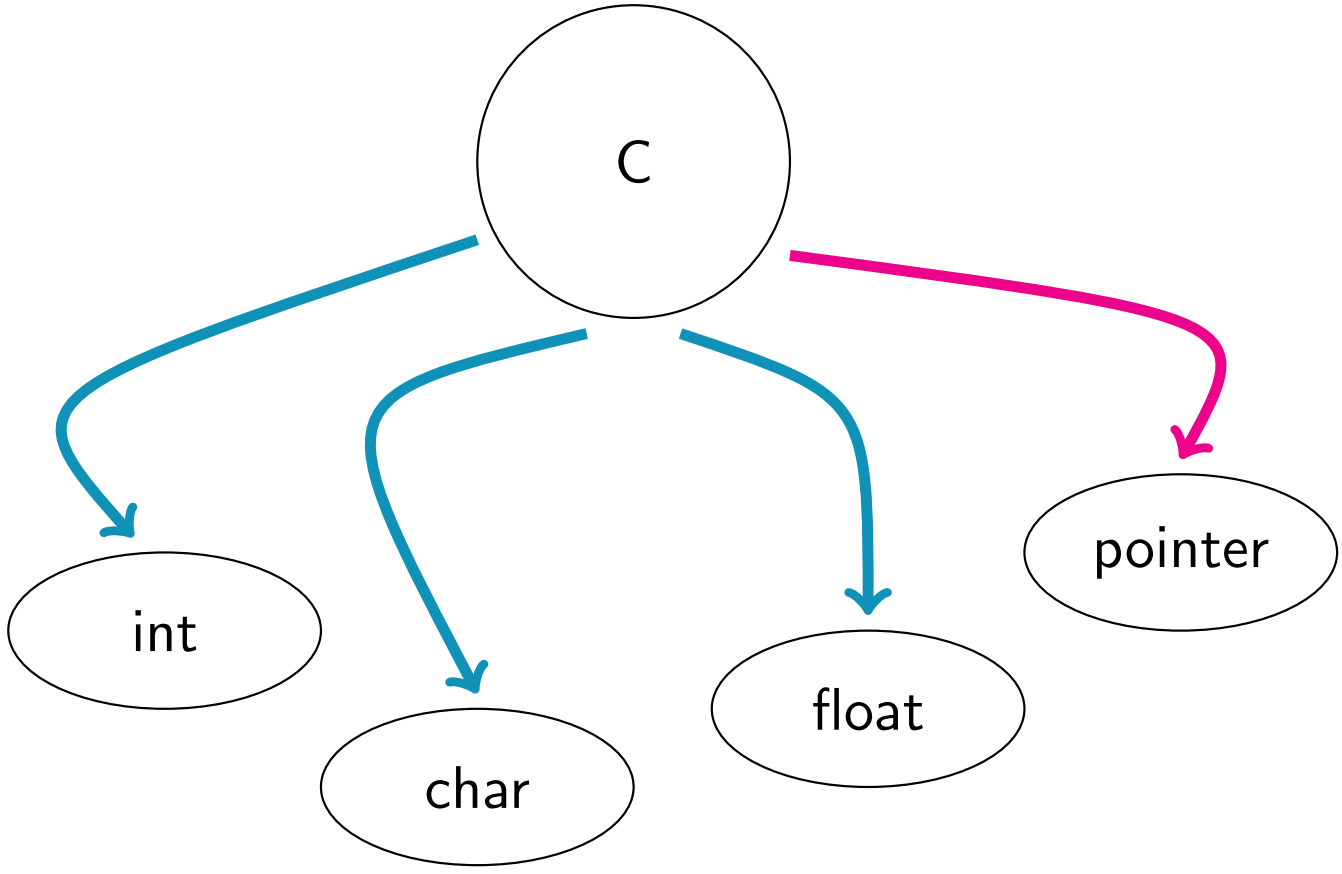


Data Types



Pointer

Data Types

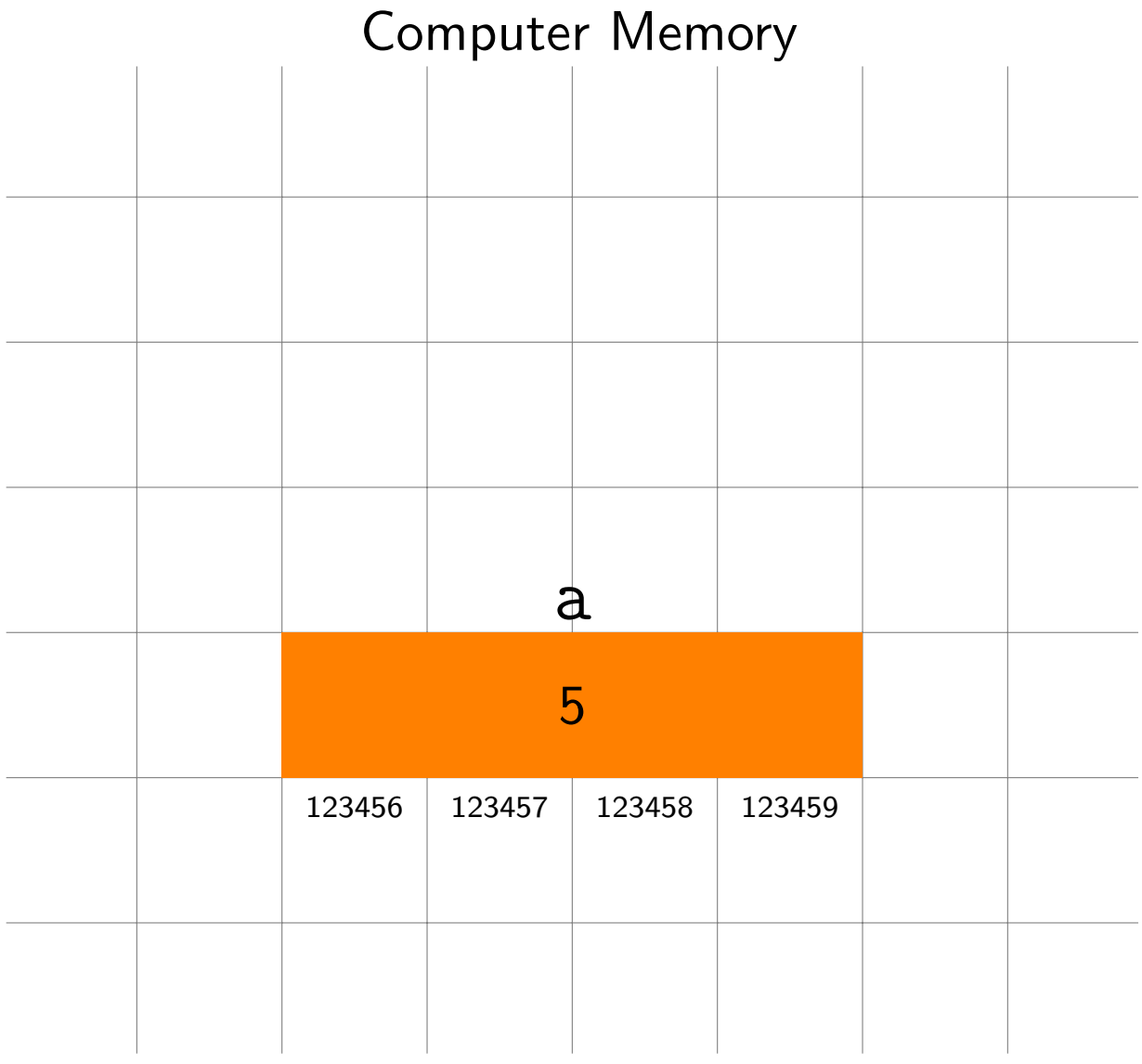


Pointer

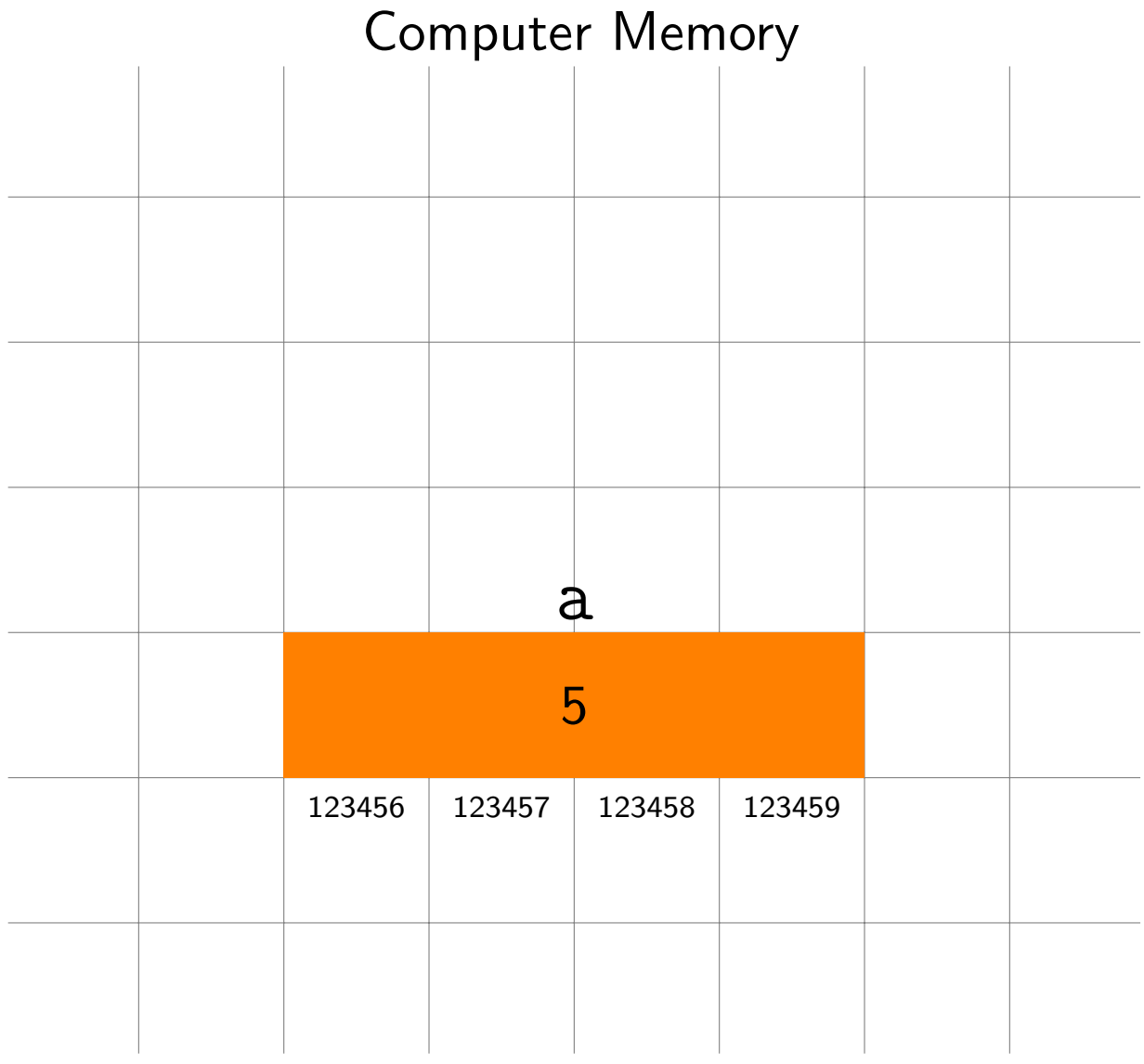
Stores address of variables.

```
int a;
```

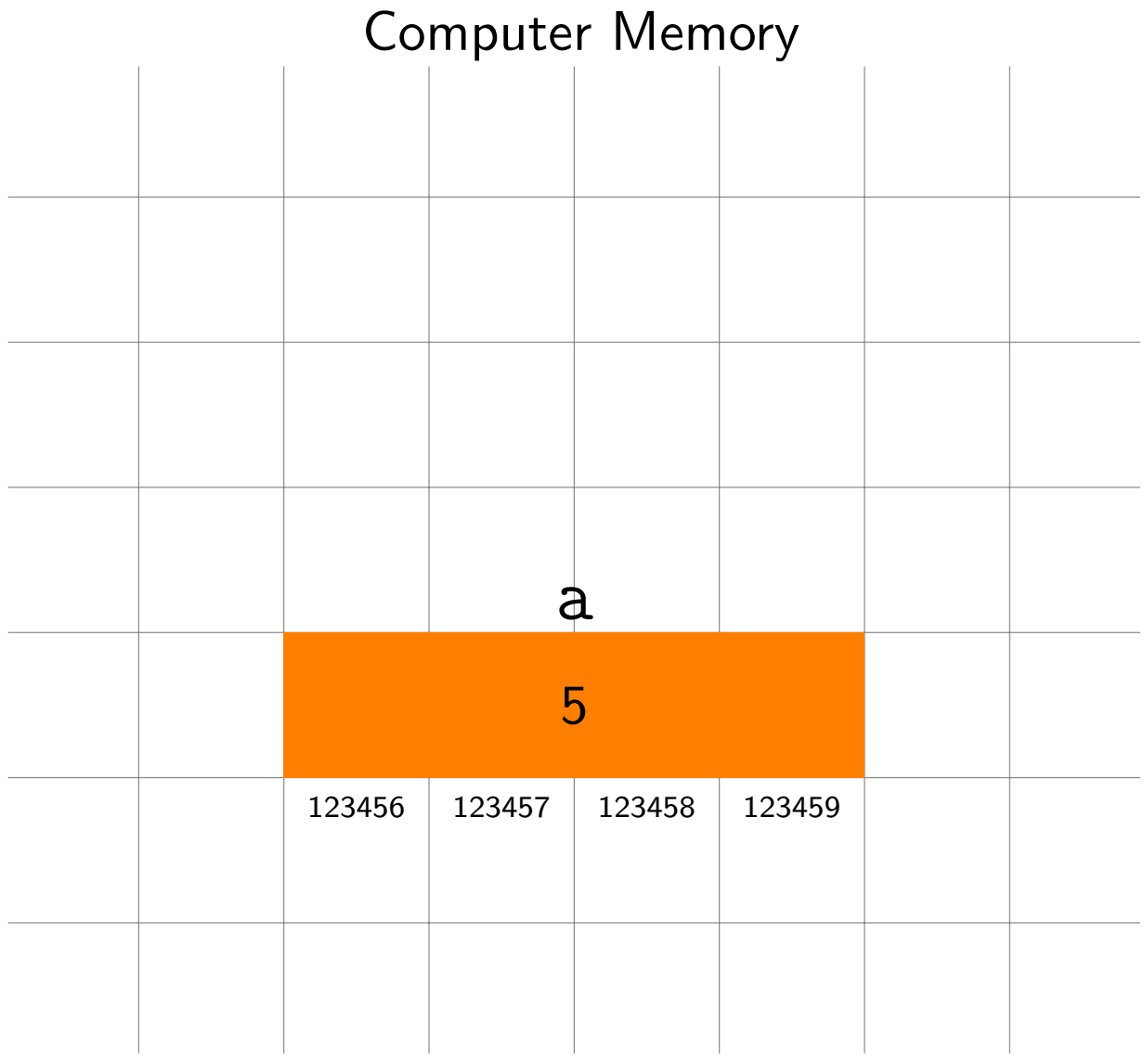
```
a = 5;
```



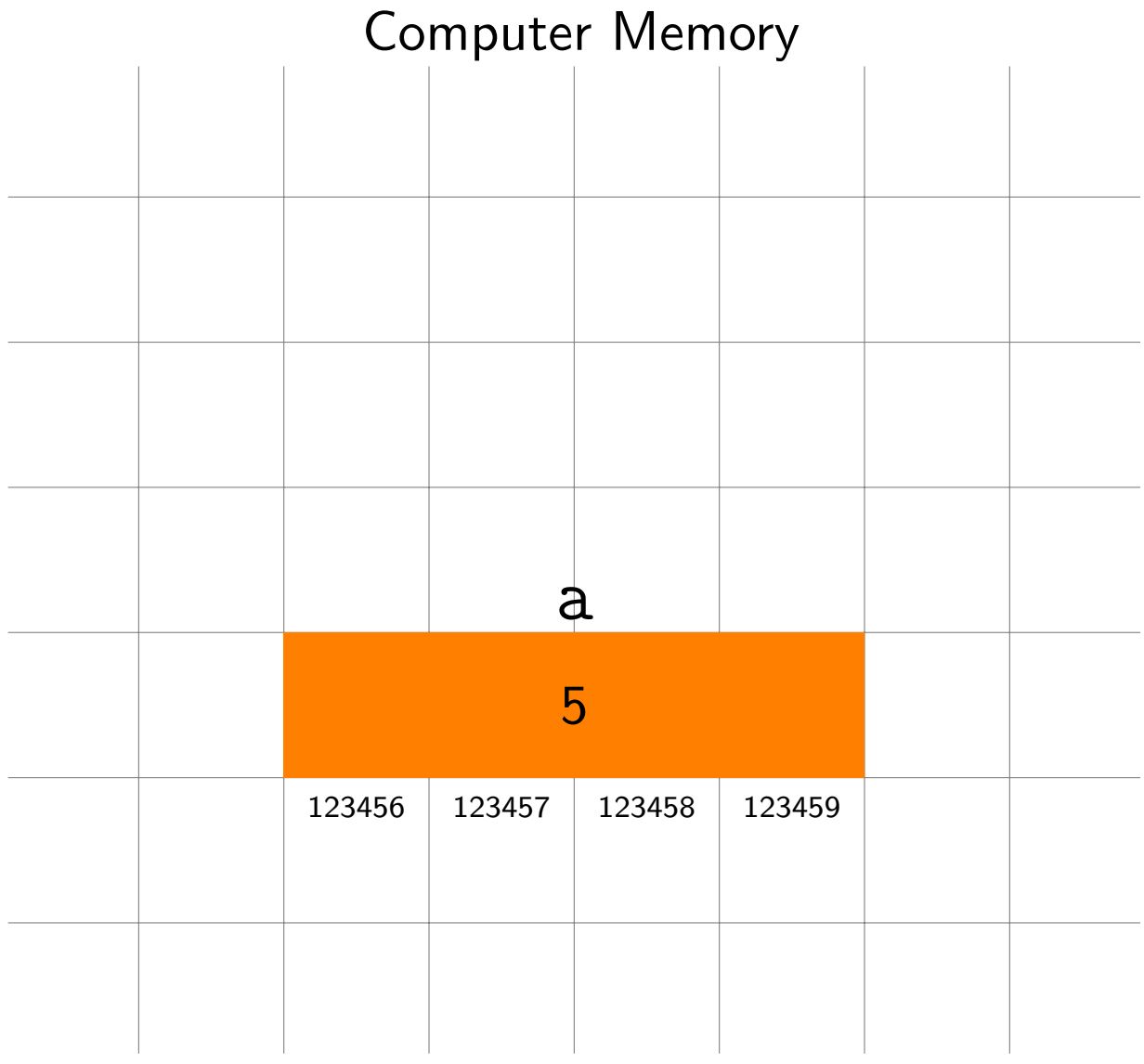
```
int a; ✓
     p
a = 5; ✓
```



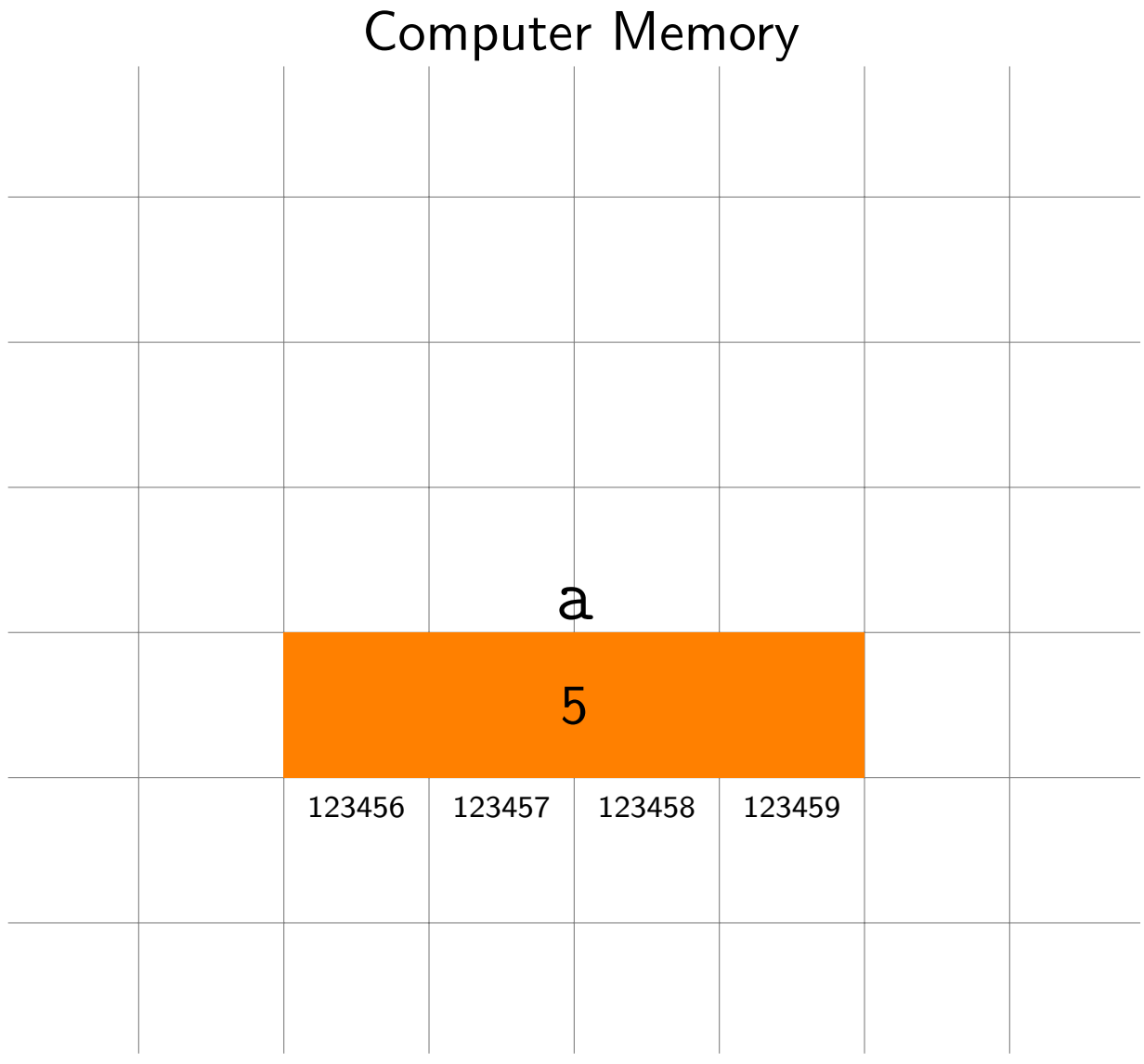
```
int a;  ✓
    *p
a = 5;  ✓
```



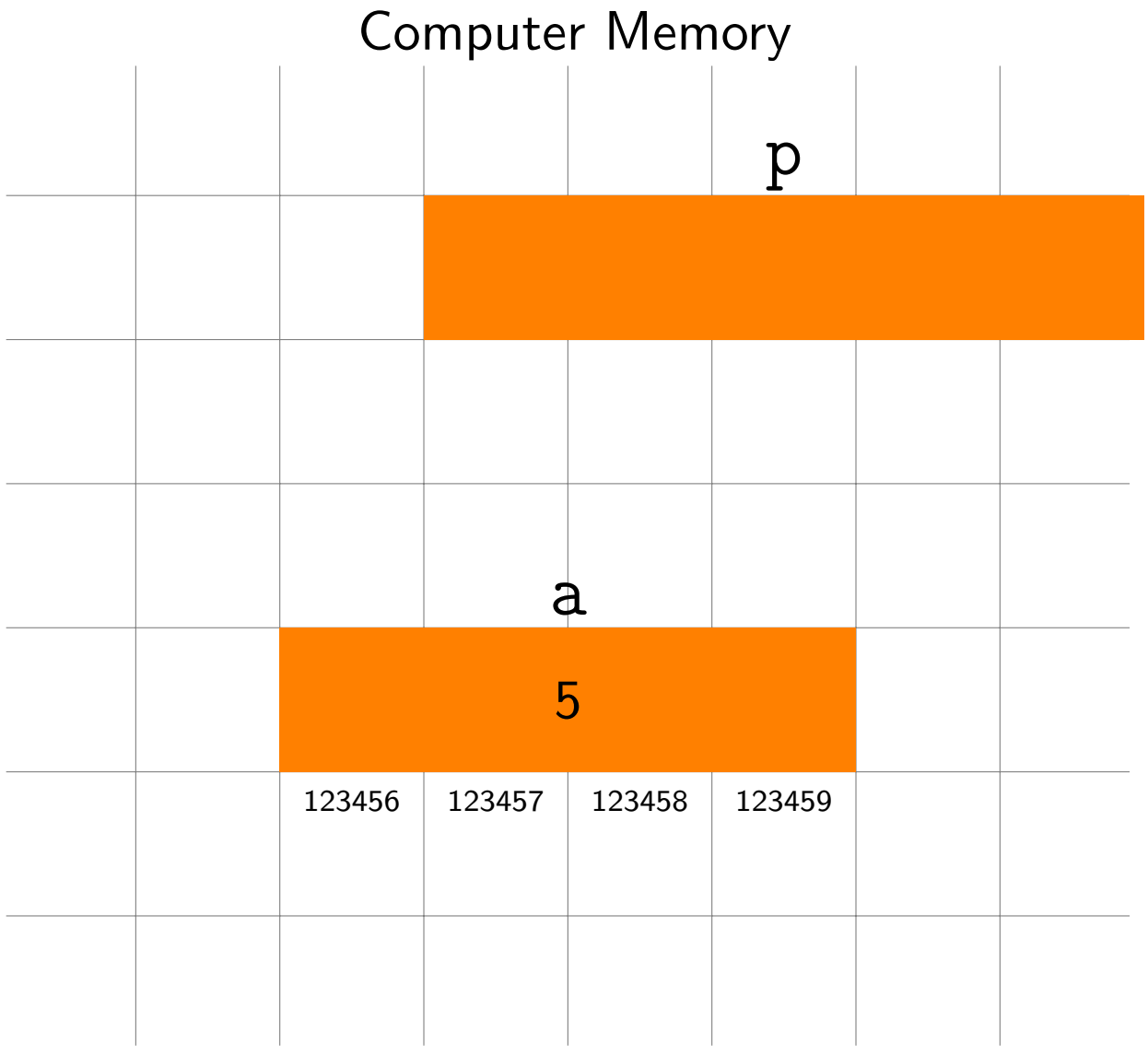
```
int a; ✓
int *p
a = 5; ✓
```



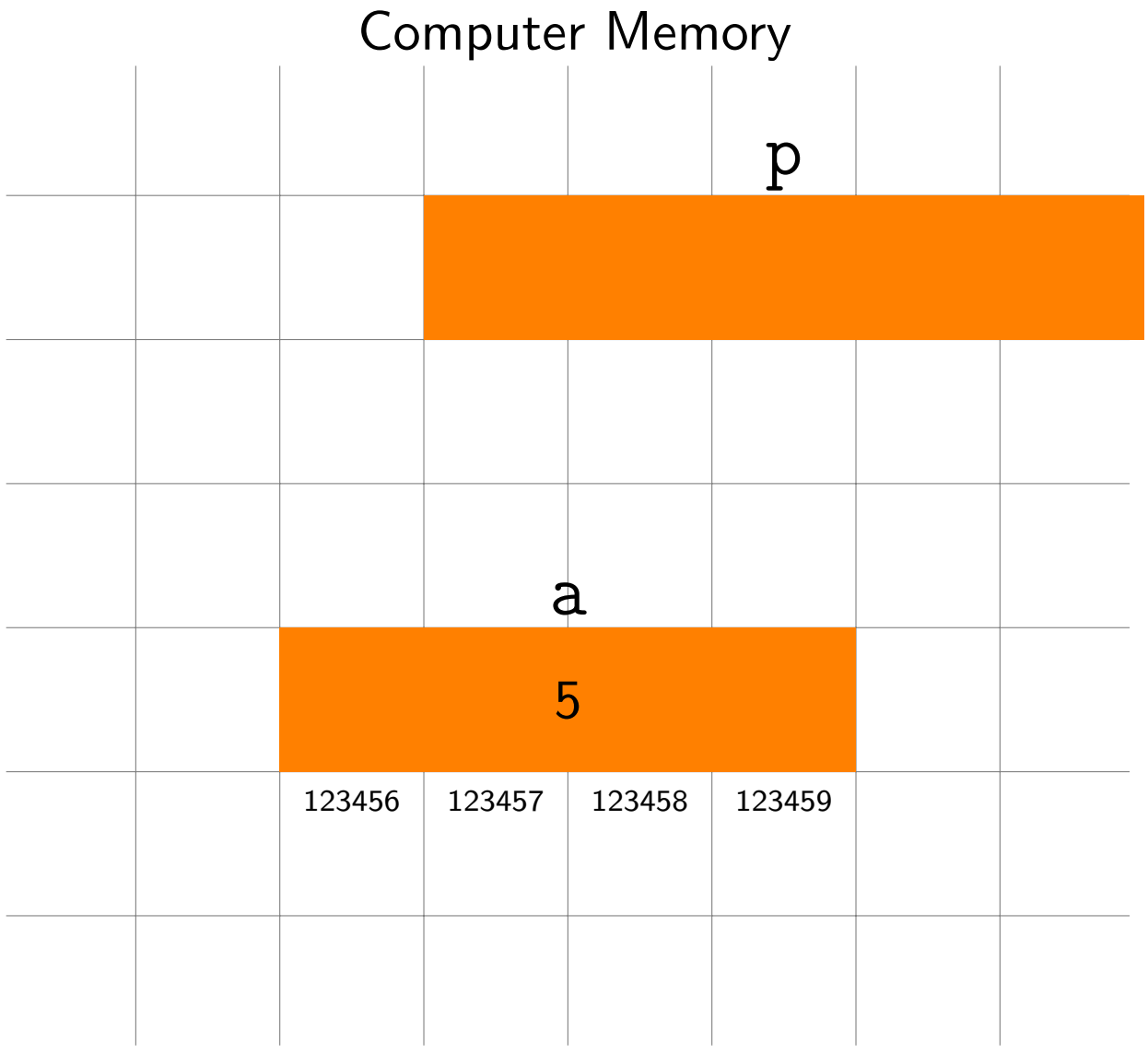
```
int a; ✓
int *p;
a = 5; ✓
```



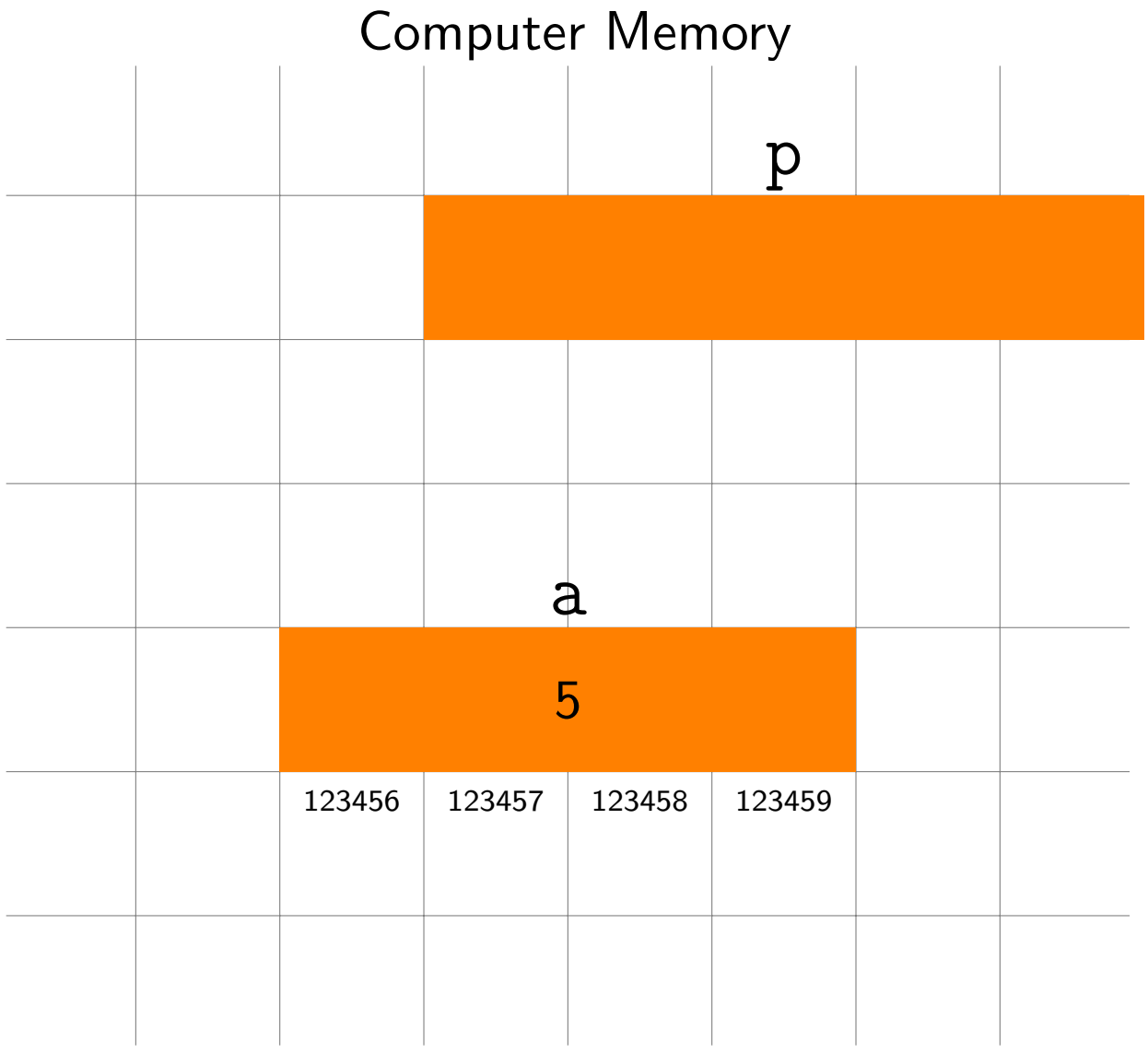
```
int a; ✓
int *p; ✓
a = 5; ✓
```



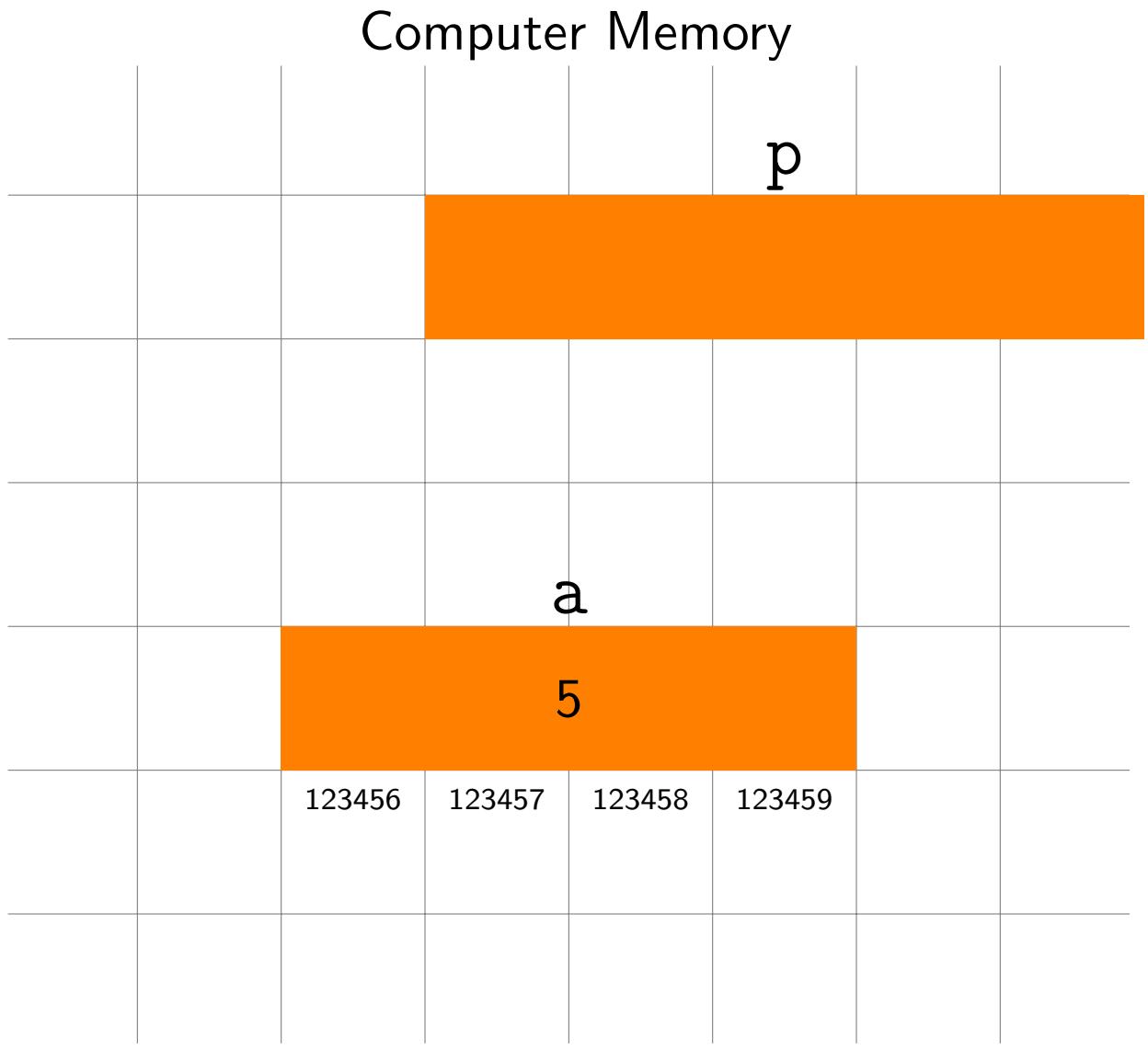
```
int a; ✓
int *p; ✓
a = 5; ✓
p =  a;
```



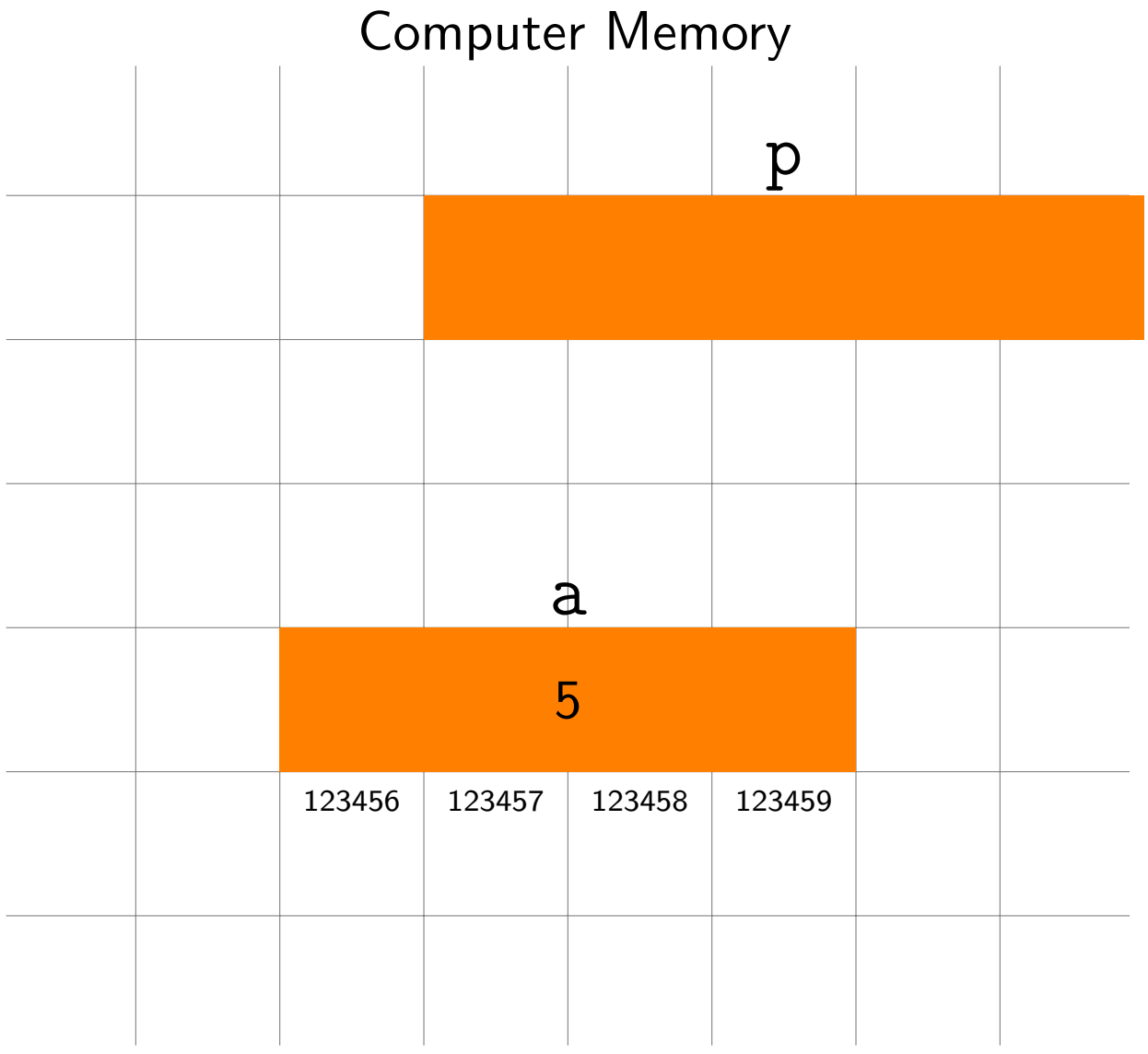
```
int a;
int *p;
a = 5;
p =  a;
```



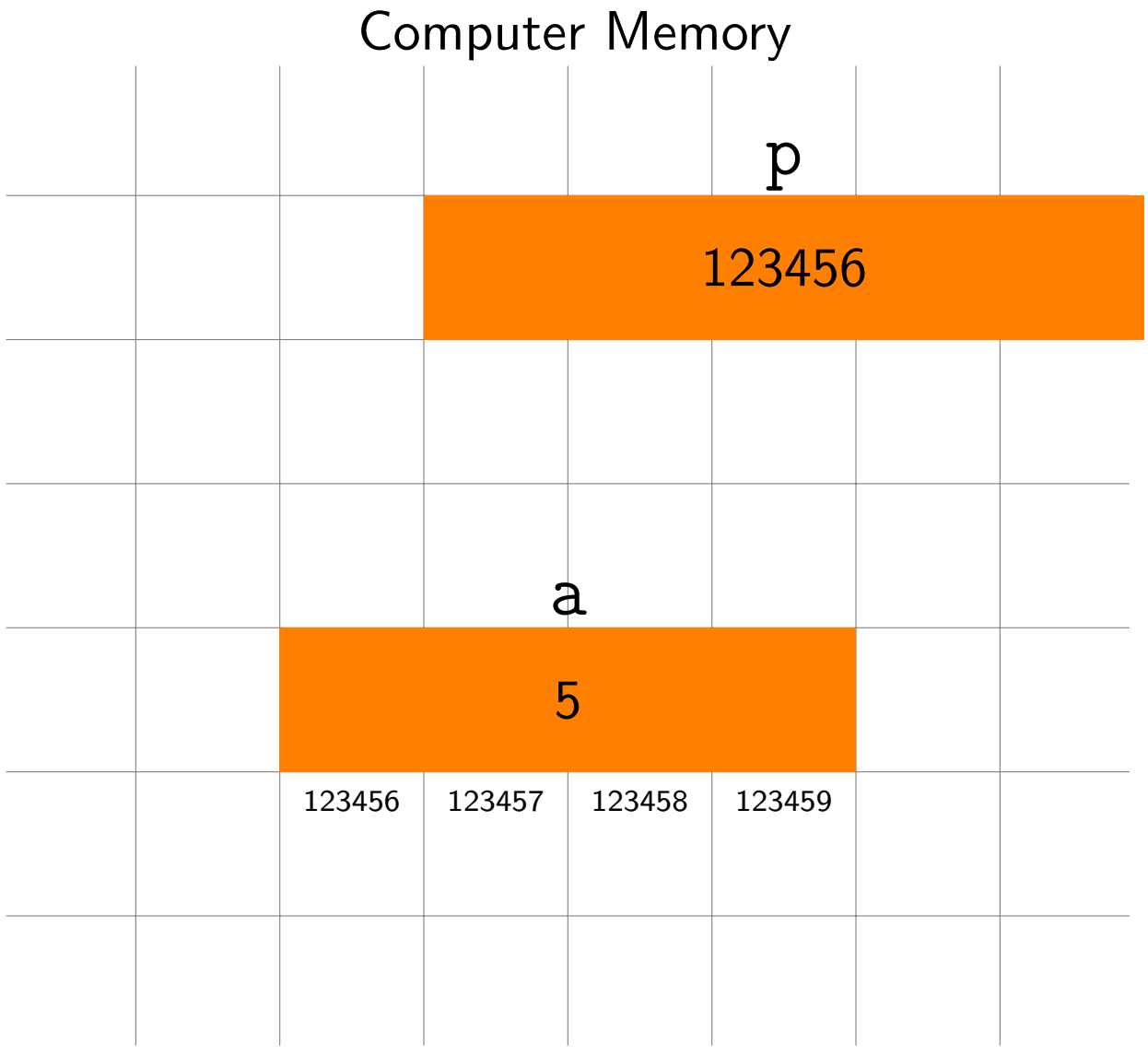
```
int a; ✓
int *p; ✓
a = 5; ✓
p = &a;
```



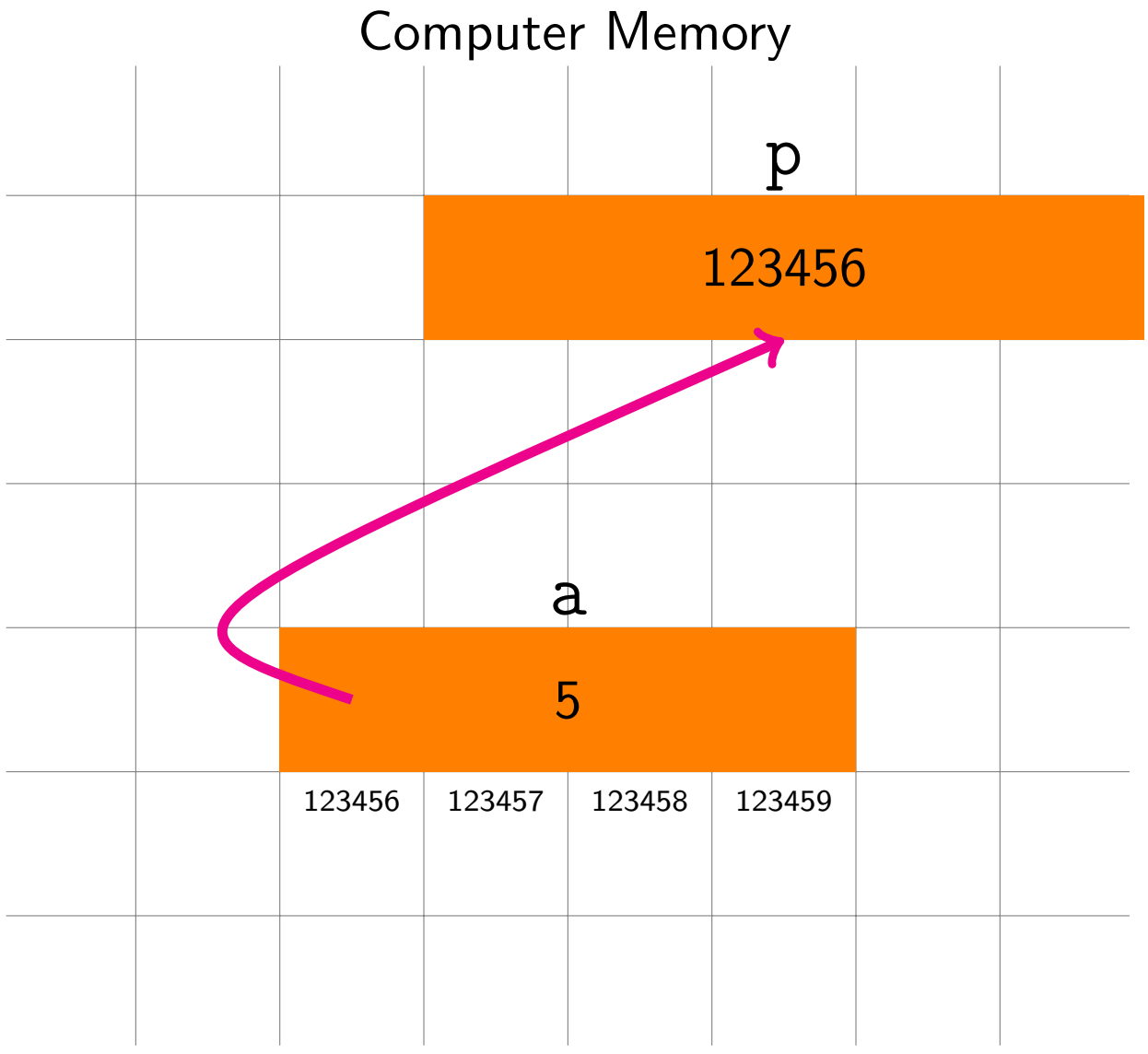
```
int a; ✓
int *p; ✓
a = 5; ✓
p = &a;
```



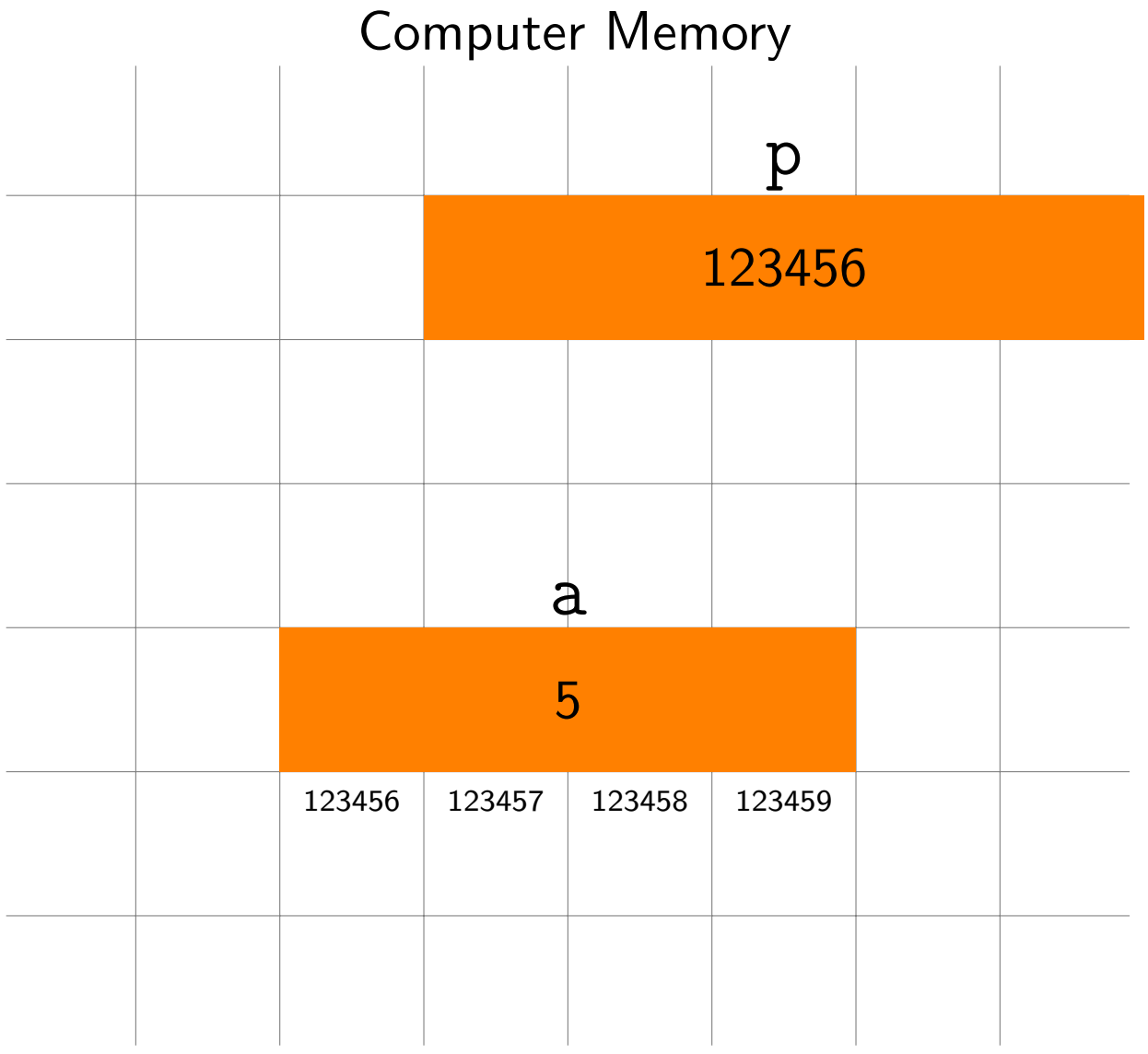
```
int a;
int *p;
a = 5;
p = &a;
```



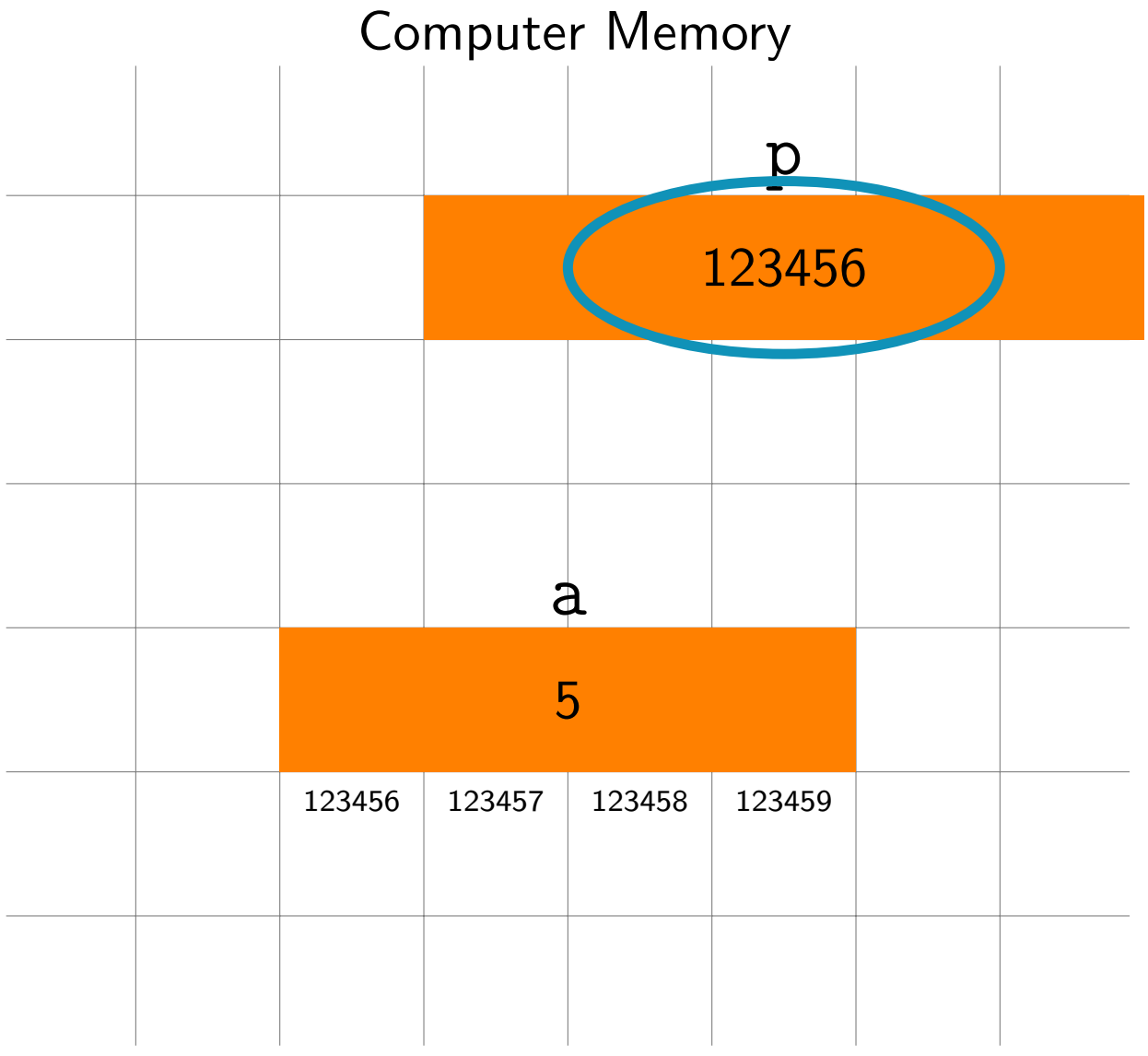
```
int a; ✓
int *p; ✓
a = 5; ✓
p = &a; ✓
```



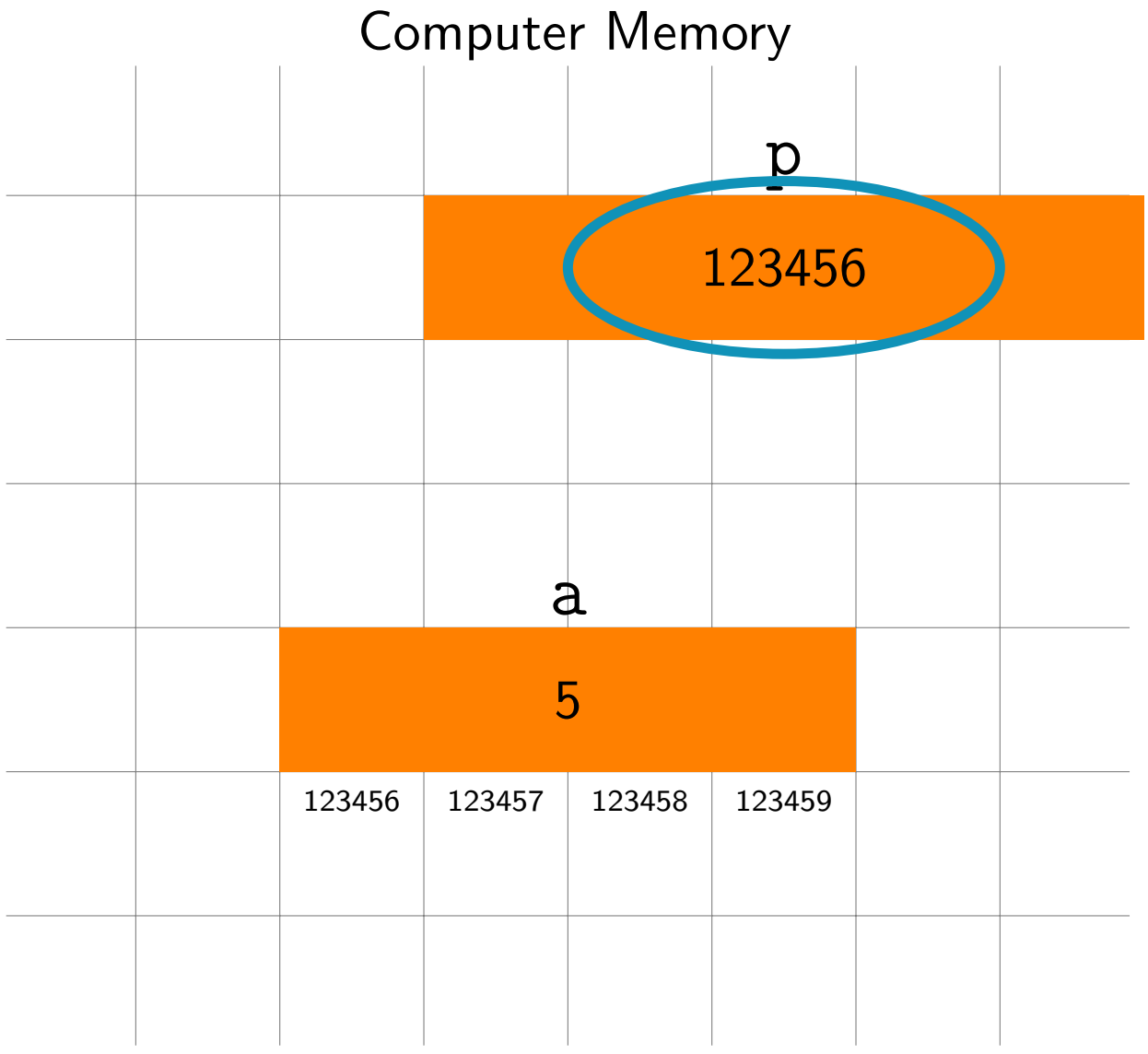
```
int a;
int *p;
a = 5;
p = &a;
 p
```



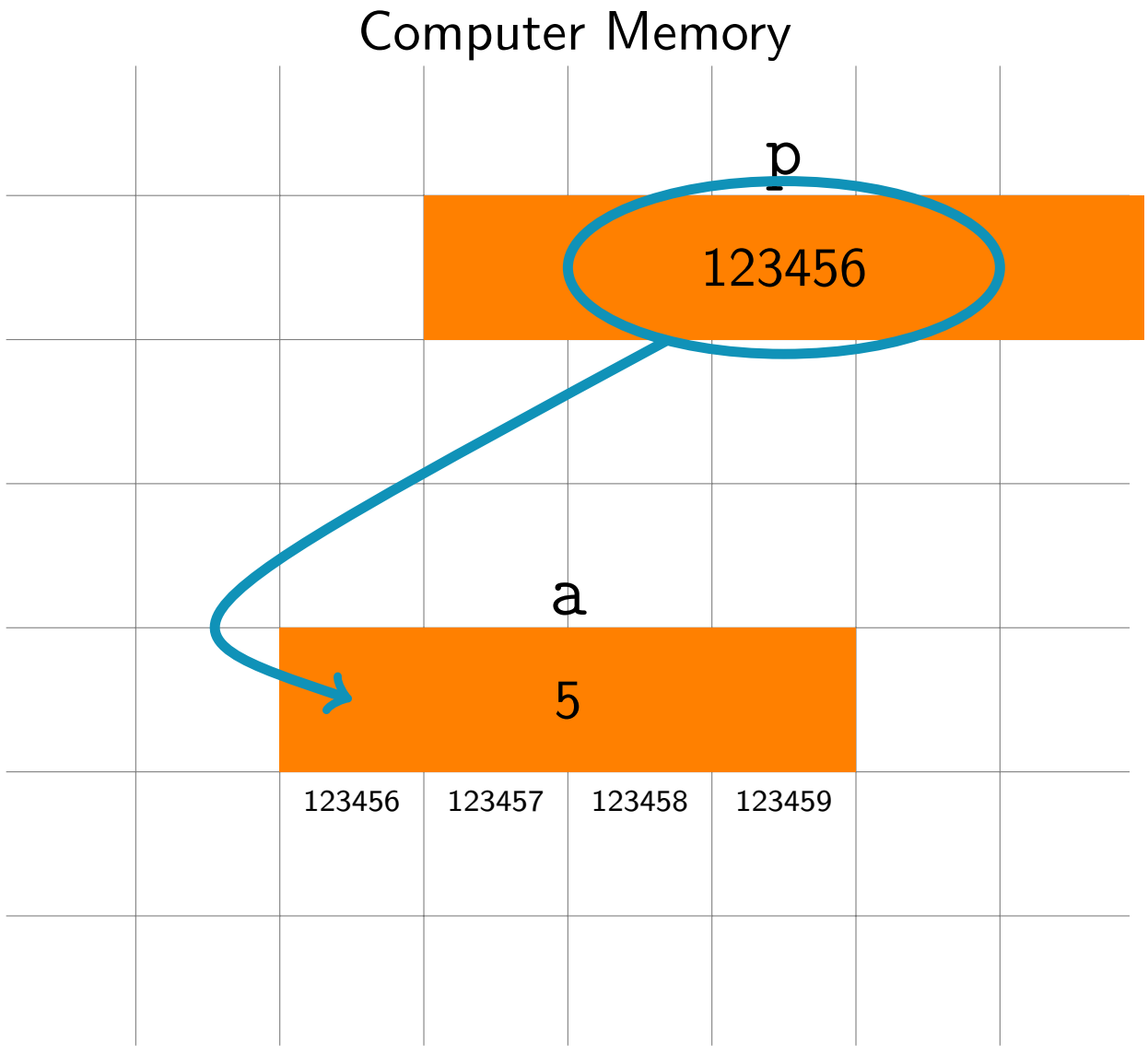
```
int a;
int *p;
a = 5;
p = &a;
 p
```



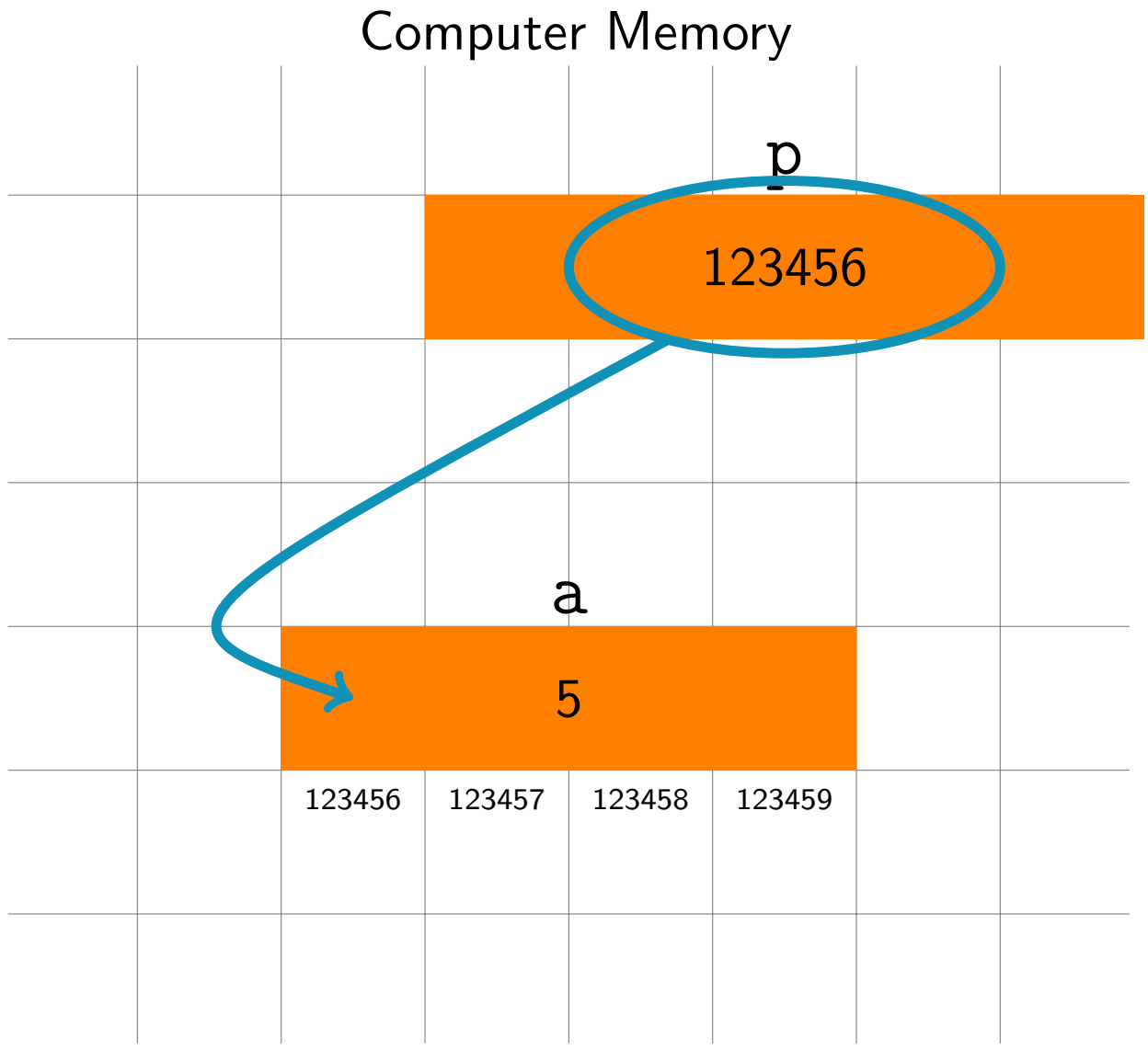
```
int a; ✓
int *p; ✓
a = 5; ✓
p = &a; ✓
*p
```



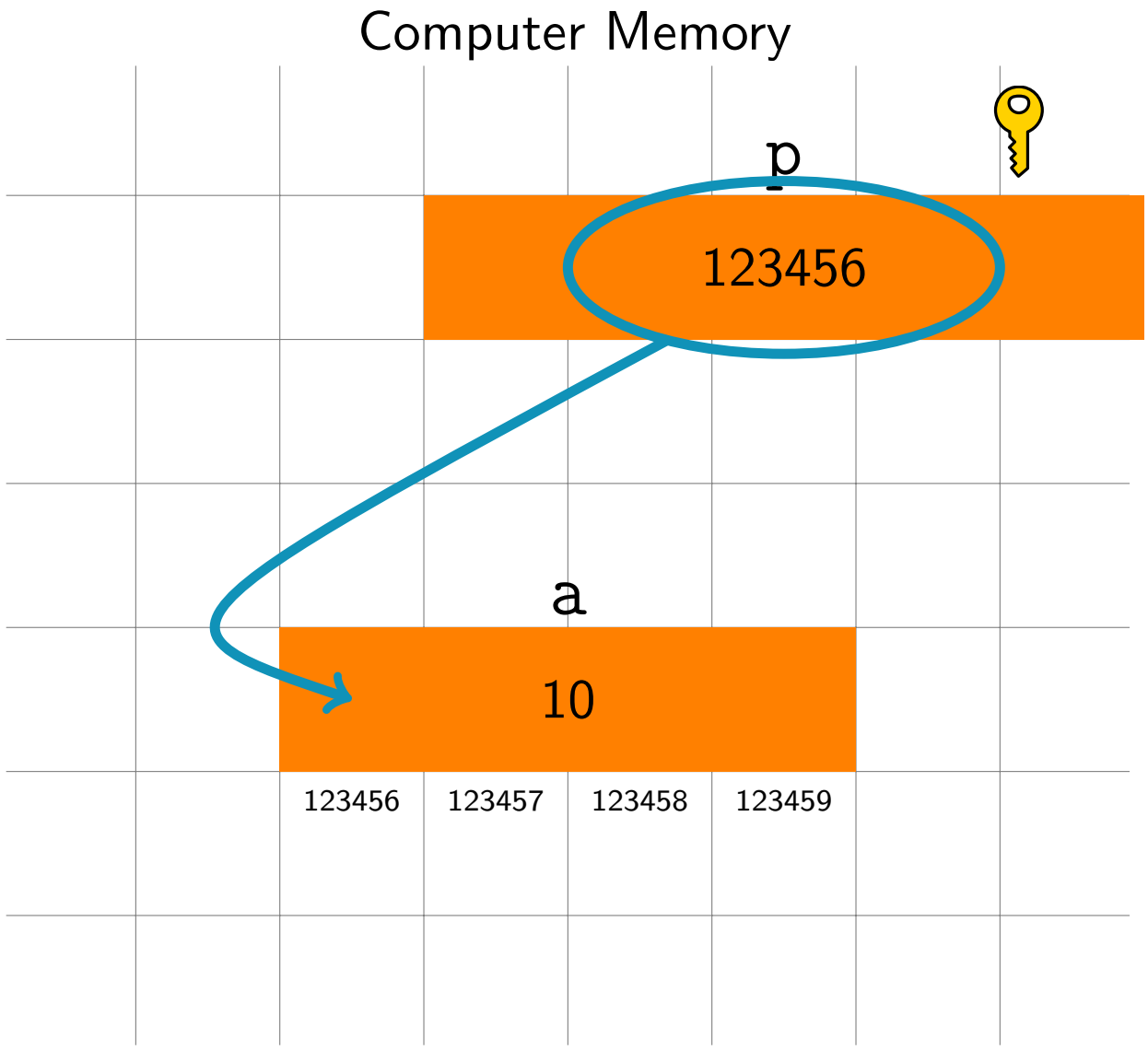
```
int a; ✓
int *p; ✓
a = 5; ✓
p = &a; ✓
*p
```



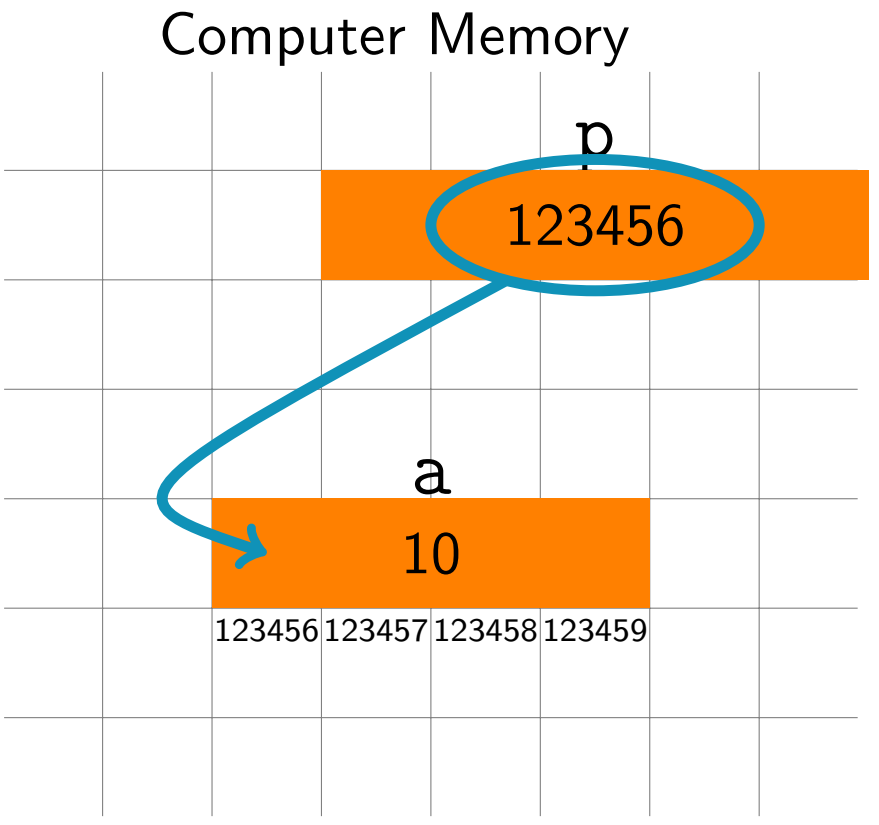
```
int a; ✓
int *p; ✓
a = 5; ✓
p = &a; ✓
*p = 10;
```



```
int a;
int *p;
a = 5;
p = &a;
*p = 10;
```

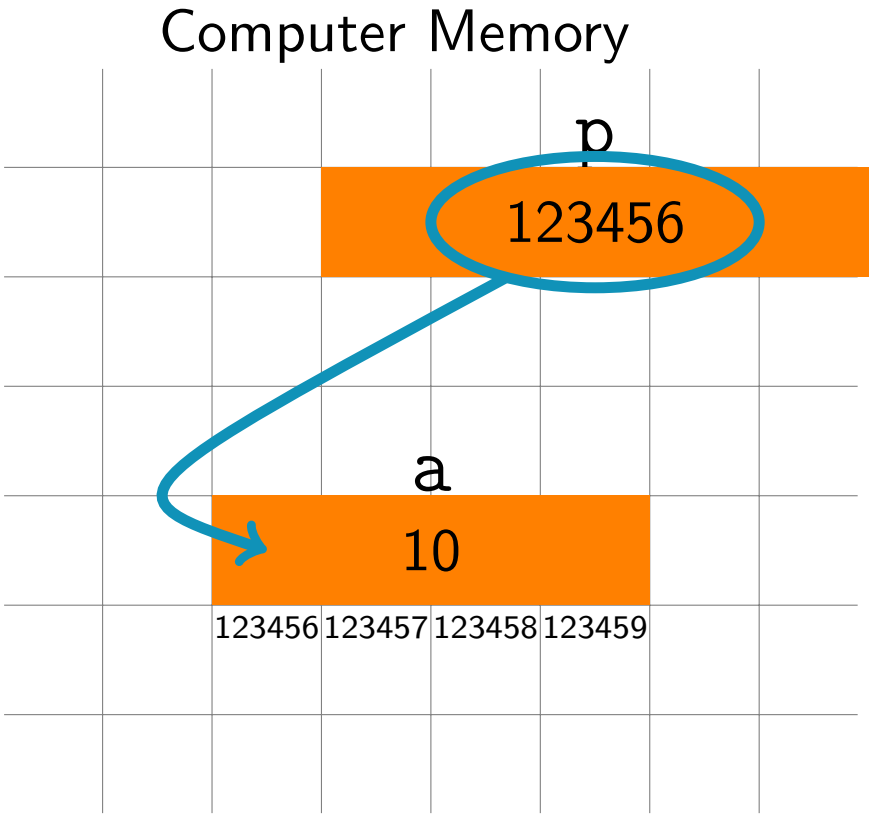


Encore!



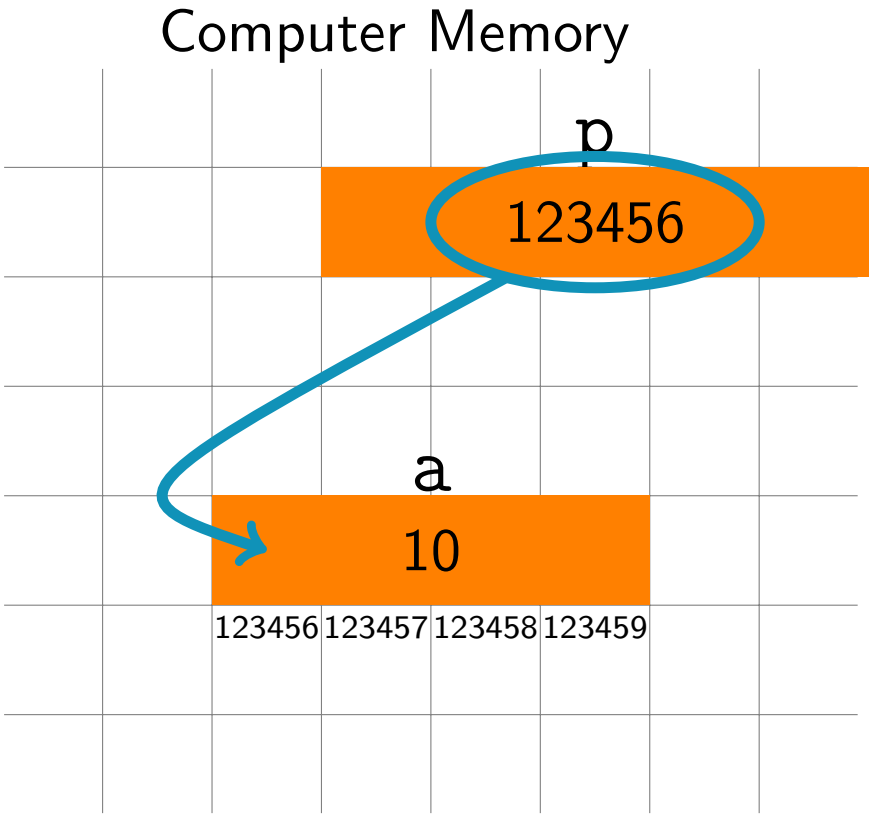
Encore!

- Every variable has some value.



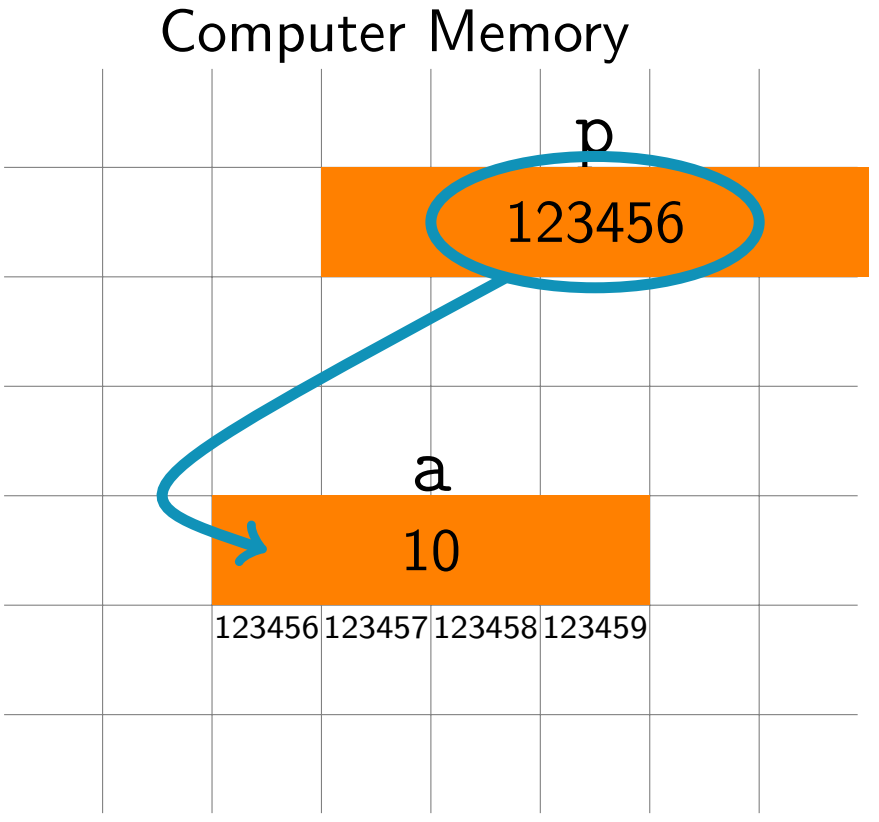
Encore!

- Every variable has some value.
  - `printf("%d", a)`



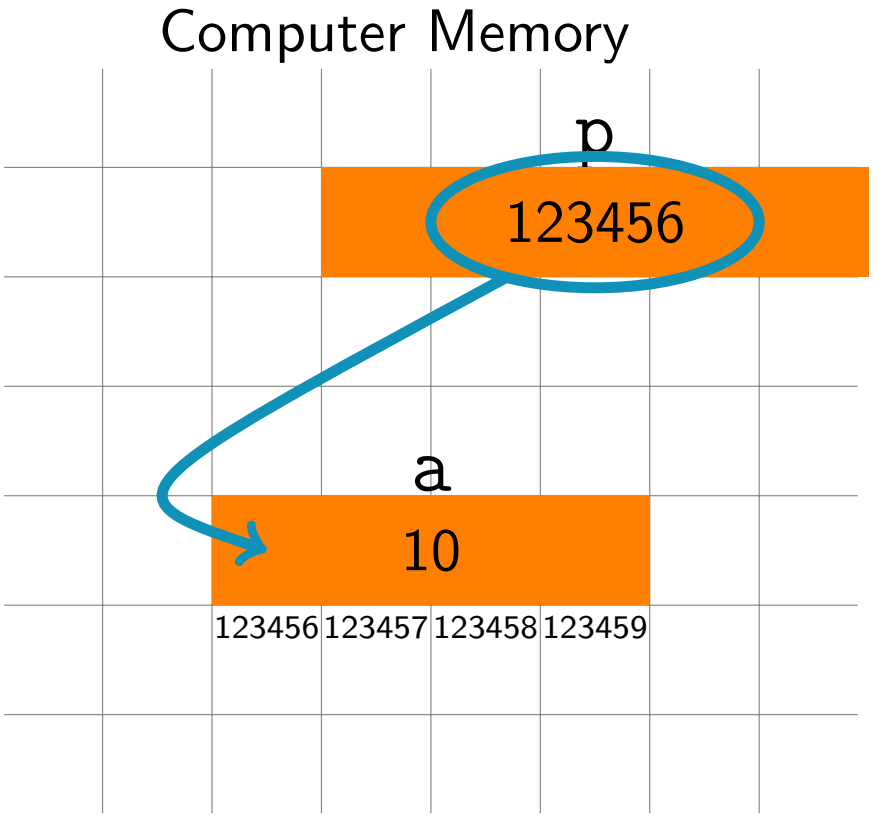
Encore!

- Every variable has some value.
  - `printf("%d", a)` $\rightarrow$ 10



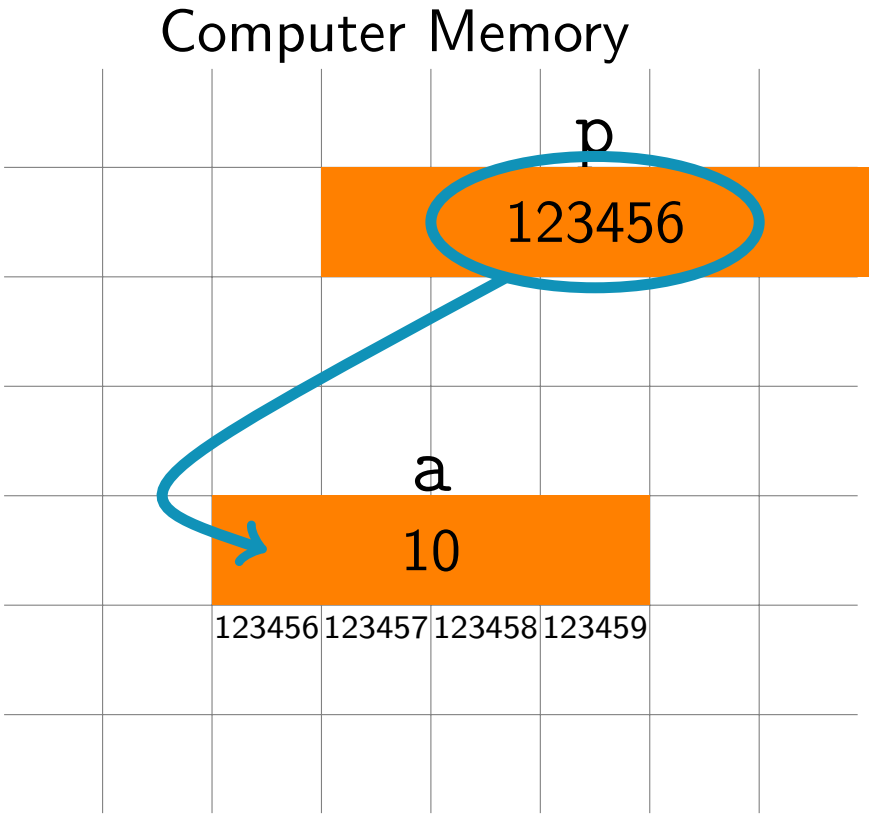
Encore!

- Every variable has some value.
  - `printf("%d", a)` $\rightarrow$ 10
  - `printf("  ", p)`



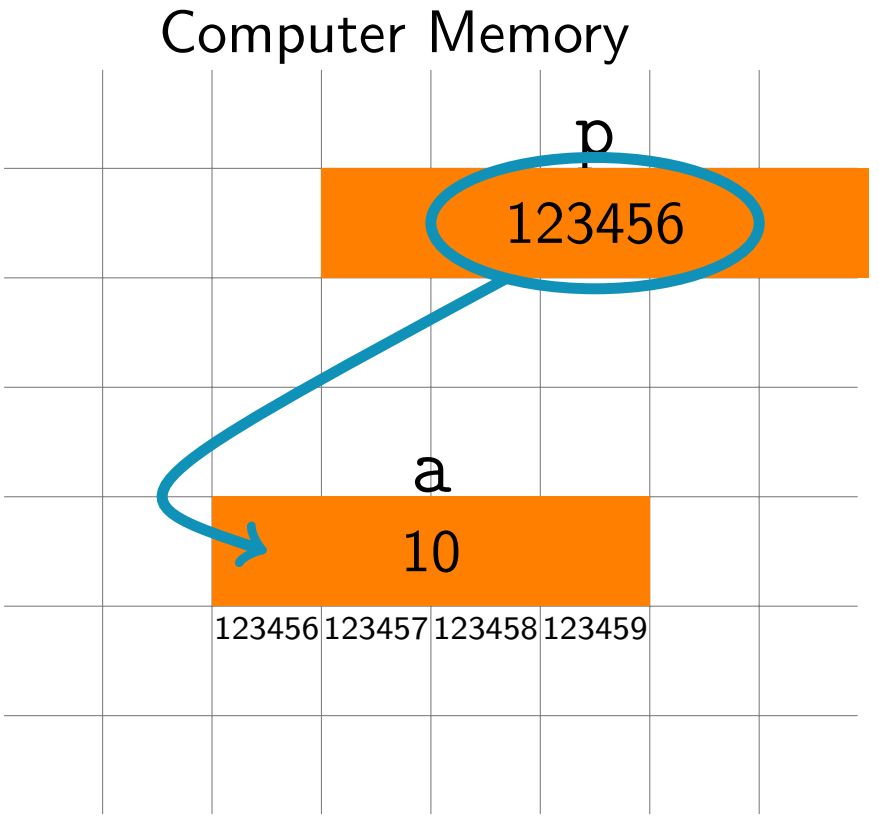
Encore!

- Every variable has some value.
  - `printf("%d", a)` $\rightarrow$ 10
  - `printf("%p", p)`



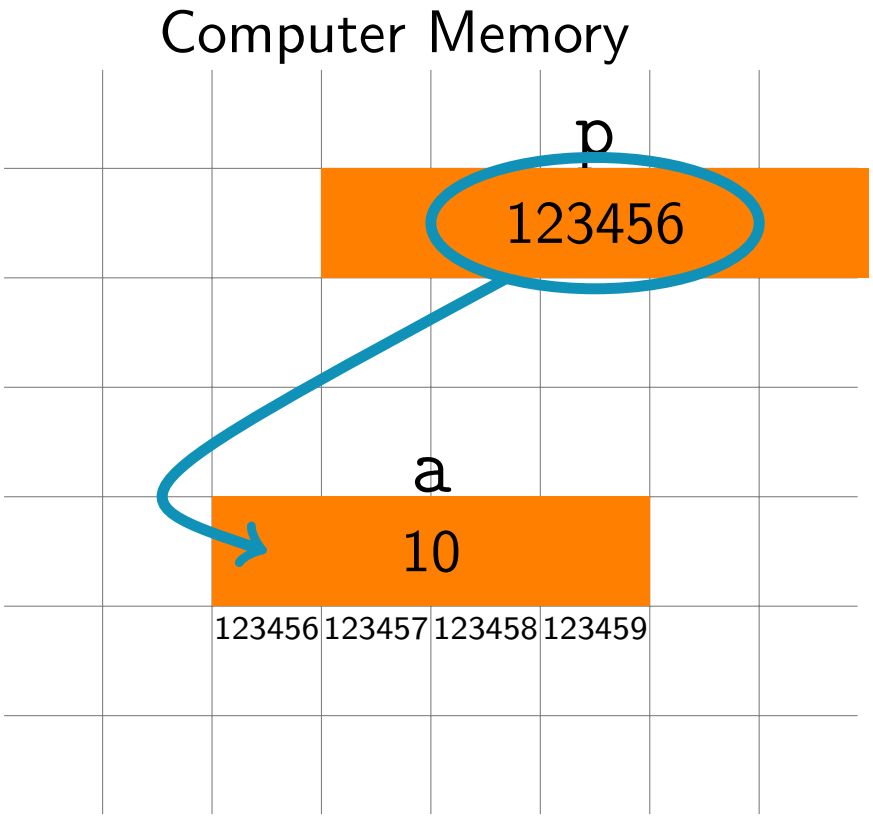
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456



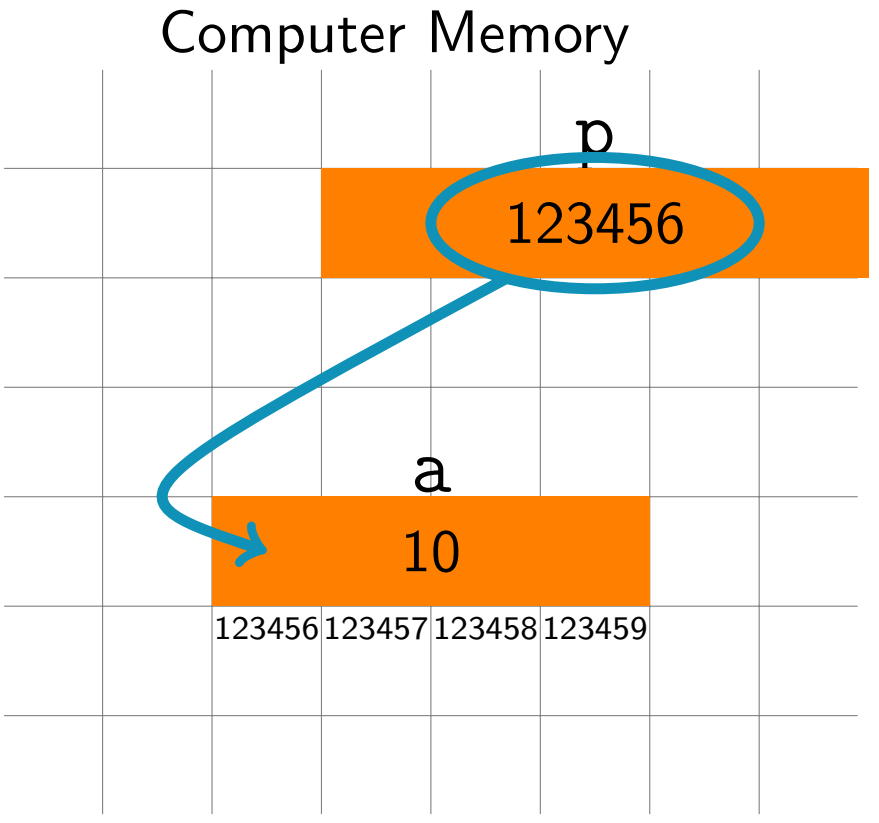
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.



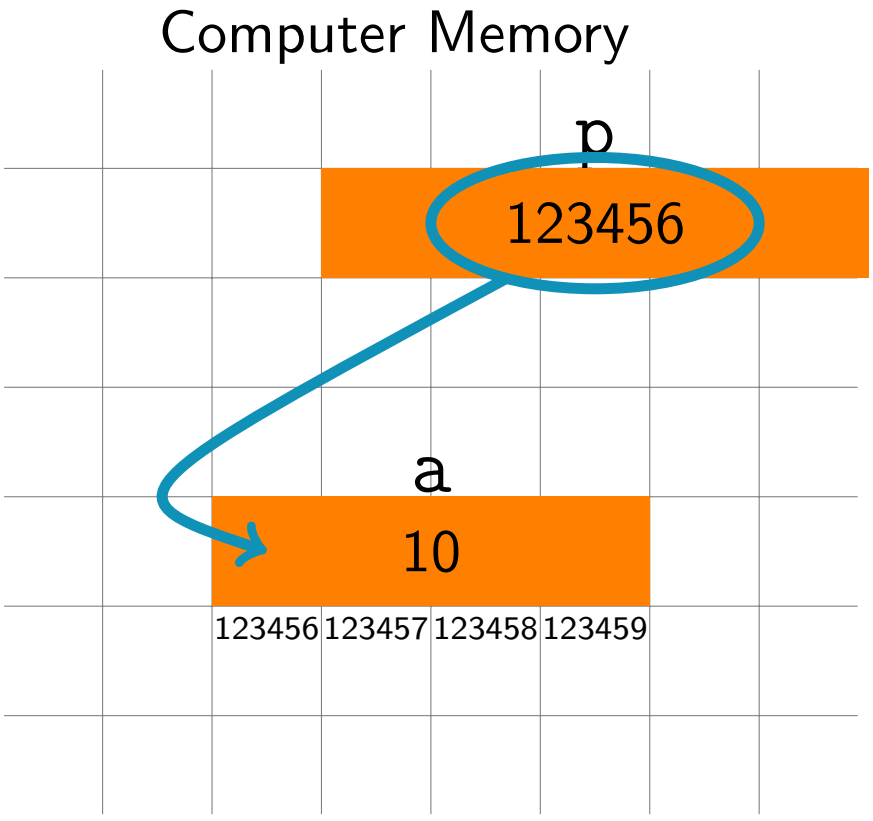
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.



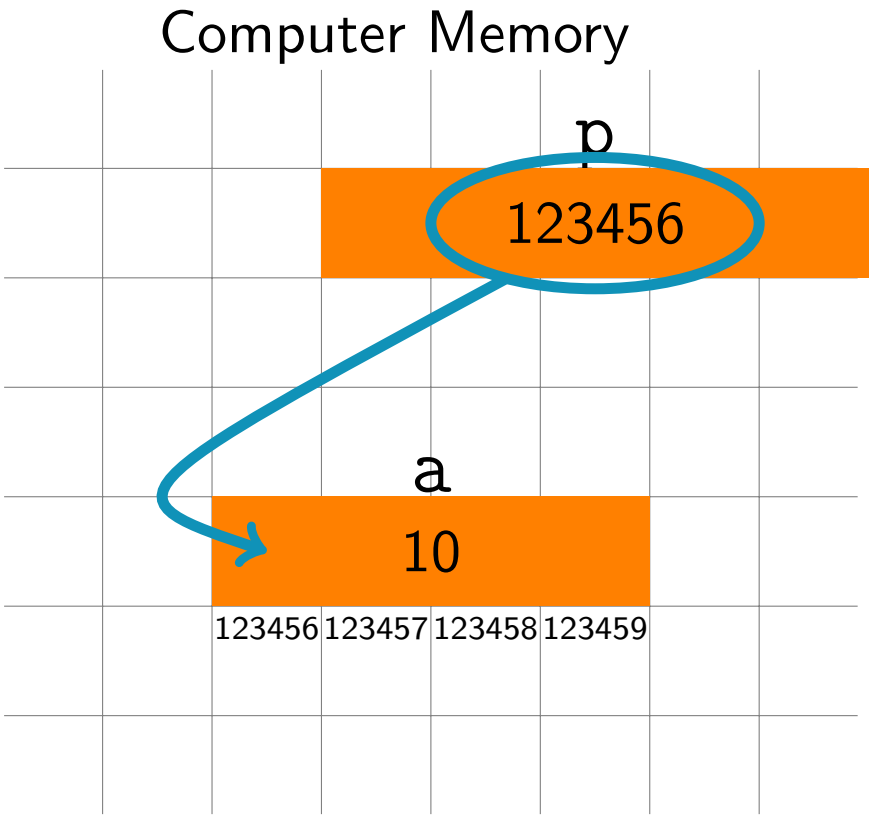
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("  ",  )`



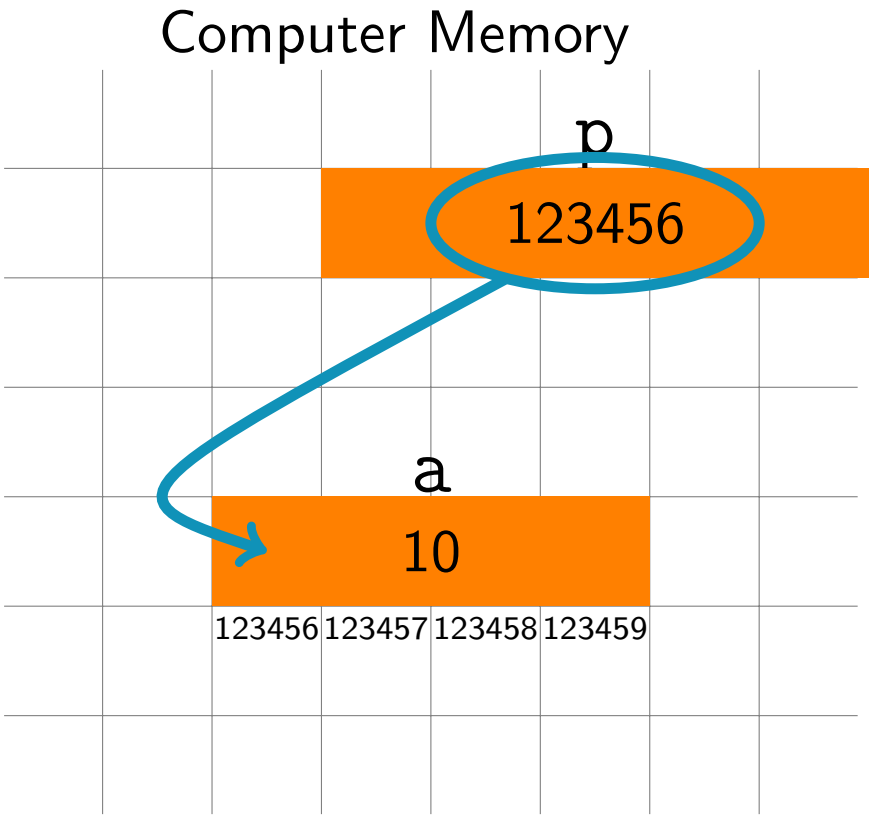
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("  ", &a)`



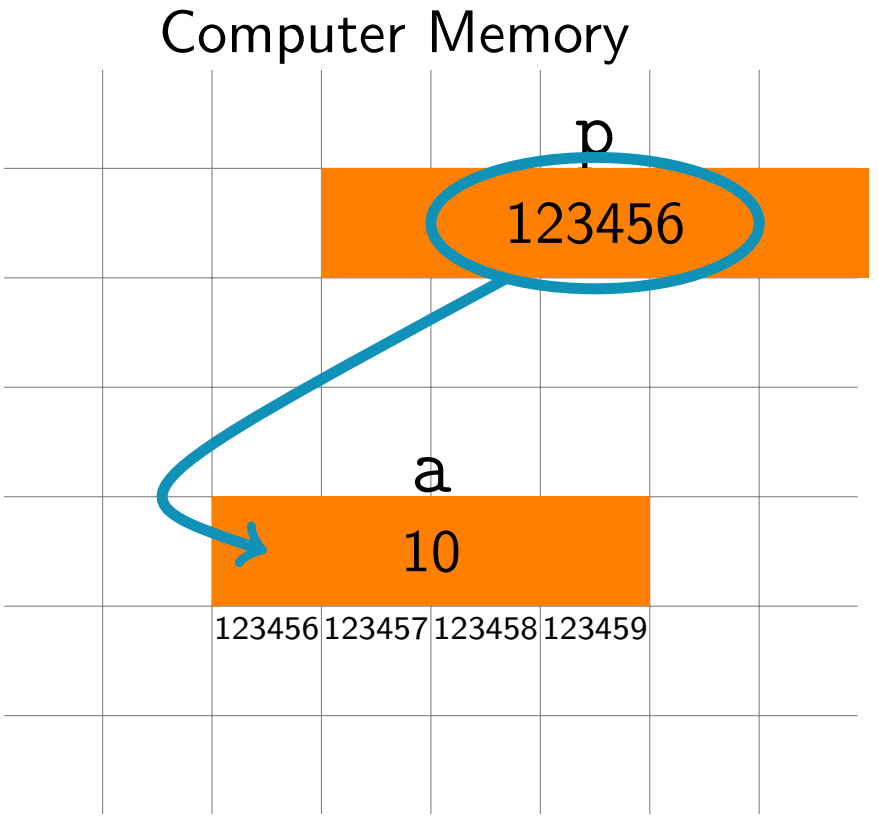
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("%p", &a)`



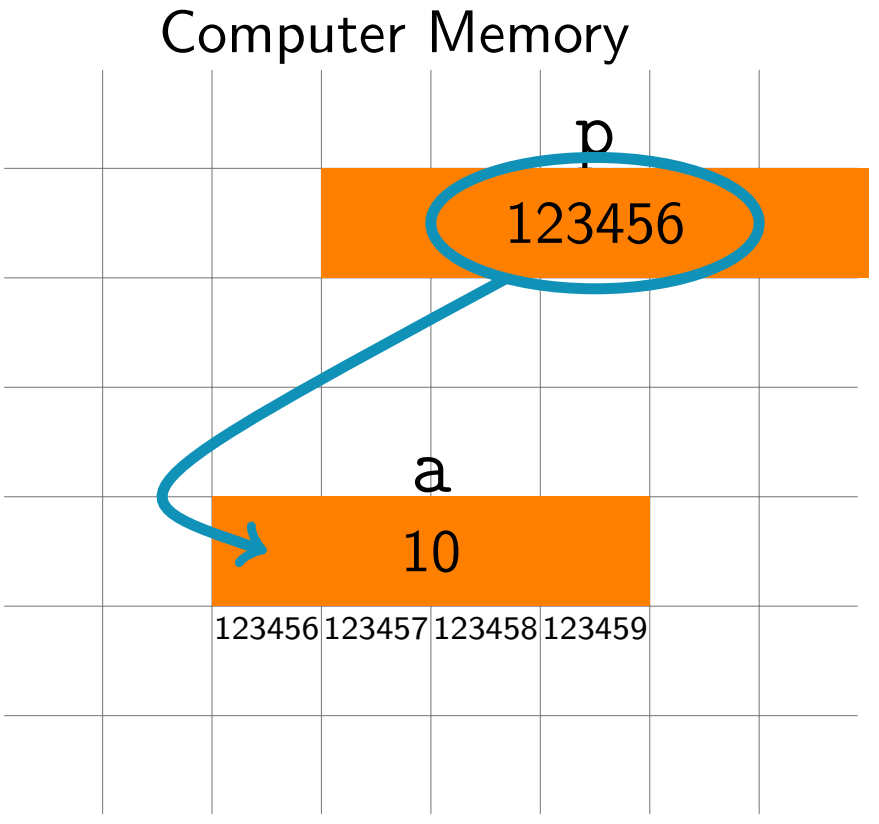
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("%p", &a)` → 123456



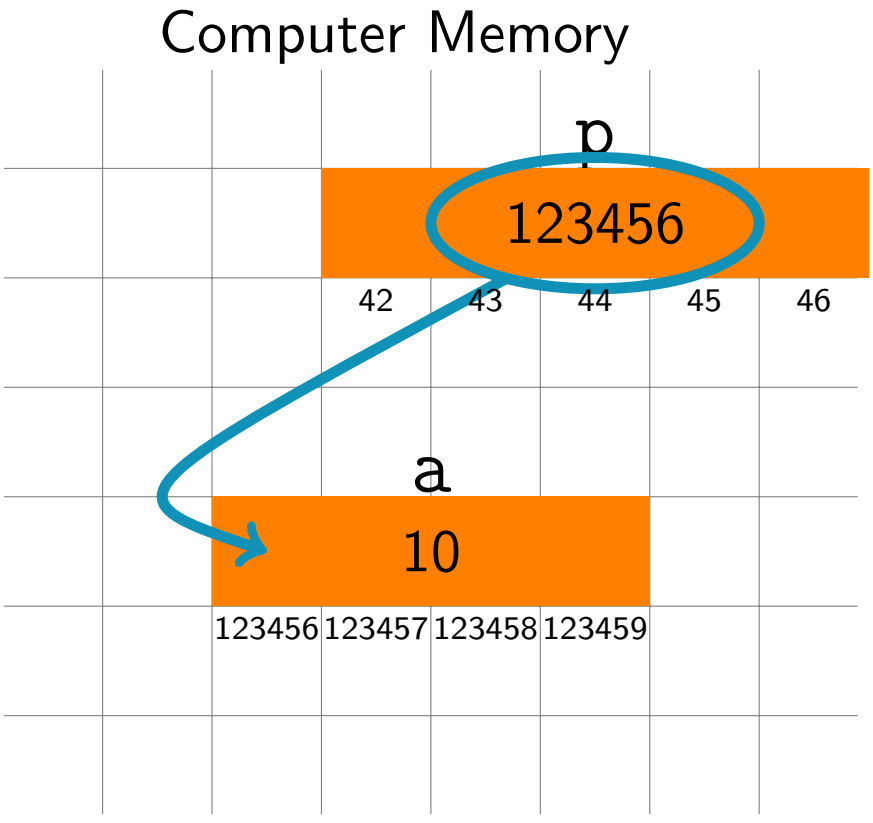
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("%p", &a)` → 123456
  - Address of `p`?



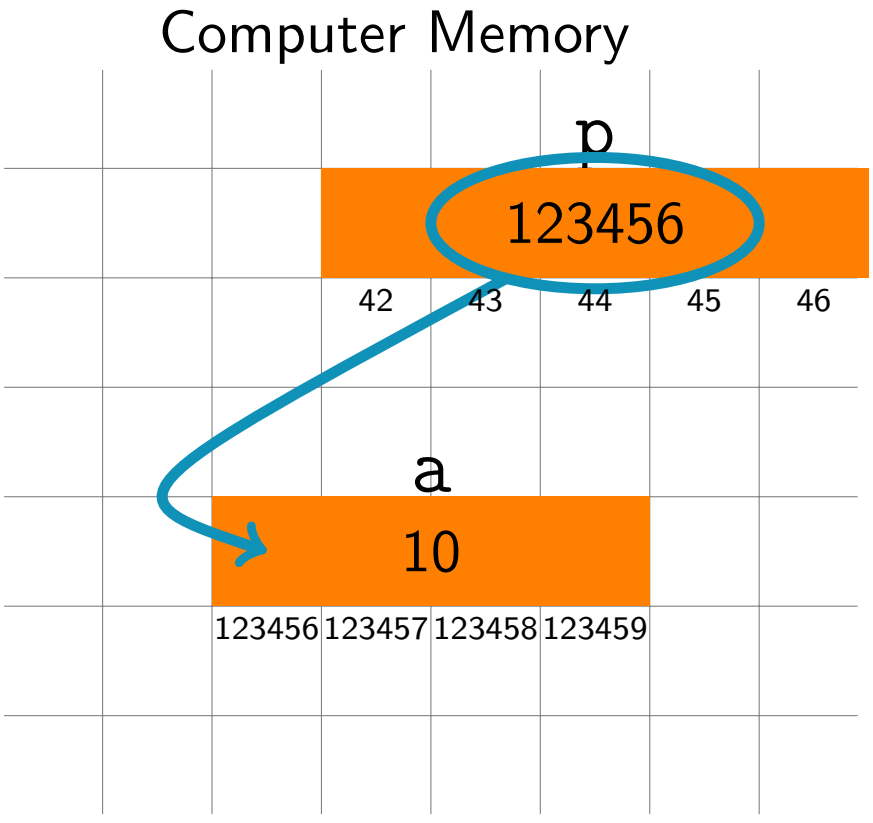
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("%p", &a)` → 123456
  - Address of `p`?



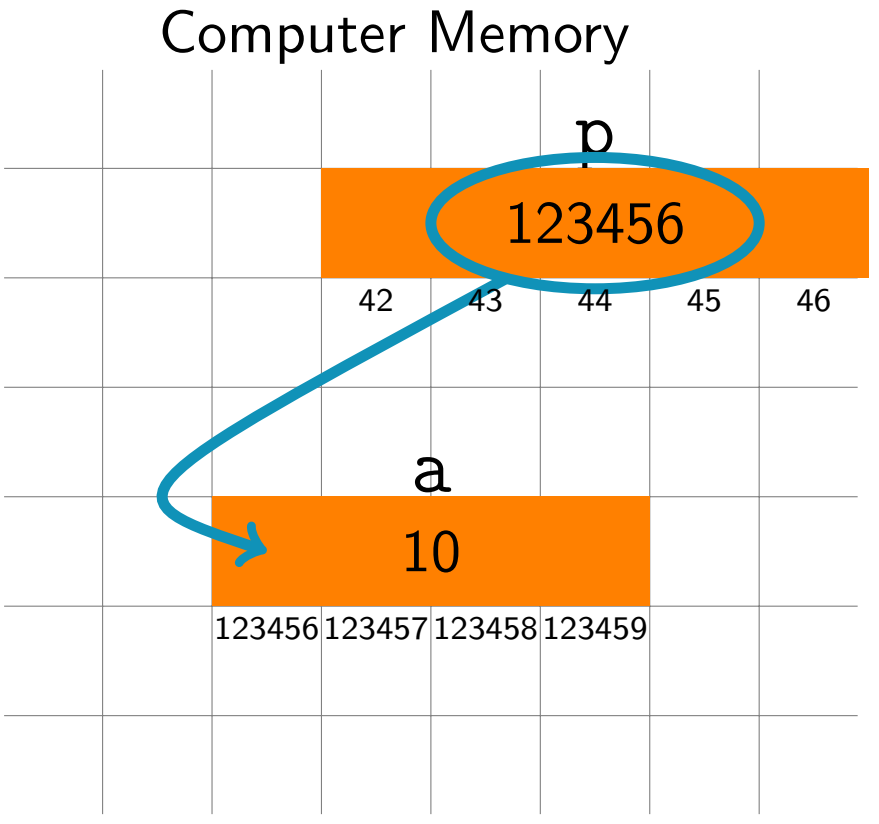
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("%p", &a)` → 123456
  - Address of `p` is 42 to 46.



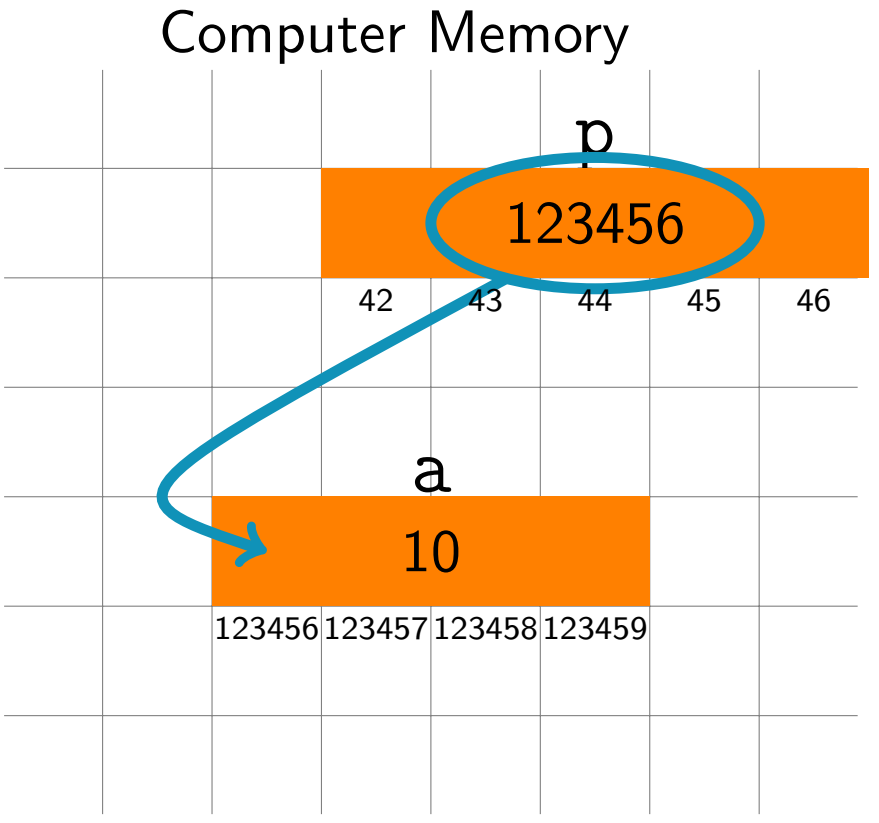
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("%p", &a)` → 123456
  - Address of `p` is 42 to 46.
    `printf("  ",  )`



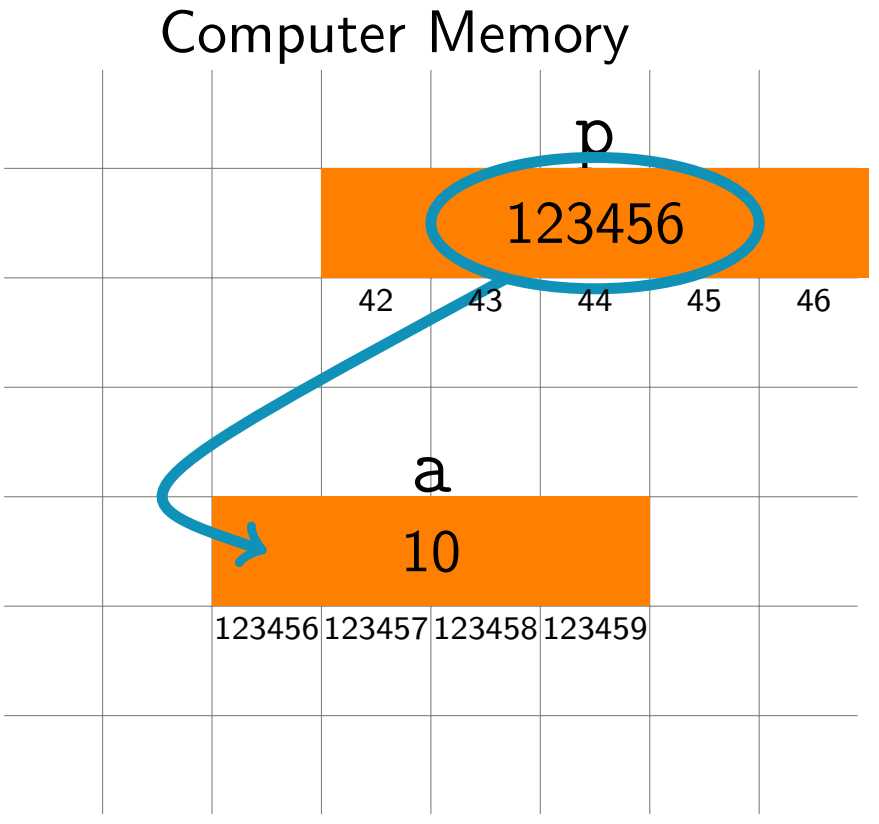
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("%p", &a)` → 123456
  - Address of `p` is 42 to 46.
    `printf("  ", &p)`



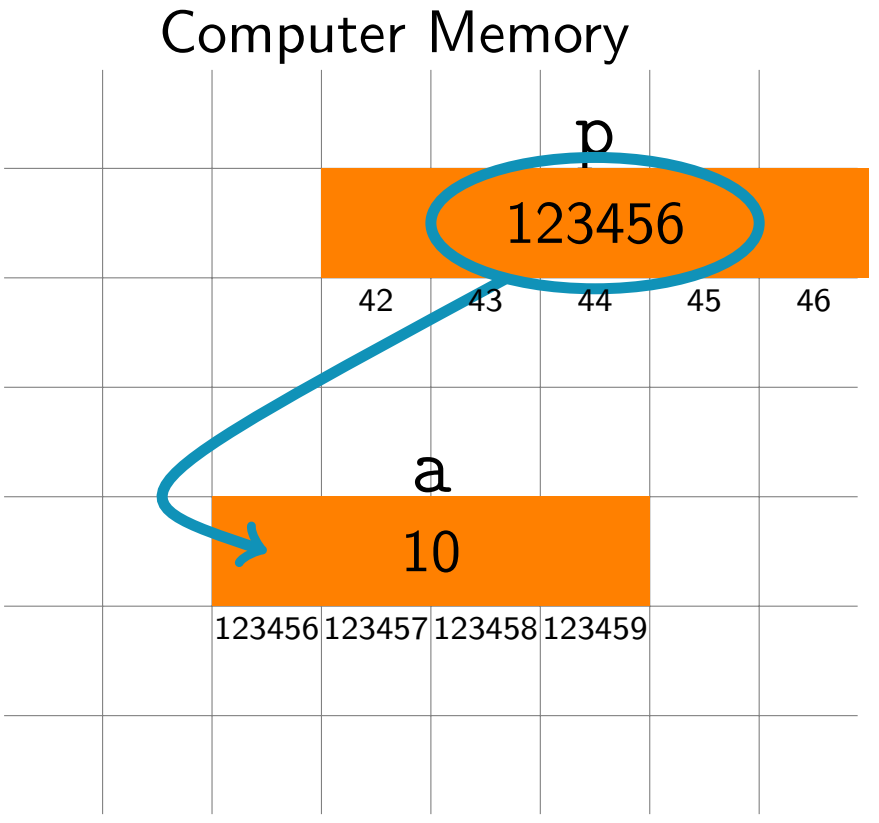
Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("%p", &a)` → 123456
  - Address of `p` is 42 to 46.
    `printf("%p", &p)`



Encore!

- Every variable has some value.
  - `printf("%d", a)` → 10
  - `printf("%p", p)` → 123456
- Every variable has an address.
  - Address of `a` is 123456 to 123459.
    `printf("%p", &a)` → 123456
  - Address of `p` is 42 to 46.
    `printf("%p", &p)` → 42



```
int a, b;
```

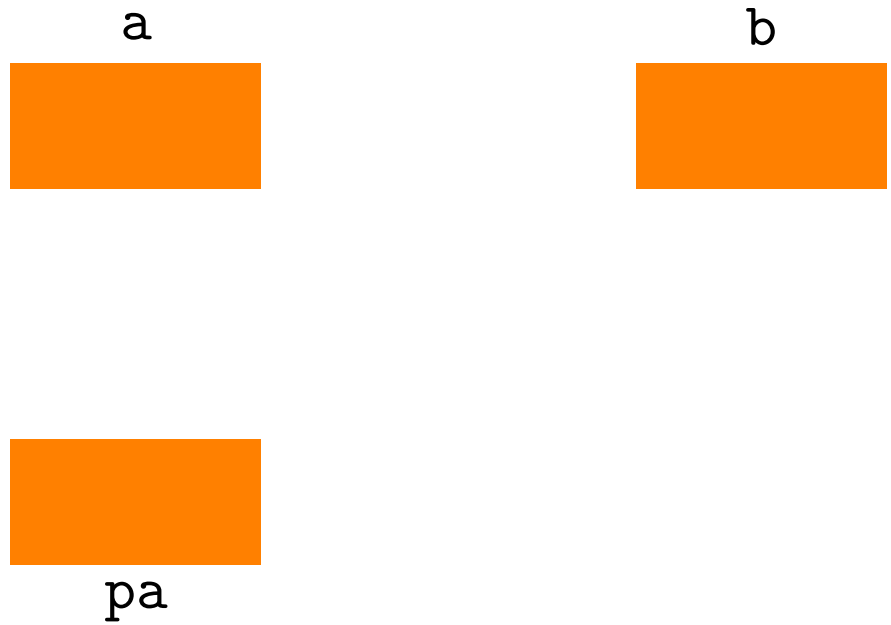
```
int a, b;
```

```
int a, b;
int *pa;
```

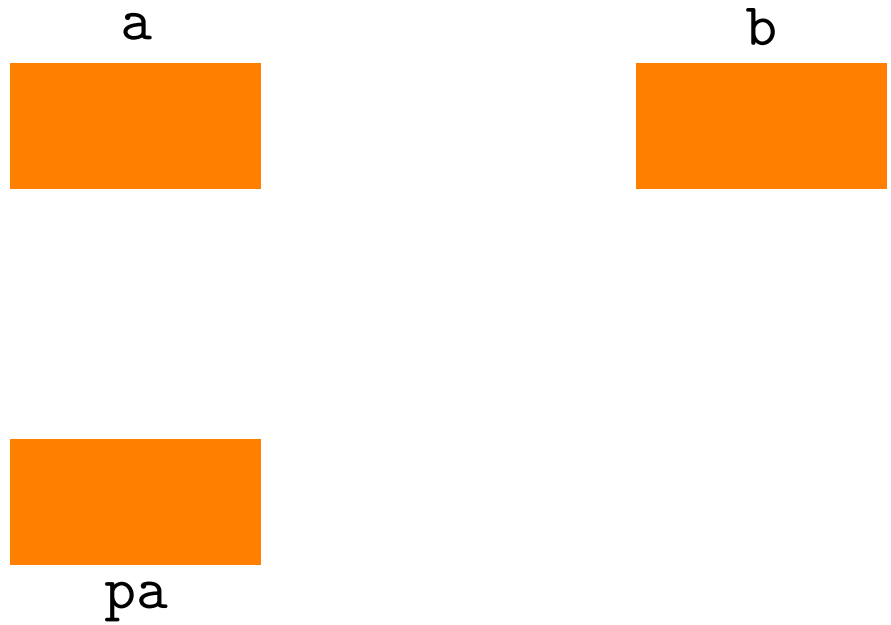
```
int a, b;
int *pa; /* pointer to a */
```

```
int a, b;
int *pa; /* address of a */
```

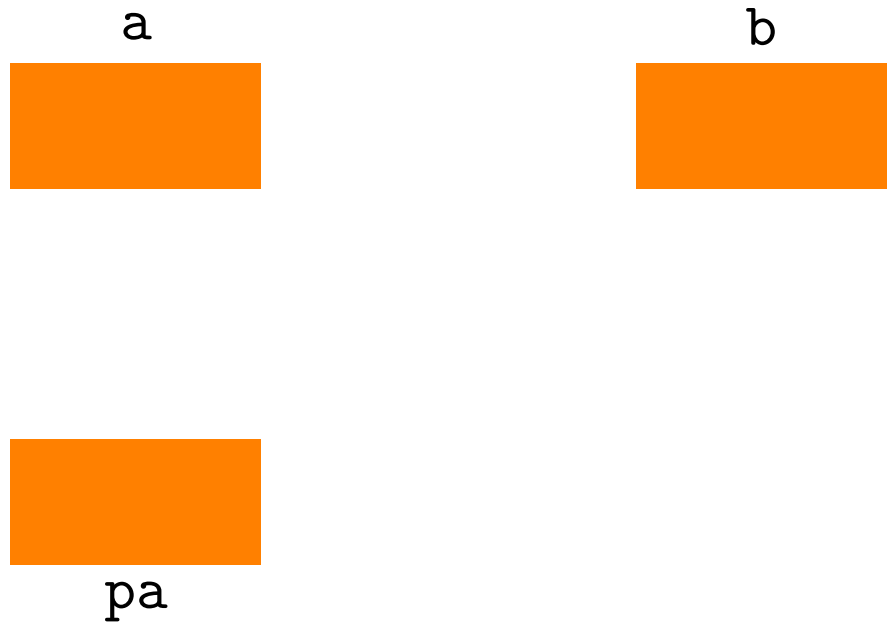
```
int a, b;
int *pa; /* address of a */
```



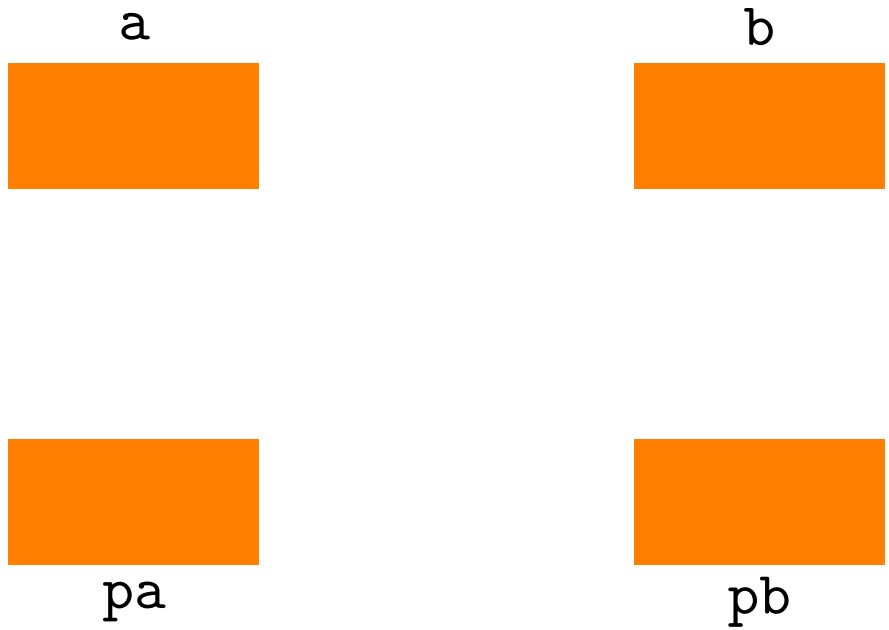
```
int a, b;
int *pa; /* address of a */
int *pb;
```



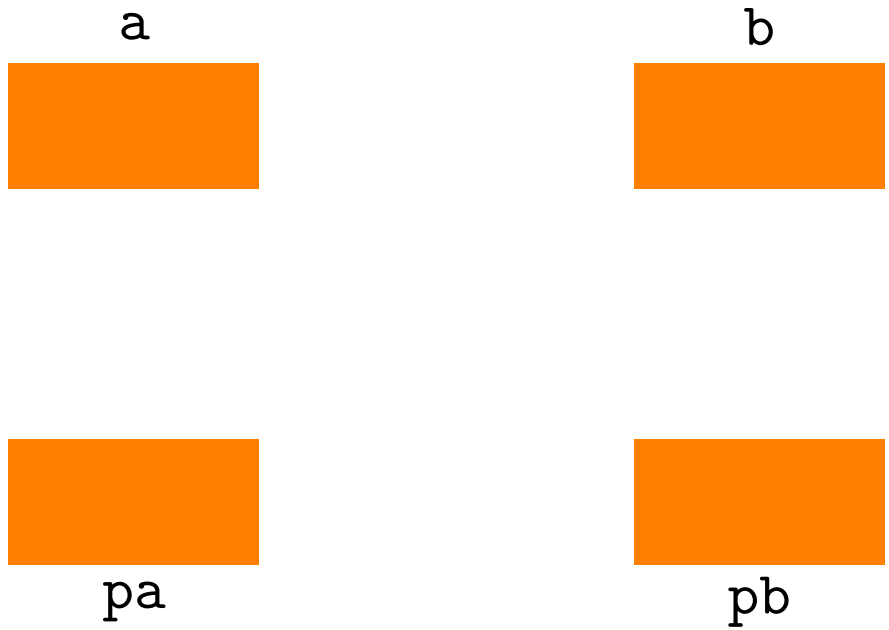
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
```



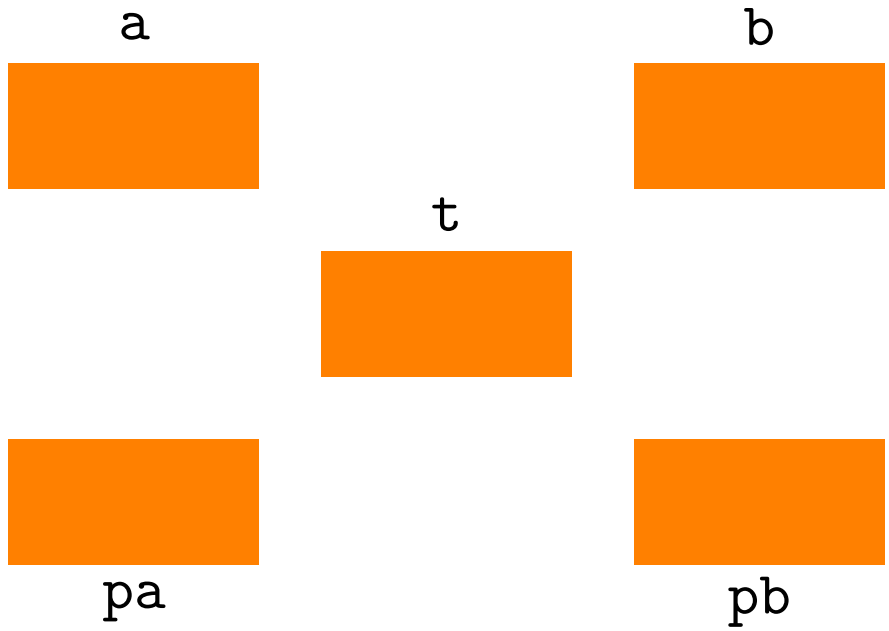
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
```



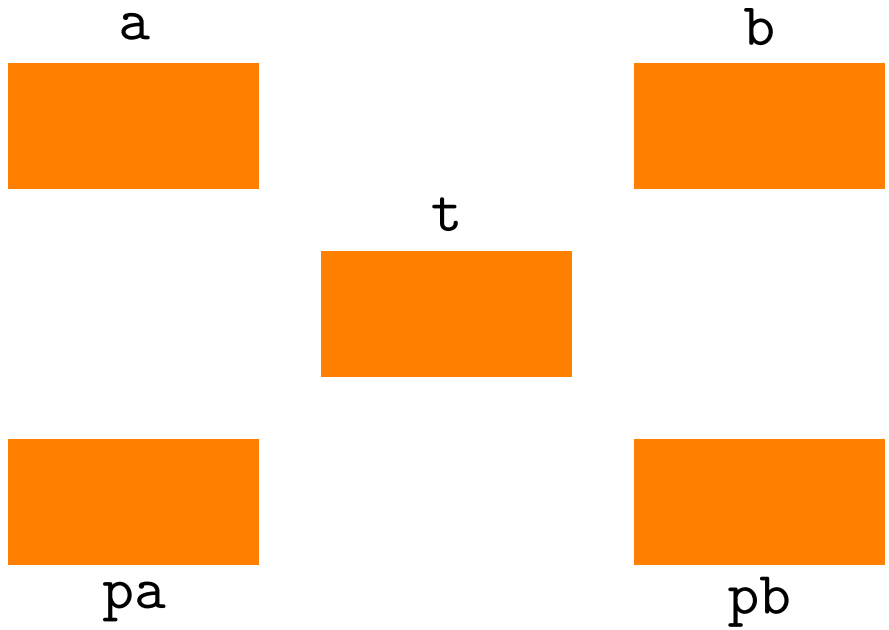
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
```



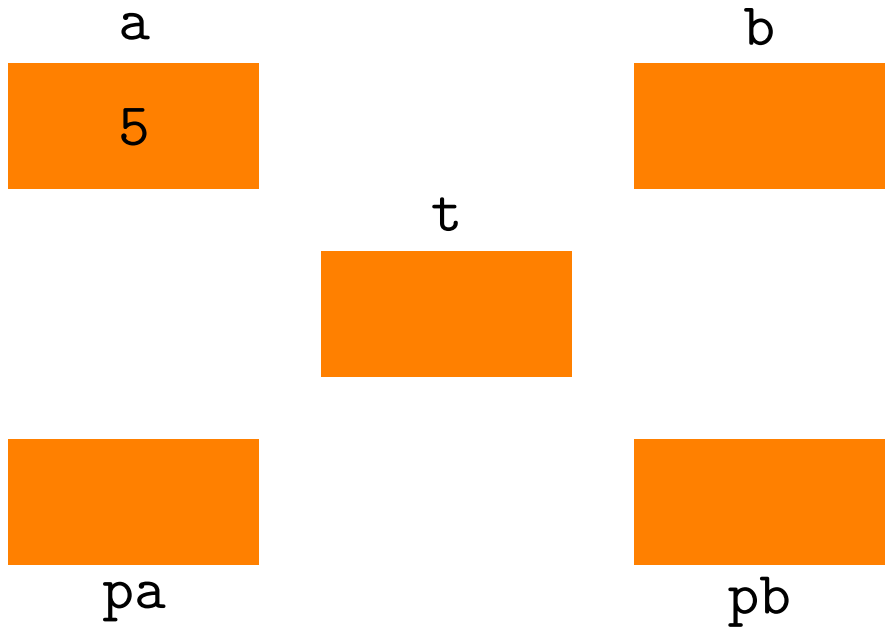
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
```



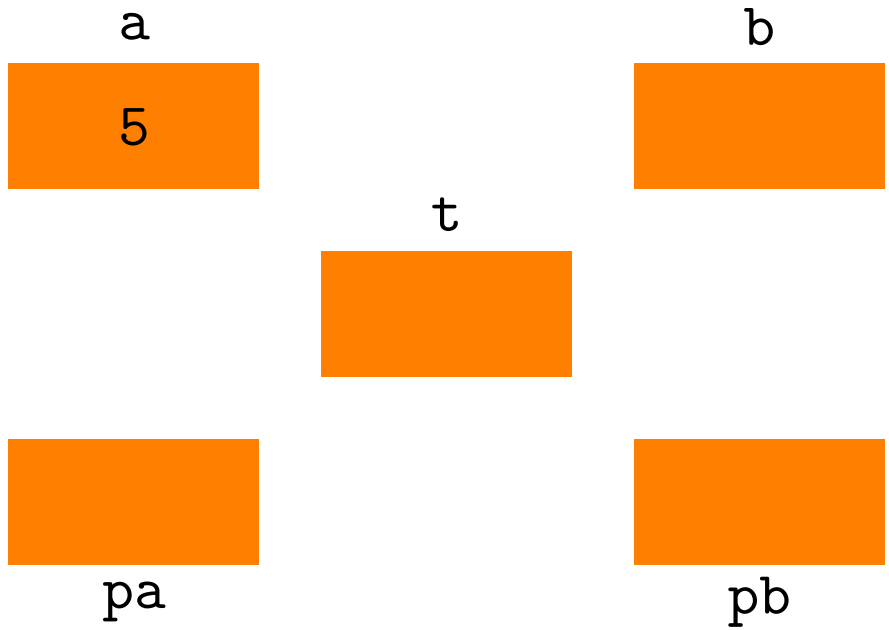
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
```



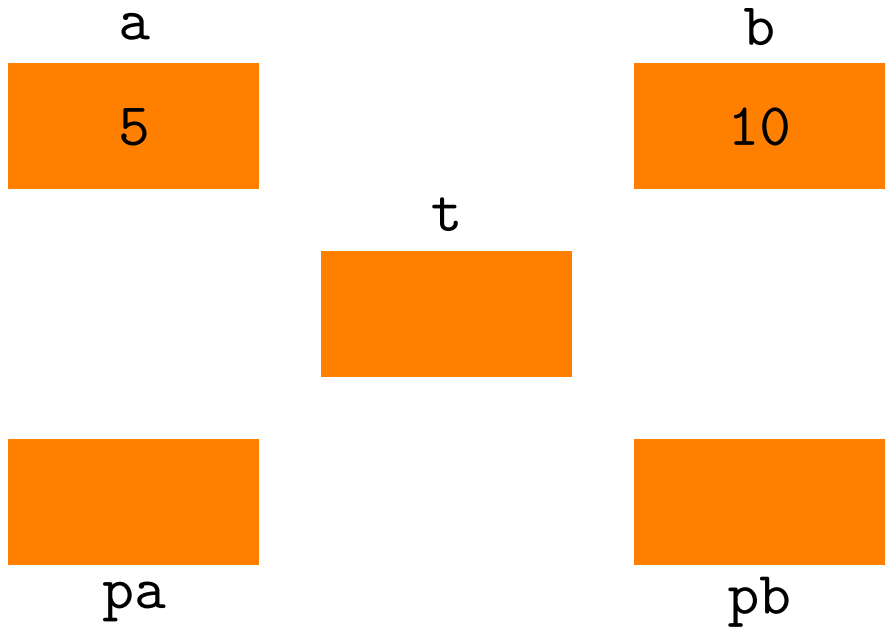
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
```



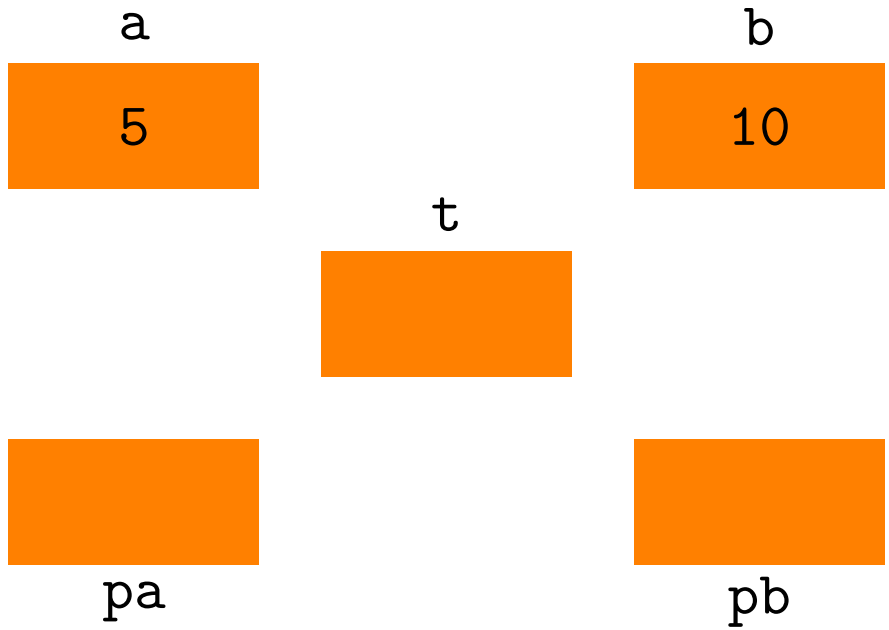
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
```



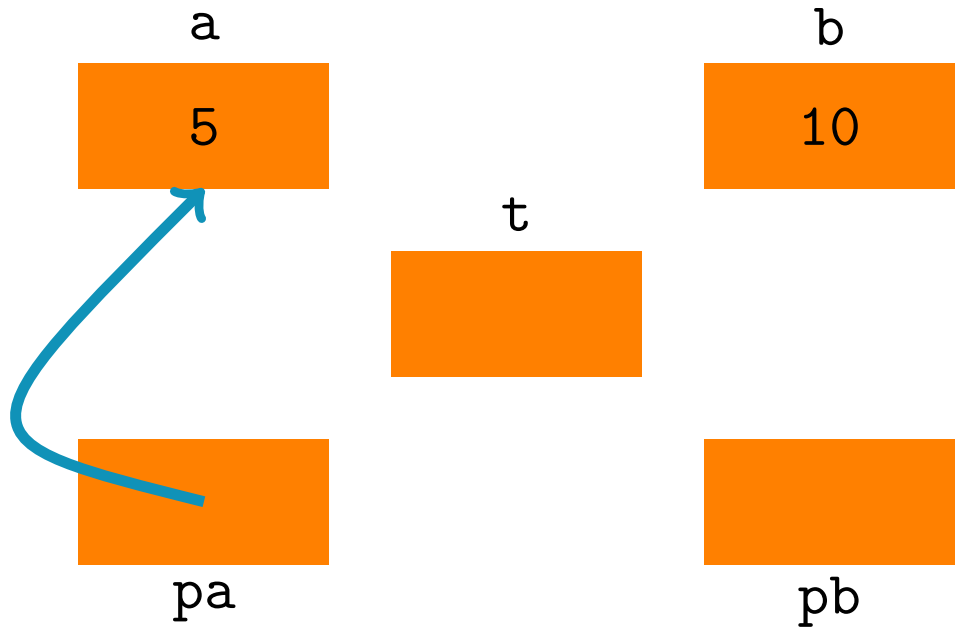
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
```



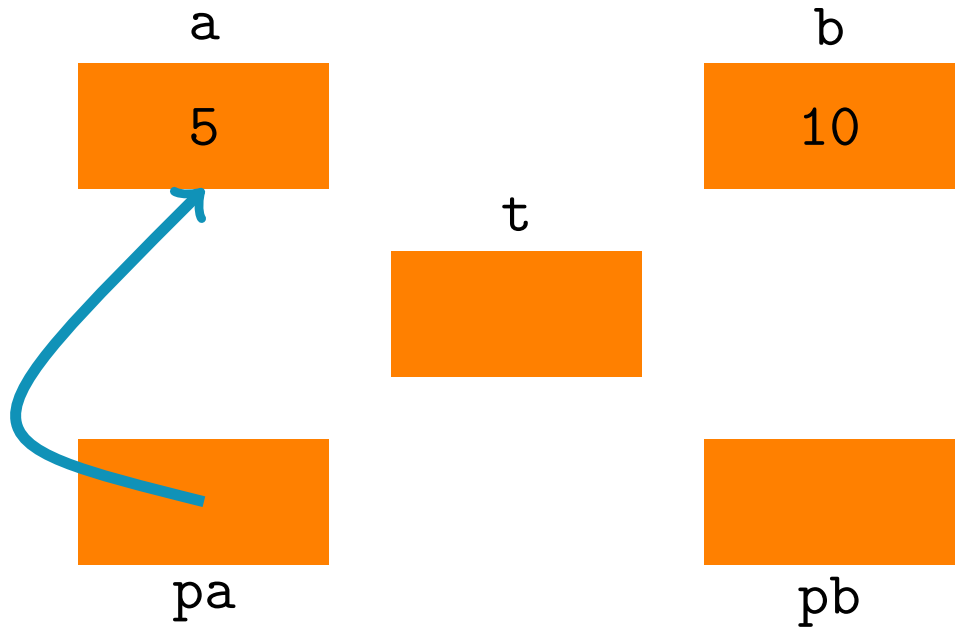
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
```



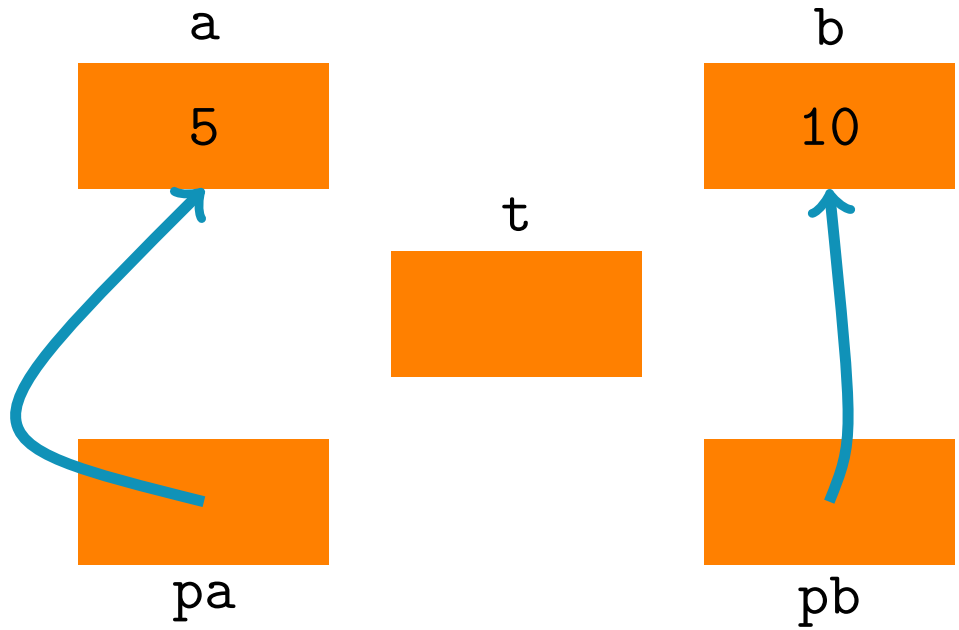
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
```



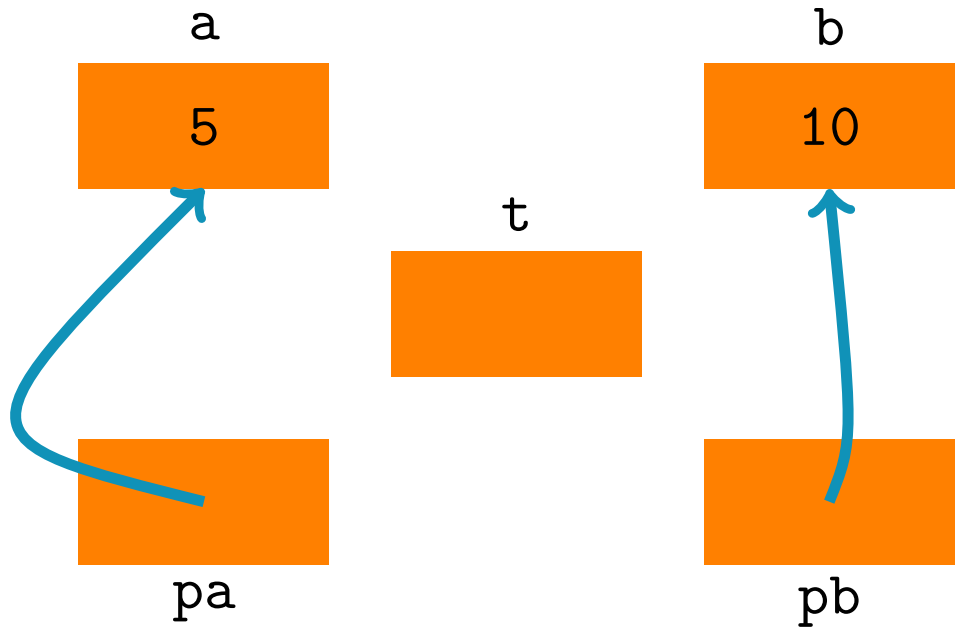
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```

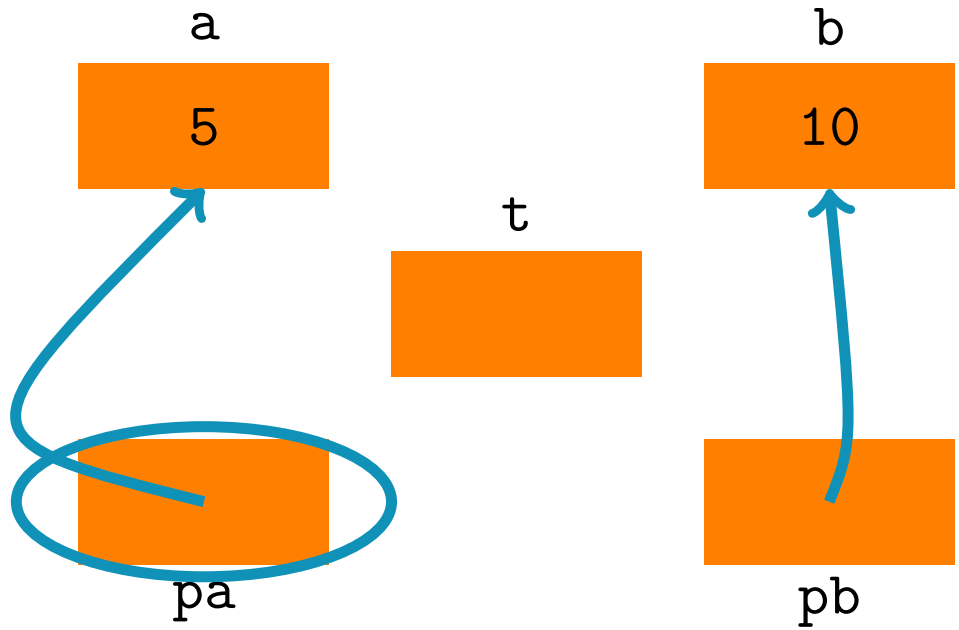


```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



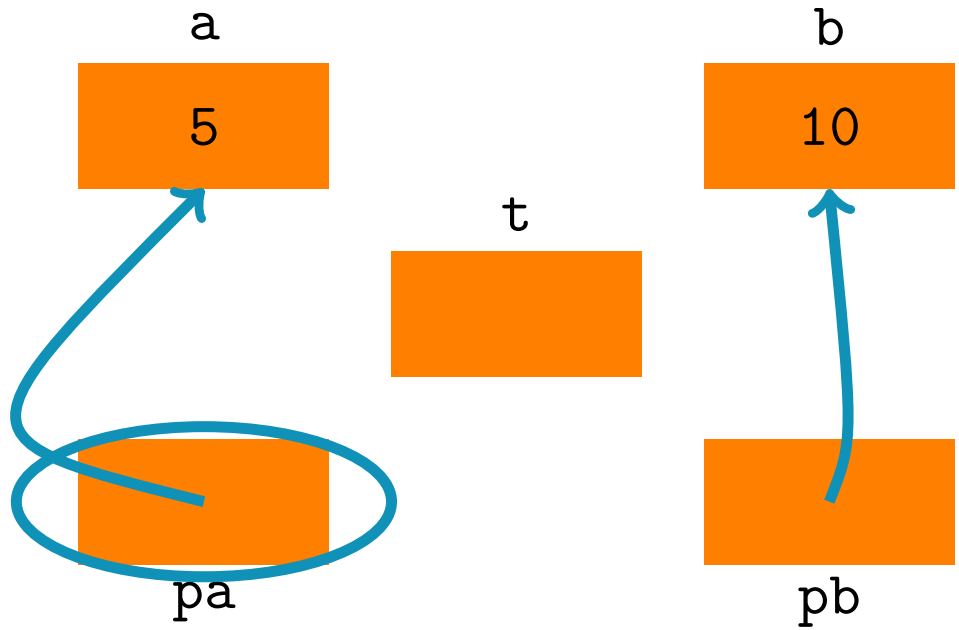
pa

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



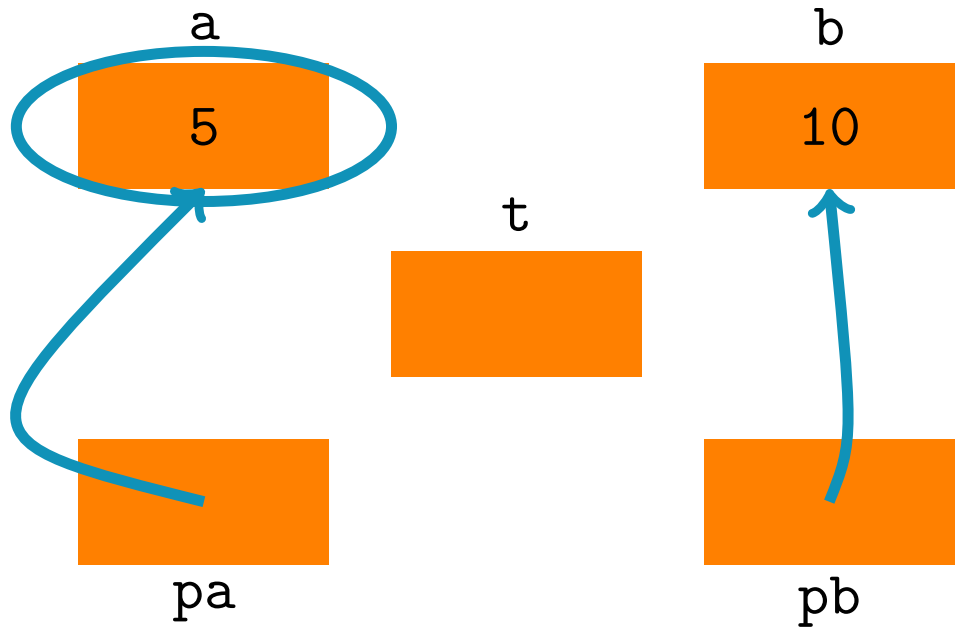
pa

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



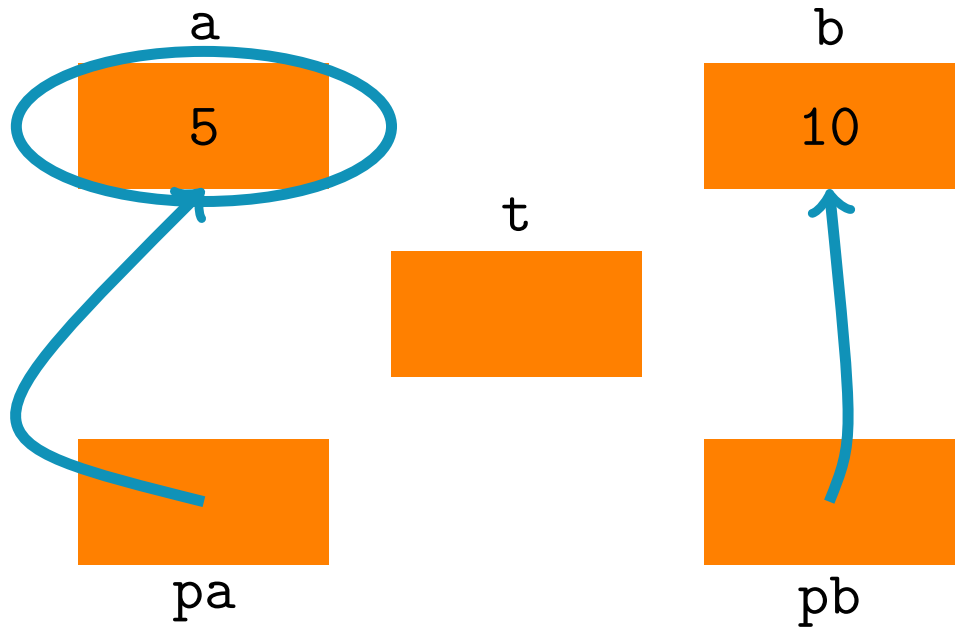

```
    *pa
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



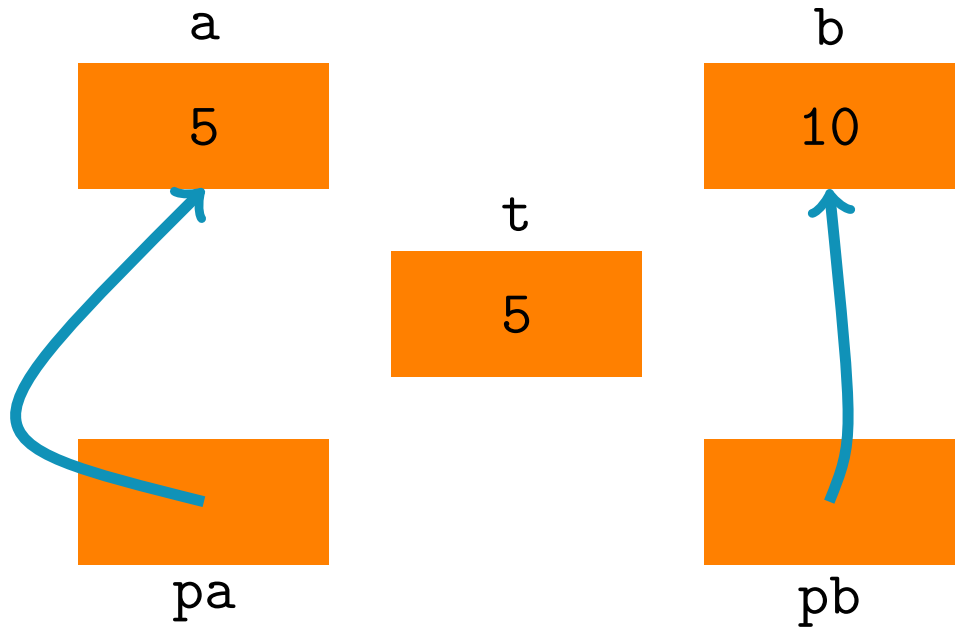

```
    *pa
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



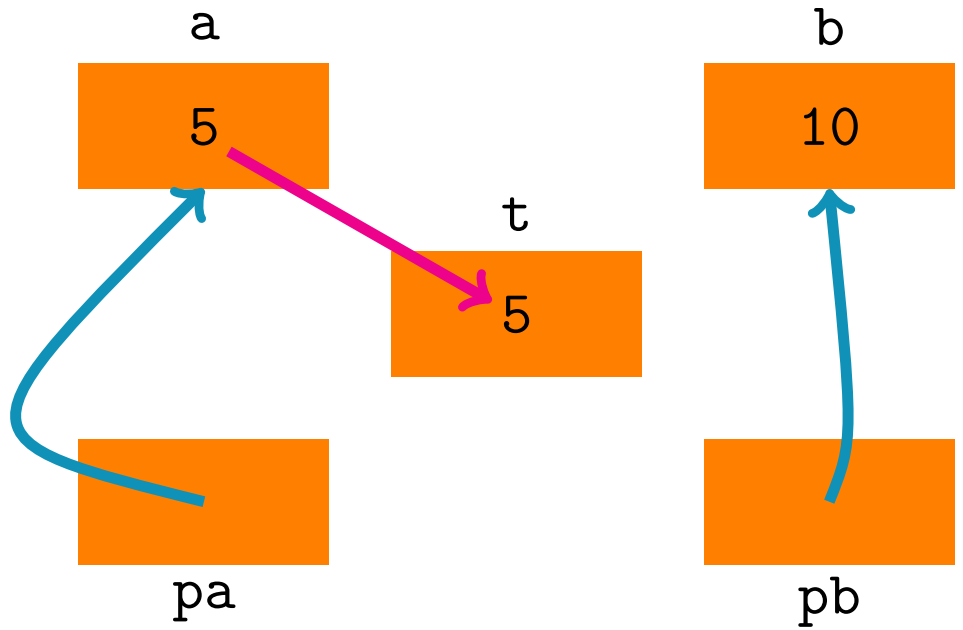

```
t = *pa;
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



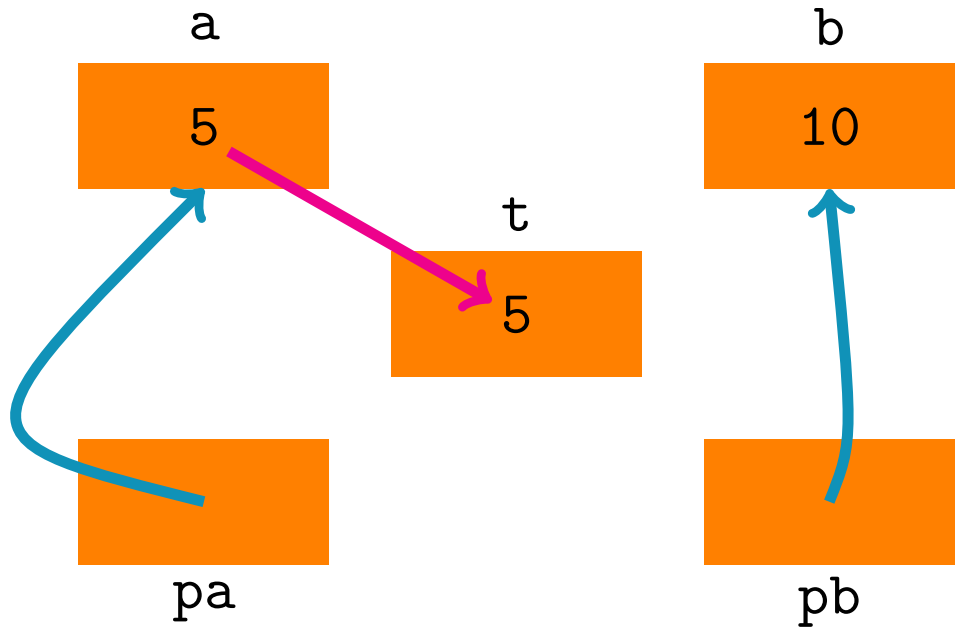

```
t = *pa;
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



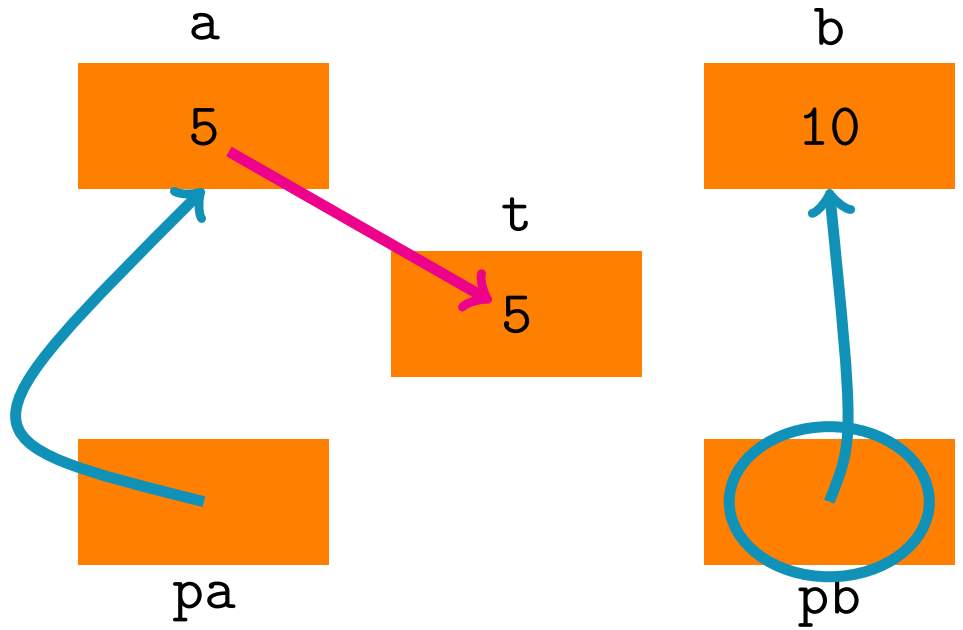

```
t = *pa;
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



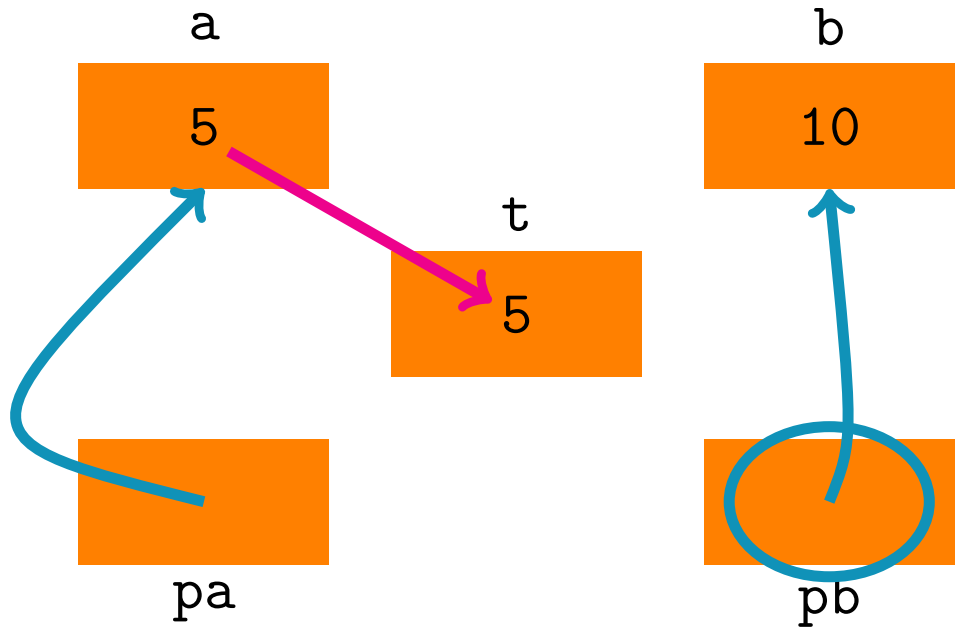

```
t = *pa;
        pb
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



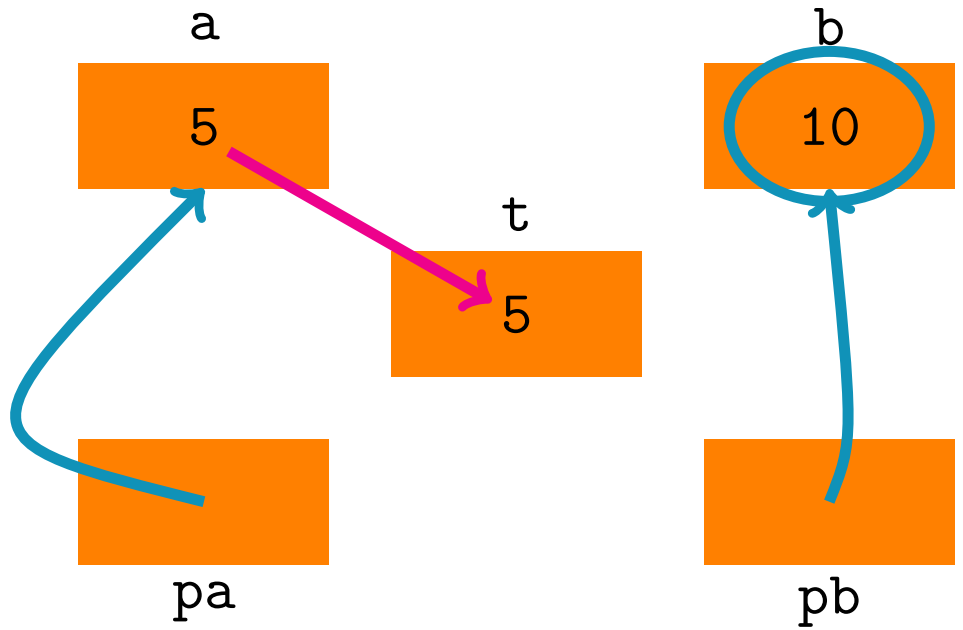

```
t = *pa;
        pb
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



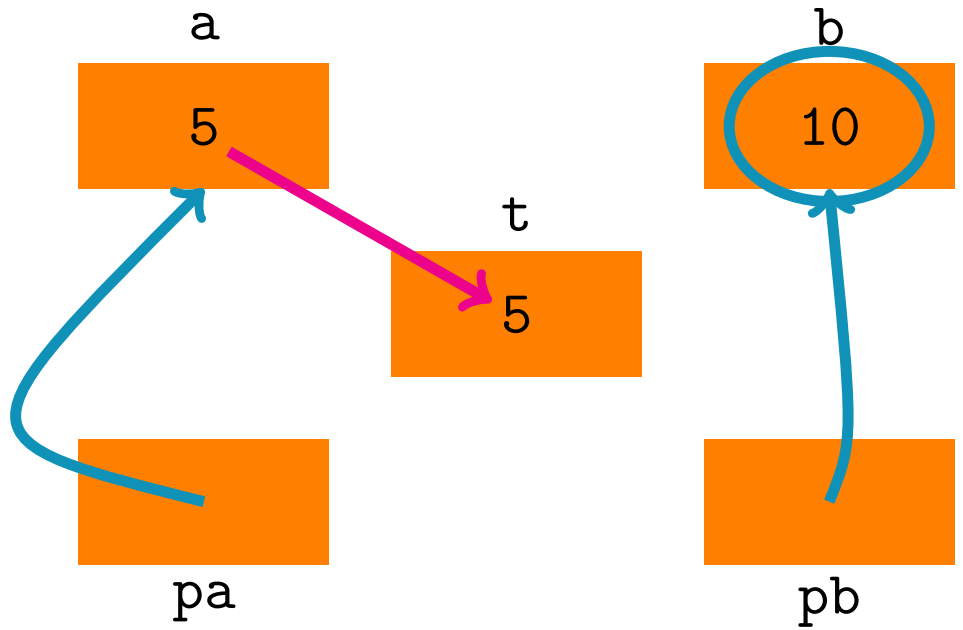

```
t = *pa;
      *pb
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



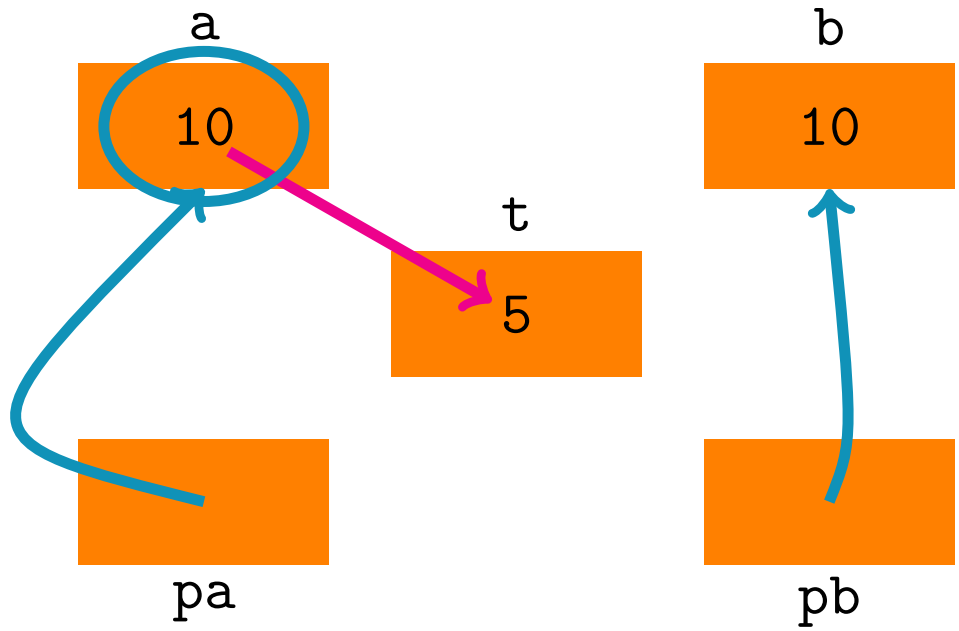

```
t = *pa;
      *pb
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



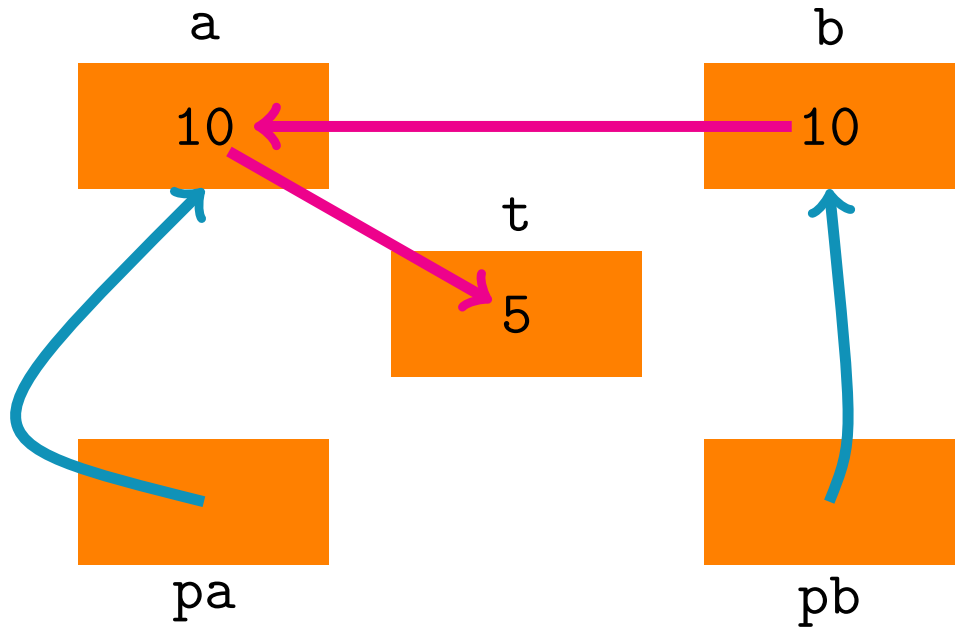

```
t = *pa;
*pa = *pb;
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



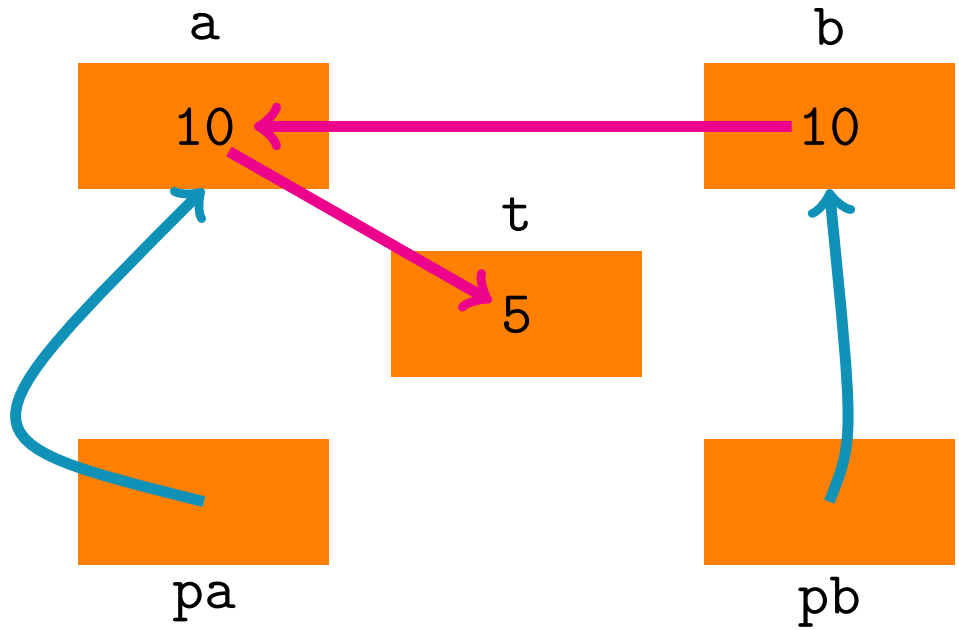

```
t = *pa;
*pa = *pb;
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



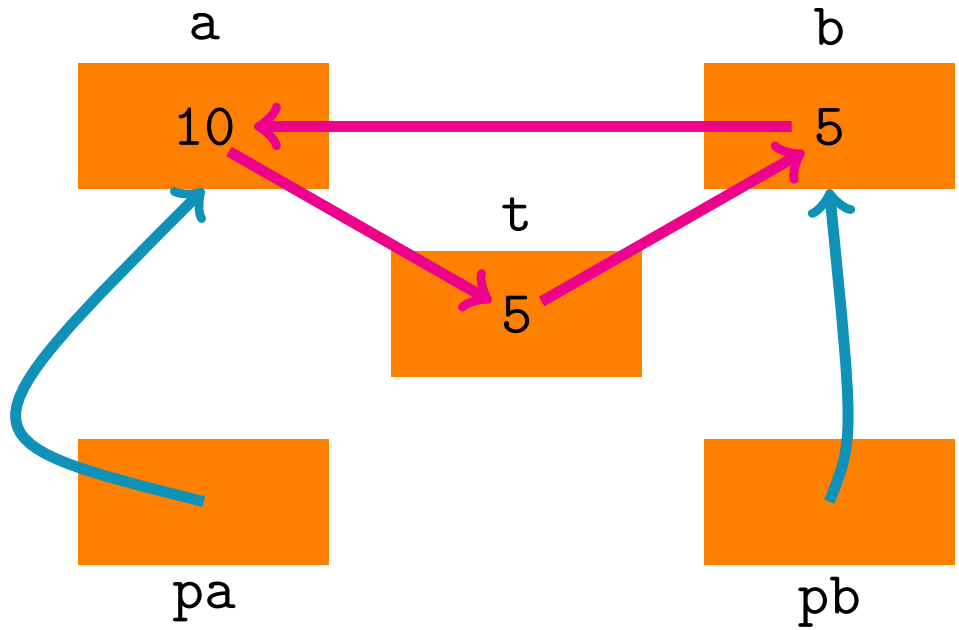

```
t = *pa;
*pa = *pb;
```

```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```

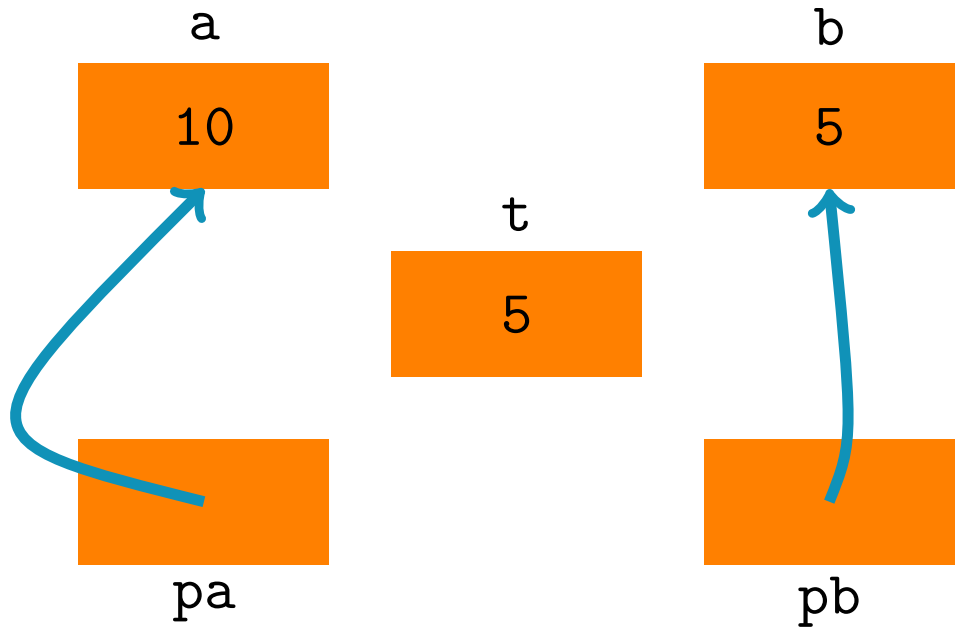



```
t = *pa;
*pa = *pb;
*pb = t;
```

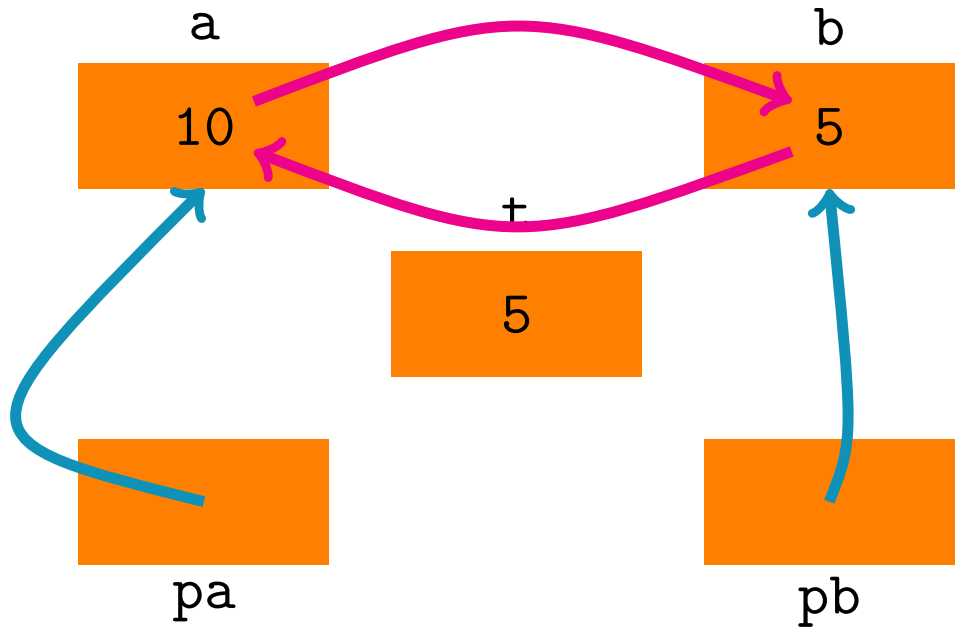
```
int a, b;
int *pa; /* address of a */
int *pb; /* address of b */
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



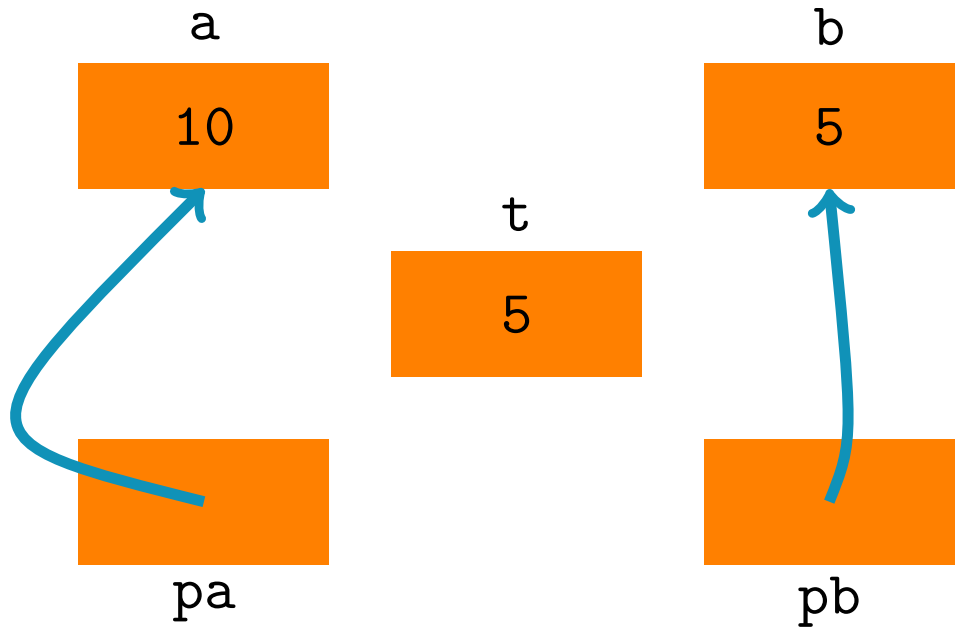

```
t = *pa;
*pa = *pb;
*pb = t;
```



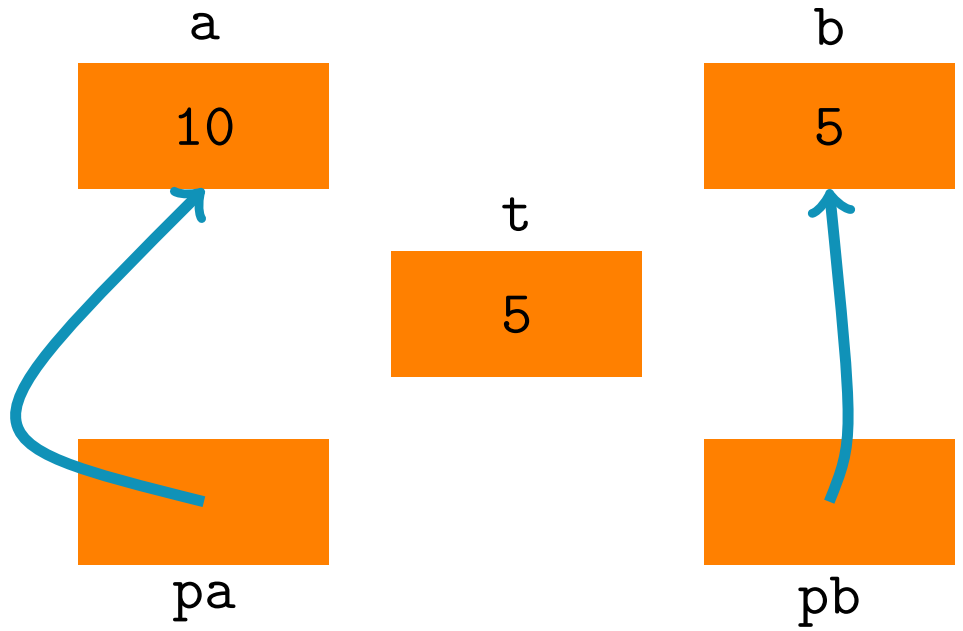

```
t = *pa;
*pa = *pb;
*pb = t;
```



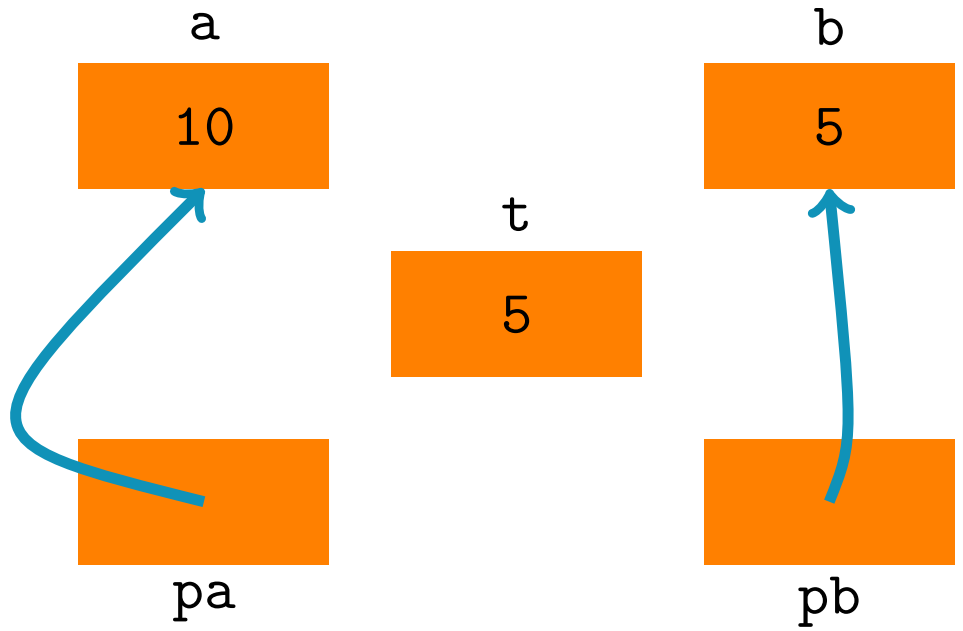

```
t = *pa;
*pa = *pb;
*pb = t;
```



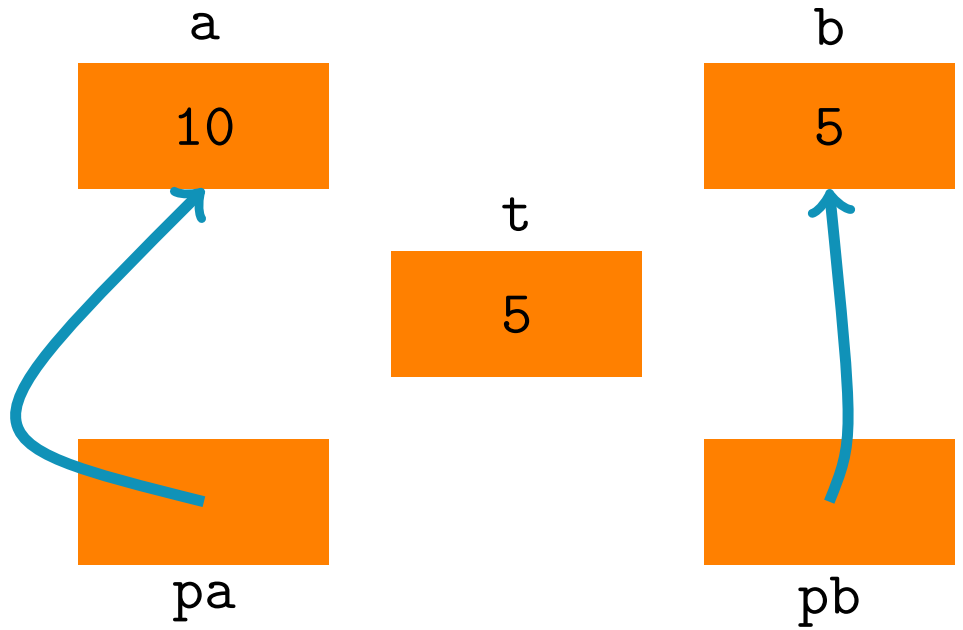

```
    swap(                    )
{
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```




```
    swap(int *pa, int *pb)
{
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

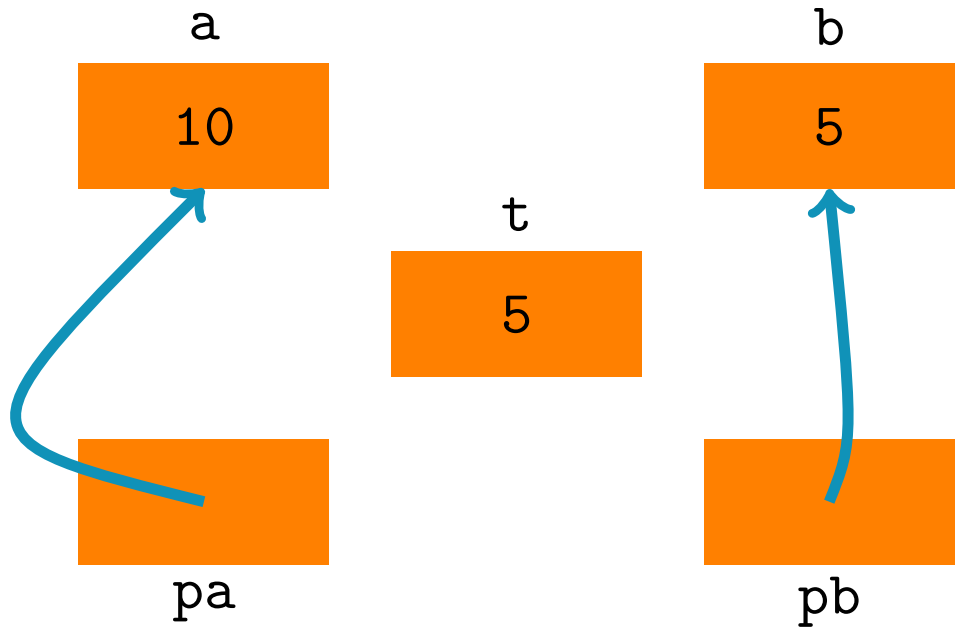



```
void swap(int *pa, int *pb)
{
   t = *pa;
   *pa = *pb;
   *pb = t;
}
```



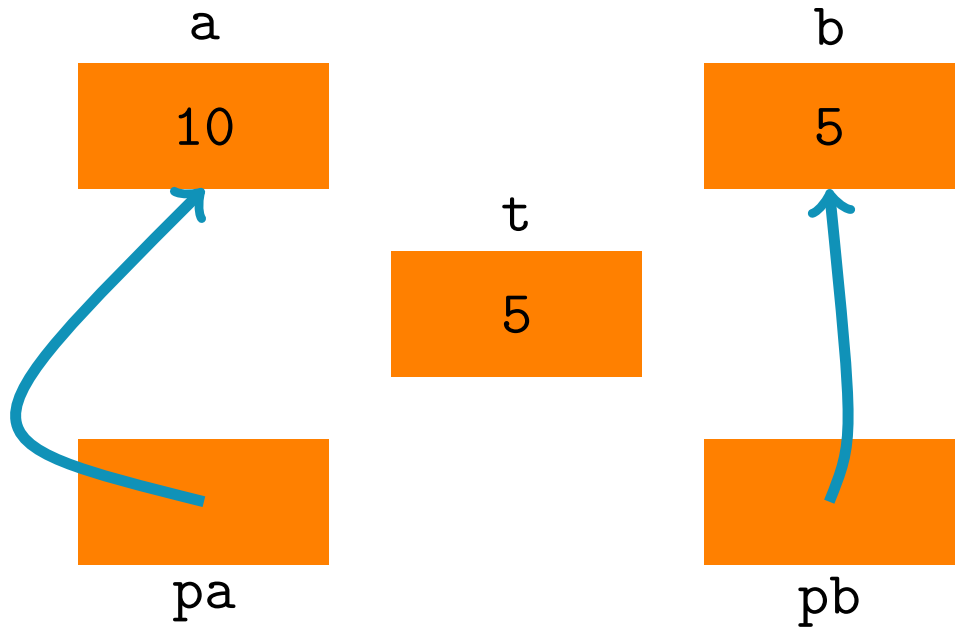

```
void swap(int *pa, int *pb)
{
   int t;
   t = *pa;
   *pa = *pb;
   *pb = t;
}
```

```
  int a, b;
  int *pa;
  int *pb;
  int t;
  a = 5;
  b = 10;
  pa = &a;
  pb = &b;
```



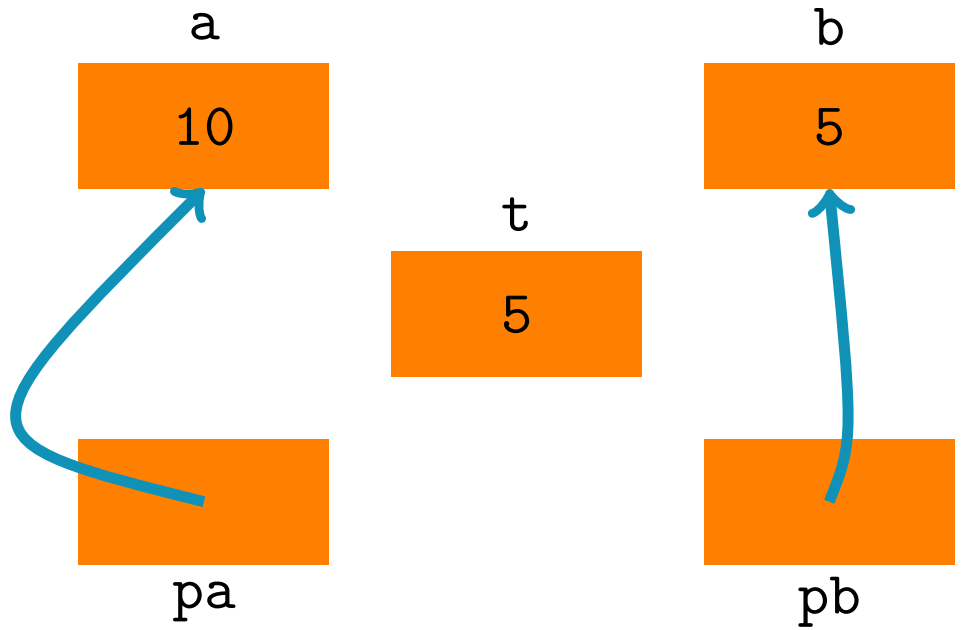

```
void swap(int *pa, int *pb)
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

```
int a, b;
int *pa;
int *pb;
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



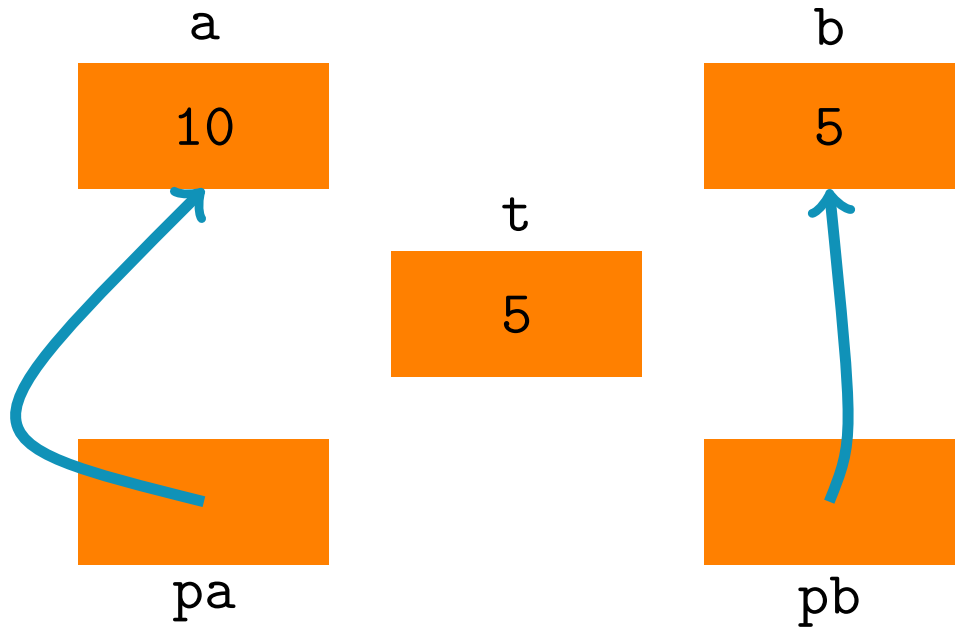

```
void swap(int *pa, int *pb)
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

```
int a, b;
int *pa;
int *pb;
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



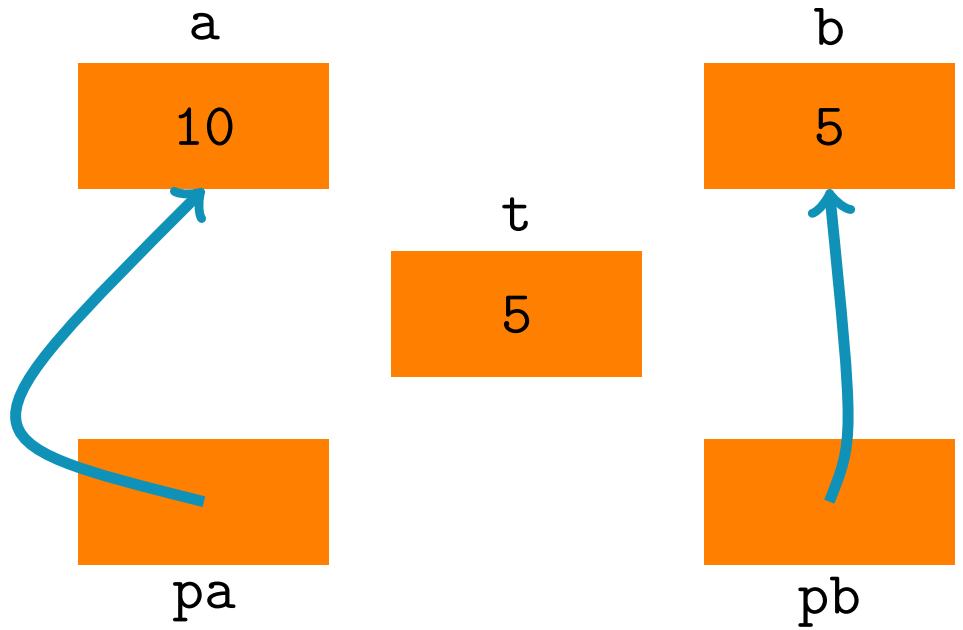

```
void swap(int *pa, int *pb)
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

```
int a, b;
int *pa;
int *pb;
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



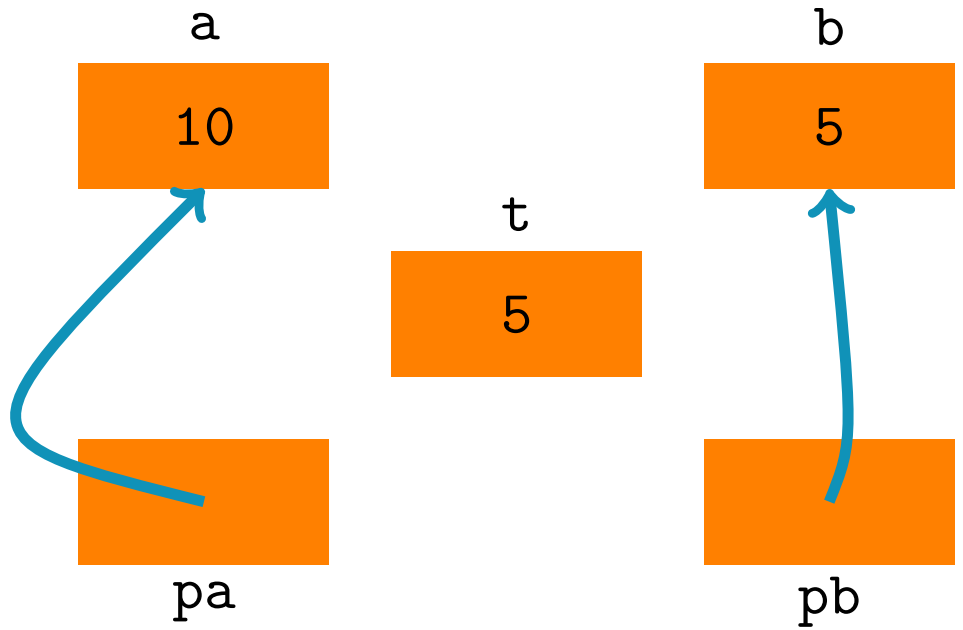

```
void swap(int *pa, int *pb)
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

```
int a, b;
int *pa;
int *pb;
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



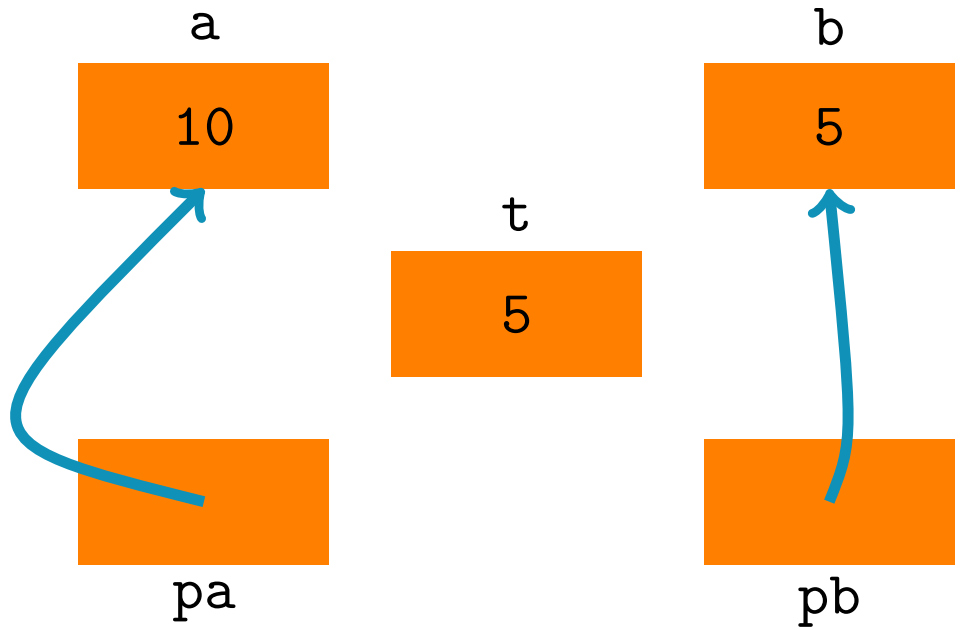

```
void swap(int *pa, int *pb)
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

```
int a, b;
int *pa;
int *pb;
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
```



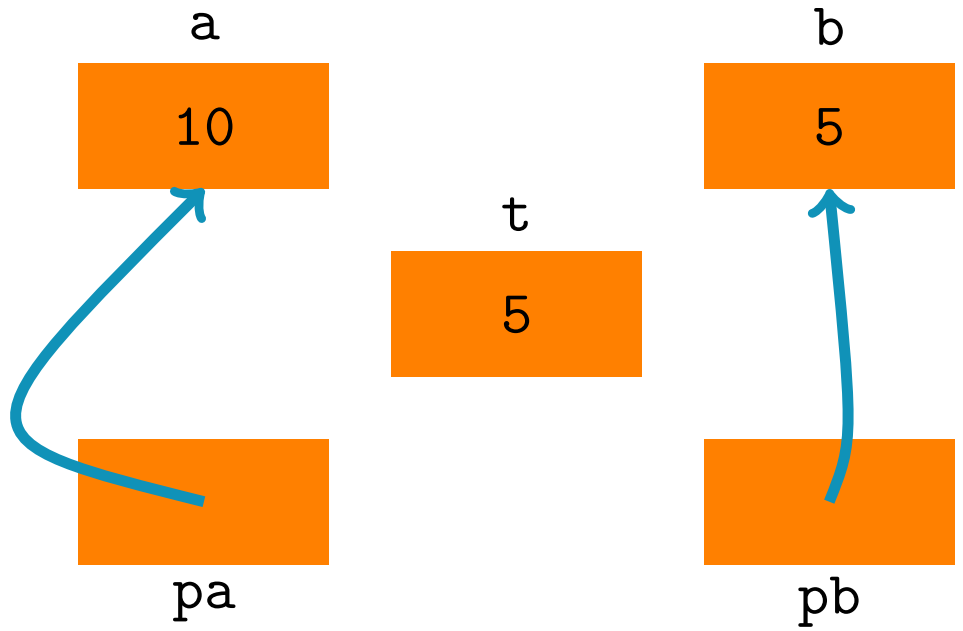

```
void swap(int *pa, int *pb)
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

```
int a, b;
int *pa;
int *pb;
int t;
a = 5;
b = 10;
pa = &a;
pb = &b;
swap(&a, &b);
```




```
void swap(int *pa, int *pb)
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

```c
int main(void)
{
  int a, b;
  int *pa;
  int *pb;
  int t;
  a = 5;
  b = 10;
  pa = &a;
  pb = &b;
  swap(&a, &b);
  return 0;
}
void swap(int *pa, int *pb)
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```



```
/* area and perimeter of a rectange */
```

```
/* area and perimeter of a rectange */
     area_peri(                                    )
{


}
```

```
/* area and perimeter of a rectange */
     area_peri(int len                              )
{


}
```

```
/* area and perimeter of a rectange */
     area_peri(int len, int bth                    )
{


}
```

```
/* area and perimeter of a rectange */
     area_peri(int len, int bth, int *ar          )
{


}
```

```
/* area and perimeter of a rectange */
     area_peri(int len, int bth, int *ar, int *pr)
{


}
```

```
/* area and perimeter of a rectange */
     area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;

}
```

```
/* area and perimeter of a rectange */
     area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}

int main(void)
{









}
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}

int main(void)
{
  int l, b;









}
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}

int main(void)
{
  int l, b;
  int a, p;









}
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}

int main(void)
{
  int l, b;
  int a, p;
  l = 2;




}
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}

int main(void)
{
  int l, b;
  int a, p;
  l = 2;
  b = 3;



}
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}

int main(void)
{
  int l, b;
  int a, p;
  l = 2;
  b = 3;
  area_peri(l, b, &a, &p);


}
```

```
/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}

int main(void)
{
  int l, b;
  int a, p;
  l = 2;
  b = 3;
  area_peri(l, b, &a, &p);
  printf("area = %d, perimeter = %d\n", a, p);

}
```

```c
#include <stdio.h>

/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}

int main(void)
{
  int l, b;
  int a, p;
  l = 2;
  b = 3;
  area_peri(l, b, &a, &p);
  printf("area = %d, perimeter = %d\n", a, p);

}
```

```c
#include <stdio.h>

/* area and perimeter of a rectange */
void area_peri(int len, int bth, int *ar, int *pr)
{
  *ar = len * bth;
  *pr = 2 * (len + bth);
}

int main(void)
{
  int l, b;
  int a, p;
  l = 2;
  b = 3;
  area_peri(l, b, &a, &p);
  printf("area = %d, perimeter = %d\n", a, p);
  return 0;
}
```