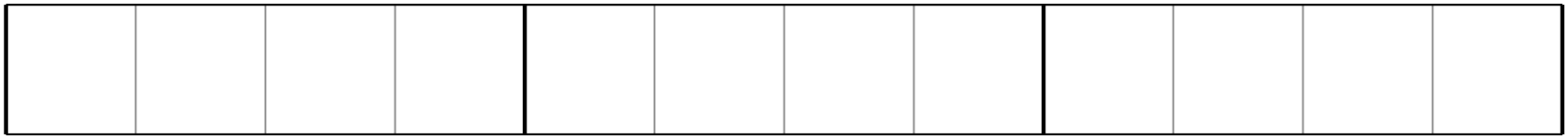
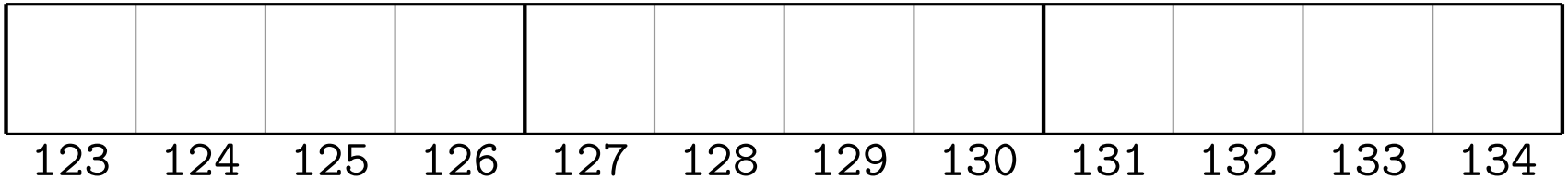
```
int a[3];
```

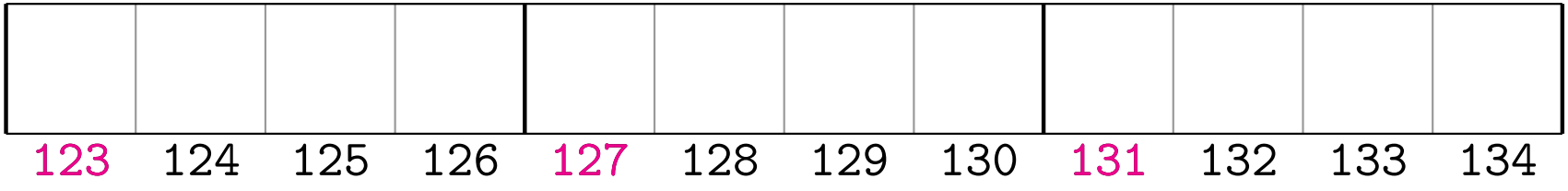
```
int a[3];
```



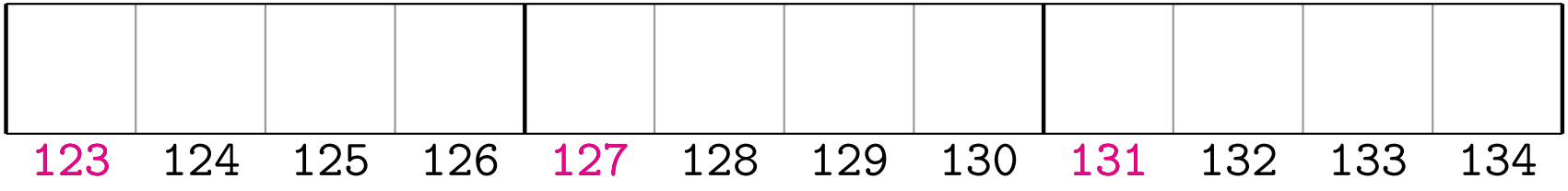
```
int a[3];
```



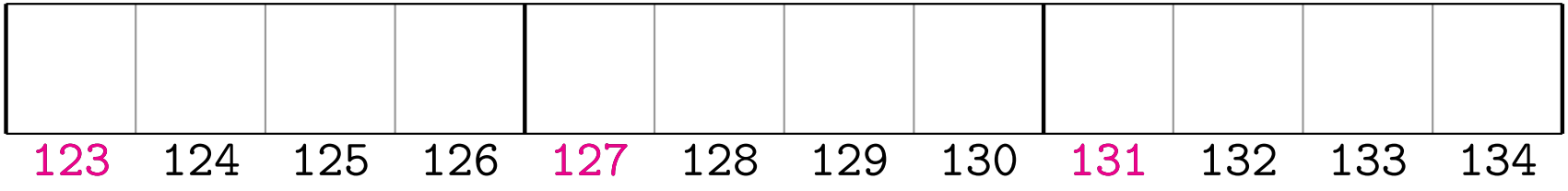
```
int a[3];
```



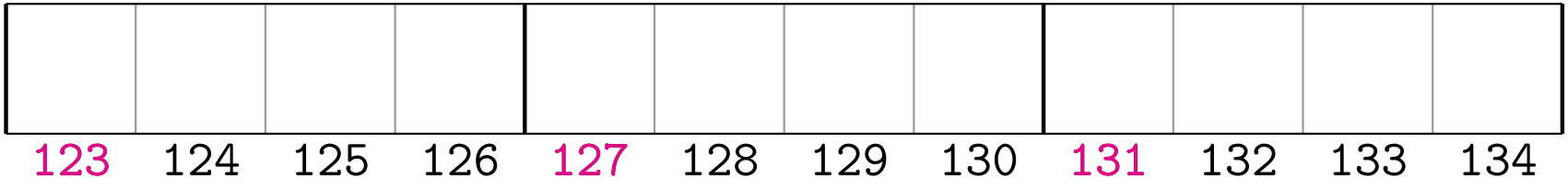
```
int a[3];
printf("%p", &a[0]);
```



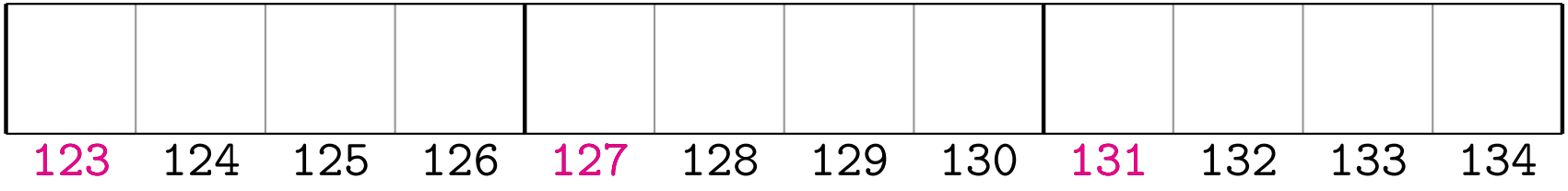
```
int a[3];
printf("%p", &a[0]); → 123
```



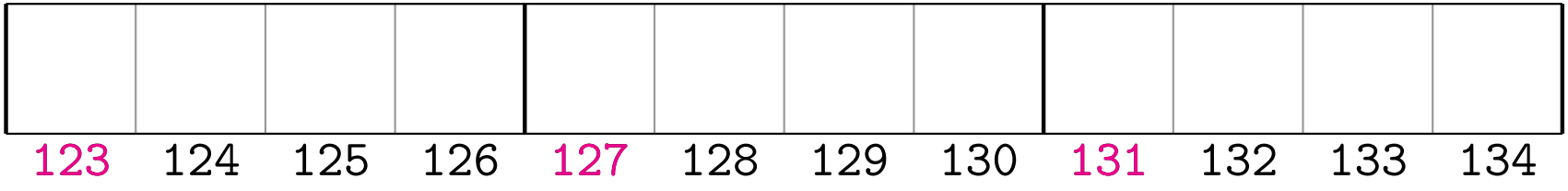
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
```



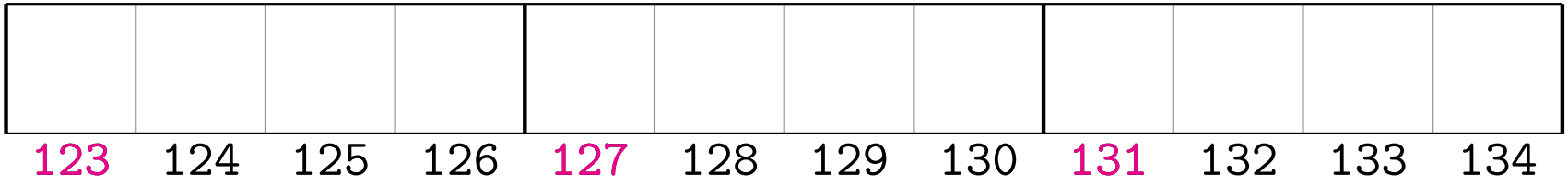
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
```



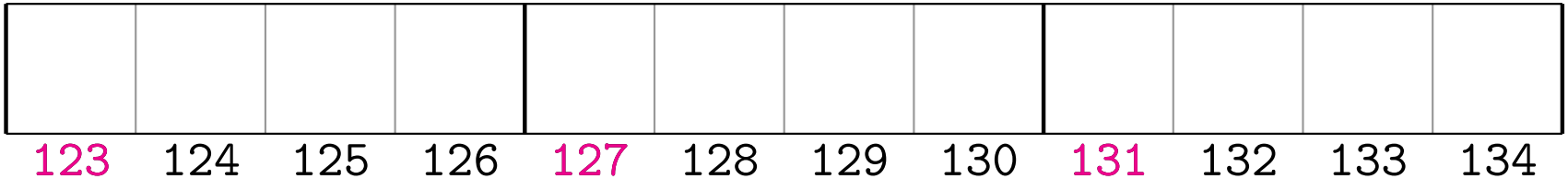
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p);
```



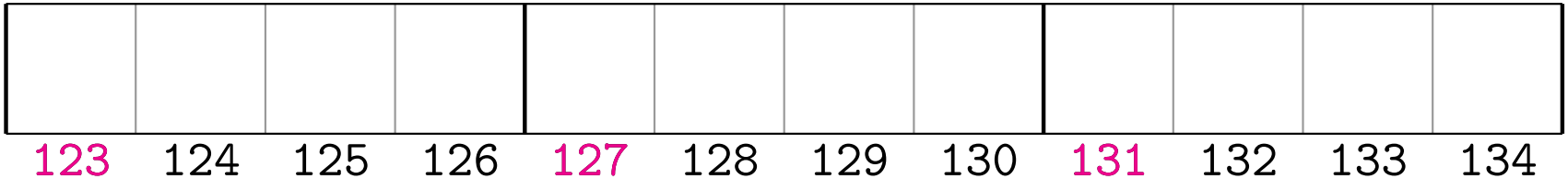
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
```



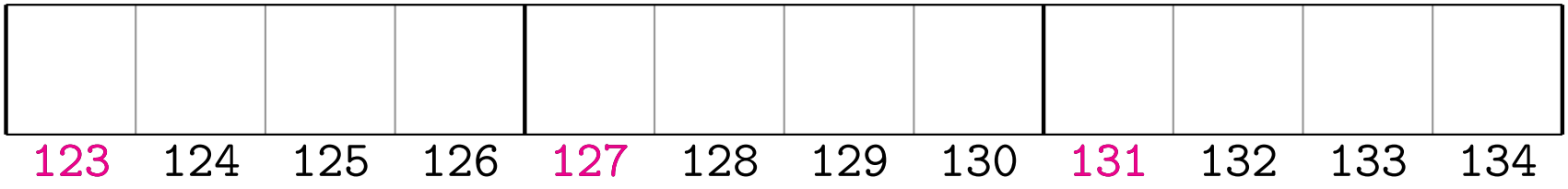
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1;
```



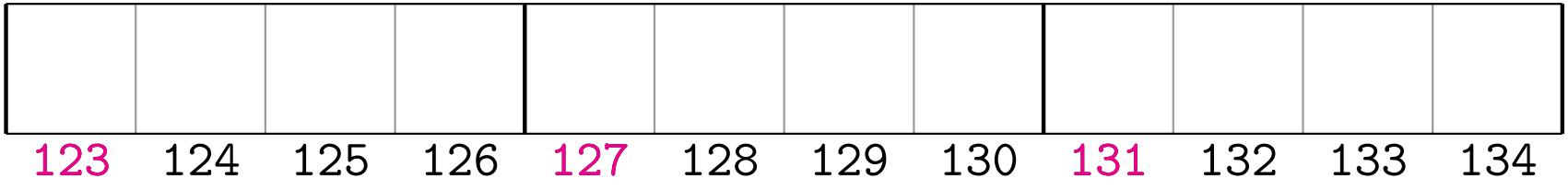
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1;
printf("%p", p);
```



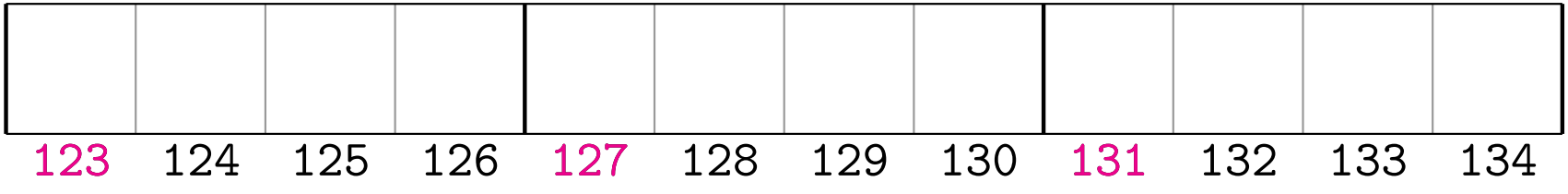
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1;
printf("%p", p); → 124
```



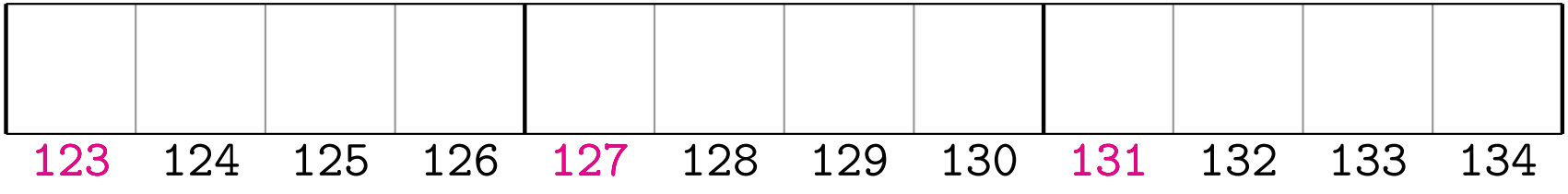
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1;
printf("%p", p); → 124
```



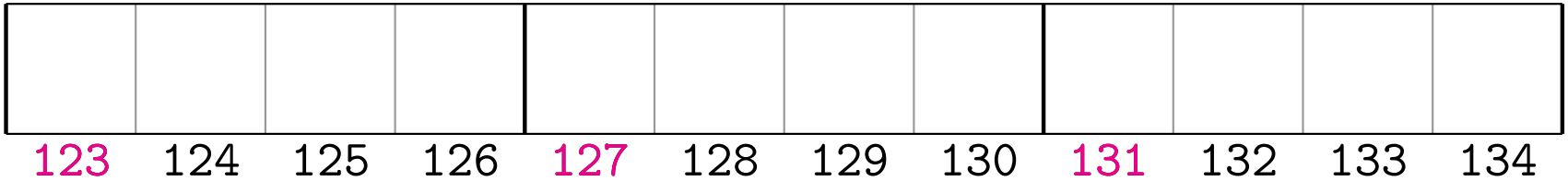
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1;
printf("%p", p); → 127✓
```



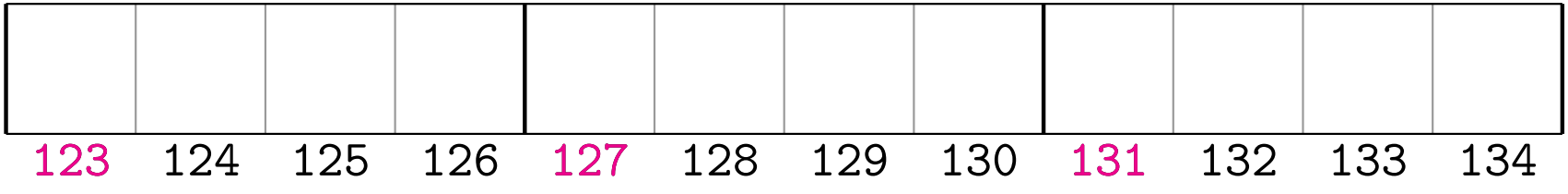
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
```



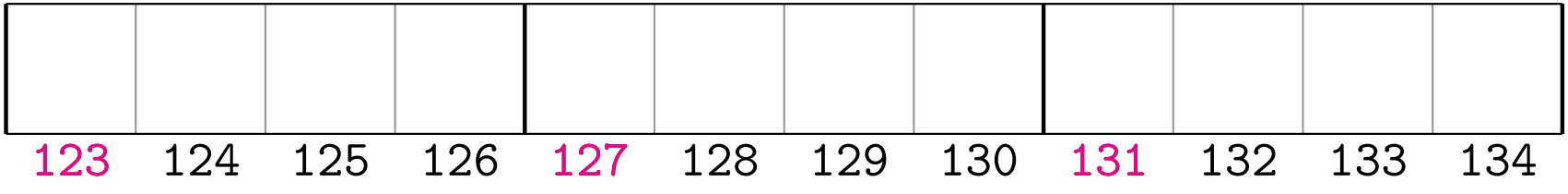
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1;
```



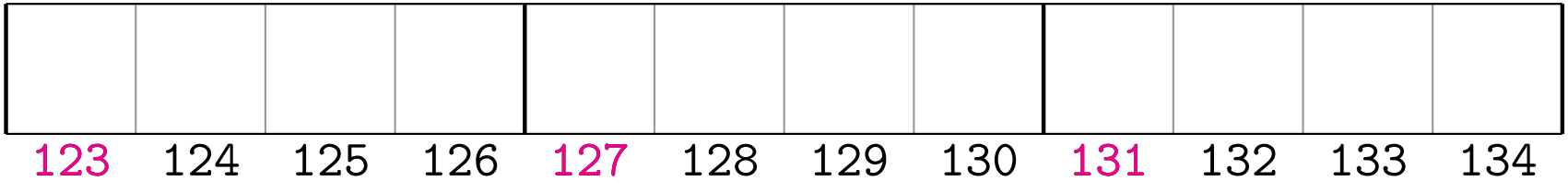
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1;
printf("%p", p);
```



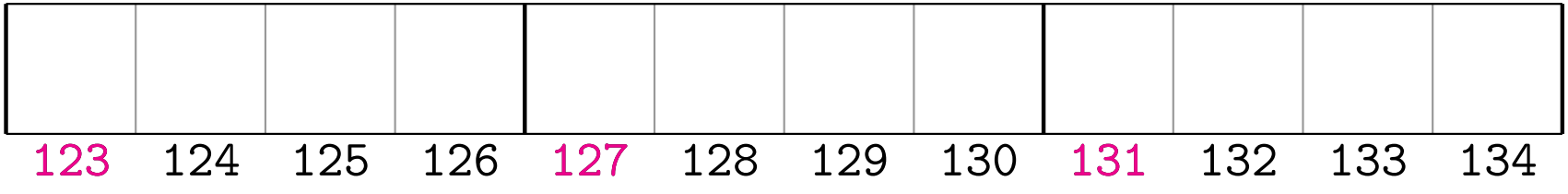
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1;
printf("%p", p); → 131
```



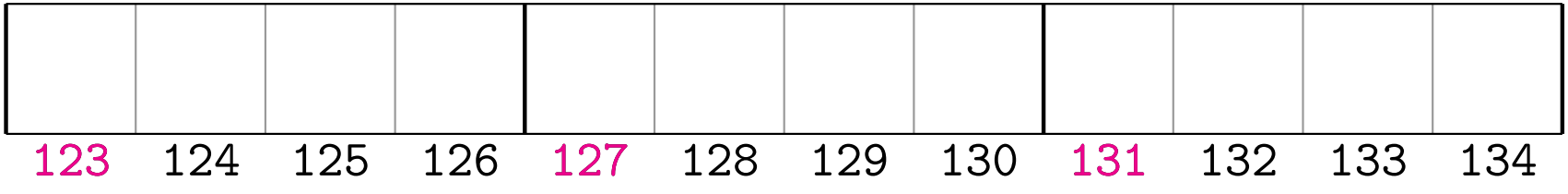
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1; /* next array location */
printf("%p", p); → 131
```



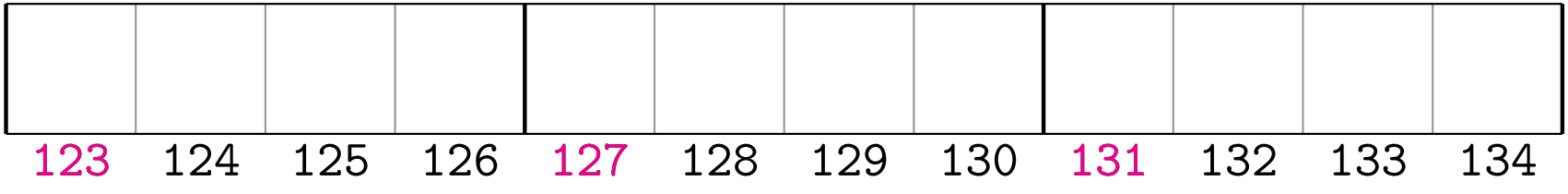
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1; /* next array location */
printf("%p", p); → 131
p = p - 1;
```



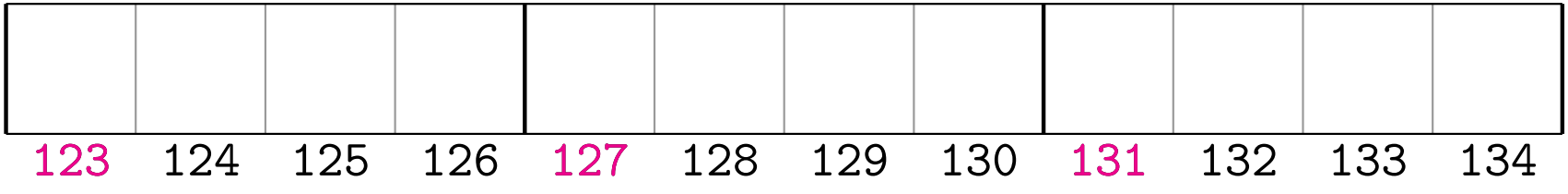
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1; /* next array location */
printf("%p", p); → 131
p = p - 1;
printf("%p", p);
```



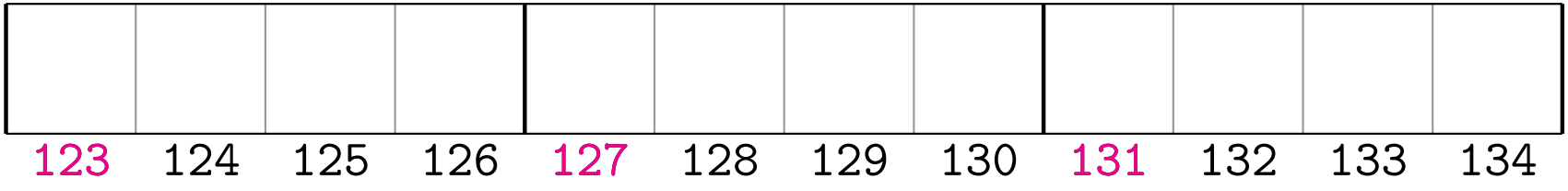
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1; /* next array location */
printf("%p", p); → 131
p = p - 1;
printf("%p", p); → 127
```



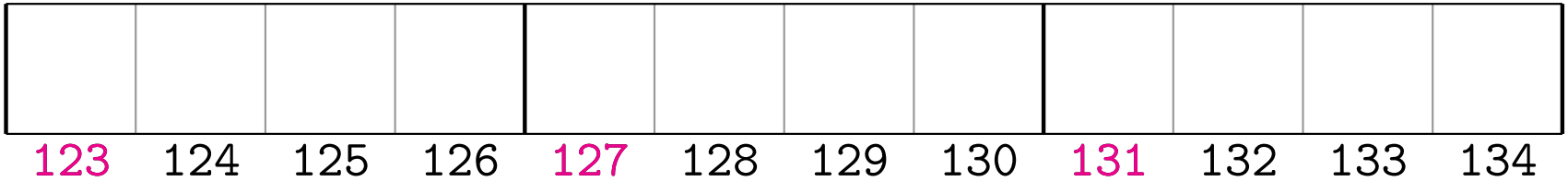
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1; /* next array location */
printf("%p", p); → 131
p = p - 1; /* previous array location */
printf("%p", p); → 127
```



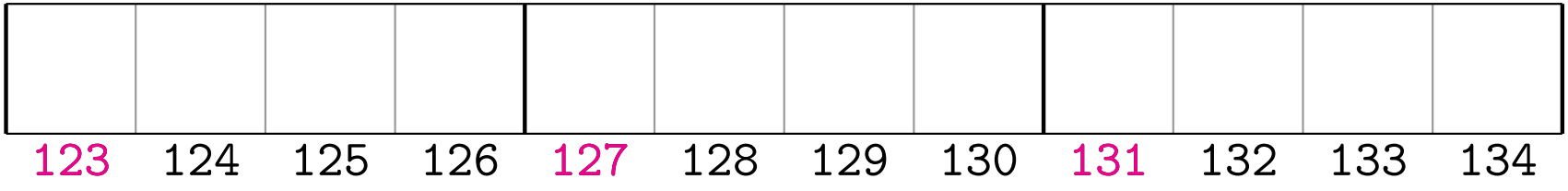
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1; /* next array location */
printf("%p", p); → 131
p = p - 1; /* previous array location */
printf("%p", p); → 127
p = p - 1;
```



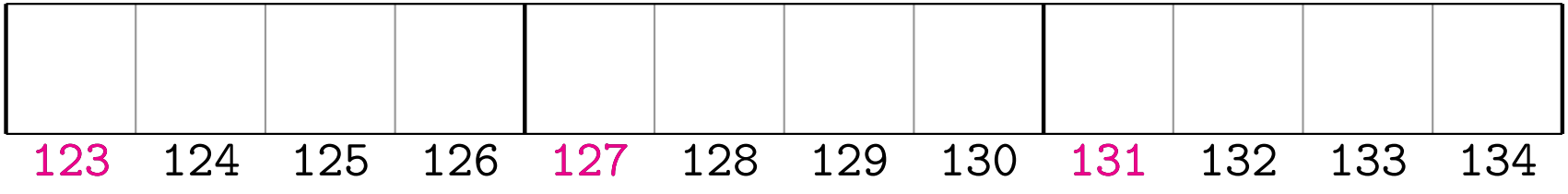
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1; /* next array location */
printf("%p", p); → 131
p = p - 1; /* previous array location */
printf("%p", p); → 127
p = p - 1;
printf("%p", p);
```

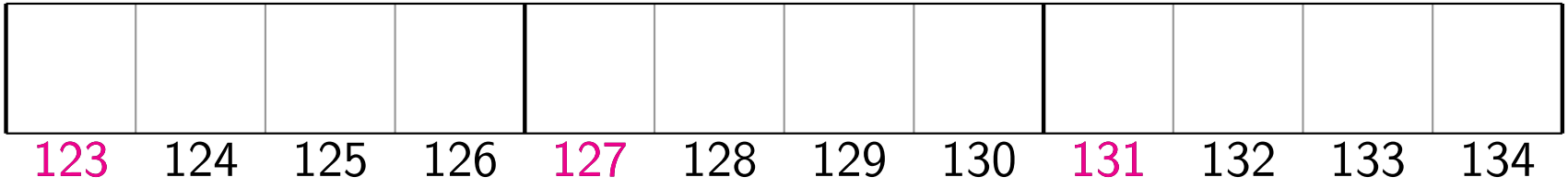


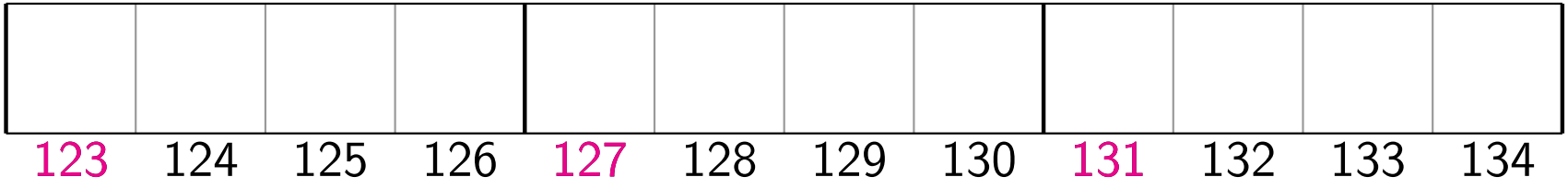
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1; /* next array location */
printf("%p", p); → 131
p = p - 1; /* previous array location */
printf("%p", p); → 127
p = p - 1;
printf("%p", p); → 123
```



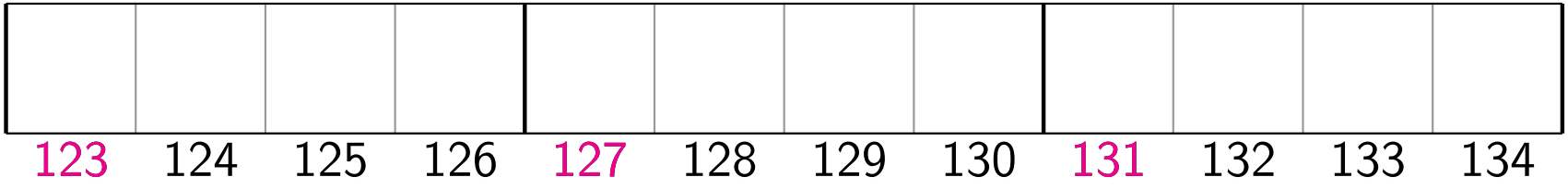
```
int a[3];
printf("%p", &a[0]); → 123
int *p;
p = &a[0];
printf("%p", p); → 123
p = p + 1; /* next array location */
printf("%p", p); → 127✓
p = p + 1; /* next array location */
printf("%p", p); → 131
p = p - 1; /* previous array location */
printf("%p", p); → 127
p = p - 1; /* previous array location */
printf("%p", p); → 123
```



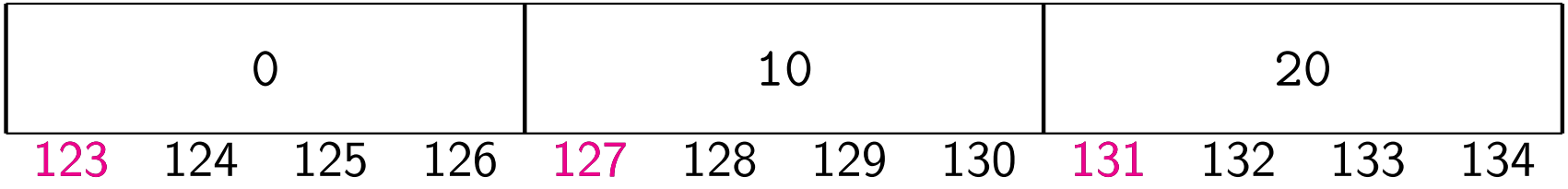

123 124 125 126 127 128 129 130 131 132 133 134
a[3]



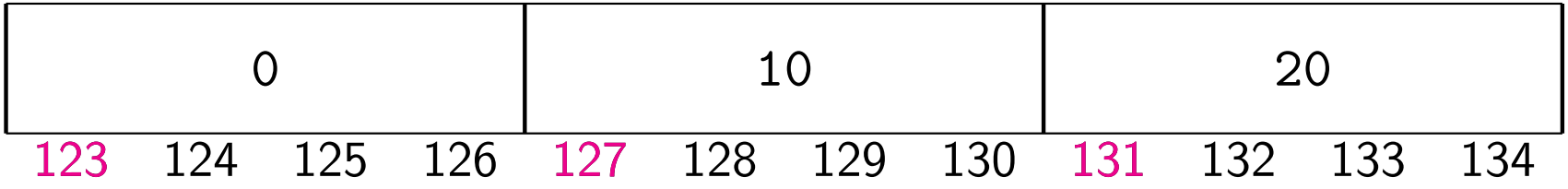

```
for (i = 0; i < 3; i++) {

}
```



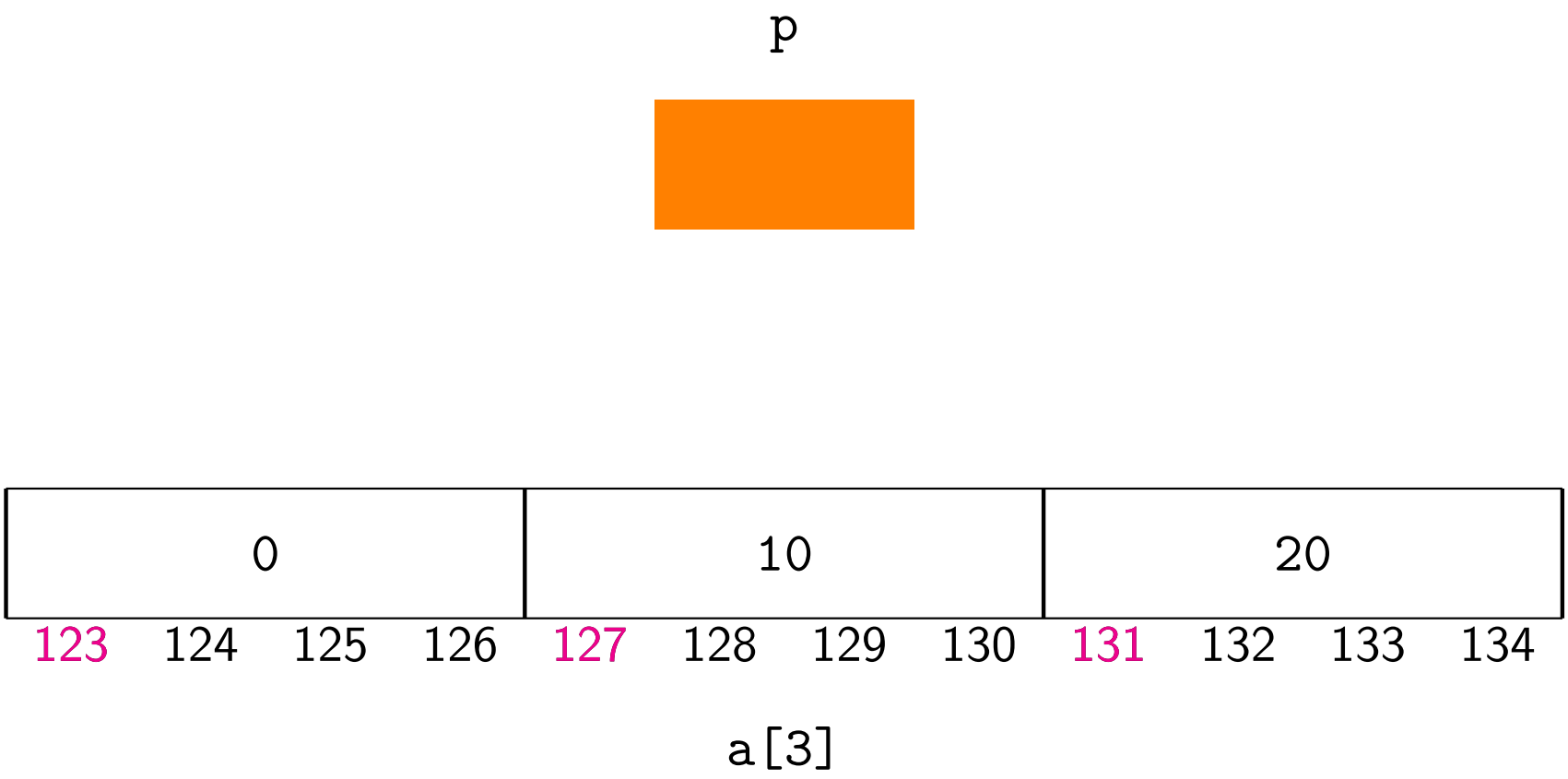

```
for (i = 0; i < 3; i++) {
  a[i] = i * 10;
}
```



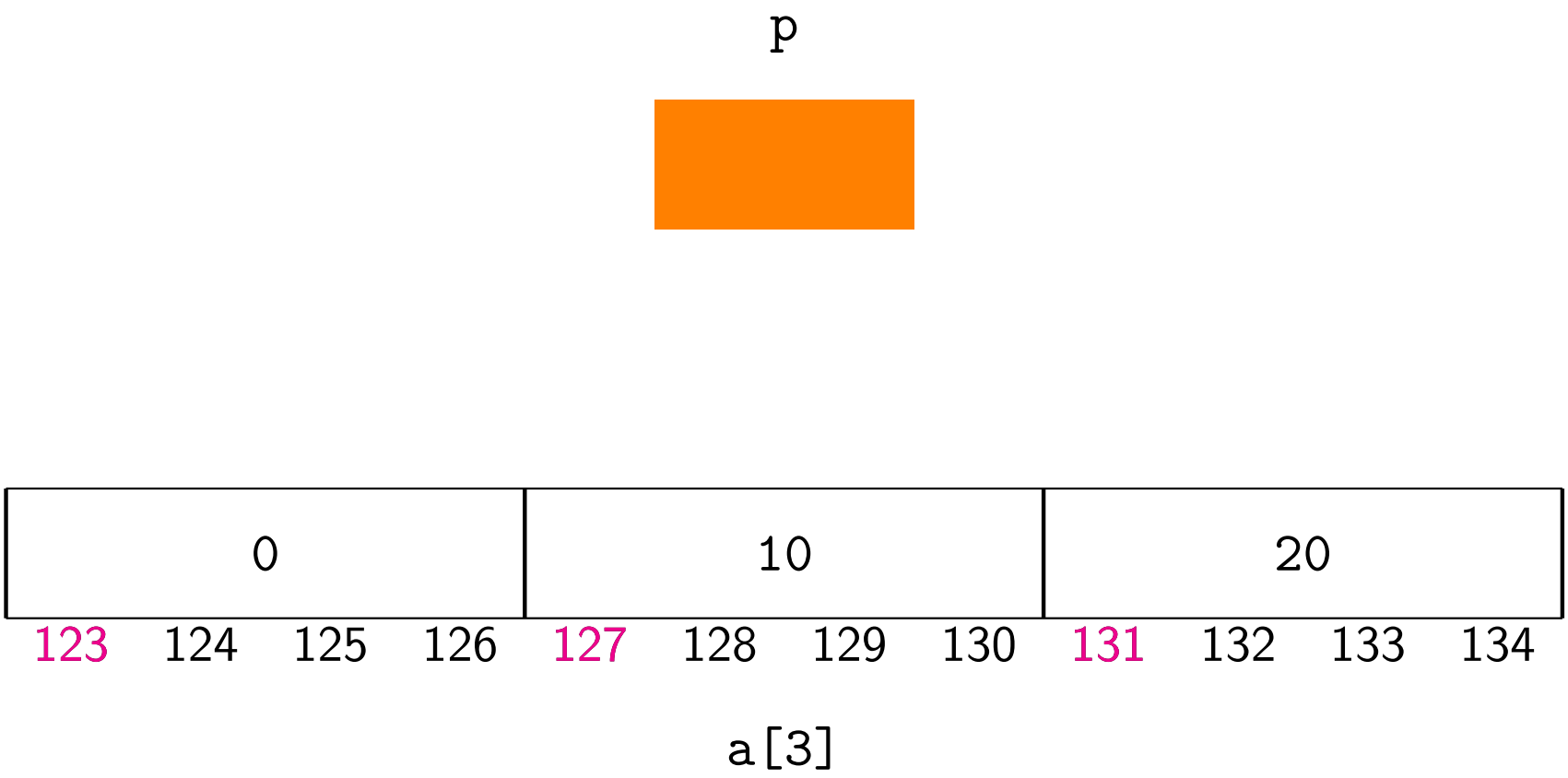

```
for (i = 0; i < 3; i++) {
  a[i] = i * 10;
}
```



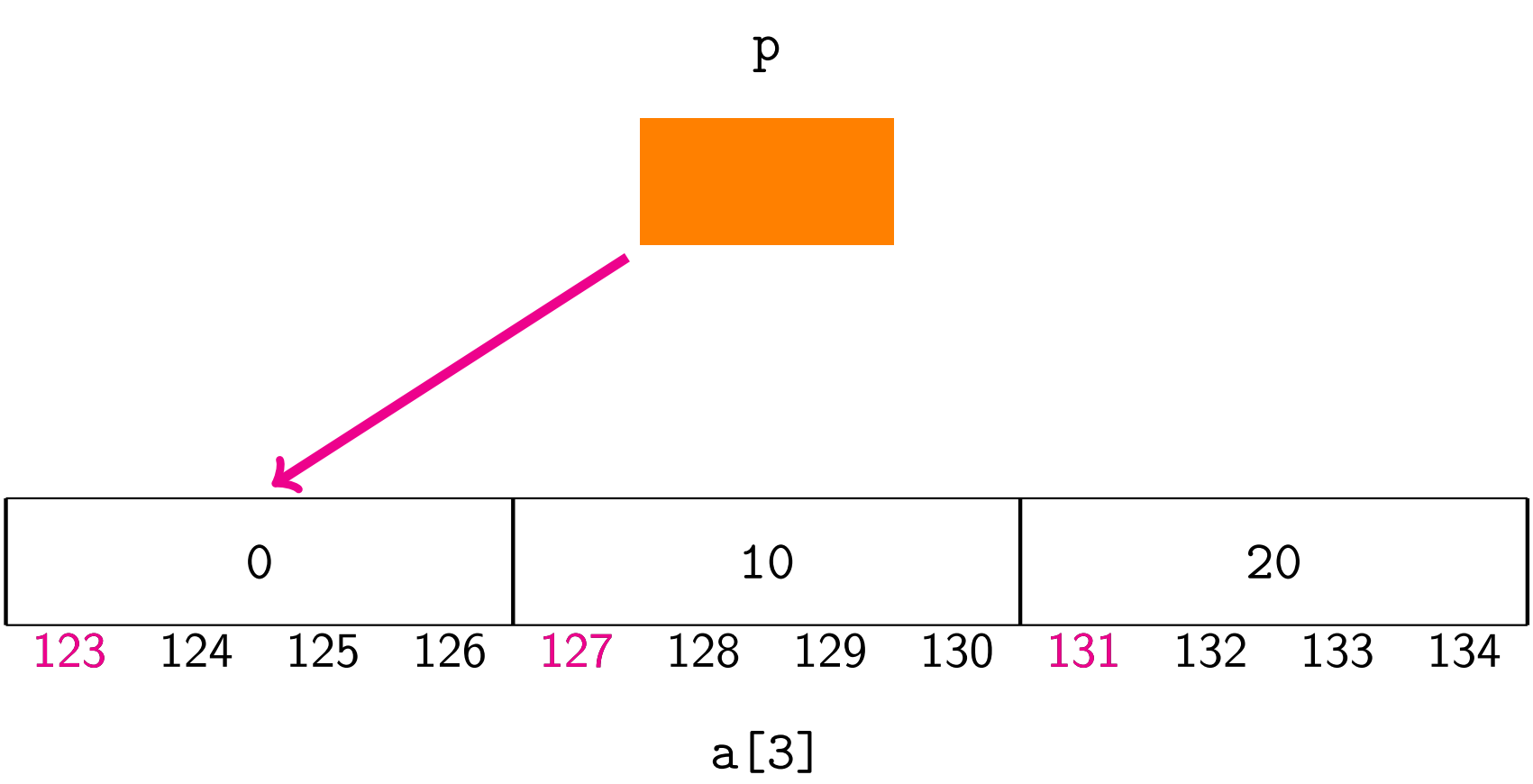

```
int *p;
```



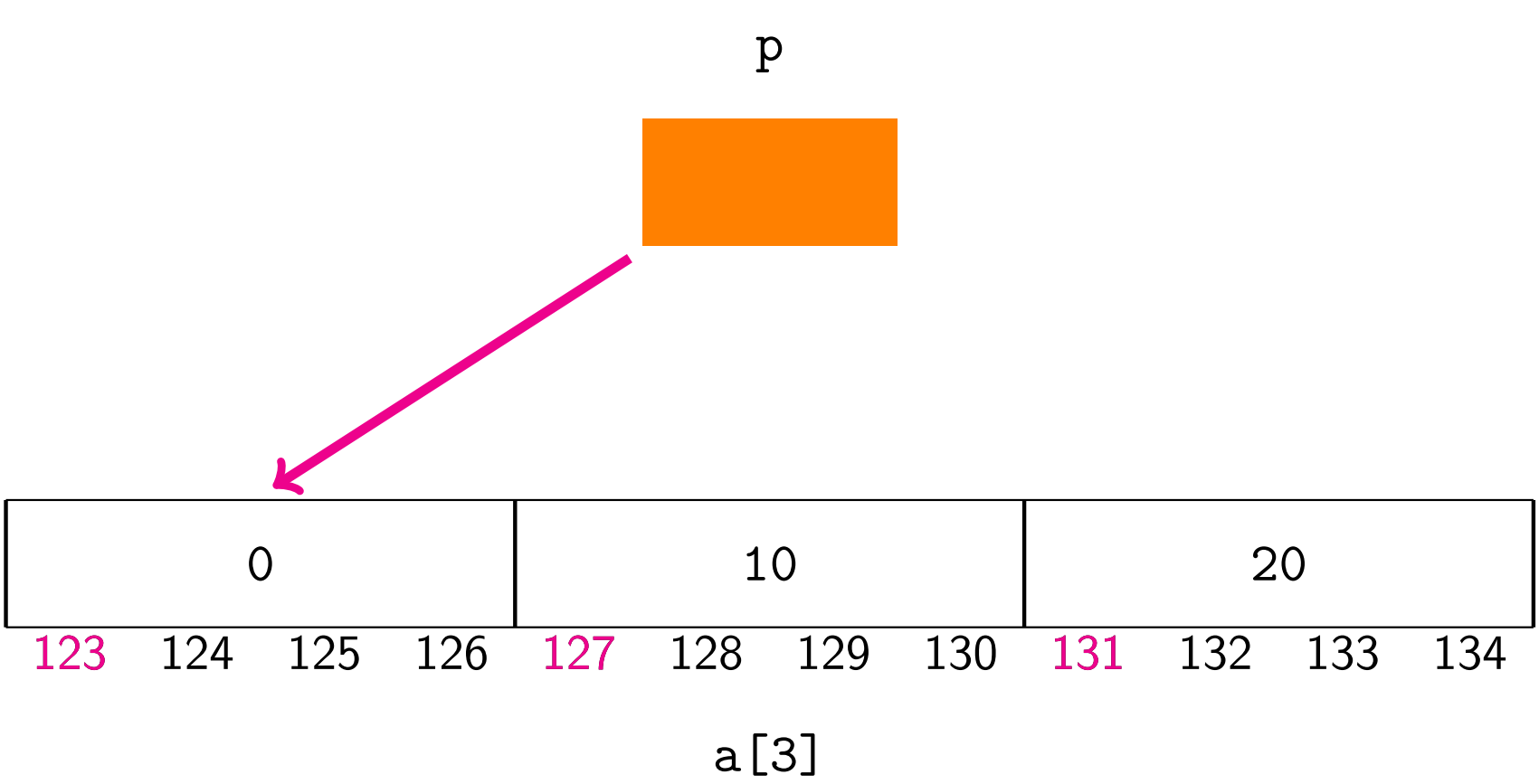

```
int *p;
```



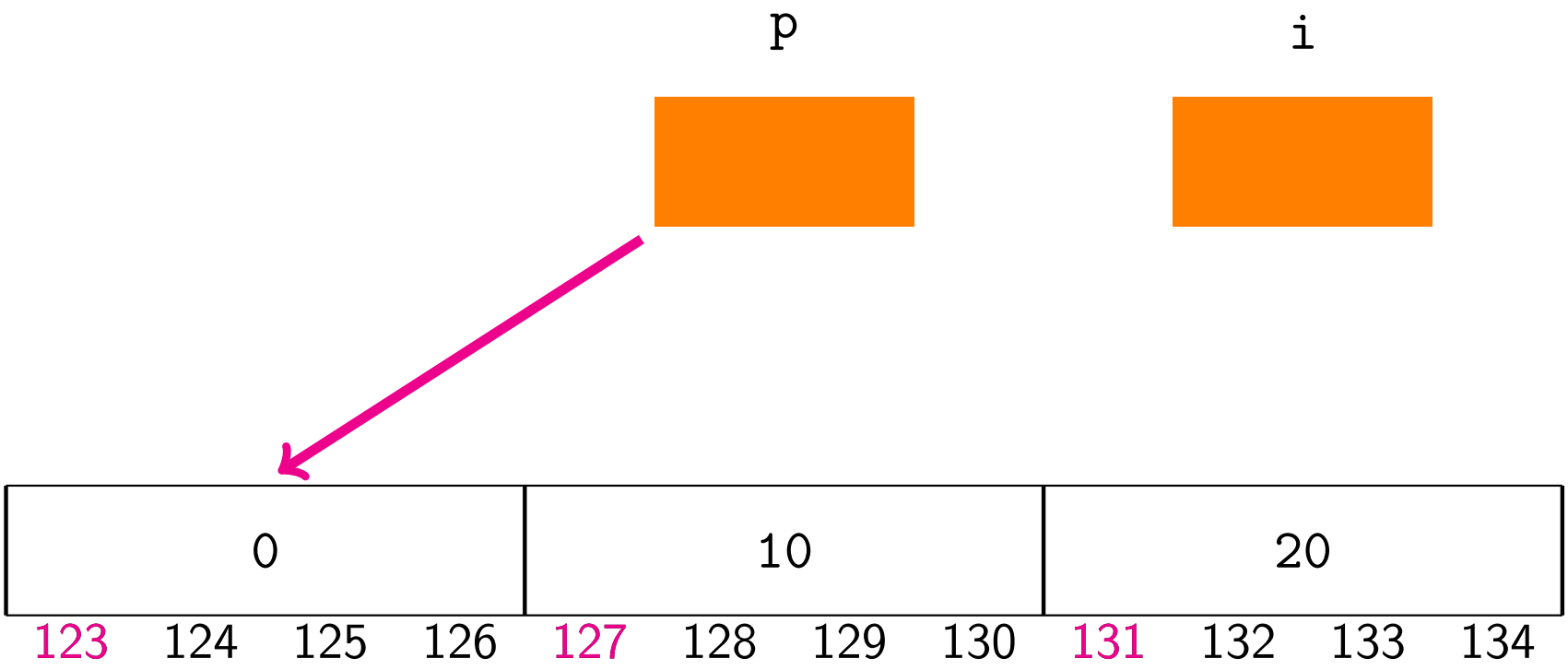

```
int *p;
p = &a[0];
```



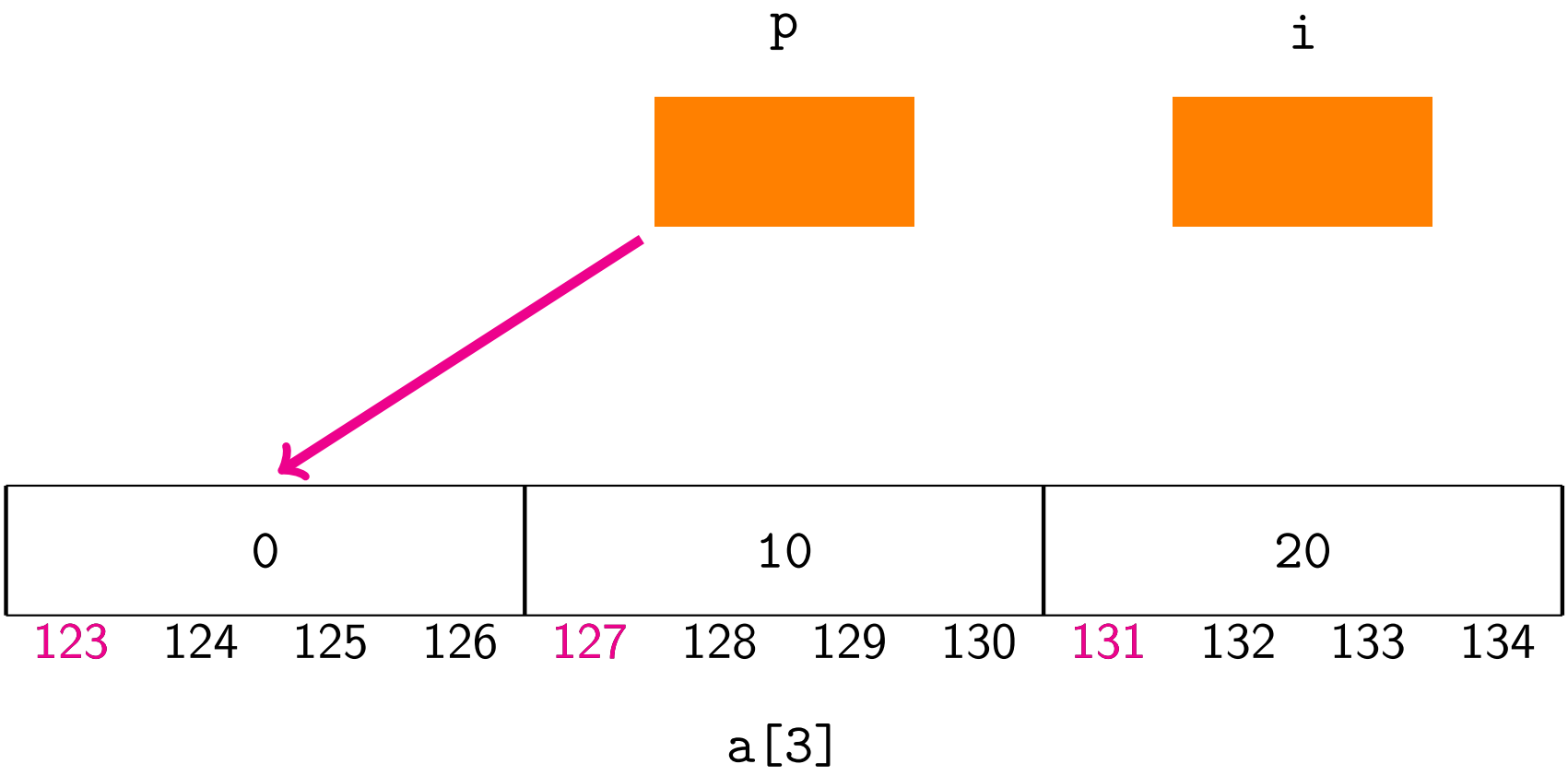

```
int *p;
p = &a[0];
```



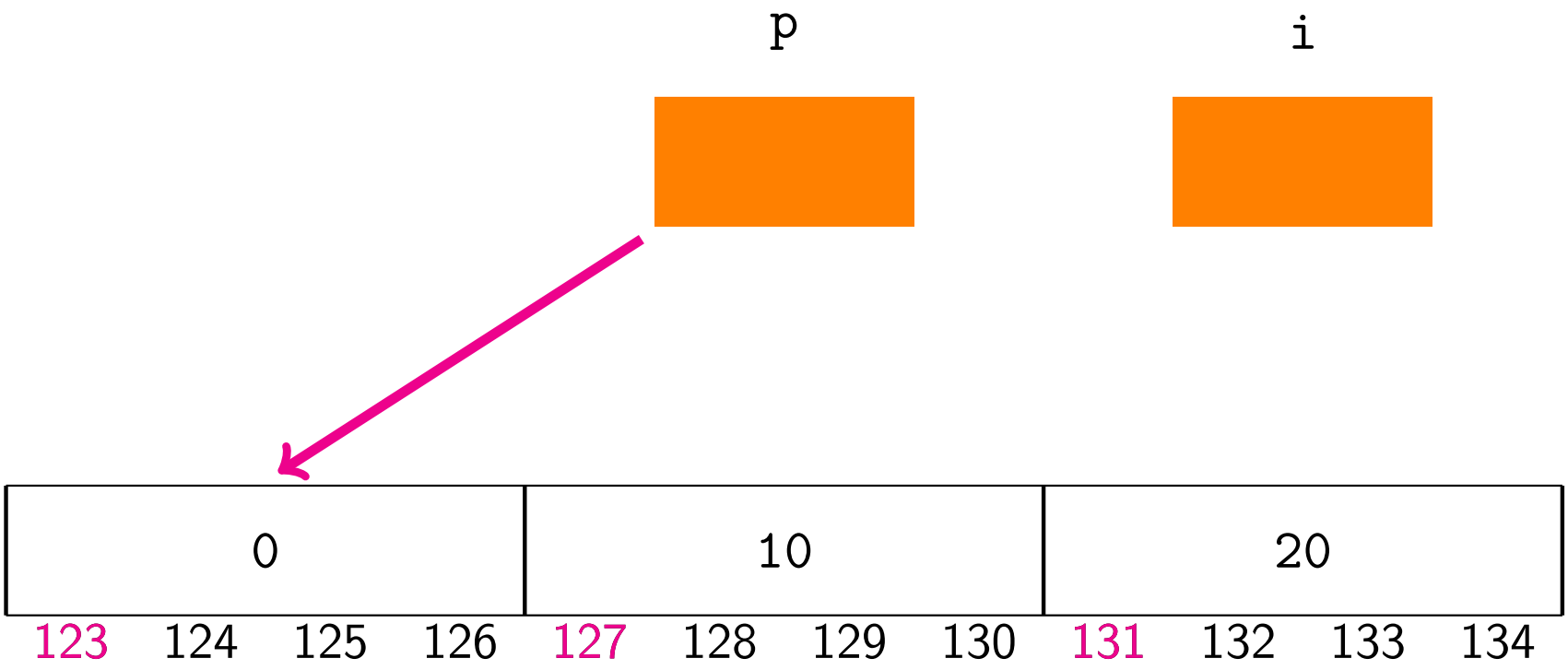

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {

}
```



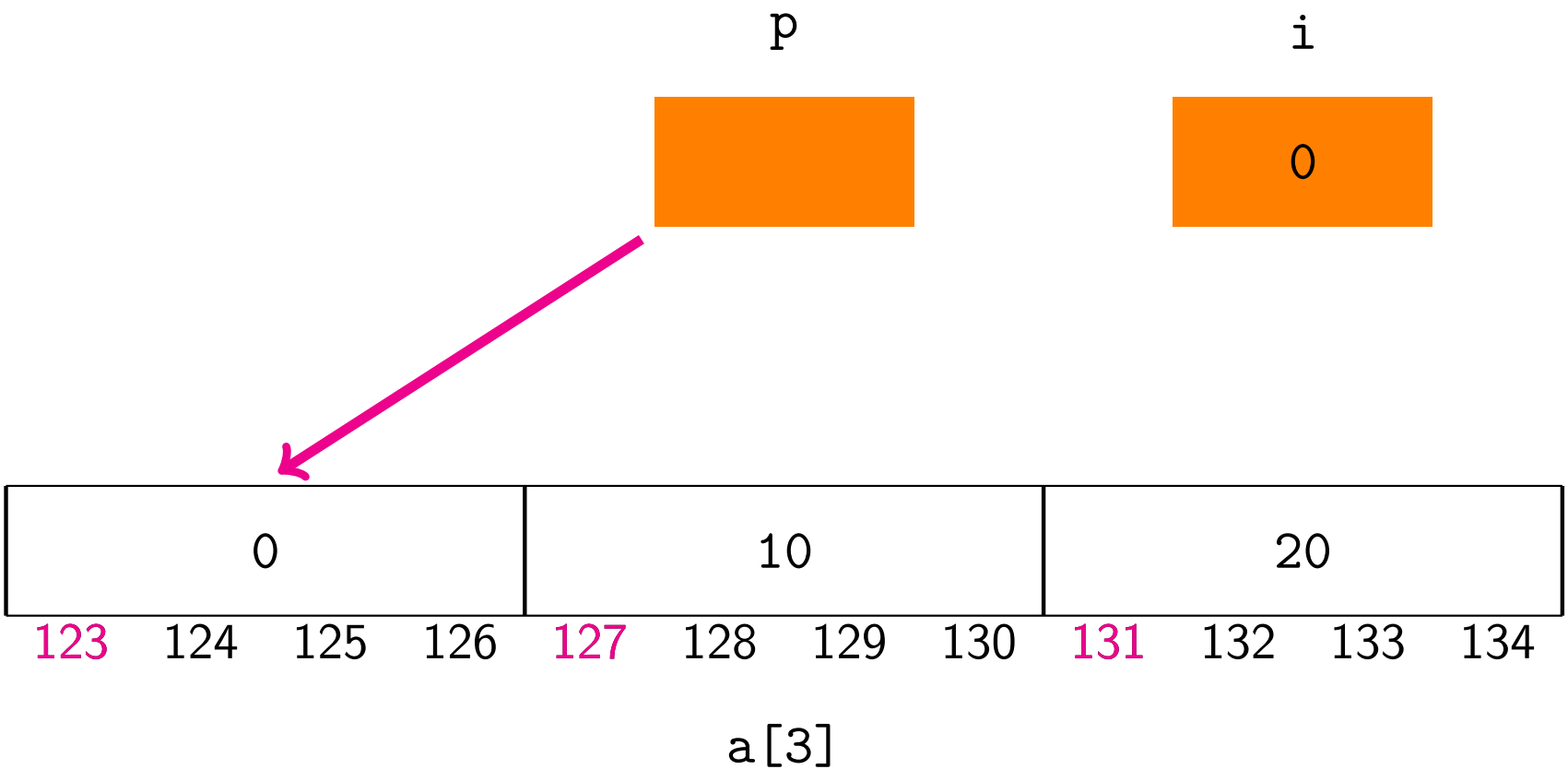

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {

}
```



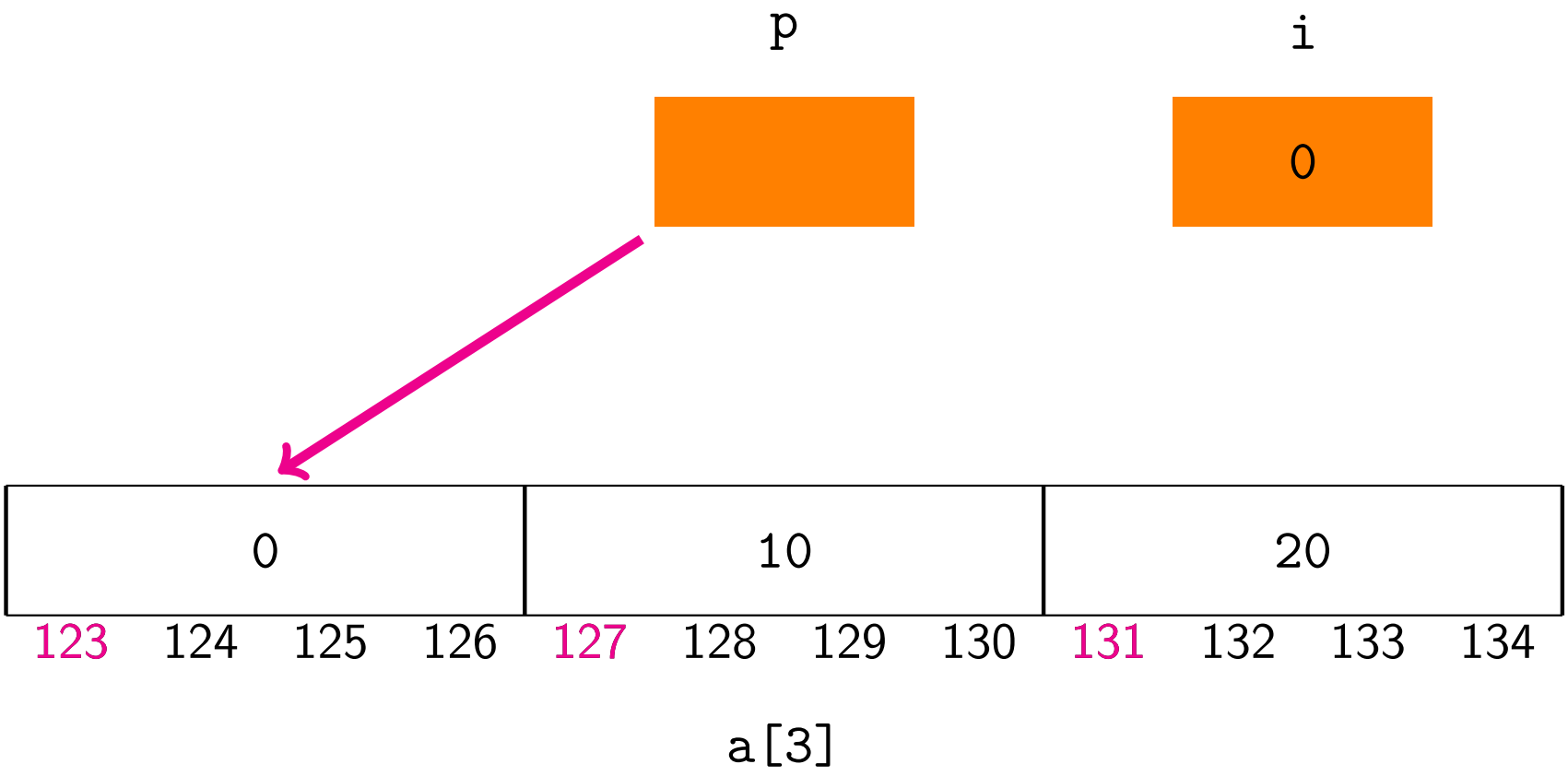

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
}
```



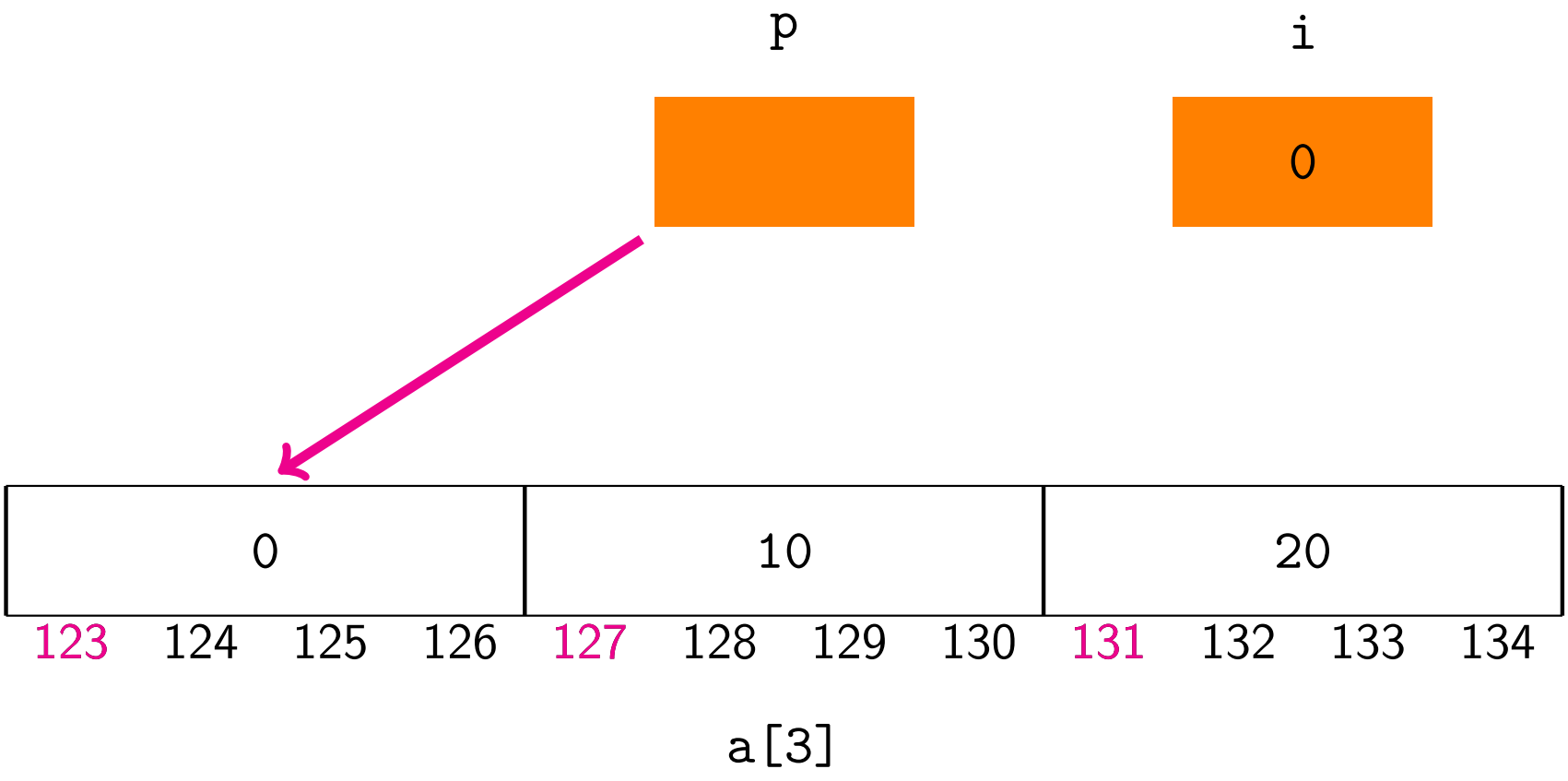

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
}
```



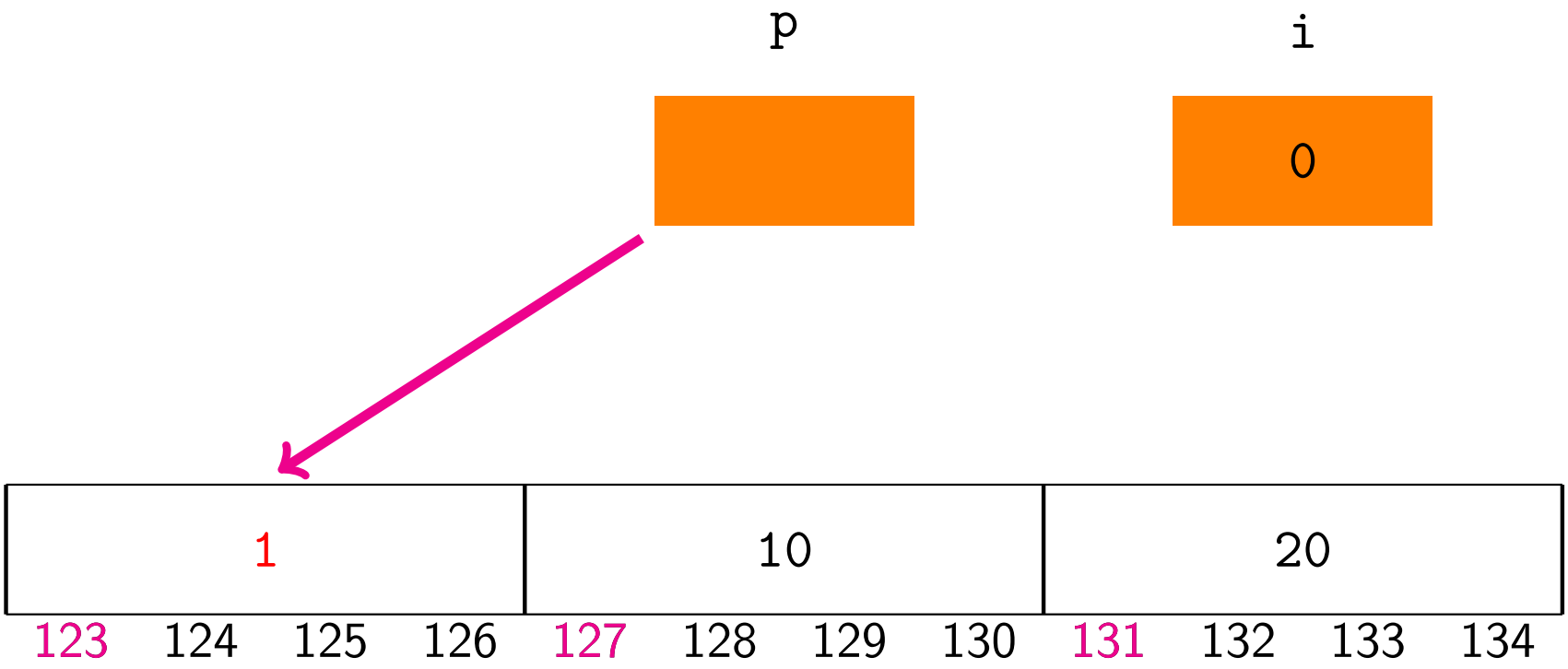

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
}
```



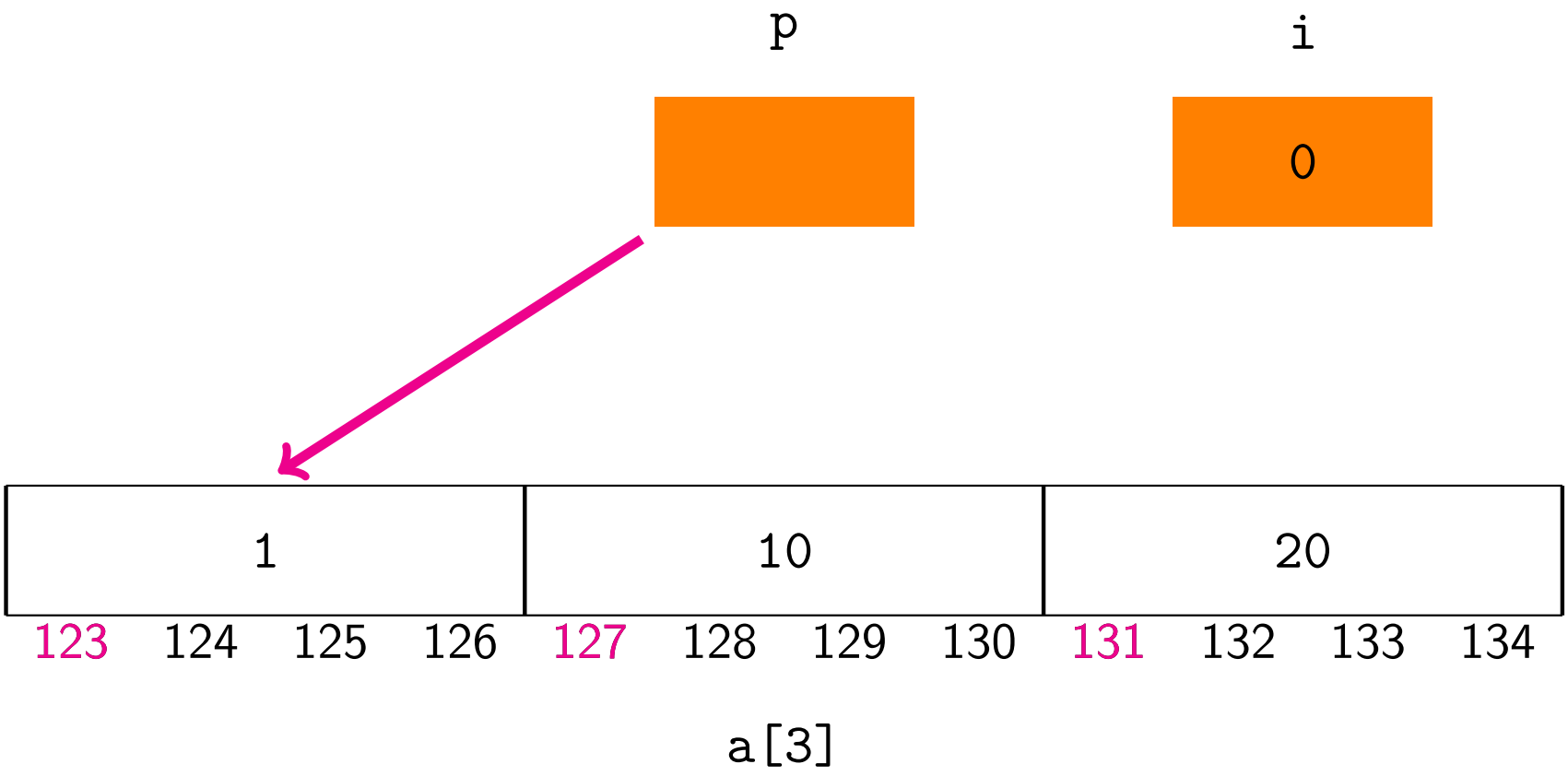

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



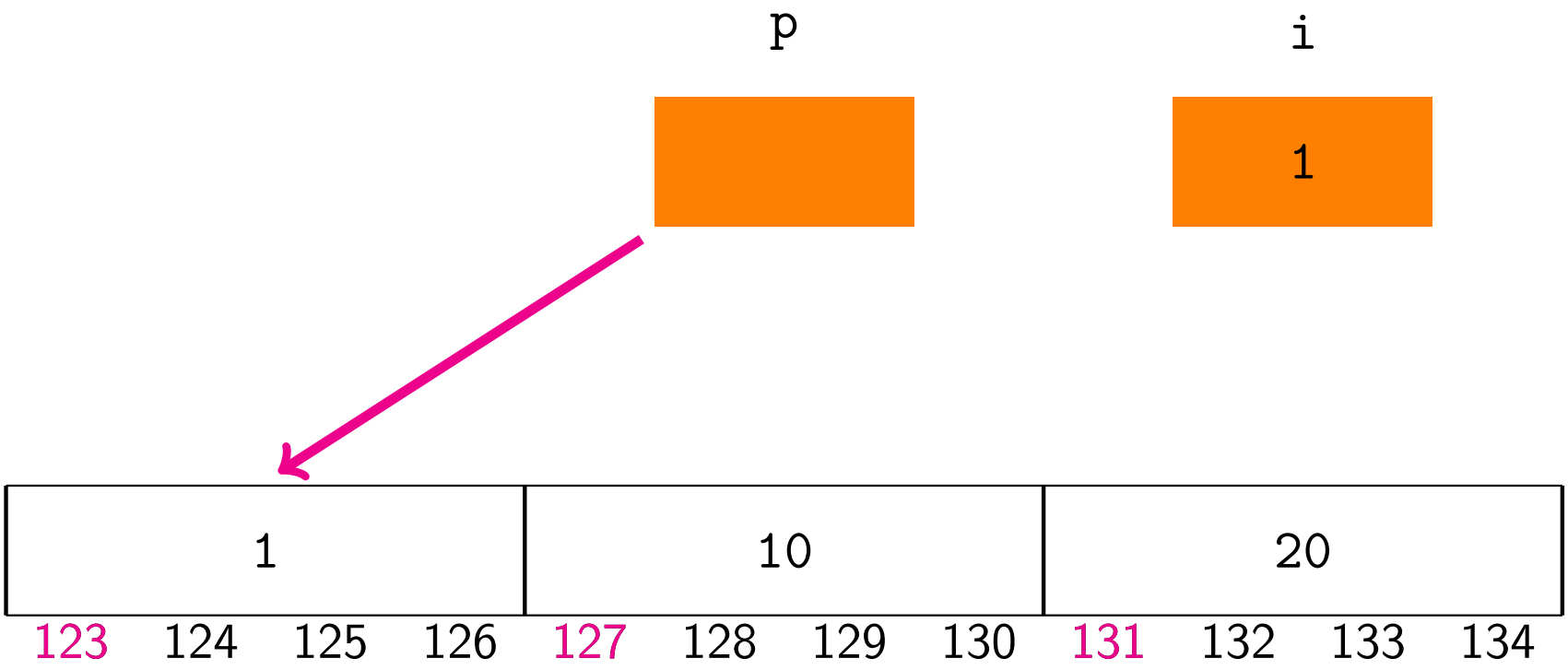

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



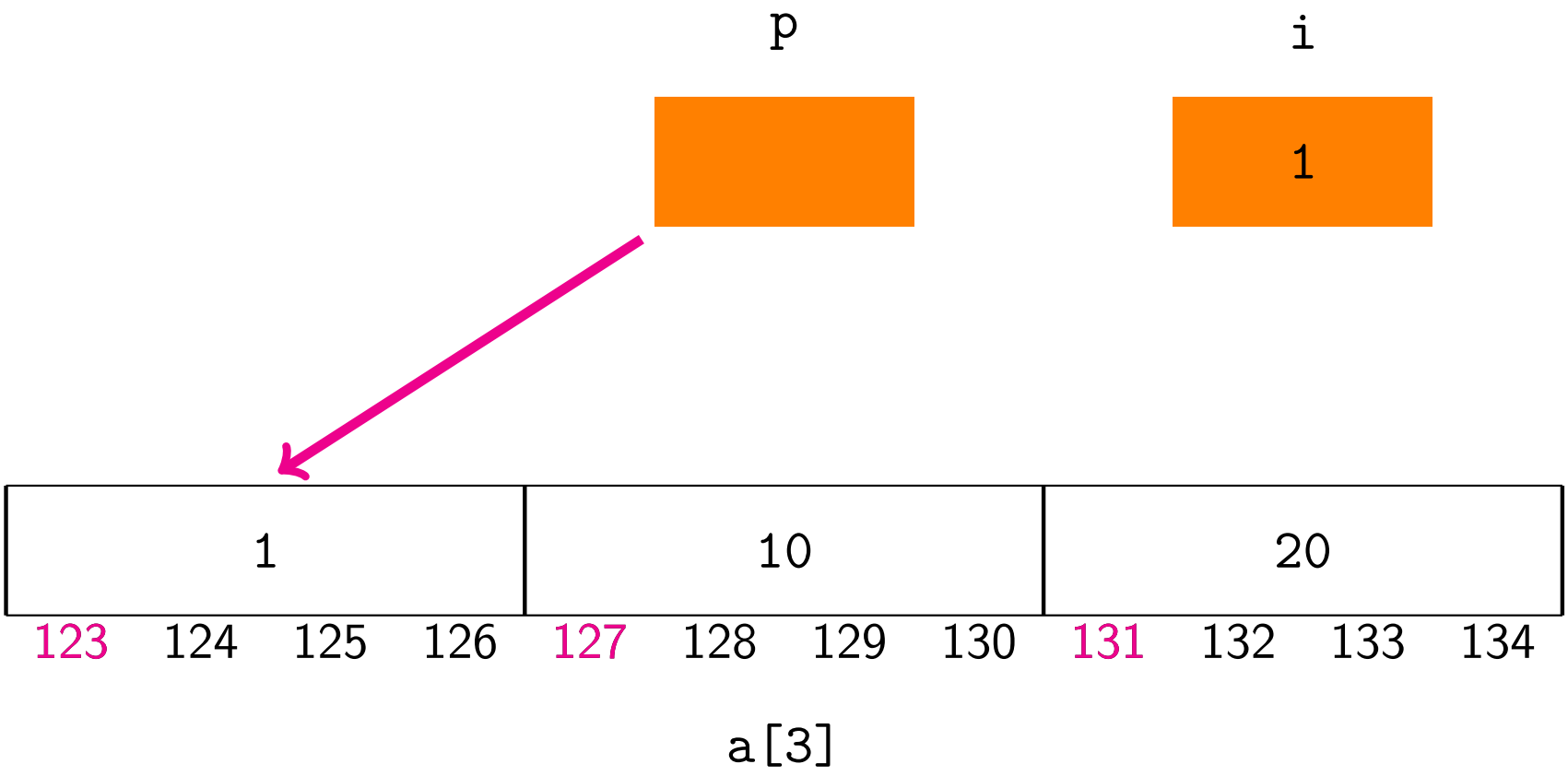

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



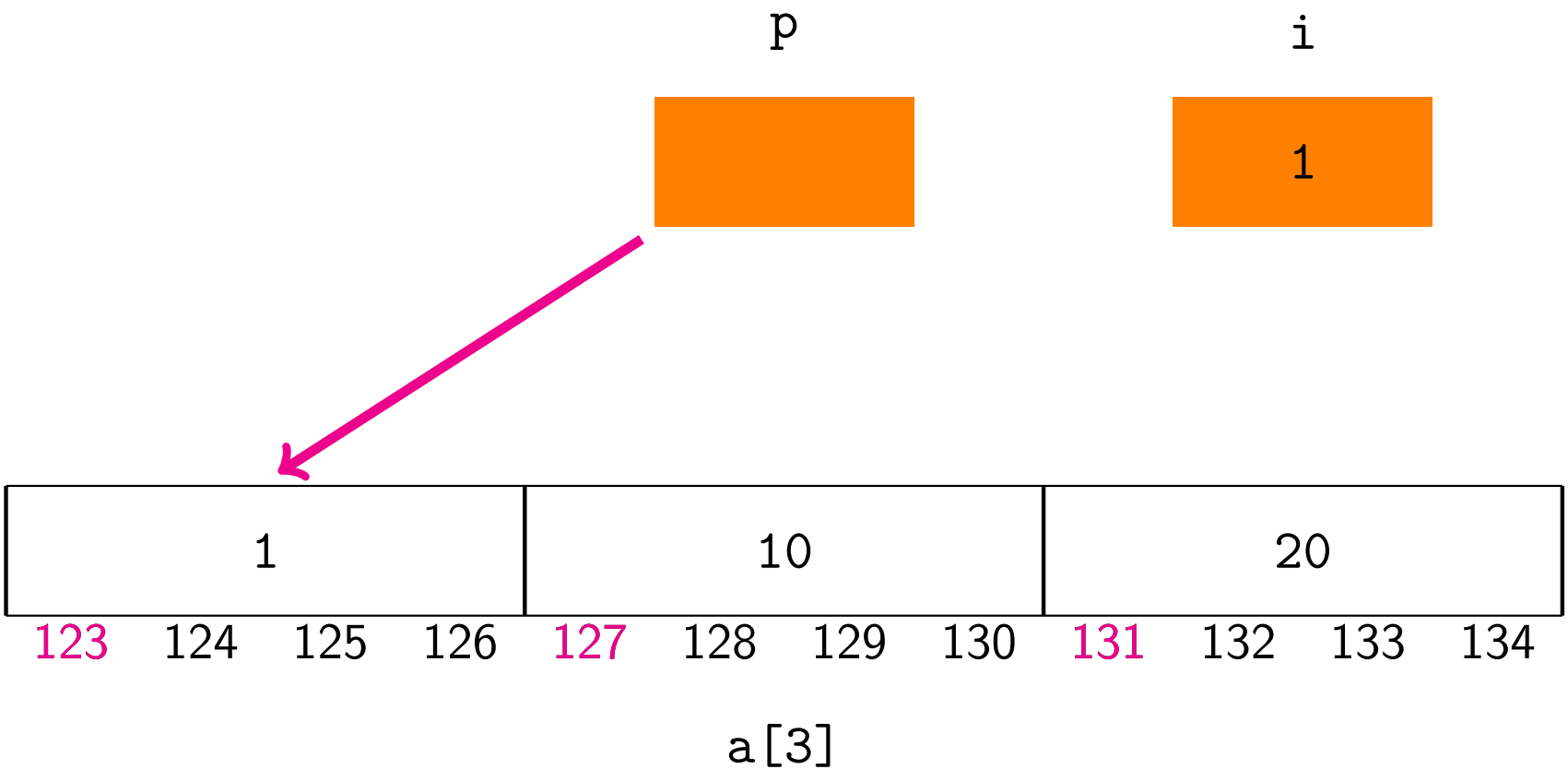

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
}
```



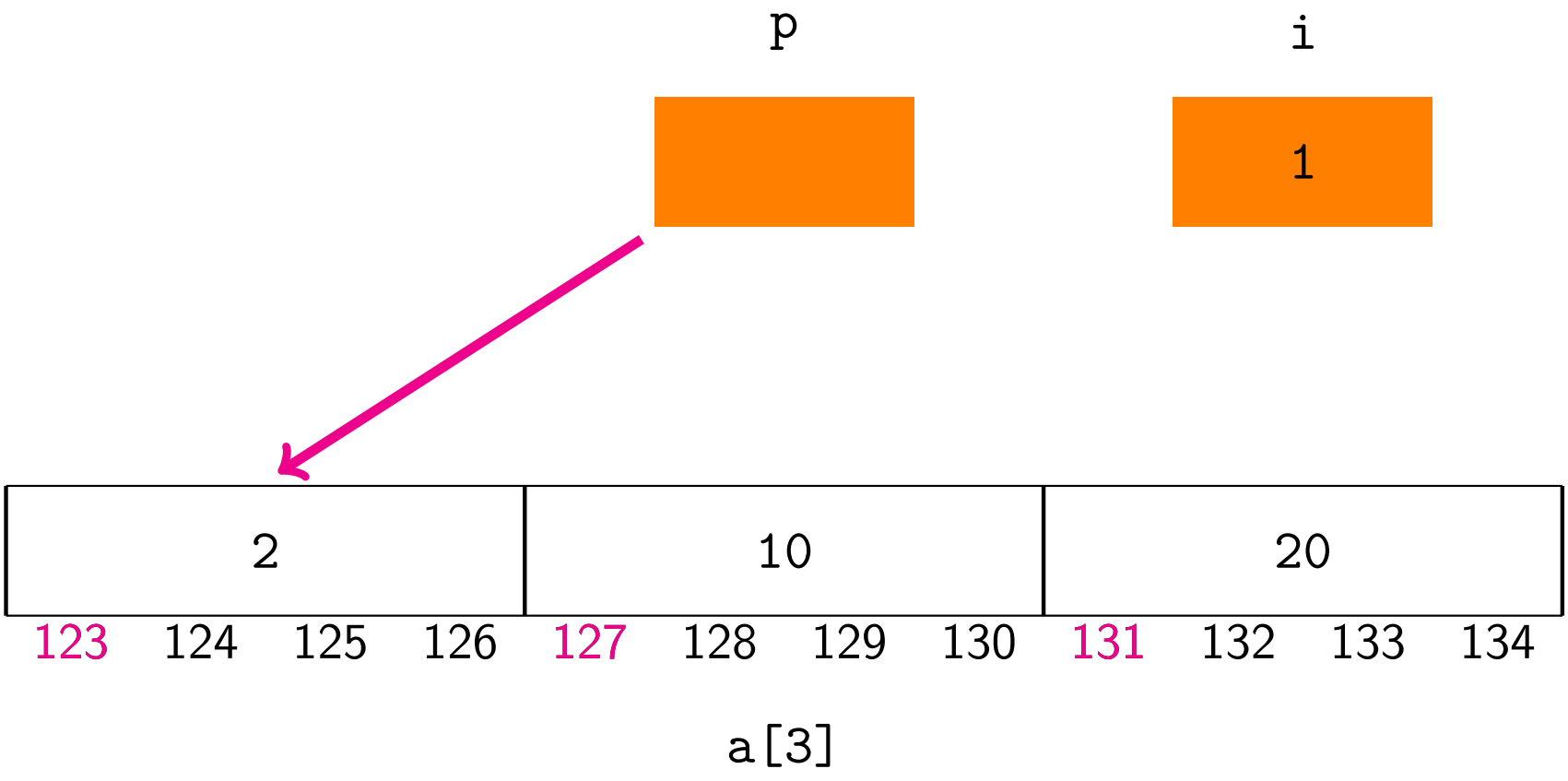

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
}
```



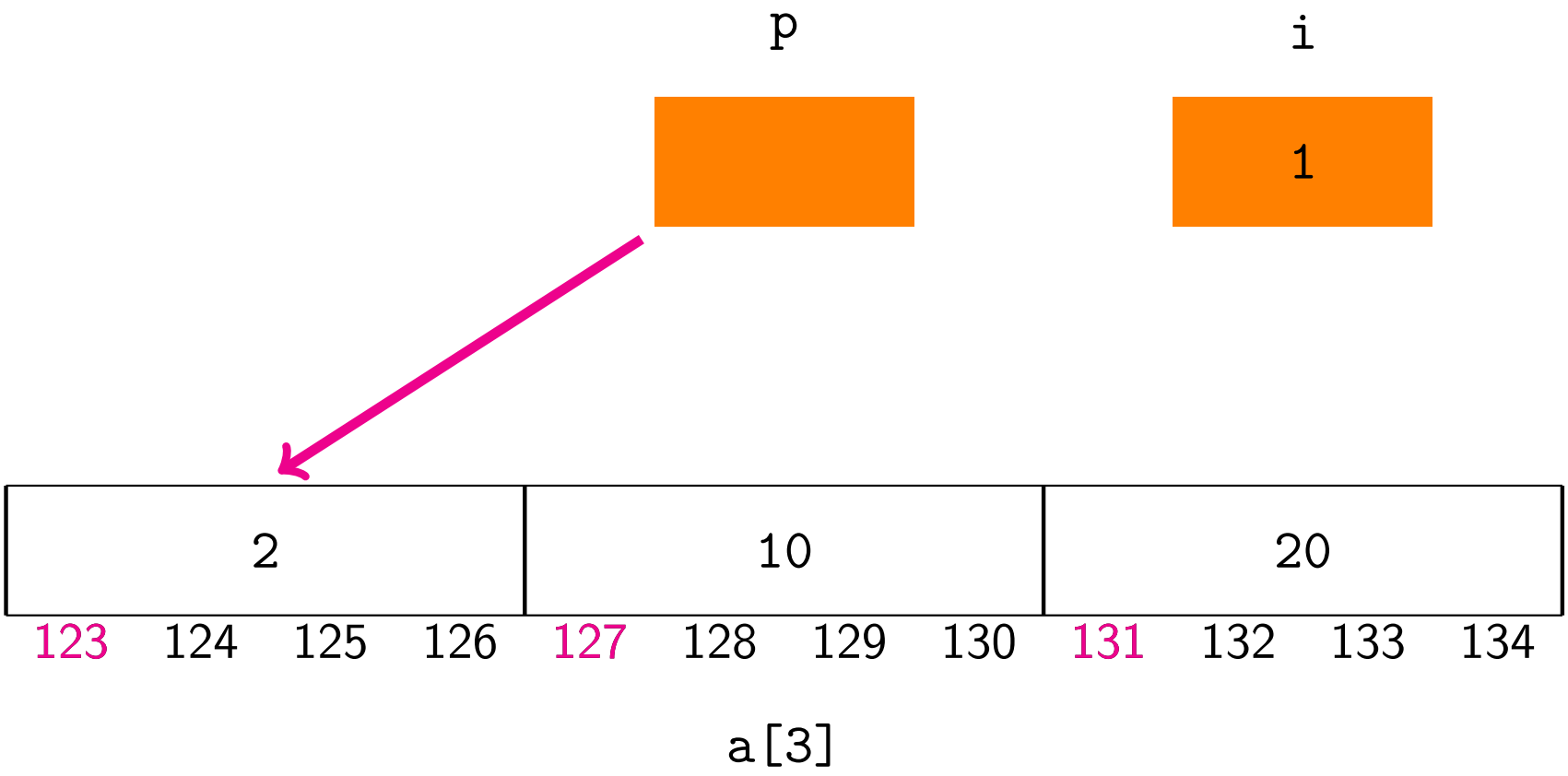

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



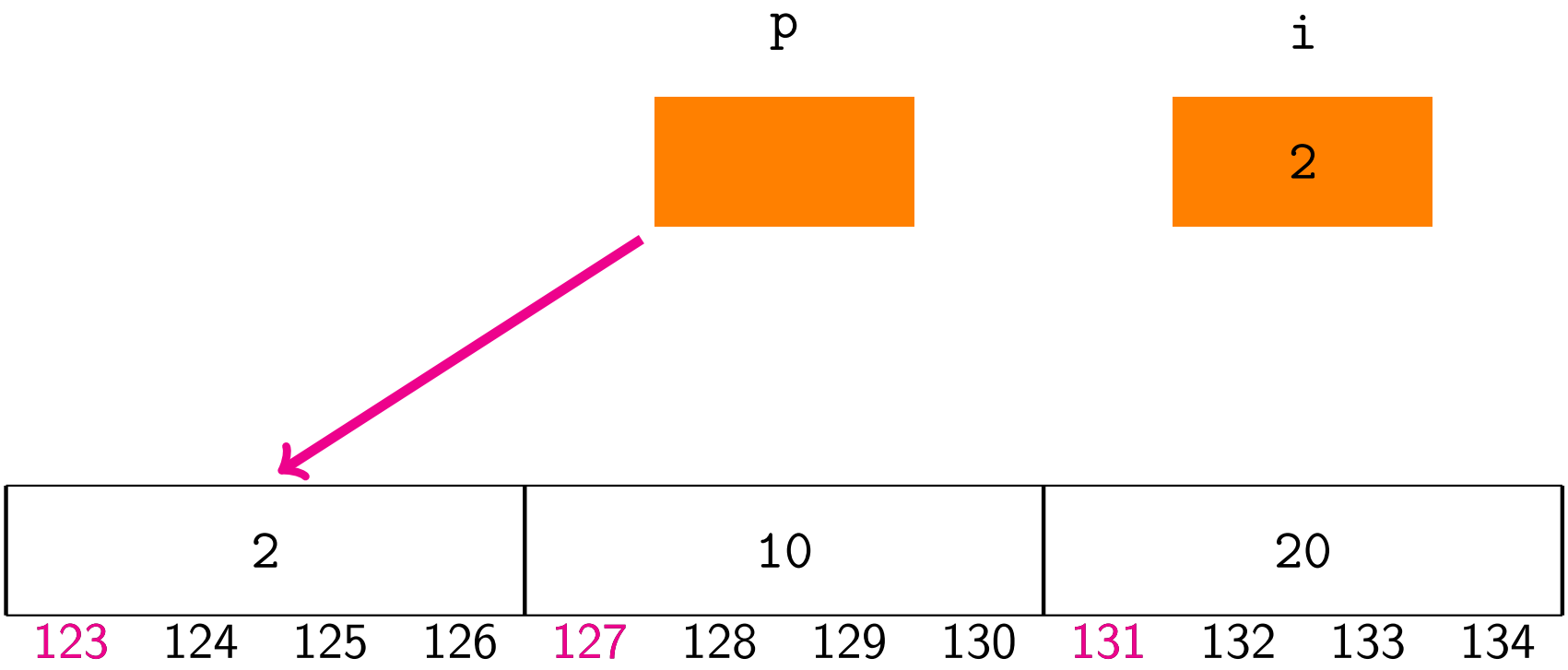

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



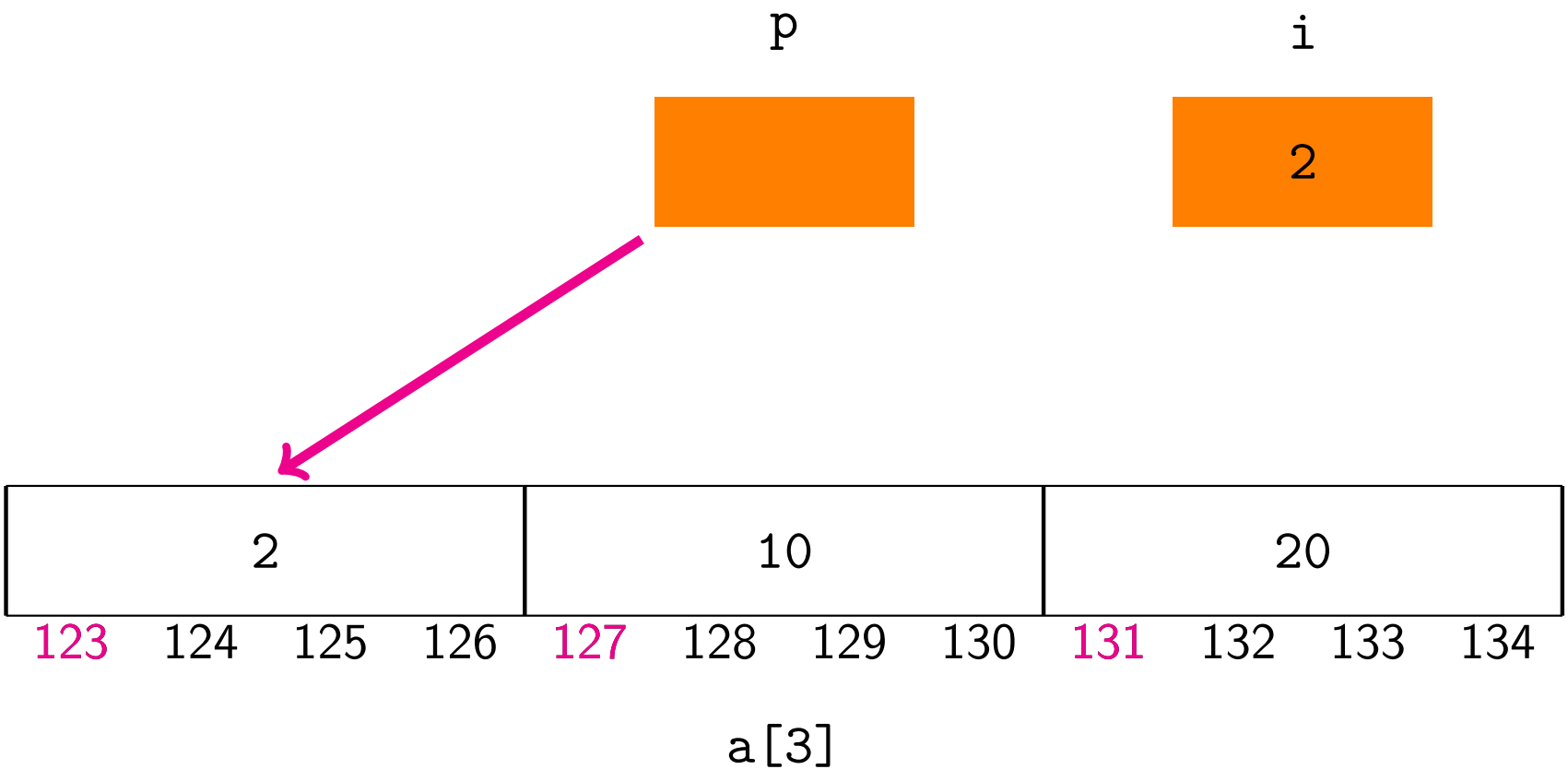

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



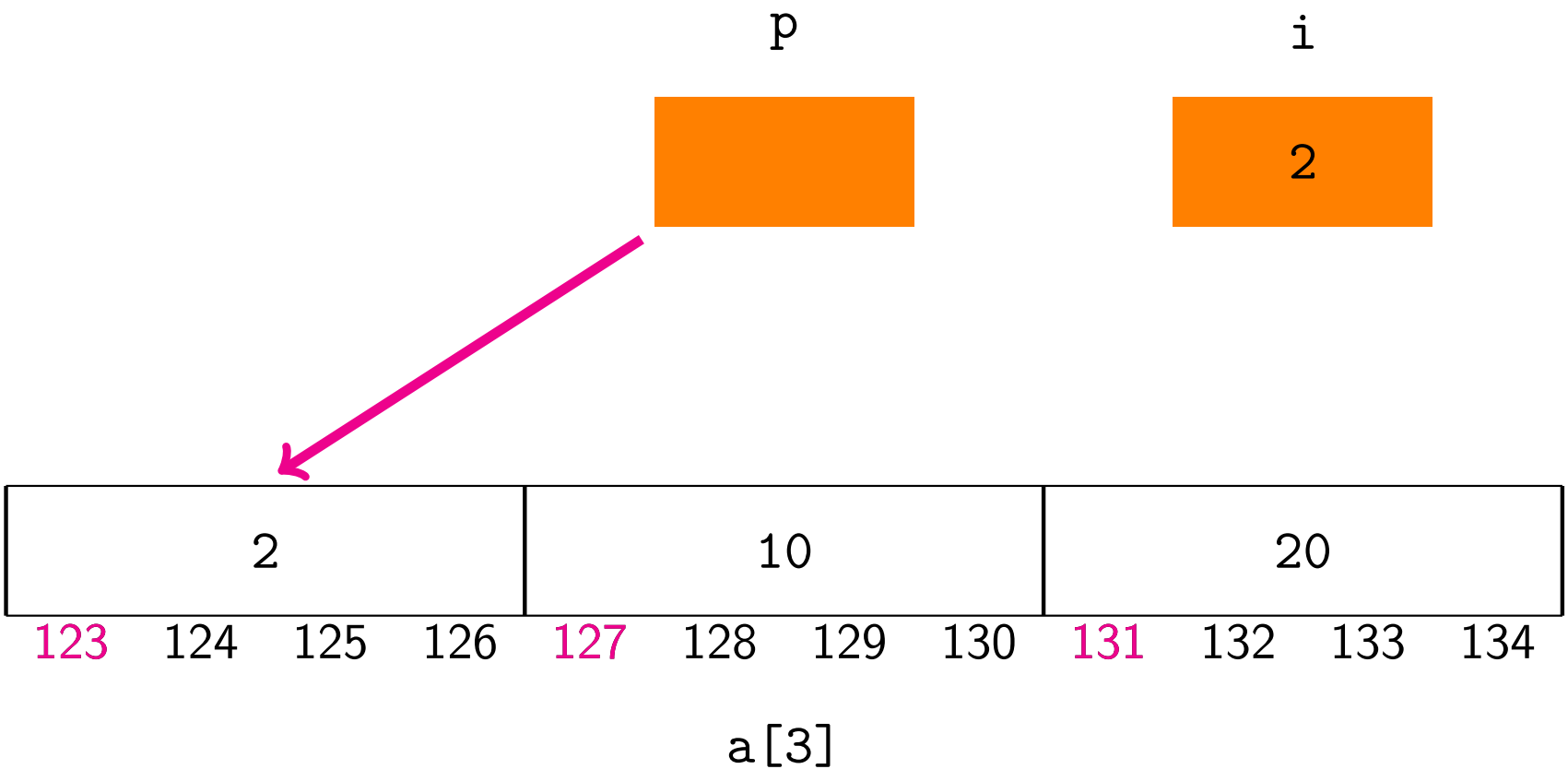

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



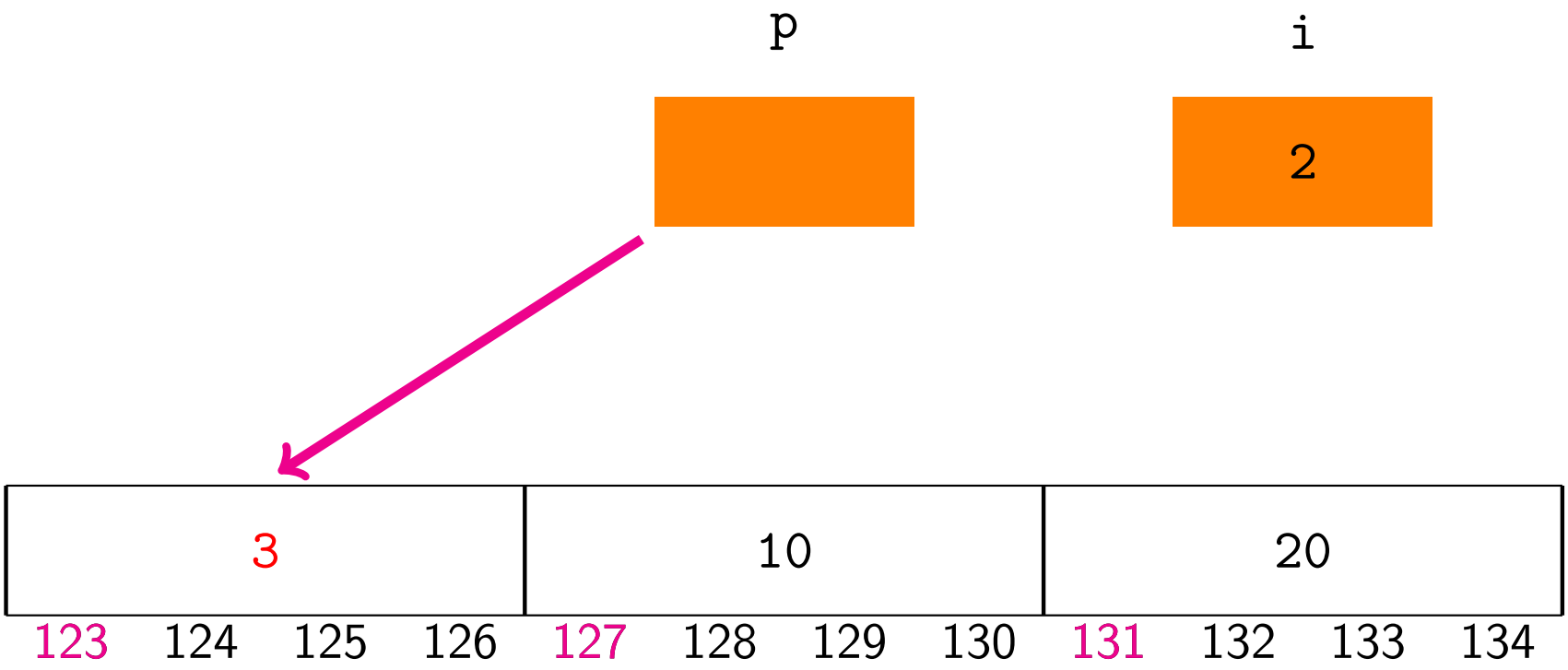

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



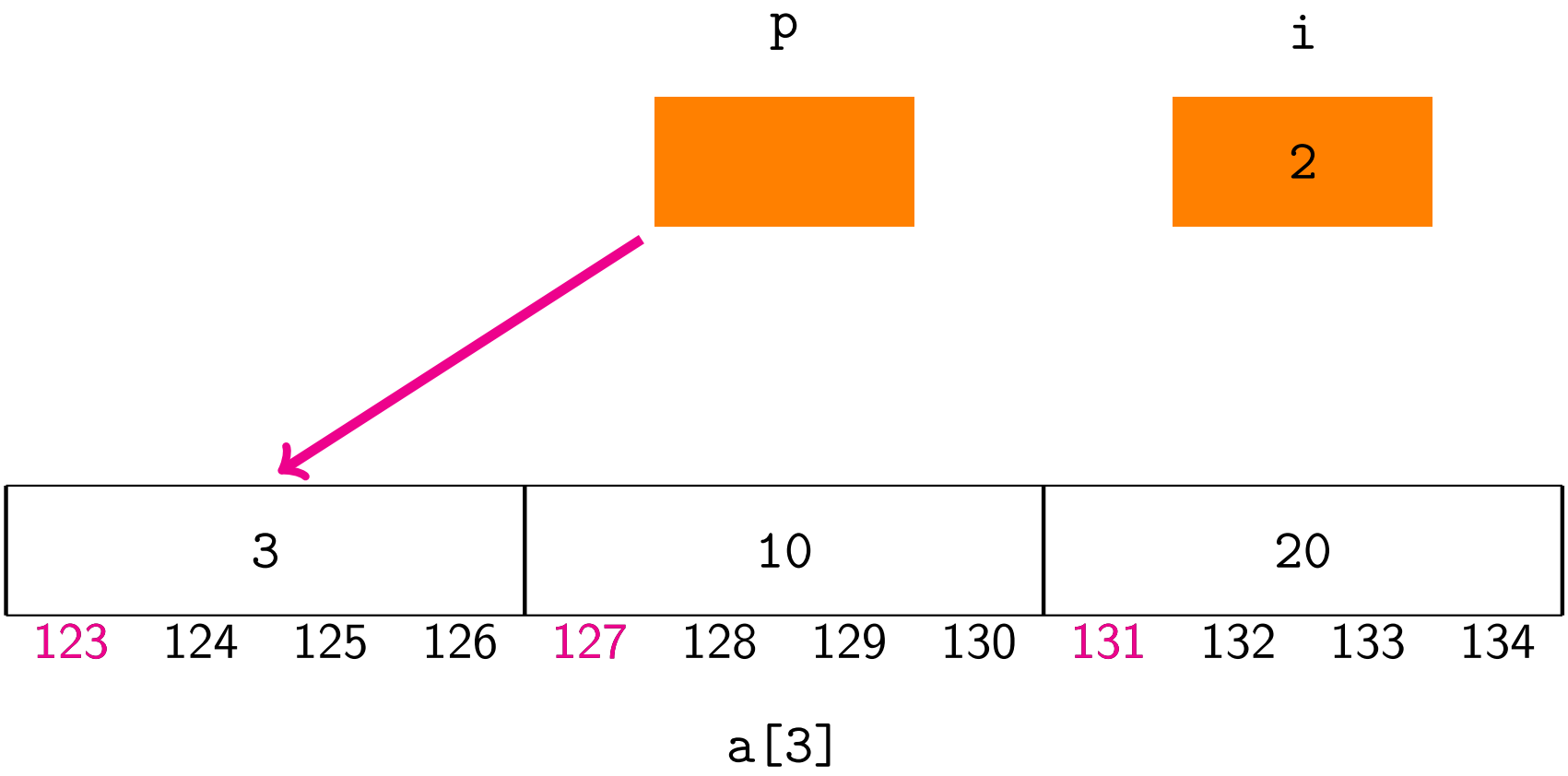

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



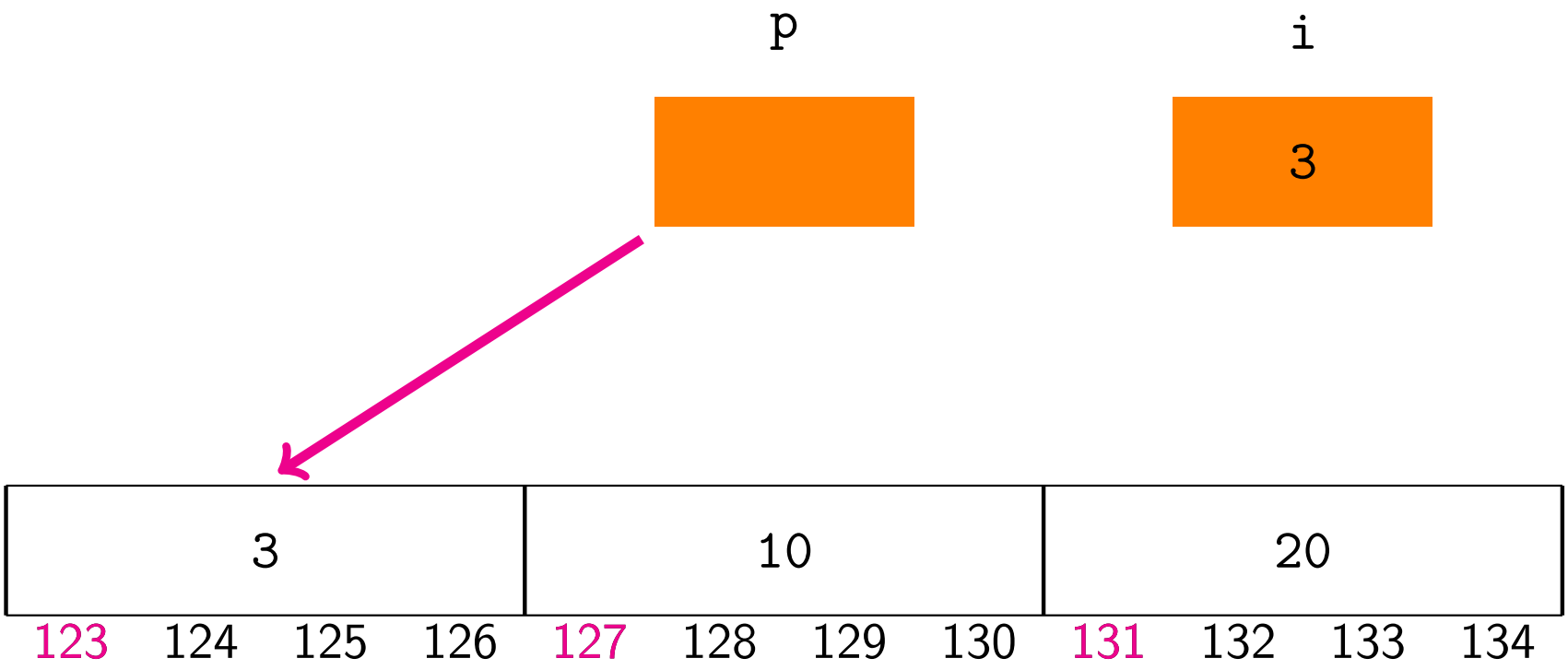

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



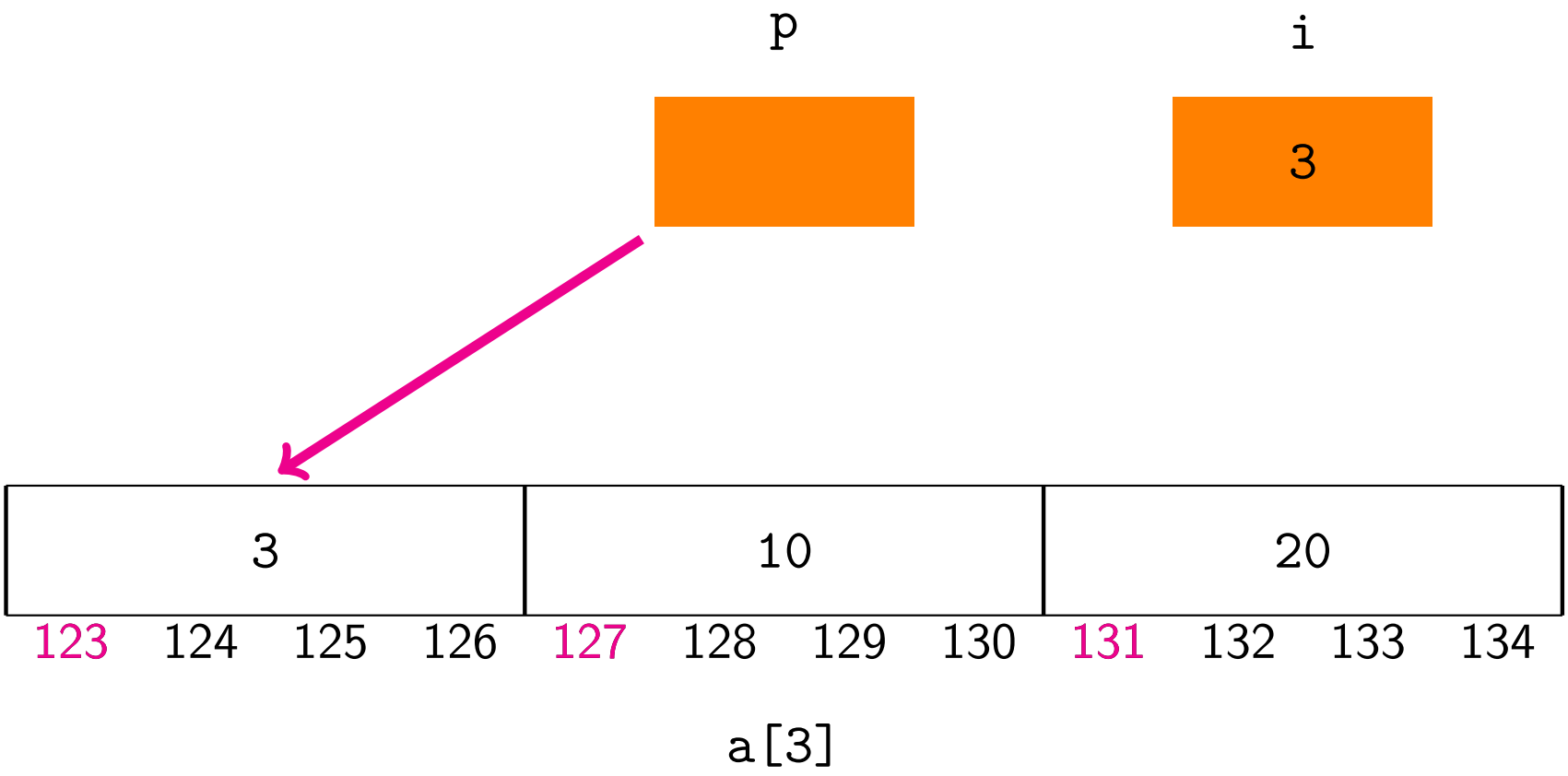

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



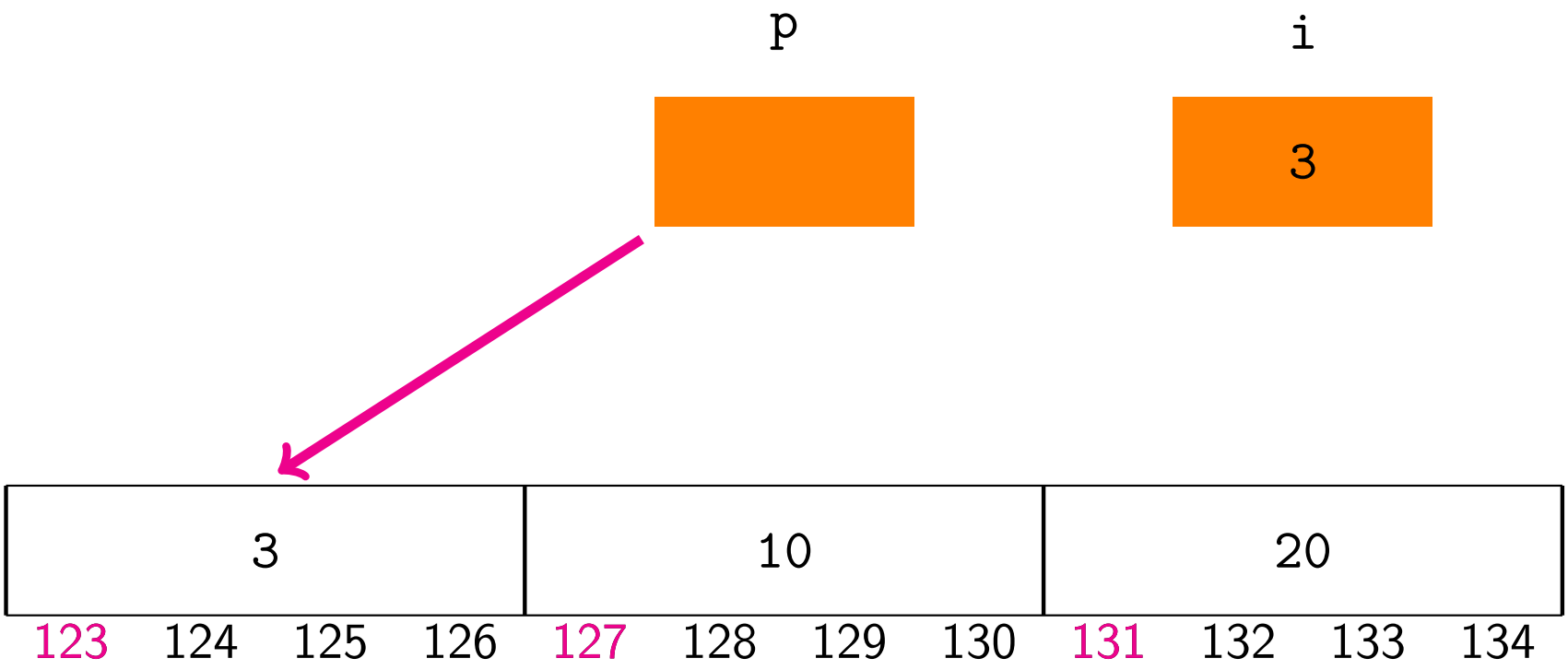

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```




```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```

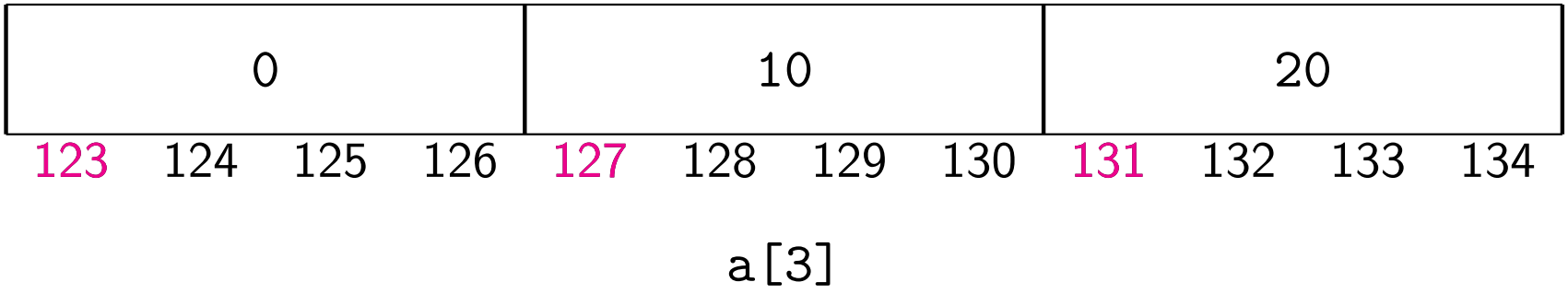


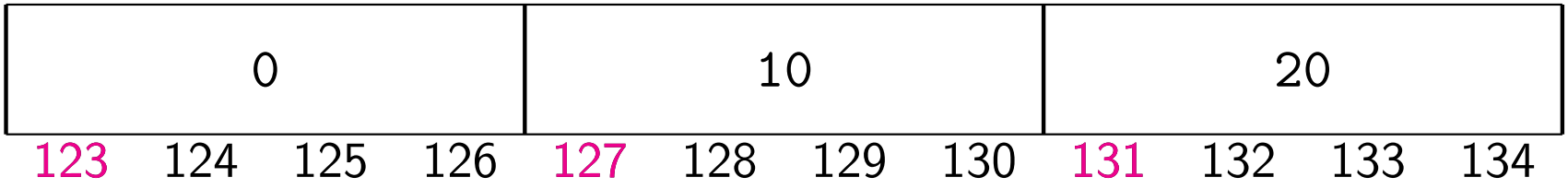

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
   *p = *p + 1;
}
```



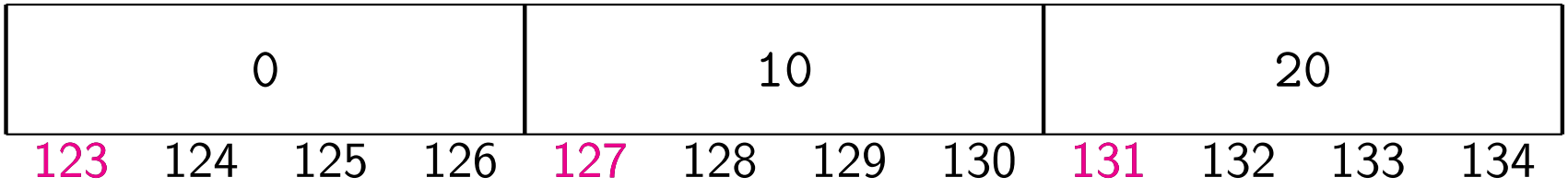

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
}
```

Done!

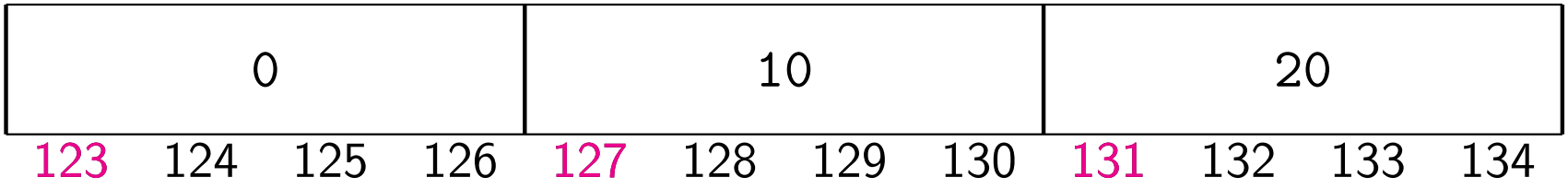

0
10
20
123
124
125
126
127
128
129
130
131
132
133
134
a[3]



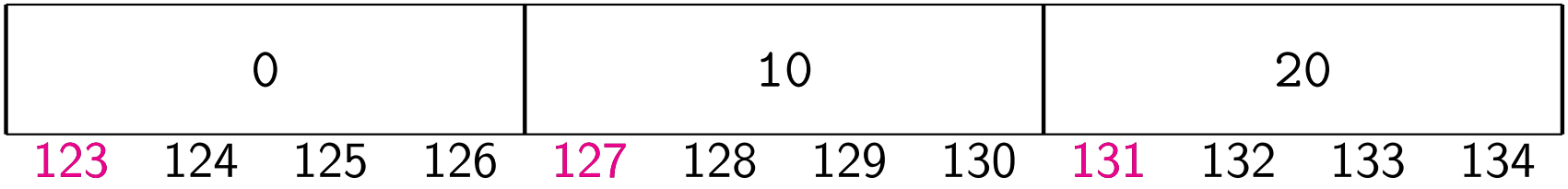

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
}
```



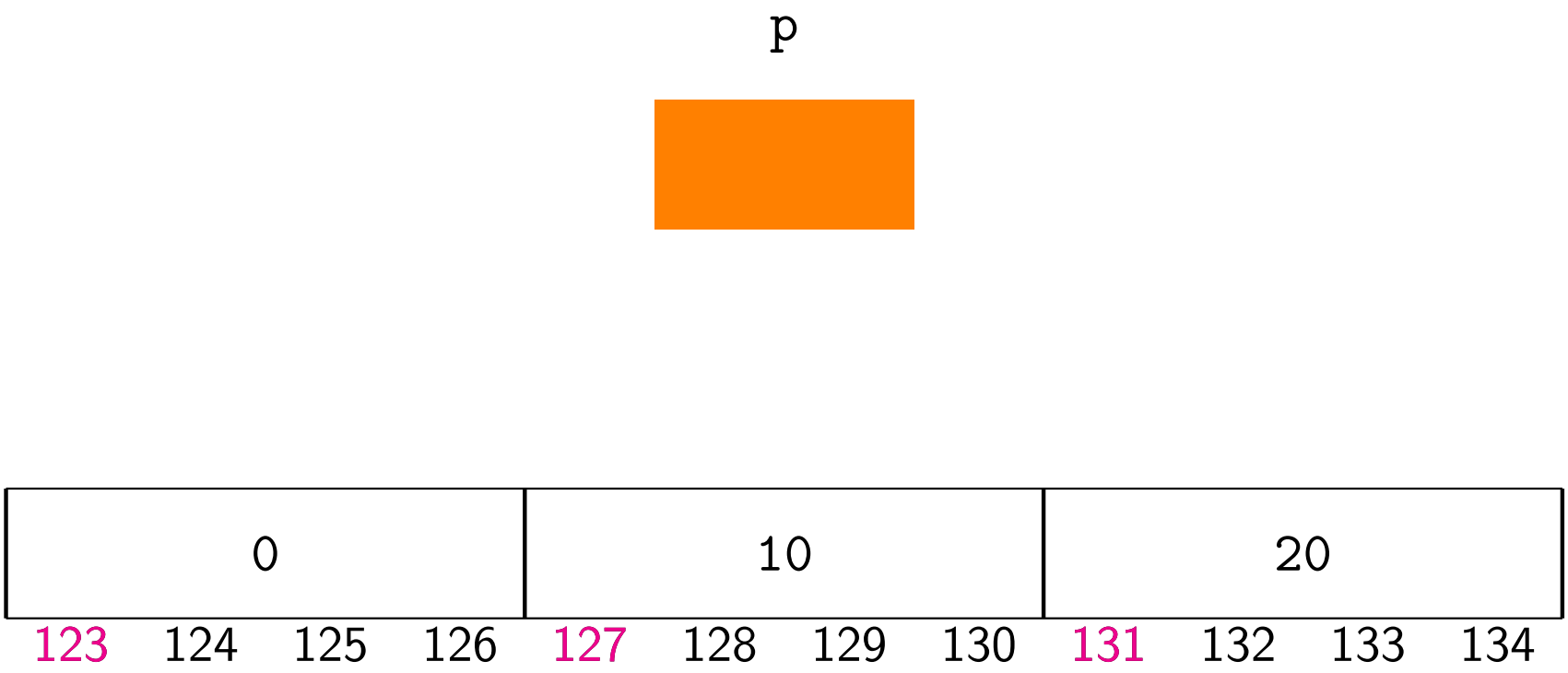

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



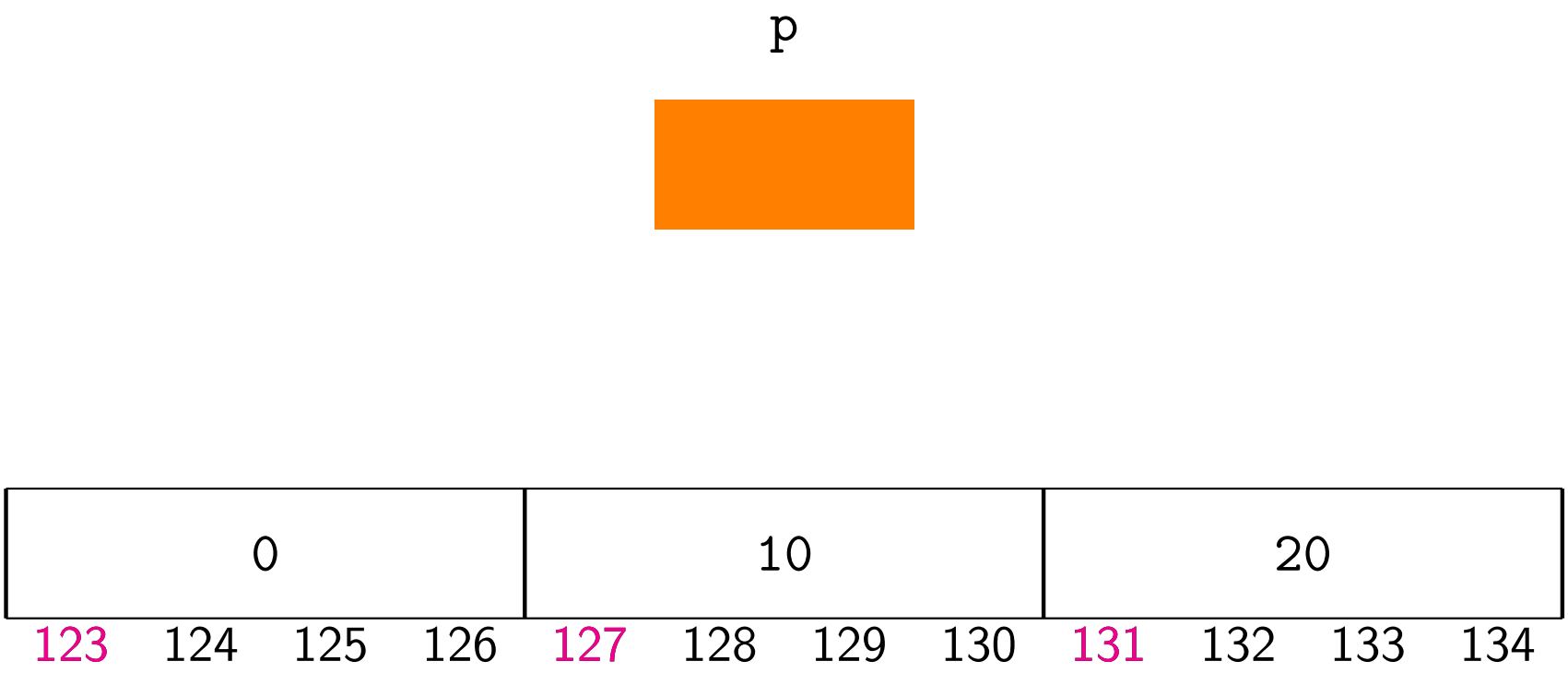

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



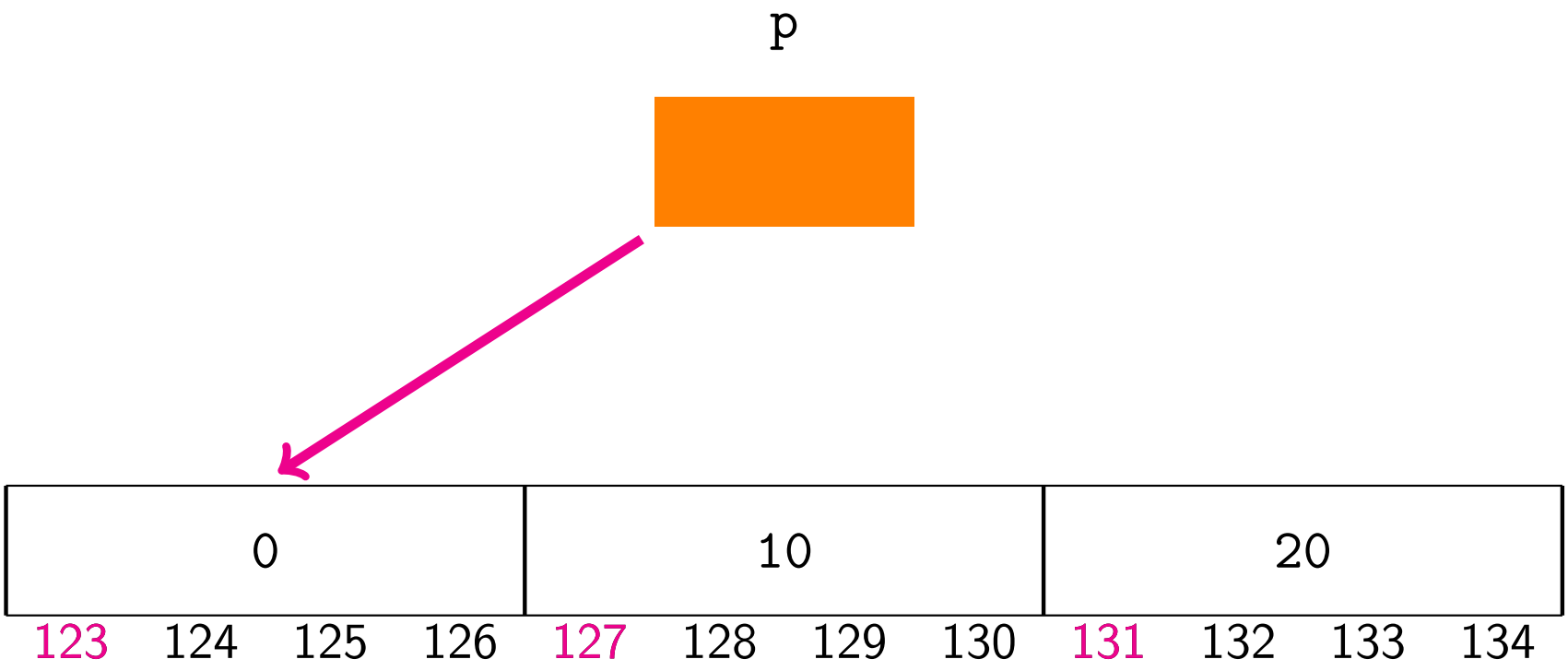

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



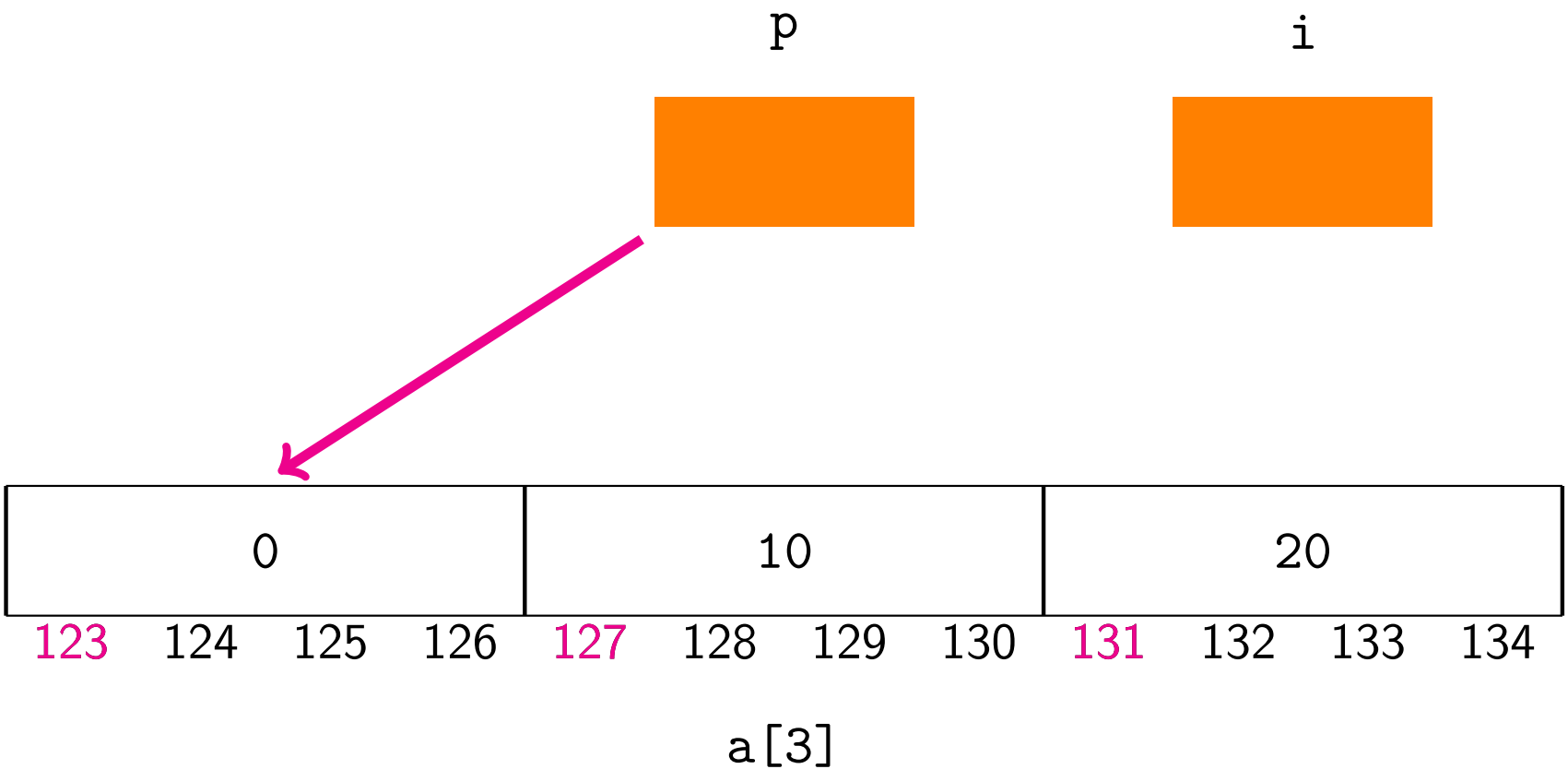

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



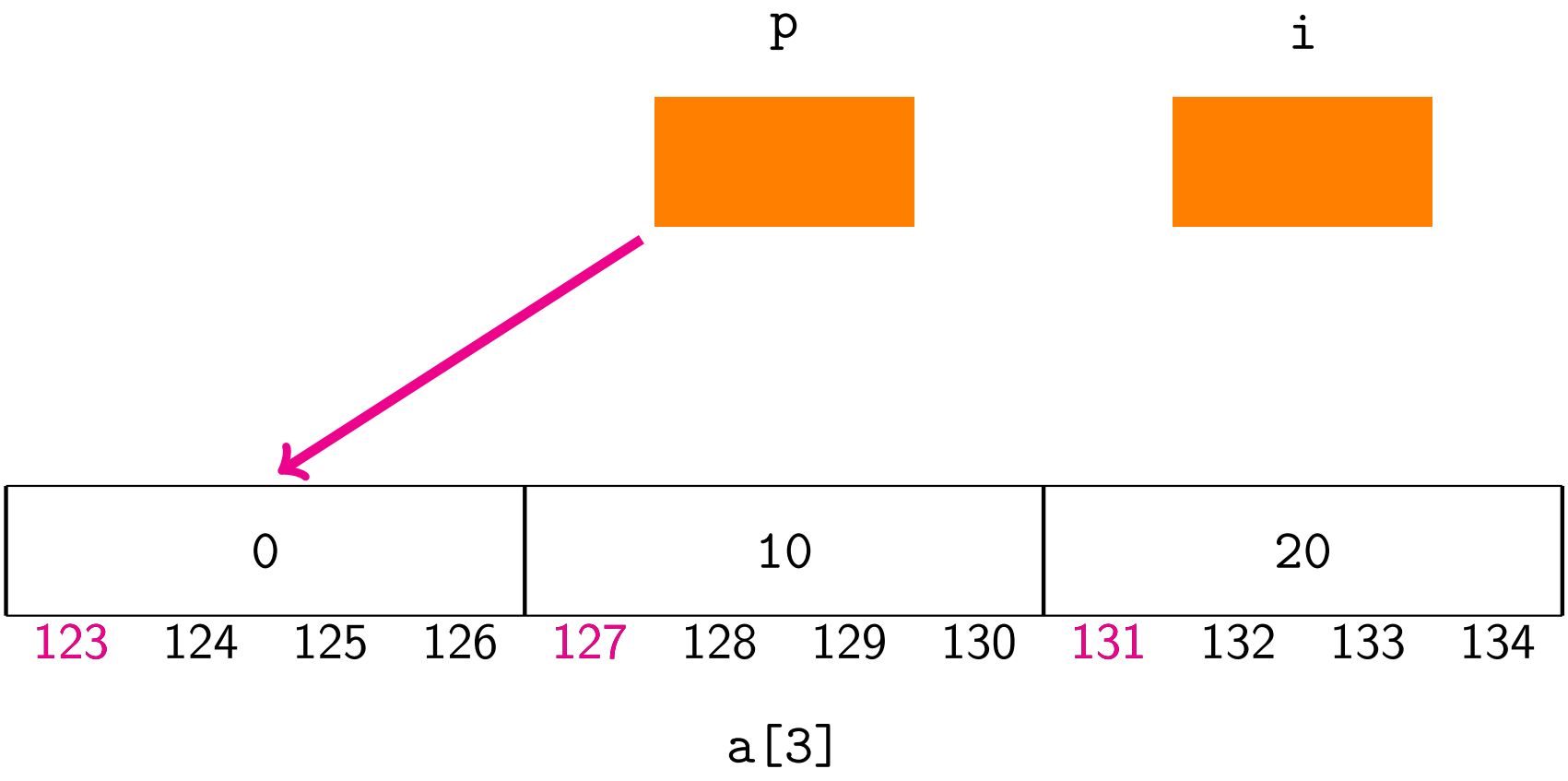

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



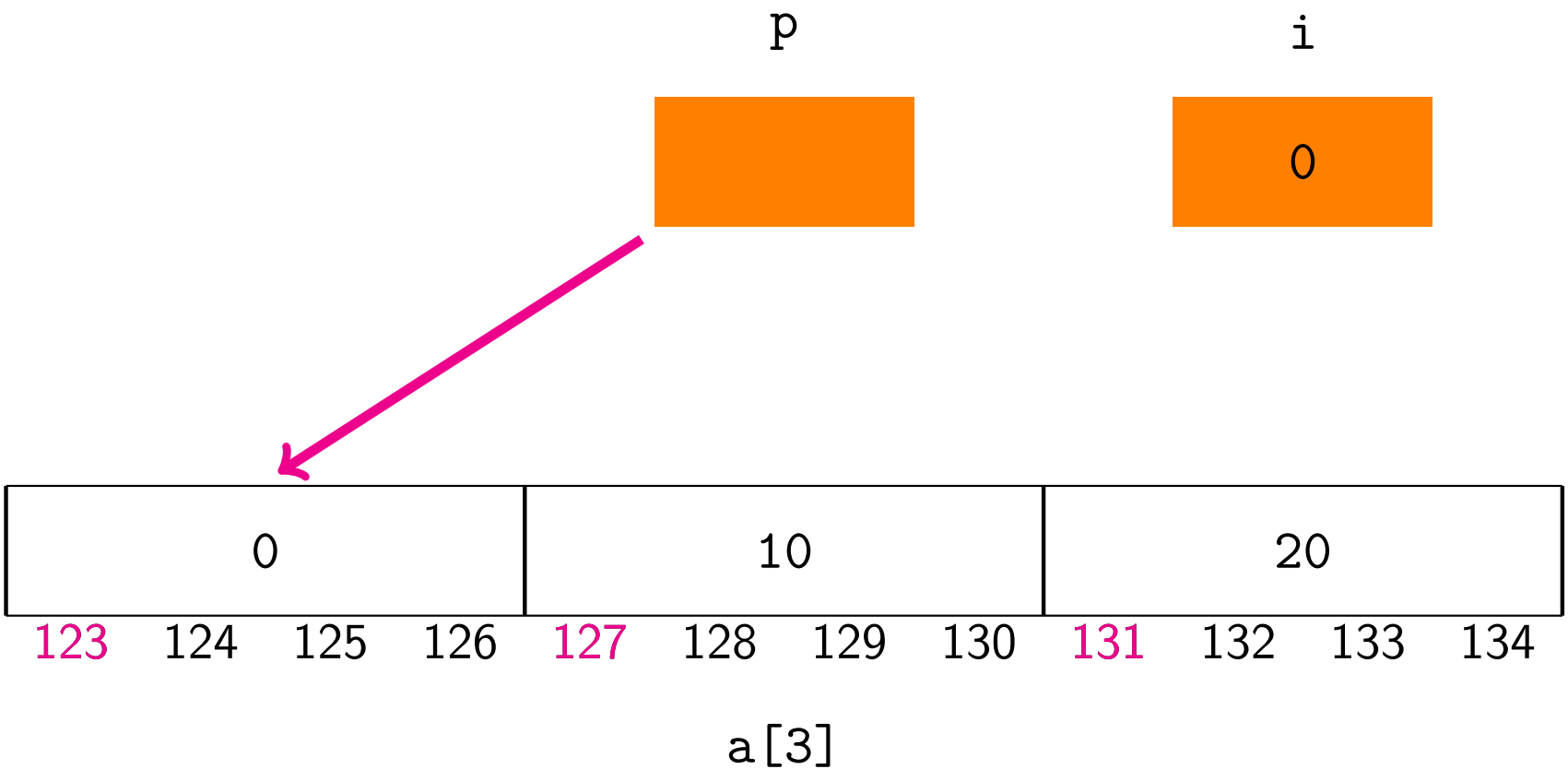

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



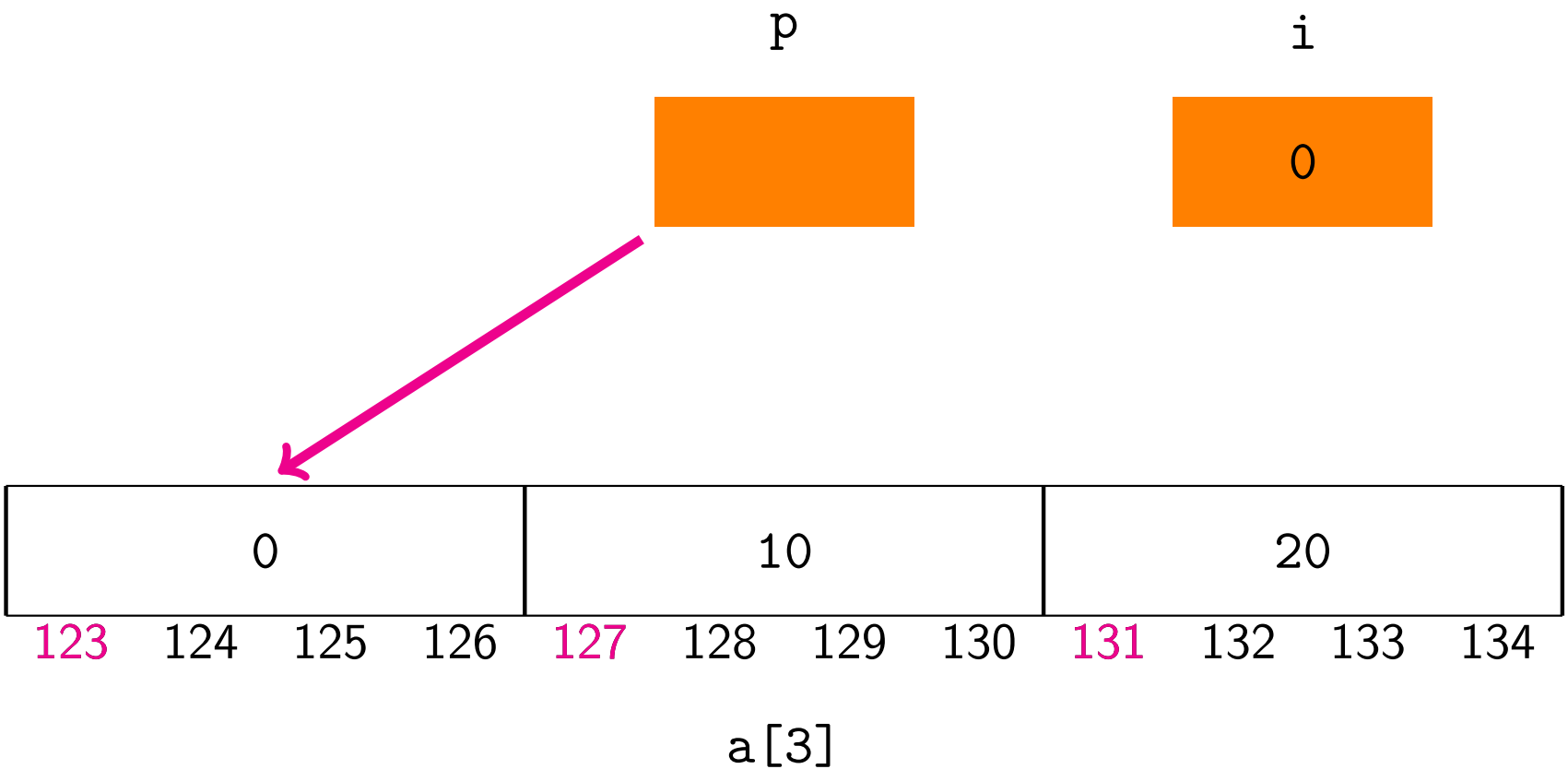

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



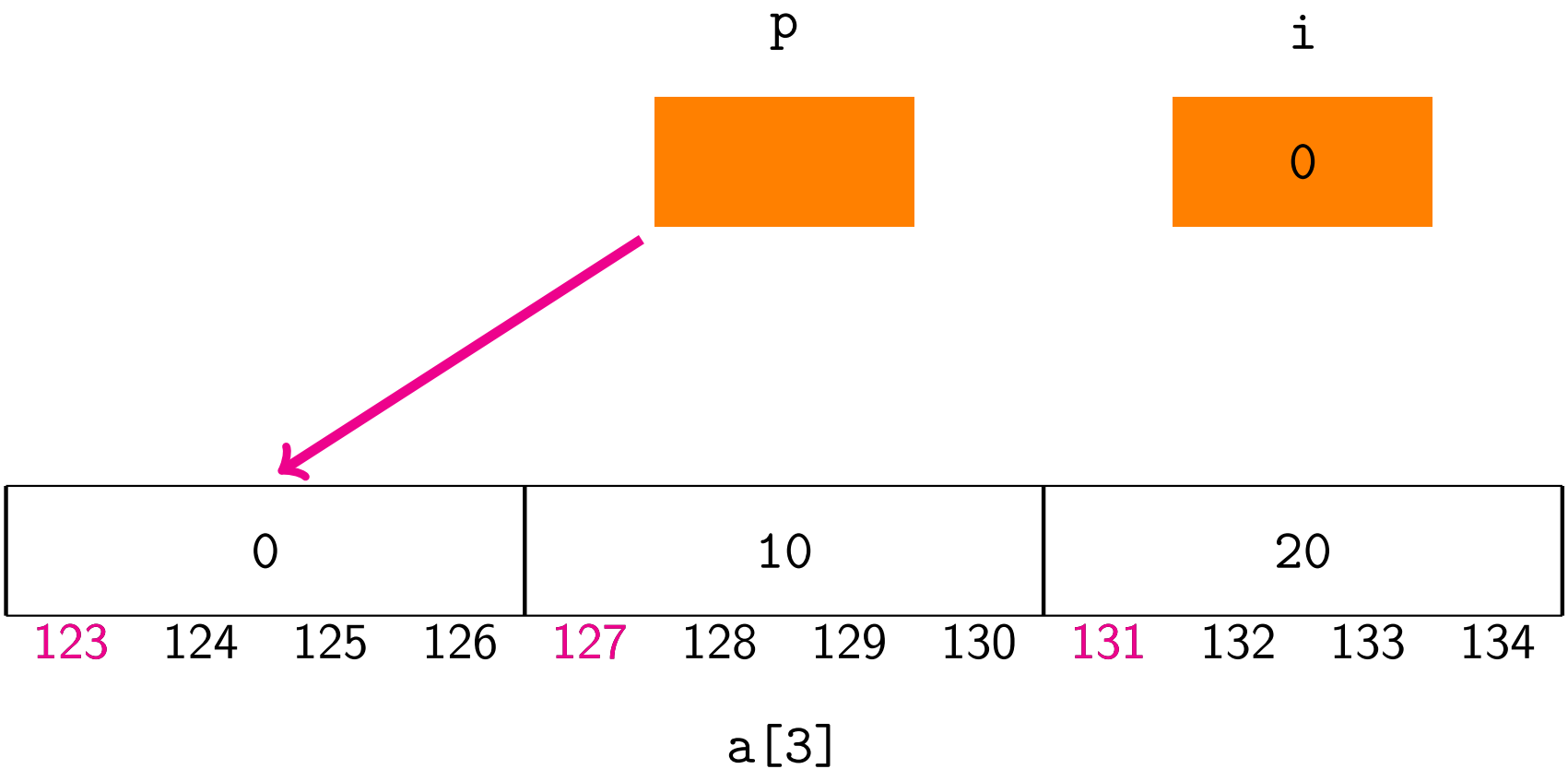

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



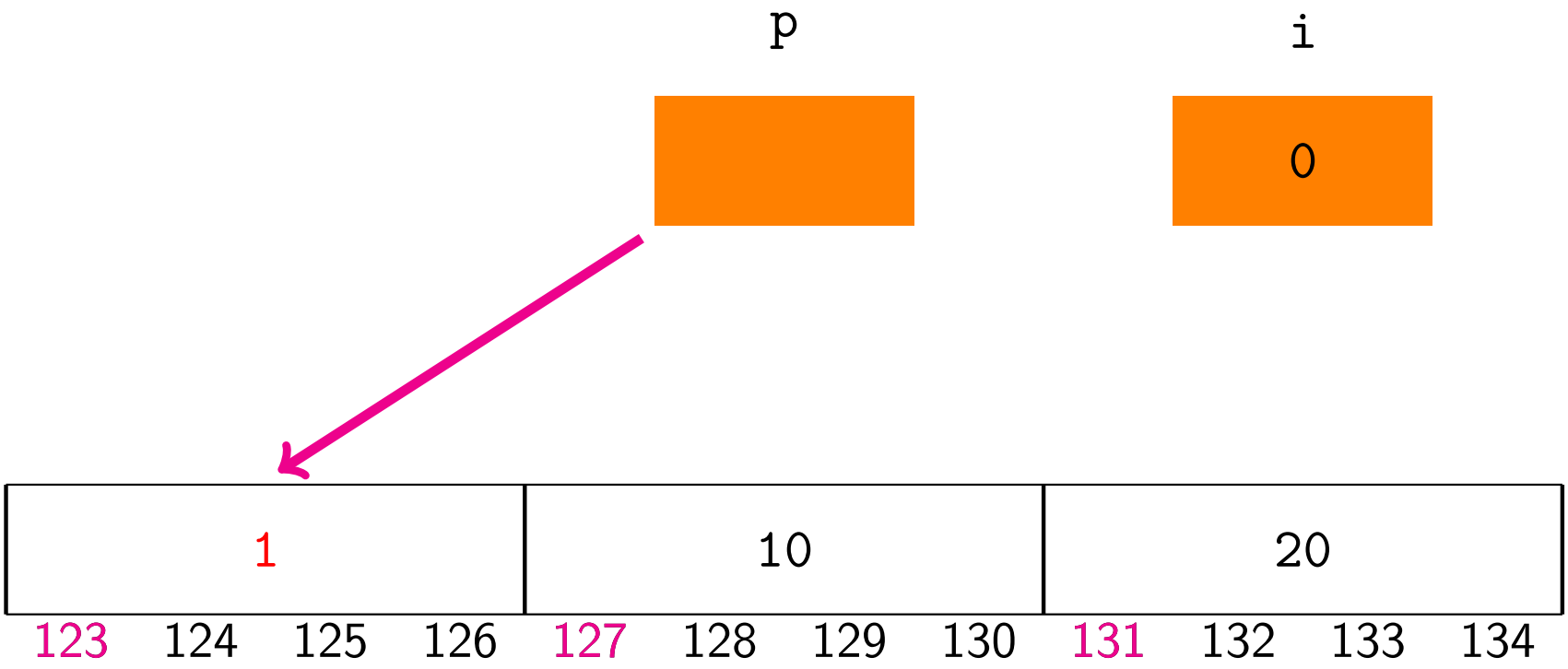

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



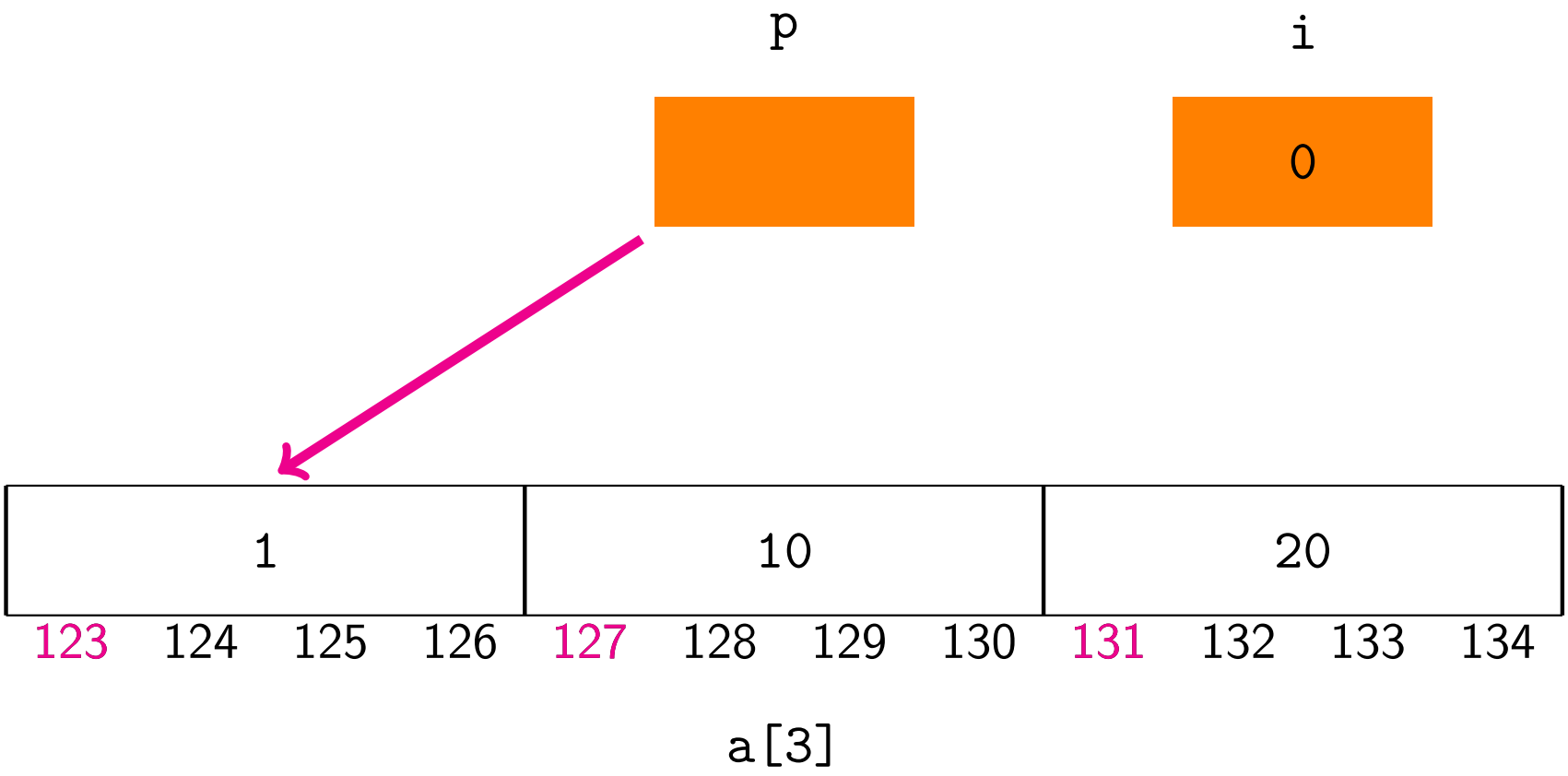

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



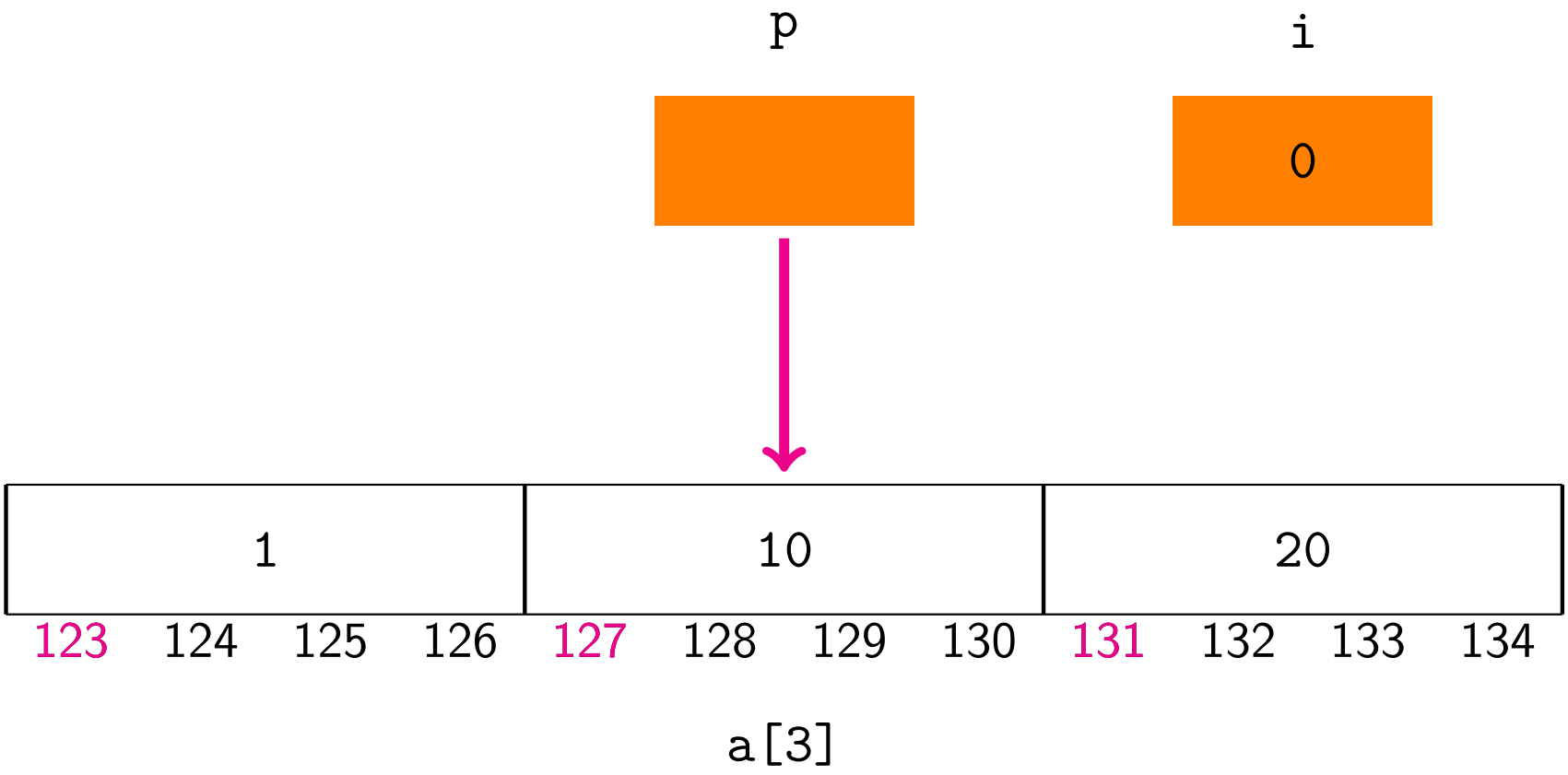

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



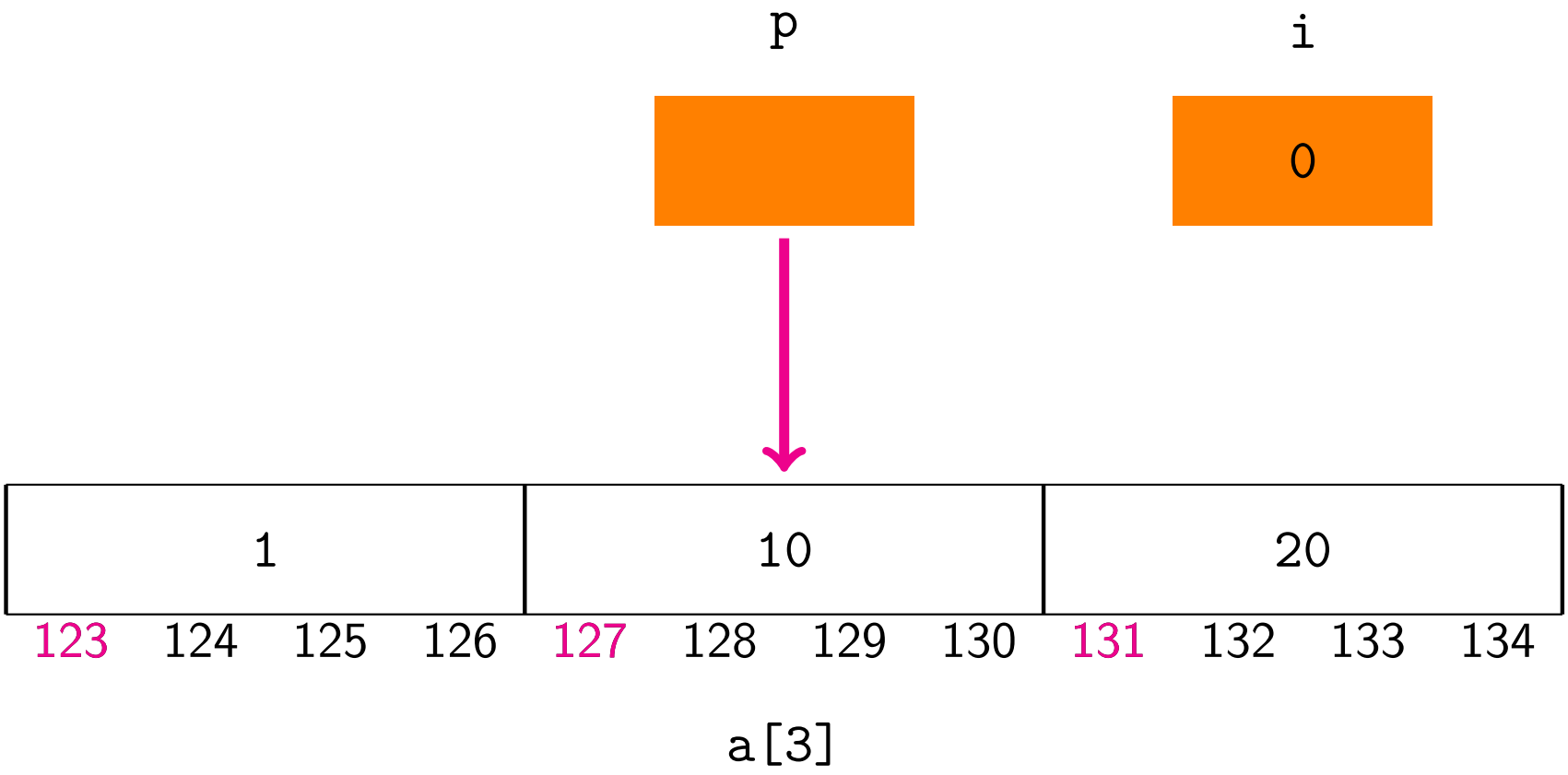

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



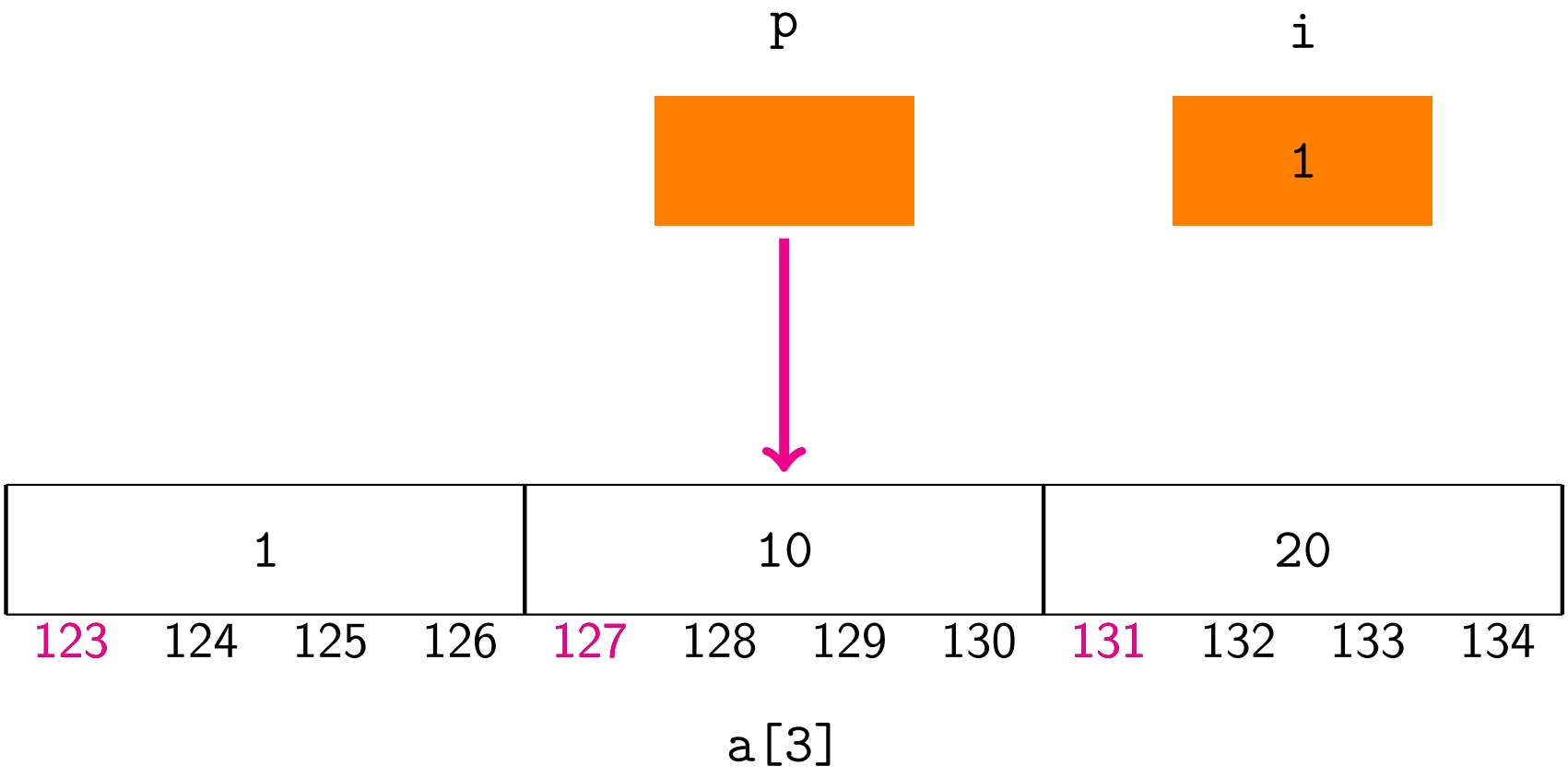

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



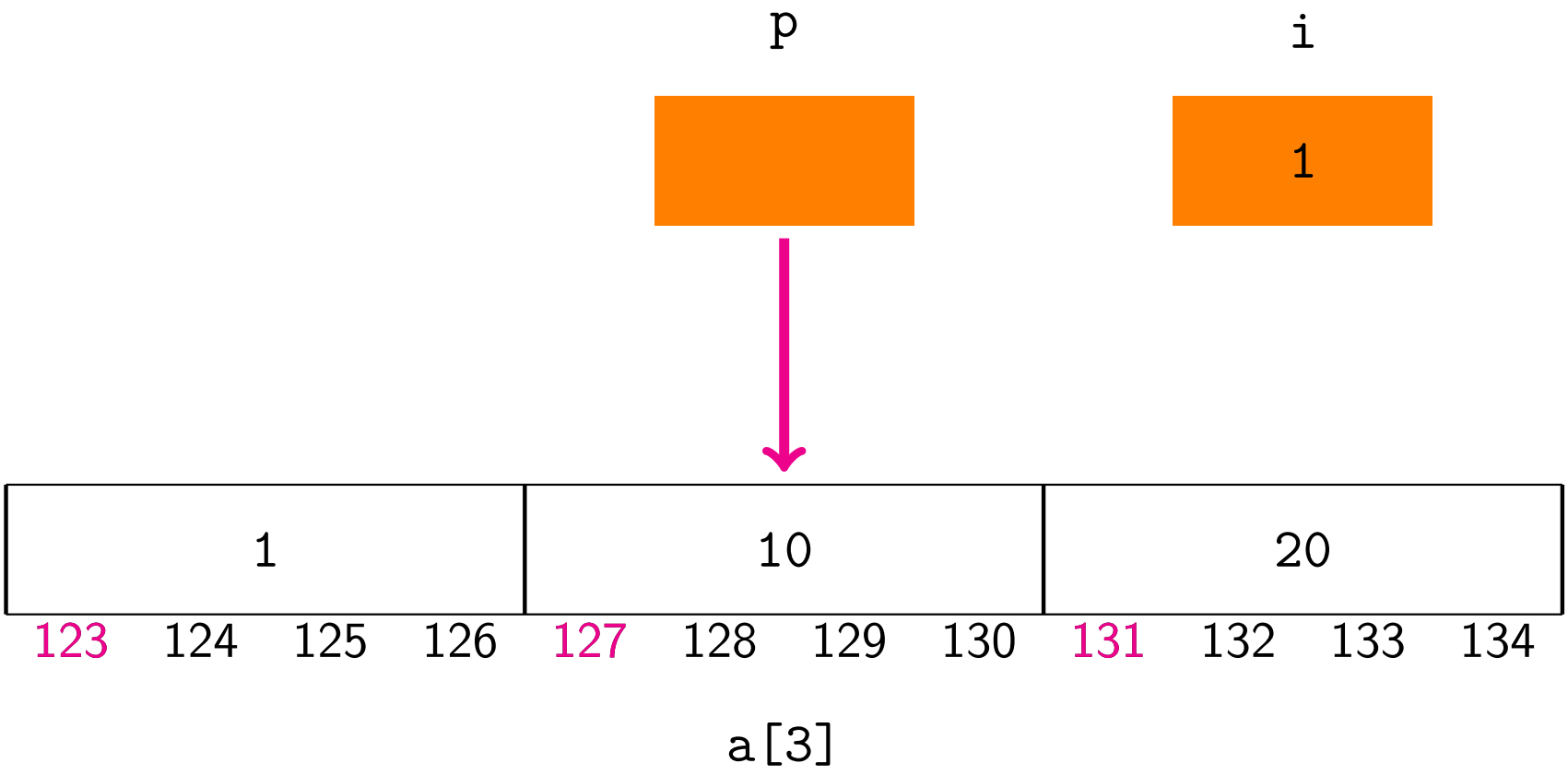

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



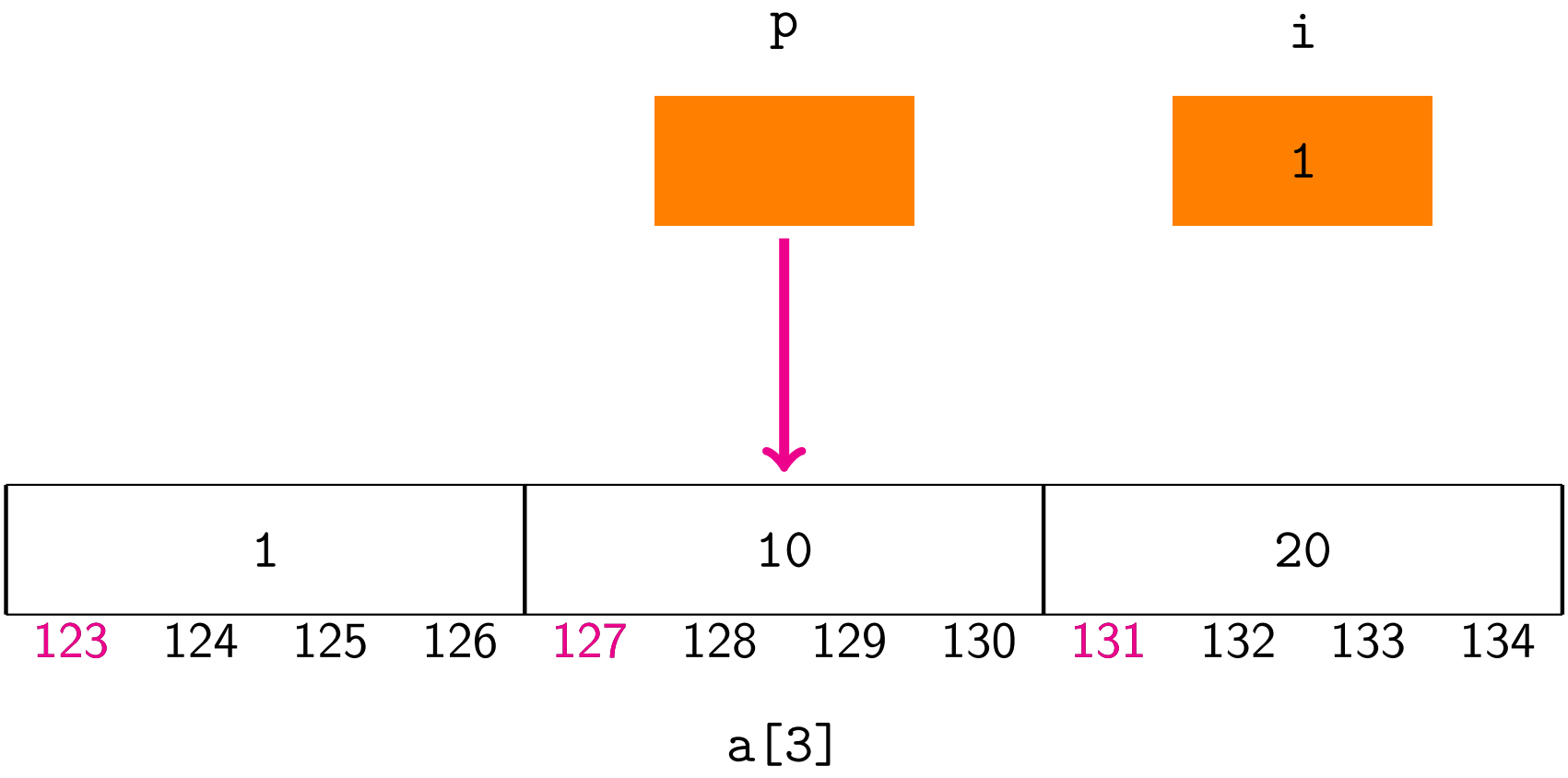

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



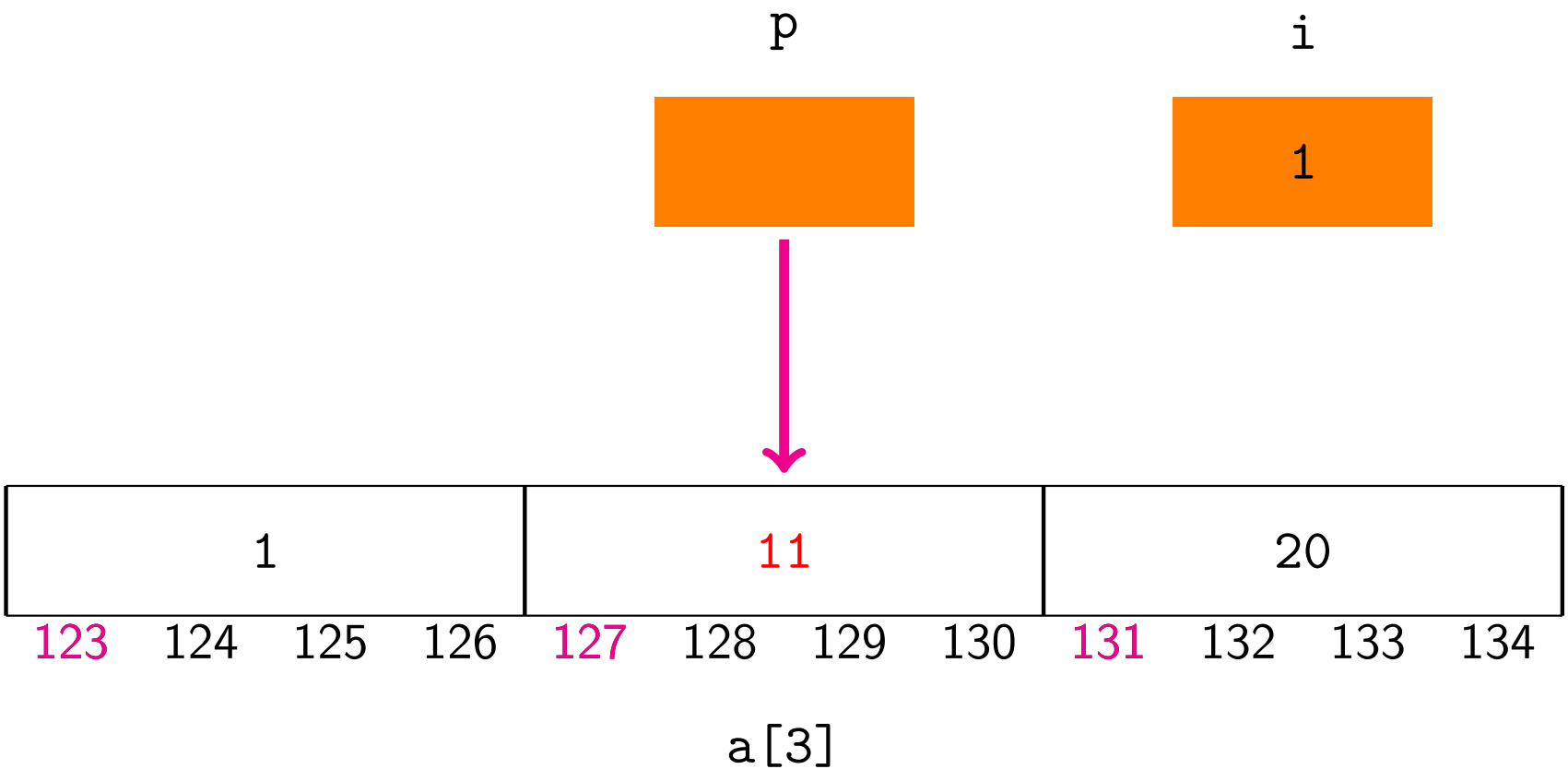

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



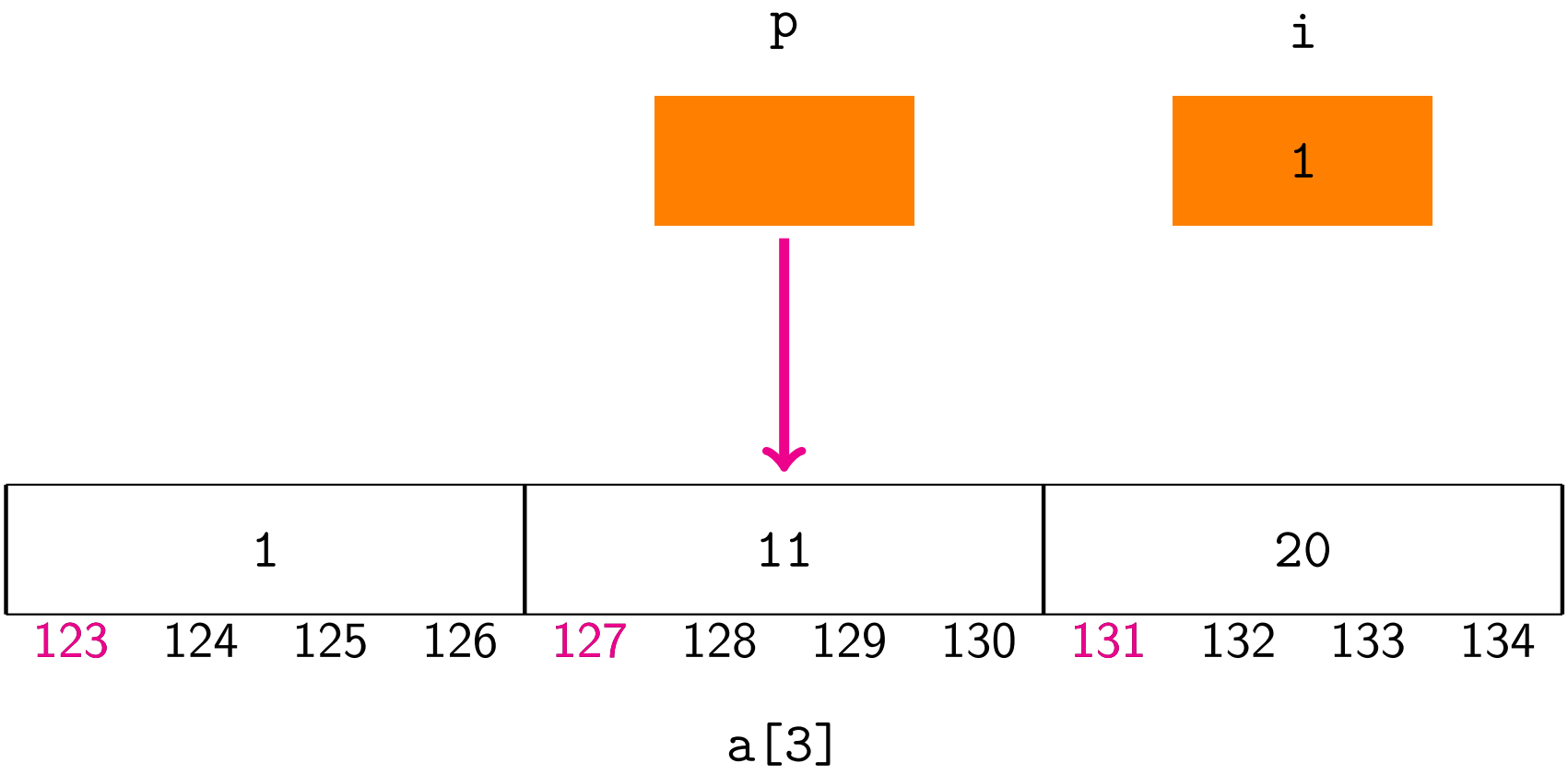

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



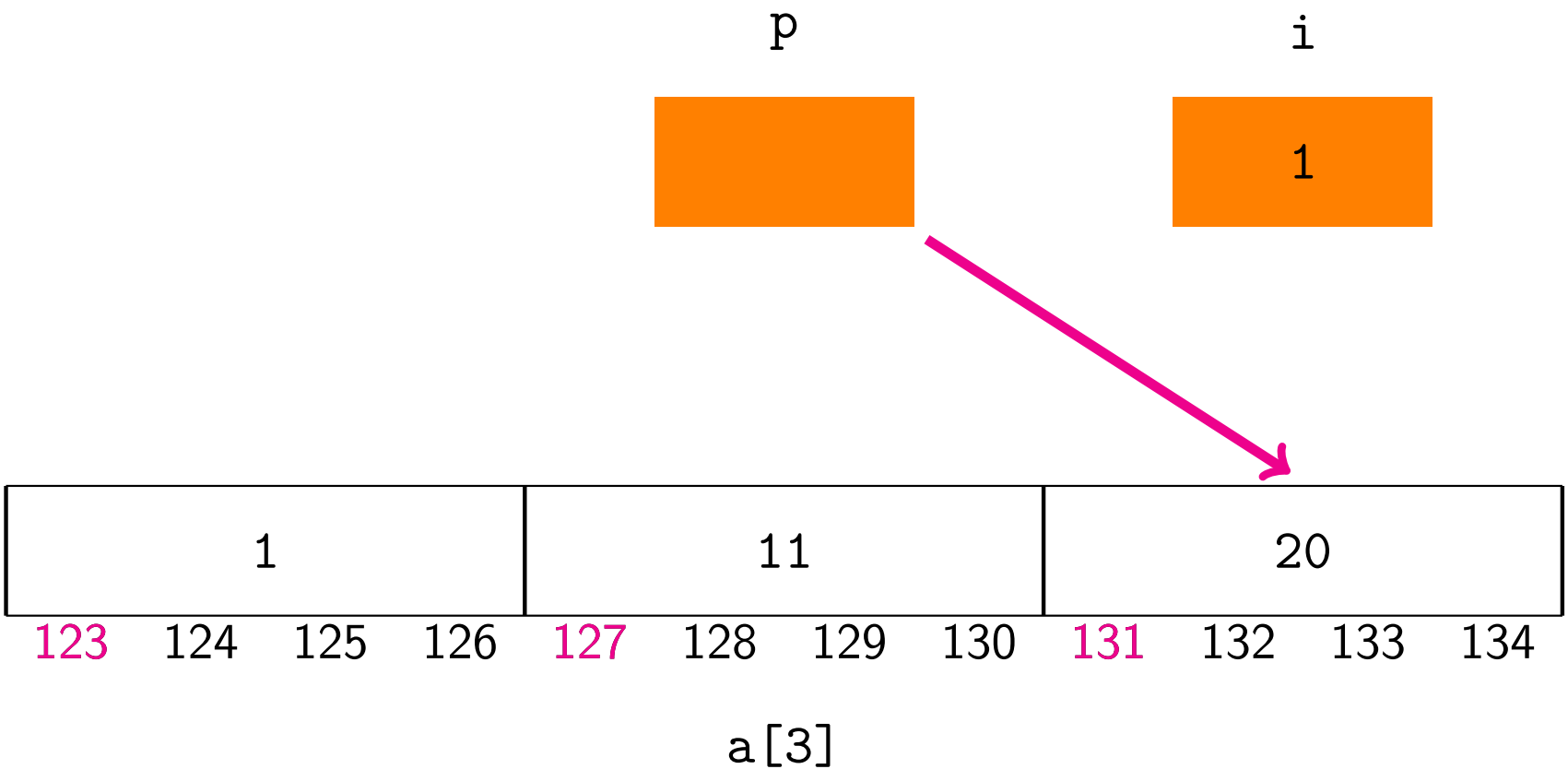

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



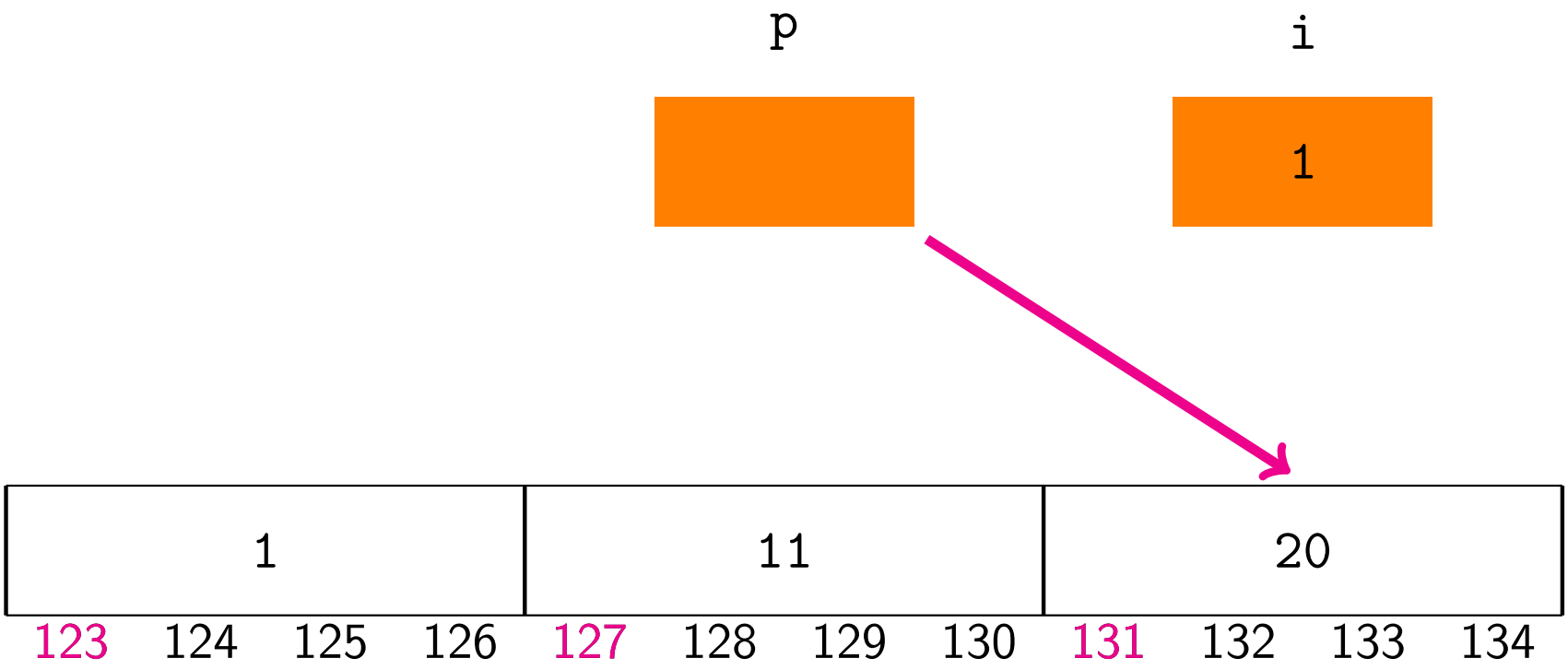

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



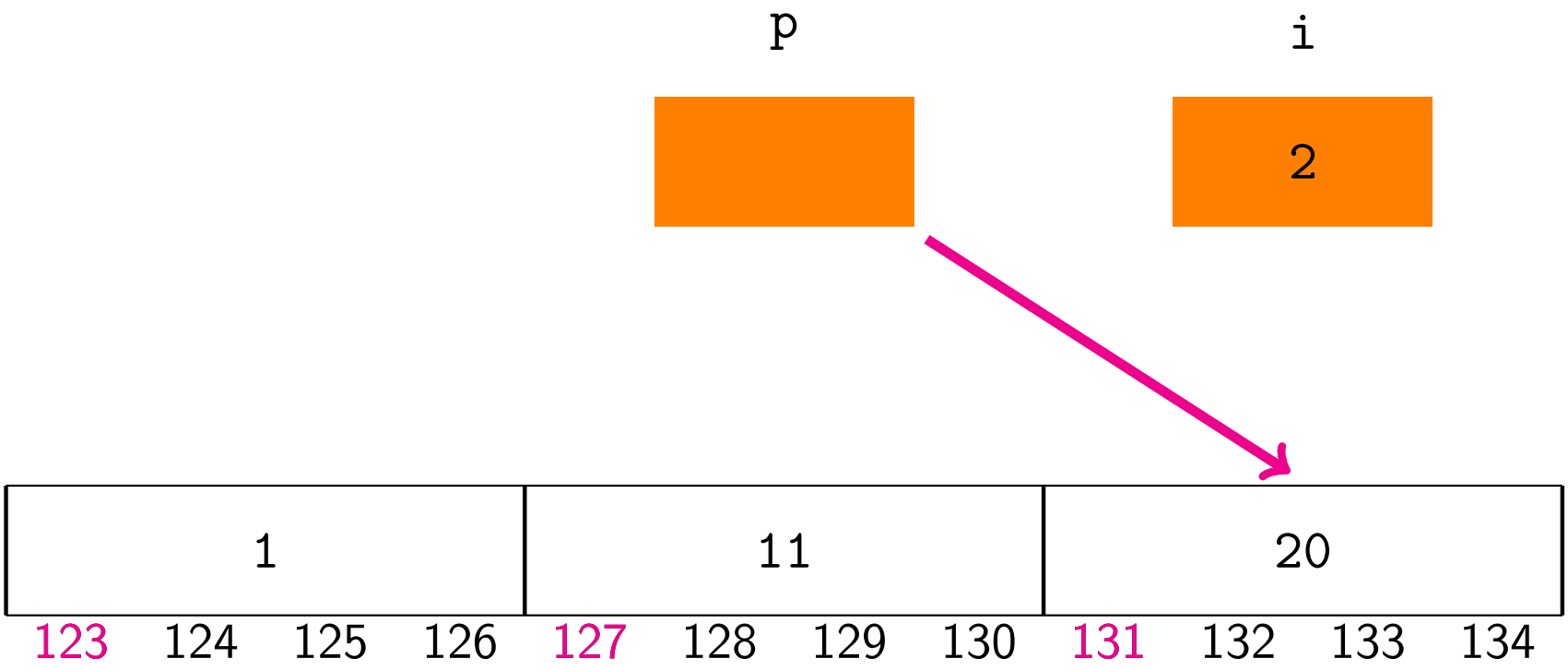

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



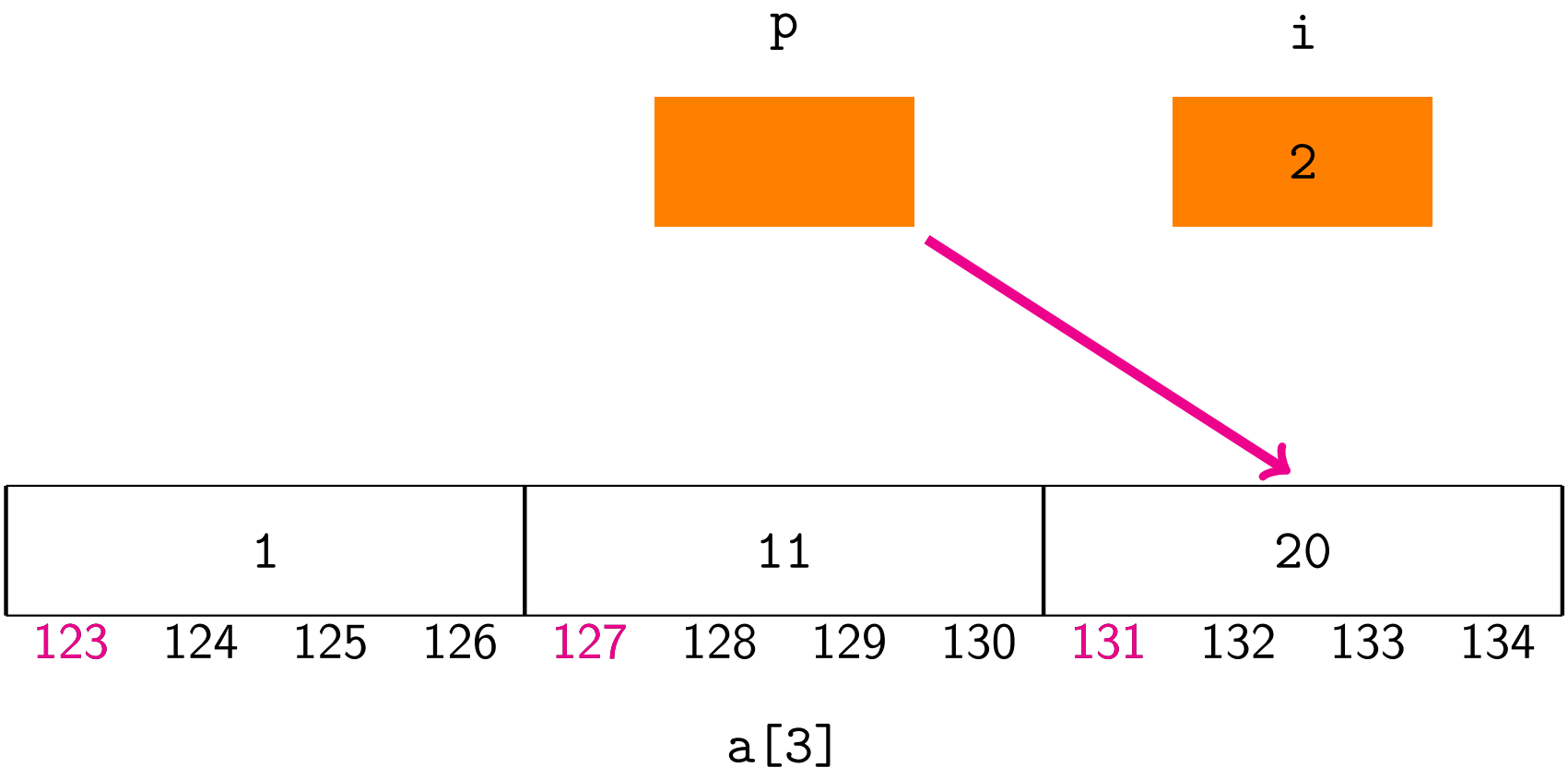

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



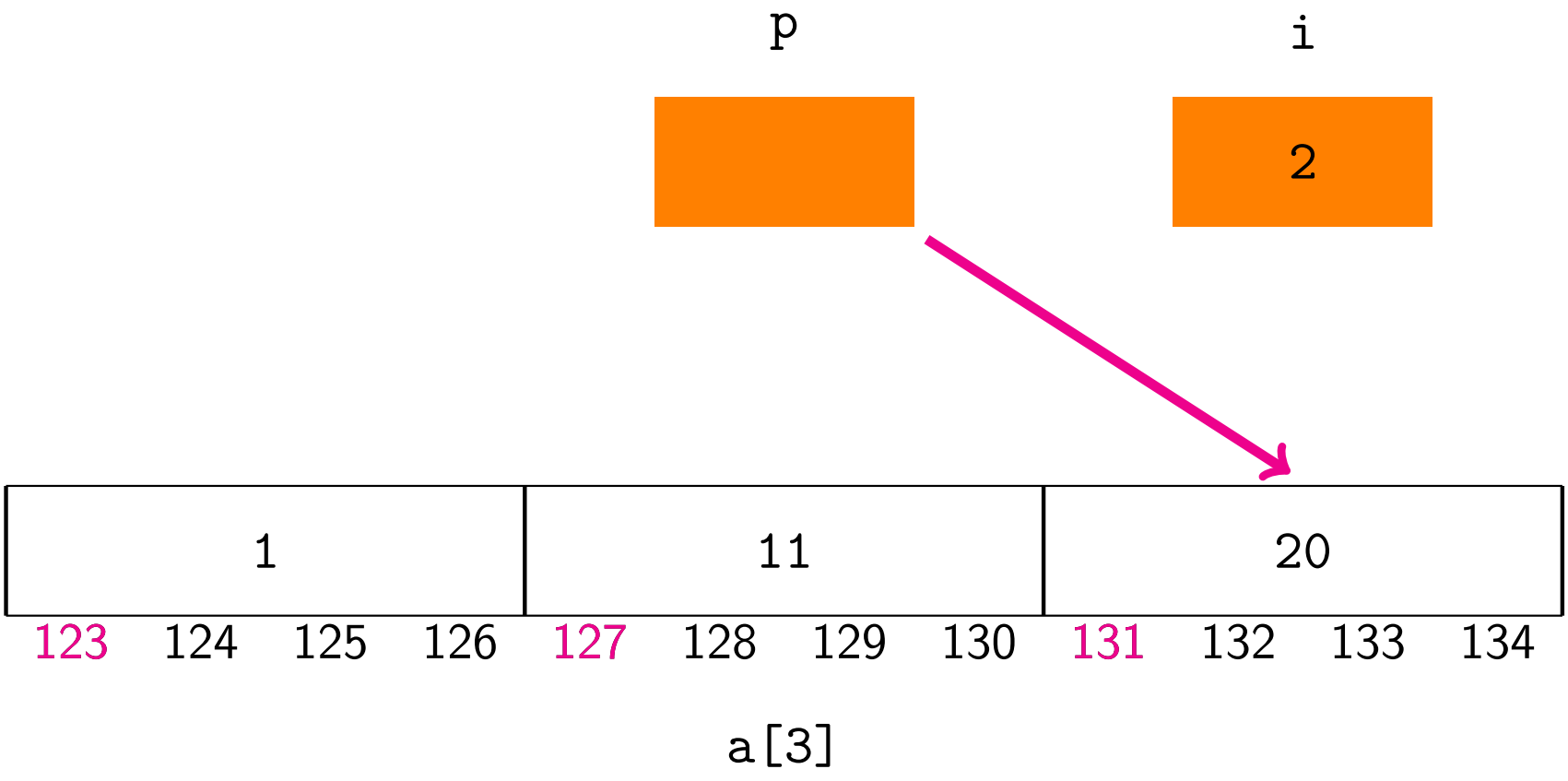

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



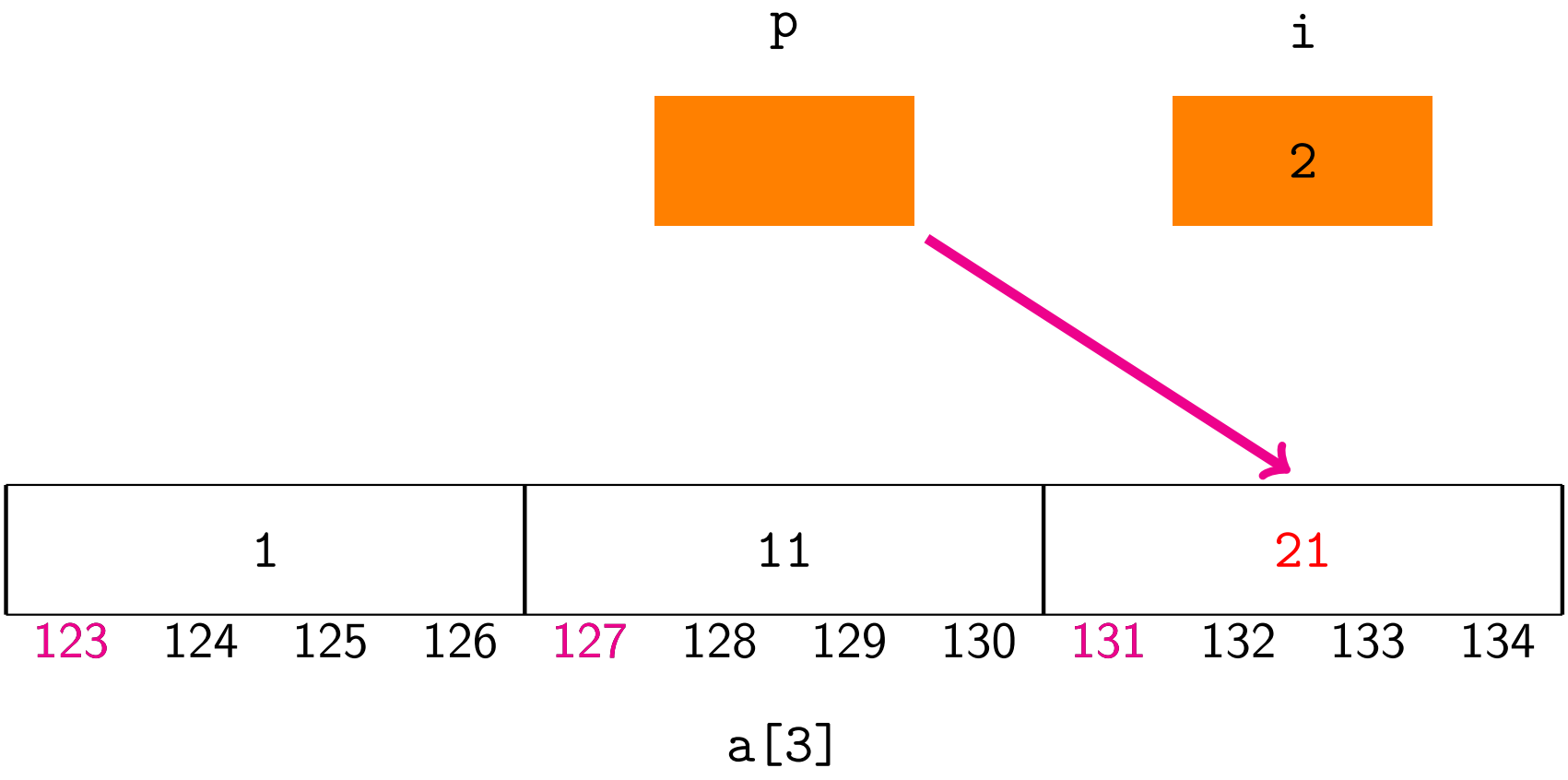

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



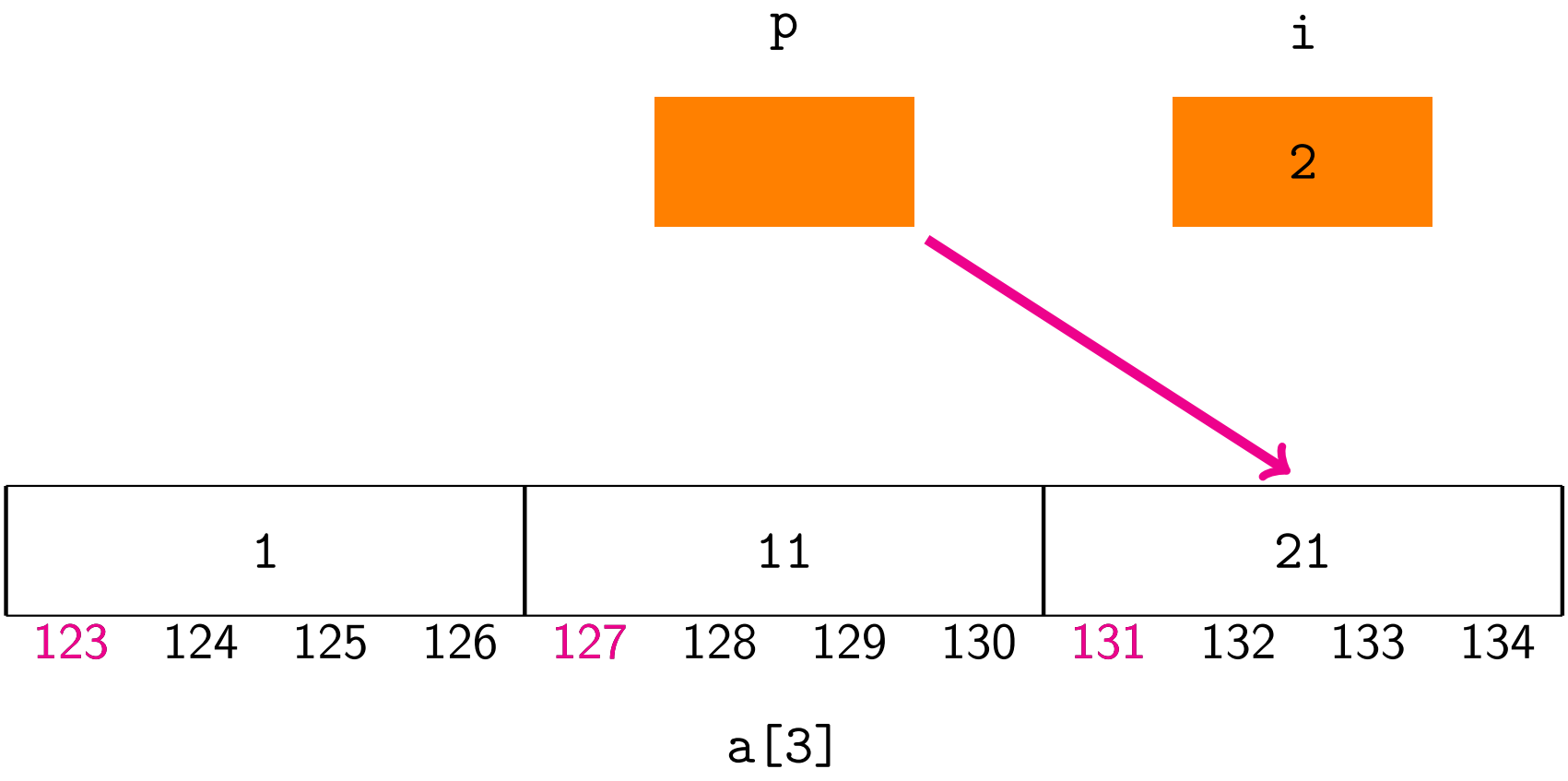

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



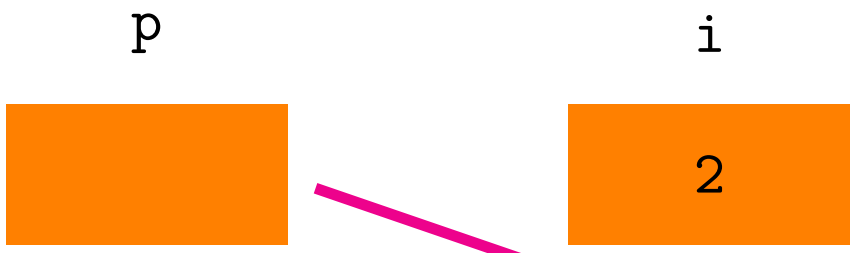

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

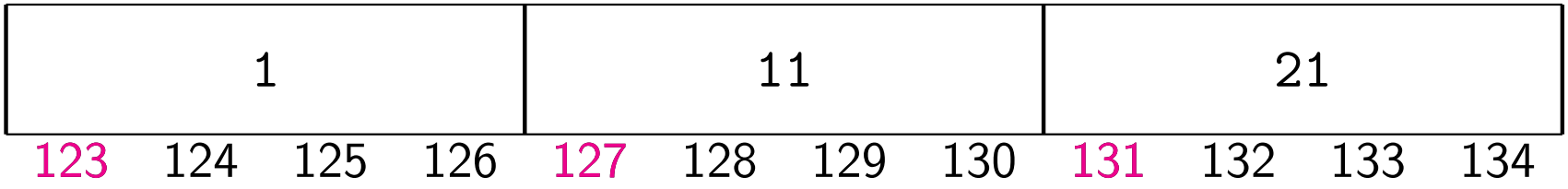



```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

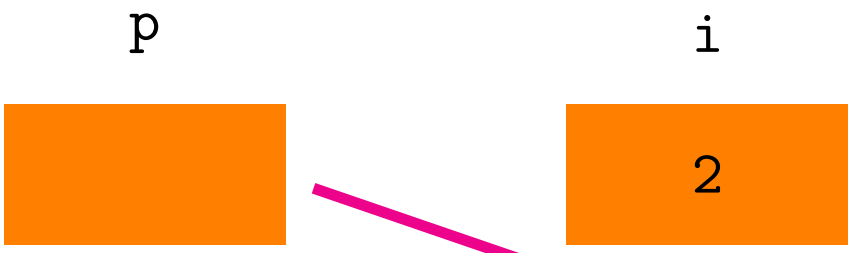


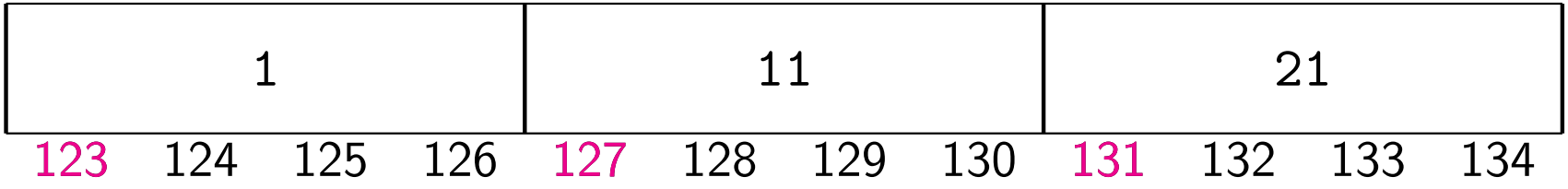

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```





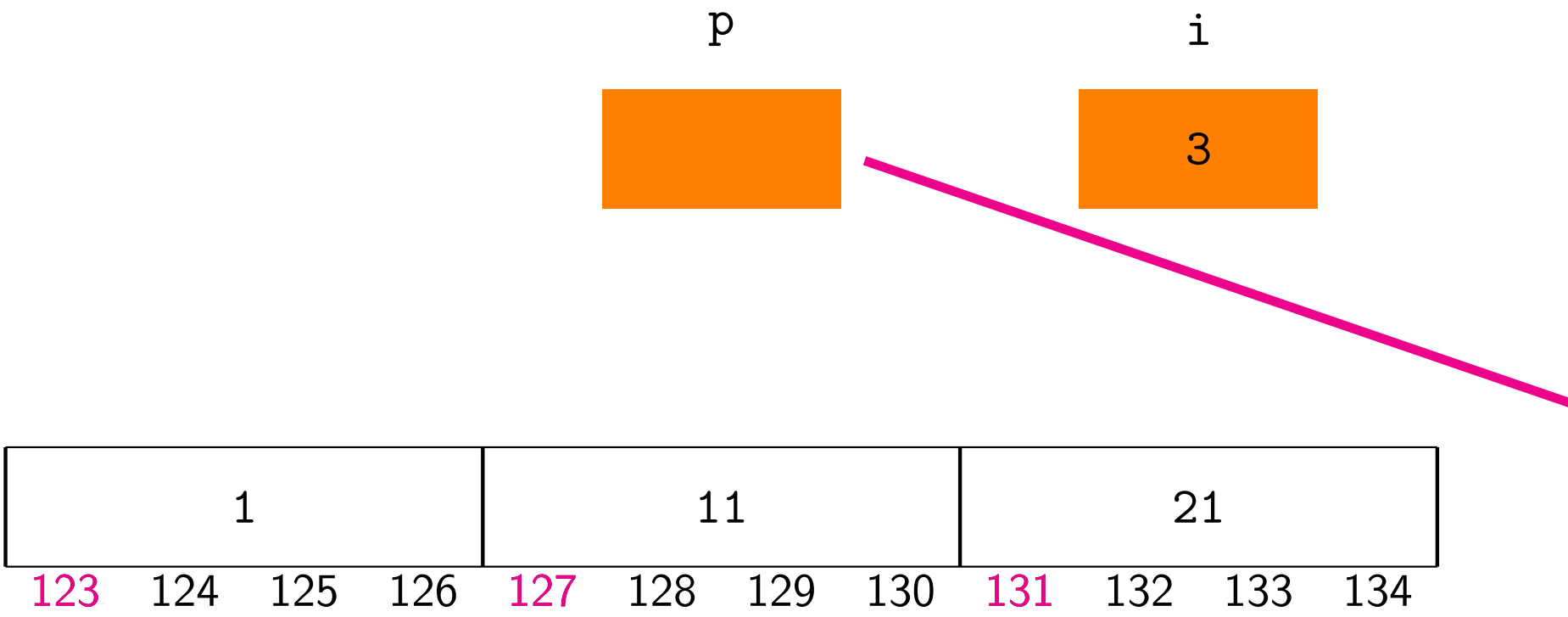

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```



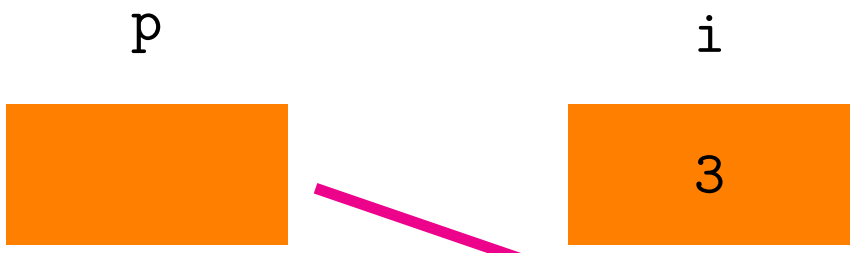


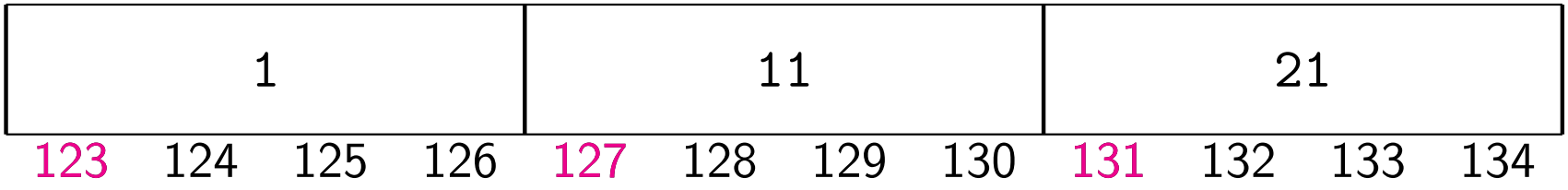
a[3]

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

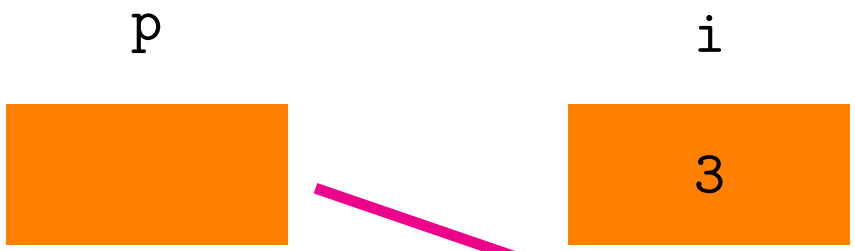


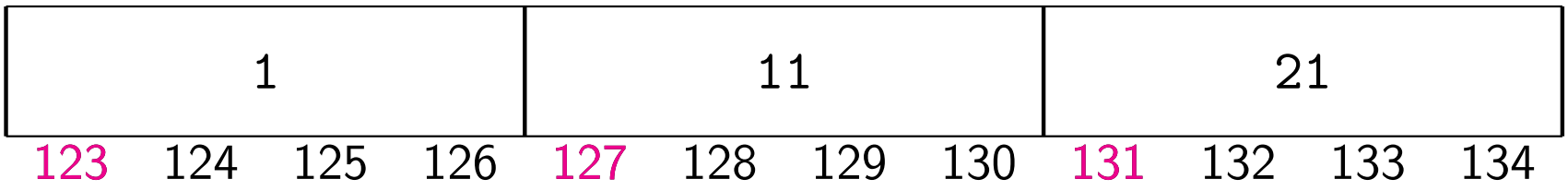

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```






```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```





a[3]

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

Done!

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

```
/* adds 1 to array elements */


  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
```

```
/* adds 1 to array elements */


  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
```

Comments

```
/* adds 1 to array elements */


  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
```

Comments

- Changed array a without using a.

```
/* adds 1 to array elements */


  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
```

Comments

- Changed array a without using a.
- What do we need?

```
/* adds 1 to array elements */


  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
```

Comments

- Changed array a without using a.
- What do we need?

```
/* adds 1 to array elements */
```

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.

```
/* adds 1 to array elements */
```

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.

```
/* adds 1 to array elements */
```

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

```
/* adds 1 to array elements */
```

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

Conclusion

```
/* adds 1 to array elements */
```

```
int *p;
p = &a[0];
for (i = 0; i < 3; i++) {
  *p = *p + 1;
  p = p + 1;
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

Conclusion

A function can do the same thing if it knows the data.

```
/* adds 1 to array elements */
     add_one(                  )
{
  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

Conclusion

A function can do the same thing if it knows the data.

```
/* adds 1 to array elements */
     add_one(int *p          )
{
  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

Conclusion

A function can do the same thing if it knows the data.

```
/* adds 1 to array elements */
     add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

Conclusion

A function can do the same thing if it knows the data.

```
/* adds 1 to array elements */
     add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

Conclusion

A function can do the same thing if it knows the data.

```
/* adds 1 to array elements */
     add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < 3; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

Conclusion

A function can do the same thing if it knows the data.

```
/* adds 1 to array elements */
     add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < len; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

Conclusion

A function can do the same thing if it knows the data.

```
/* adds 1 to array elements */
void add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < len; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
```

Comments

- Changed array a without using a.
- What do we need?
  - Address of `a[0]`.
  - Length of array a.

Conclusion

A function can do the same thing if it knows the data.

```
/* adds 1 to array elements */
void add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < len; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
```

```
/* adds 1 to array elements */
void add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < len; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
int main(void)
{









  return 0;
}
```

```
/* adds 1 to array elements */
void add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < len; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
int main(void)
{
  int a[3];



  return 0;
}
```

```
/* adds 1 to array elements */
void add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < len; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
int main(void)
{
  int a[3];
  a[0] = 1, a[1] = 2, a[2] = 3;


  return 0;
}
```

```
/* adds 1 to array elements */
void add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < len; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
int main(void)
{
  int a[3];
  a[0] = 1, a[1] = 2, a[2] = 3;
  add_one(&a[0], 3);

  return 0;
}
```

```
/* adds 1 to array elements */
void add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < len; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
int main(void)
{
  int a[3];
  a[0] = 1, a[1] = 2, a[2] = 3;
  add_one(&a[0], 3);
  printf("%d %d %d", a[0], a[1], a[2]);
  return 0;
}
```

```
/* adds 1 to array elements */
void add_one(int *p, int len)
{
  int *p;
  p = &a[0];
  for (i = 0; i < len; i++) {
    *p = *p + 1;
    p = p + 1;
  }
}
int main(void)
{
  int a[3];
  a[0] = 1, a[1] = 2, a[2] = 3;
  add_one(&a[0], 3);
  printf("%d %d %d", a[0], a[1], a[2]); → 2 3 4
  return 0;
}
```

Array name

Array name

```
int a[3];
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

```
printf("%d", *a);
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

```
printf("%d", *a); → 1
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

```
printf("%d", *a); → 1
                a + 1
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

```
printf("%d", *a); → 1
                *(a + 1)
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

```
printf("%d", *a); → 1
printf("%d", *(a + 1));
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

```
printf("%d", *a); → 1
printf("%d", *(a + 1)); → 2
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

Fact 2

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

Fact 2

The name of the array is not a variable.

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

Fact 2

The name of the array is not a variable.

```
int b;
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

Fact 2

The name of the array is not a variable.

```
int b;
a = &b;
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

Fact 2

The name of the array is not a variable.

```
int b;
a = &b;
```

Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

Fact 2

The name of the array is not a variable.

`a` is this sense is constant.

Array index

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

```
printf("%d", a[2]);
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

`printf("%d", a[2]);` or `printf("%d", *(a + 2);`

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

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

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
printf("%d", p[2]);
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
printf("%d", p[2]);
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
printf("%d", p[2]); → 3
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

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

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
for (i = 0; i < 3; i++) {

}
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
for (i = 0; i < 3; i++) {
  printf("%d", a[i]);
}
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
for (i = 0; i < 3; i++) {
  printf("%d", p[i]);
}
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
for (i = 0; i < 3; i++) {
  printf("%d", p[i]); → 1 2 3
}
```