Pointers on Arrays

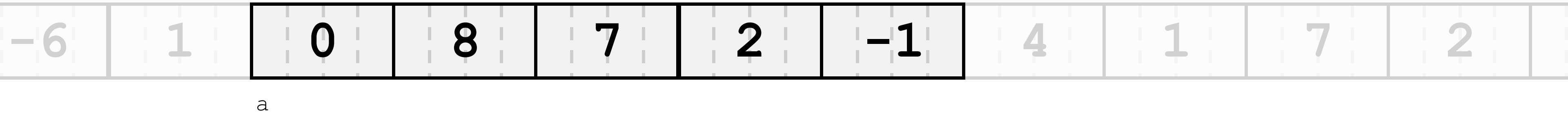
Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                         // array-to-pointer conv
++ptr;                                // shift to the right
int my_int = *ptr;                    // read target
ptr += 2;                             // shift by 2 elements
*ptr = 18;                            // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```

-6	1	3	-8	1	5	-3	4	1	7	2

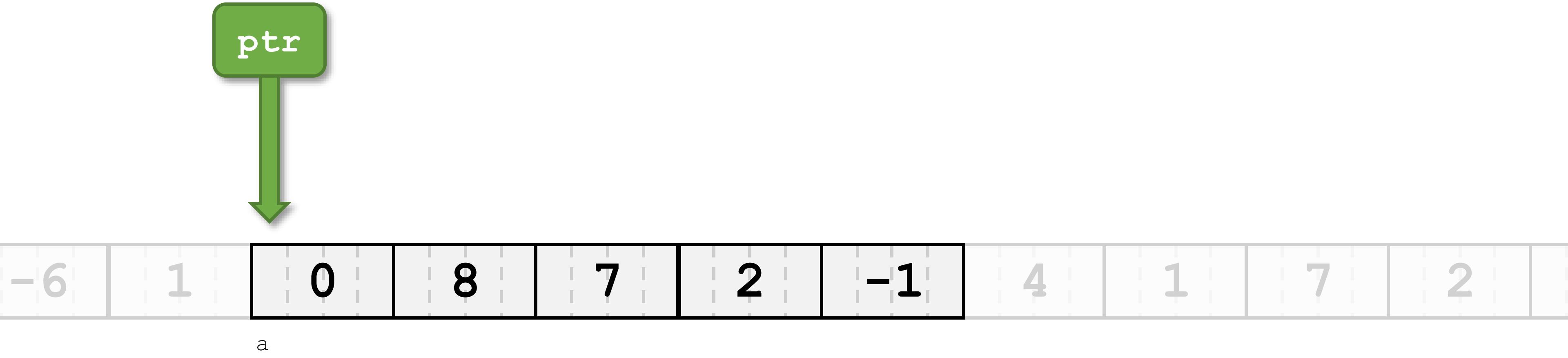
Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                      // array-to-pointer conv
++ptr;                             // shift to the right
int my_int = *ptr;                 // read target
ptr += 2;                          // shift by 2 elements
*ptr = 18;                         // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```



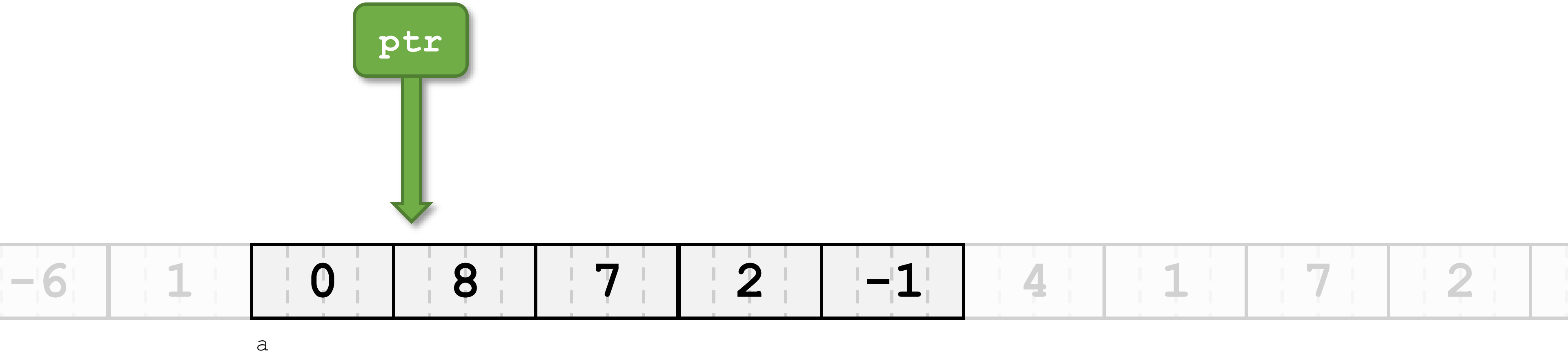
Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                       // array-to-pointer conv
++ptr;                              // shift to the right
int my_int = *ptr;                  // read target
ptr += 2;                           // shift by 2 elements
*ptr = 18;                          // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```



Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                      // array-to-pointer conv
++ptr;                             // shift to the right
int my_int = *ptr;                 // read target
ptr += 2;                          // shift by 2 elements
*ptr = 18;                         // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```



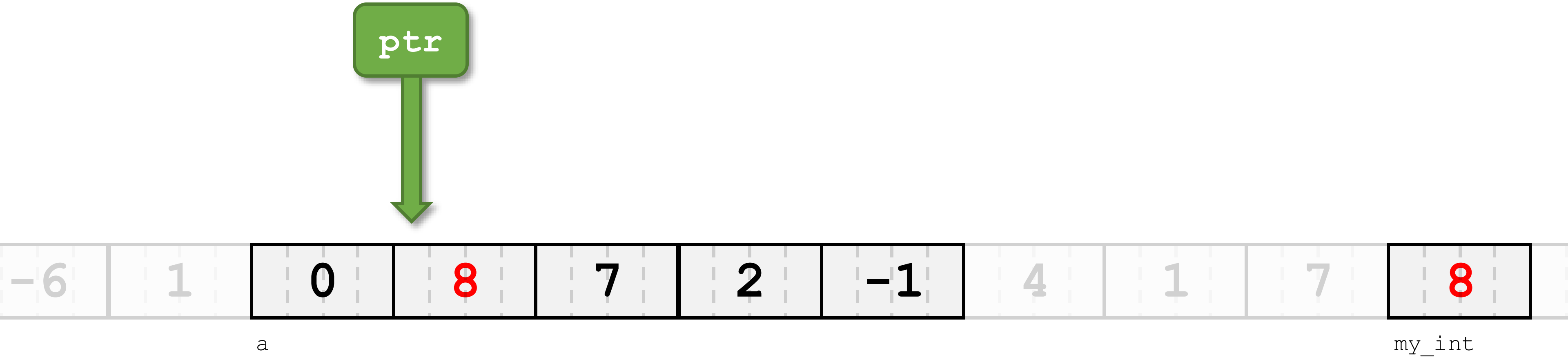
ptr

-6 1 0 8 7 2 -1 4 1 7 2

a

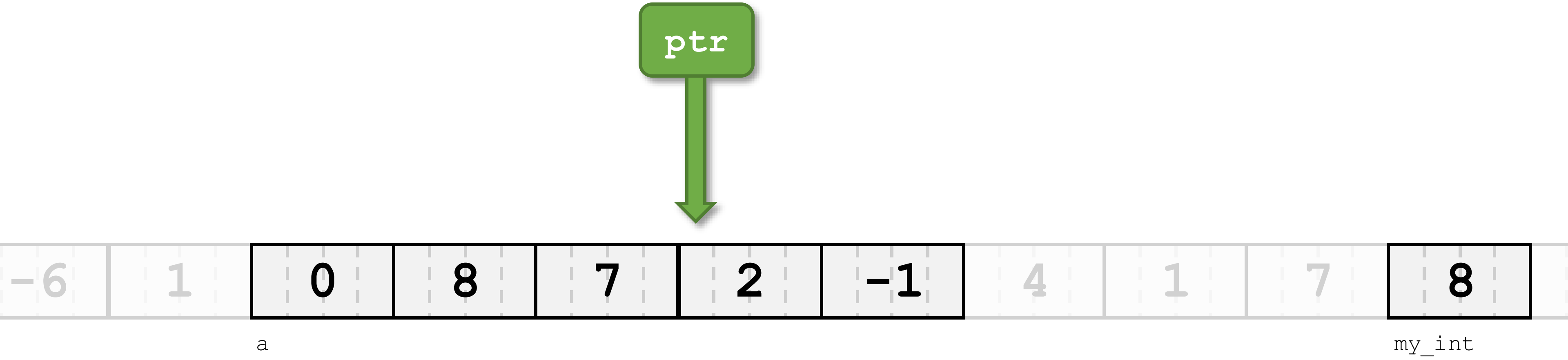
Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                       // array-to-pointer conv
++ptr;                              // shift to the right
int my_int = *ptr;                  // read target
ptr += 2;                           // shift by 2 elements
*ptr = 18;                          // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```



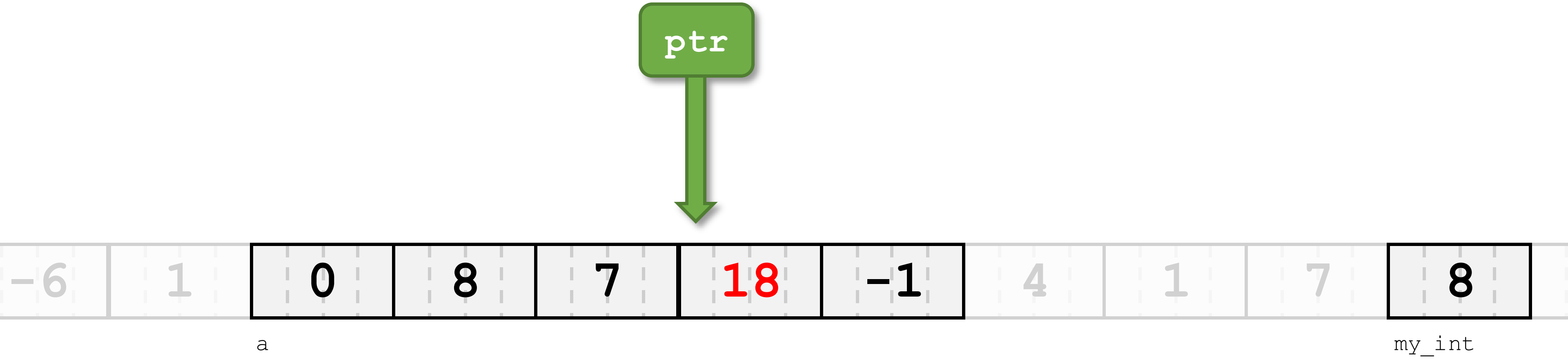
Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                      // array-to-pointer conv
++ptr;                             // shift to the right
int my_int = *ptr;                 // read target
ptr += 2;                          // shift by 2 elements
*ptr = 18;                         // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```



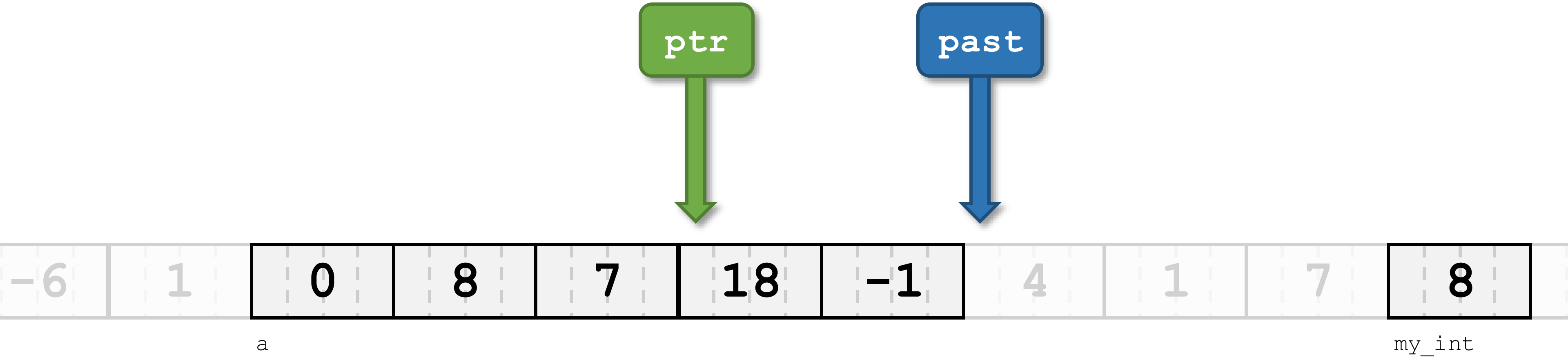
Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                       // array-to-pointer conv
++ptr;                              // shift to the right
int my_int = *ptr;                  // read target
ptr += 2;                           // shift by 2 elements
*ptr = 18;                          // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```



Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                          // array-to-pointer conv
++ptr;                                 // shift to the right
int my_int = *ptr;                     // read target
ptr += 2;                              // shift by 2 elements
*ptr = 18;                             // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```

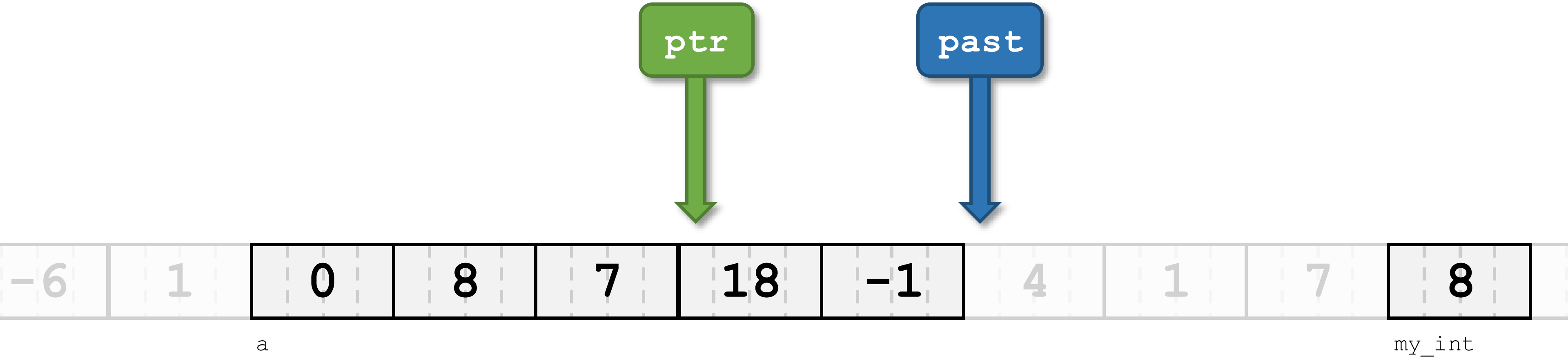


Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                                   // arr
++ptr;                                          // shi
int my_int = *ptr;                              // read
ptr += 2;                                       // shift
*ptr = 18;                                      // overw
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```

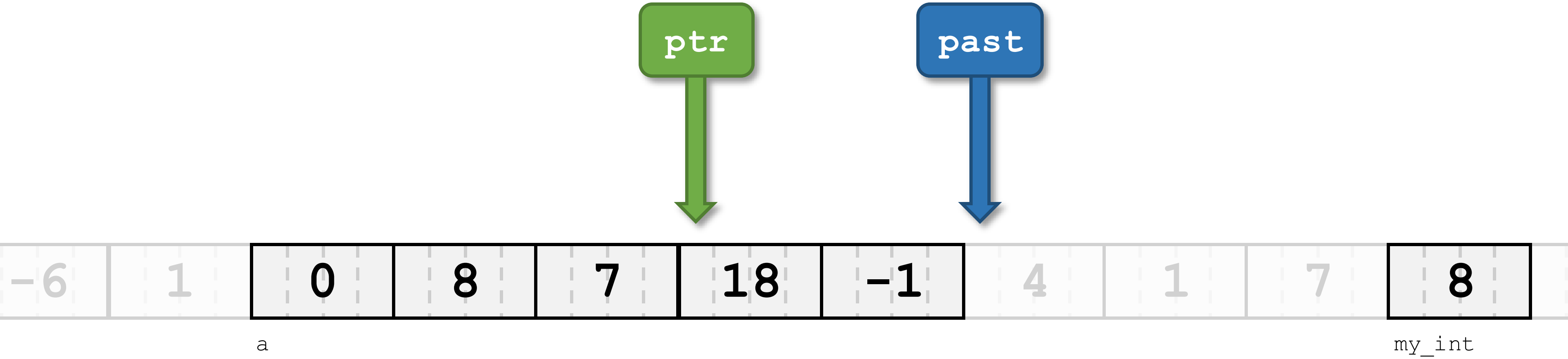
Output: true

Because ptr is "to the left" of past.



Pointer Program

```
int a[5] = {0, 8, 7, 2, -1};
int* ptr = a;                          // array-to-pointer conv
++ptr;                                 // shift to the right
int my_int = *ptr;                     // read target
ptr += 2;                              // shift by 2 elements
*ptr = 18;                             // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n"; // compare pointers
```



Exercise!

Pointer Program

Find and fix at least 3 problems in the following program.

```
#include <iostream>
int main () {
    int a[7] = {0, 6, 5, 3, 2, 4, 1}; // static array
    int b[7];
    int* c = b;

    // copy a into b using pointers
    for (int* p = a; p <= a+7; ++p)
        *c++ = *p;

    // cross-check with random access
    for (int i = 0; i <= 7; ++i)
        if (a[i] != c[i])
            std::cout << "Oops, copy error...\n";

    return 0;
}
```

(From: Script Exercise 117)

Pointer Program

```
#include <iostream>
int main () {
    int a[7] = {0, 6, 5, 3, 2, 4, 1}; // static array
    int b[7];
    int* c = b;

    // copy a into b using pointers
    for (int* p = a; p <= a+7; ++p)
        *c++ = *p;

    // cross-check with random access
    for (int i = 0; i <= 7; ++i)
        if (a[i] != c[i])
            std::cout << "Oops, copy error...\n";

    return 0;
}
```

`p = a+7` is dereferenced

Solution:
Use `<` instead of `<=`

(From: Script Exercise 117)

Pointer Program

```
#include <iostream>
int main () {
    int a[7] = {0, 6, 5, 3, 2, 4, 1}; // static array
    int b[7];
    int* c = b;

    // copy a into b using pointers
    for (int* p = a; p <= a+7; ++p)
        *c++ = *p;

    // cross-check with random access
    for (int i = 0; i <= 7; ++i)
        if (a[i] != c[i])
            std::cout << "Oops, copy er

    return 0;
}
```

`p = a+7` is dereferenced

Solution:
Use `<` instead of `<=`

Same problem as above

(From: Script Exercise 117)

Pointer Program

```
#include <iostream>
int main () {
    int a[7] = {0, 6, 5, 3, 2, 4, 1}; // static array
    int b[7];
    int* c = b;

    // copy a into b using pointers
    for (int* p = a; p <= a+7; ++p)
        *c++ = *p;

    // cross-check with random access
    for (int i = 0; i <= 7; ++i)
        if (a[i] != c[i])
            std::cout << "Oops, copy er

    return 0;
}
```

`c` doesn't point to `b[0]` anymore.

Solution:
Use `b` instead of `c`

`p = a+7` is dereferenced

Solution:
Use `<` instead of `<=`

Same problem as above

(From: Script Exercise 117)