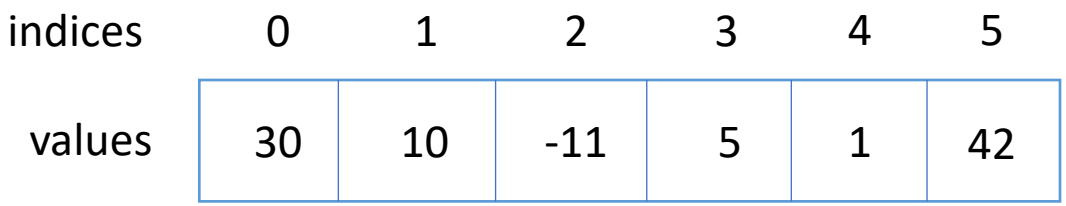
Array Bounds

Indexing into Collections

- Arrays (C/C++) or lists (Python) can give us direct access to data stored at a given index (position)



- The first index is 0
- The size of the array above is 6, but the last index is 5
- The largest possible index is always size - 1

An Example in C

- Declaring and initializing an array in C:

```
int array[4] = { 3, -10, 14, 4};
```

An Example in C

- Declaring and initializing an array in C:

```
int array[4] = { 3, -10, 14, 4};
```

- What is the index of value 3?

An Example in C

- Declaring and initializing an array in C:

```
int array[4] = { 3, -10, 14, 4};
```

- What is the index of value 3? 0
- What is the index of value 14?

An Example in C

- Declaring and initializing an array in C:

```
int array[4] = { 3, -10, 14, 4};
```

- What is the index of value 3? 0
- What is the index of value 14? 2
- What value is at index 4?

An Example in C

- Declaring and initializing an array in C:

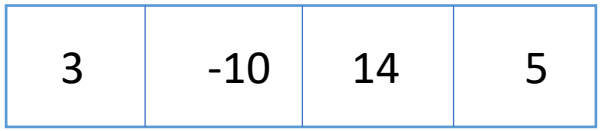
```
int array[4] = { 3, -10, 14, 4};
```

- What is the index of value 3? 0
- What is the index of value 14? 2
- What value is at index 4? ???

How Arrays are Stored

- Arrays are stored in **contiguous** block of memory
  - Values in the same array are always right next to each other
- The index is calculated based on where the first element is stored
  - The first is 0 away from the first, 1 away from the second, etc.

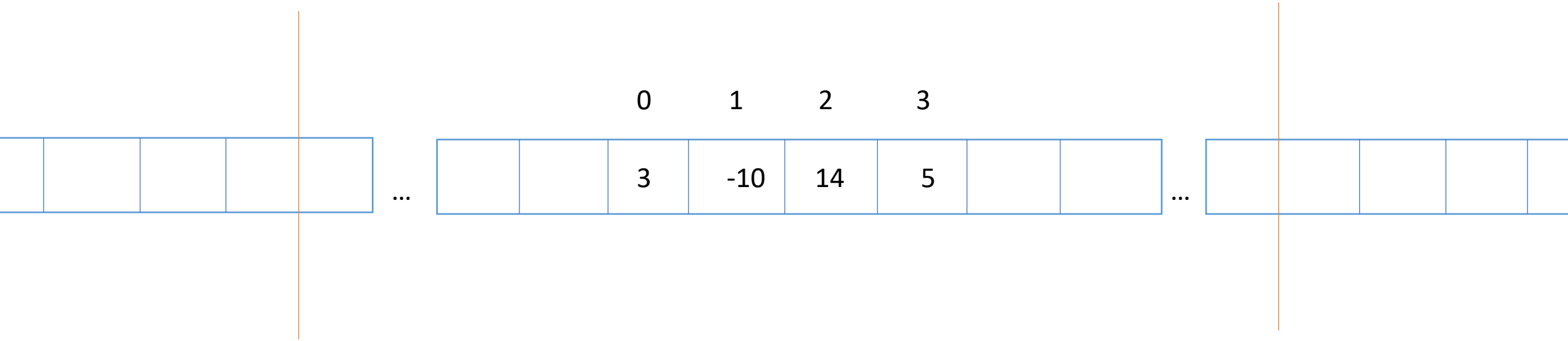
```
int array[4] = { 3, -10, 14, 4};
```



- What happens if we ask for a position not in the array?

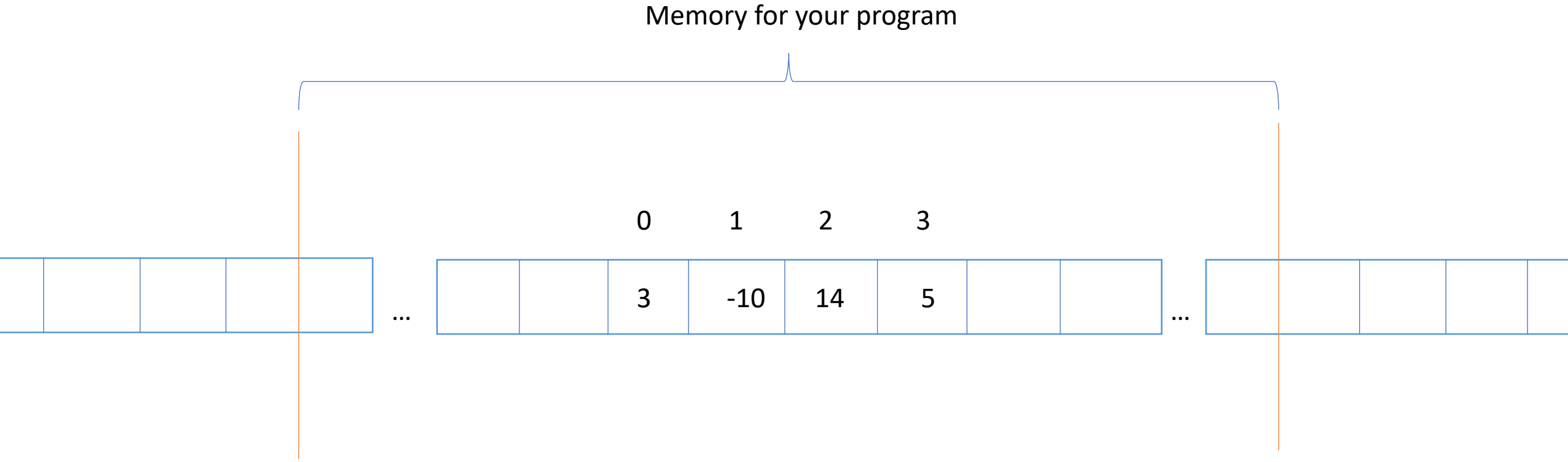
An Array in Memory

```
int array[4] = { 3, -10, 14, 4};
```



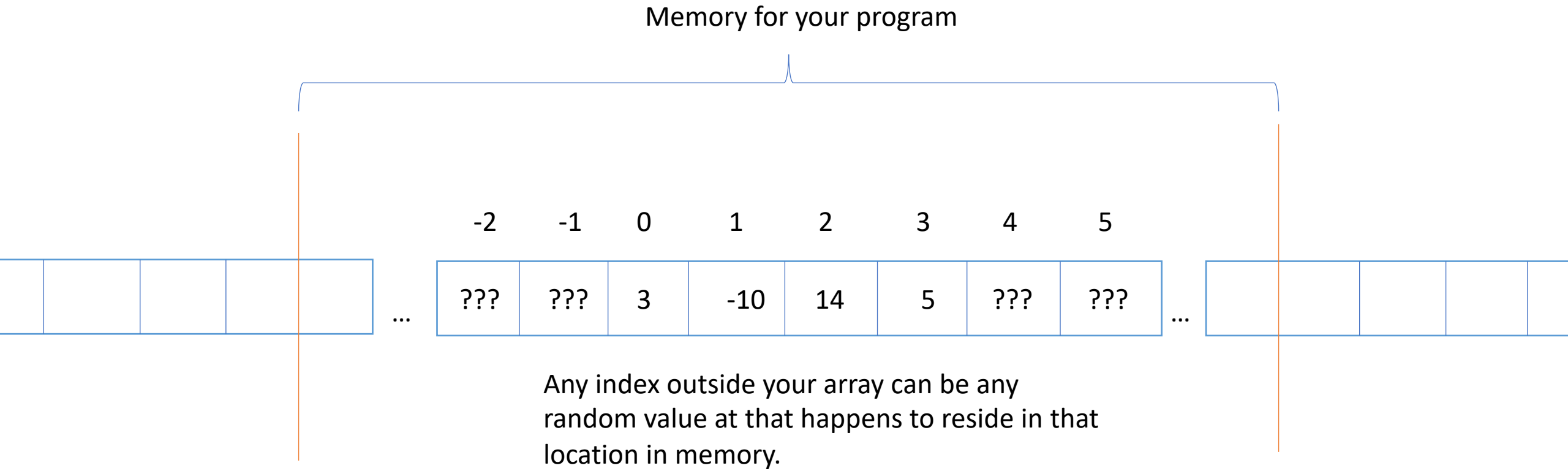
An Array in Memory

```
int array[4] = { 3, -10, 14, 4};
```



Out of Bounds

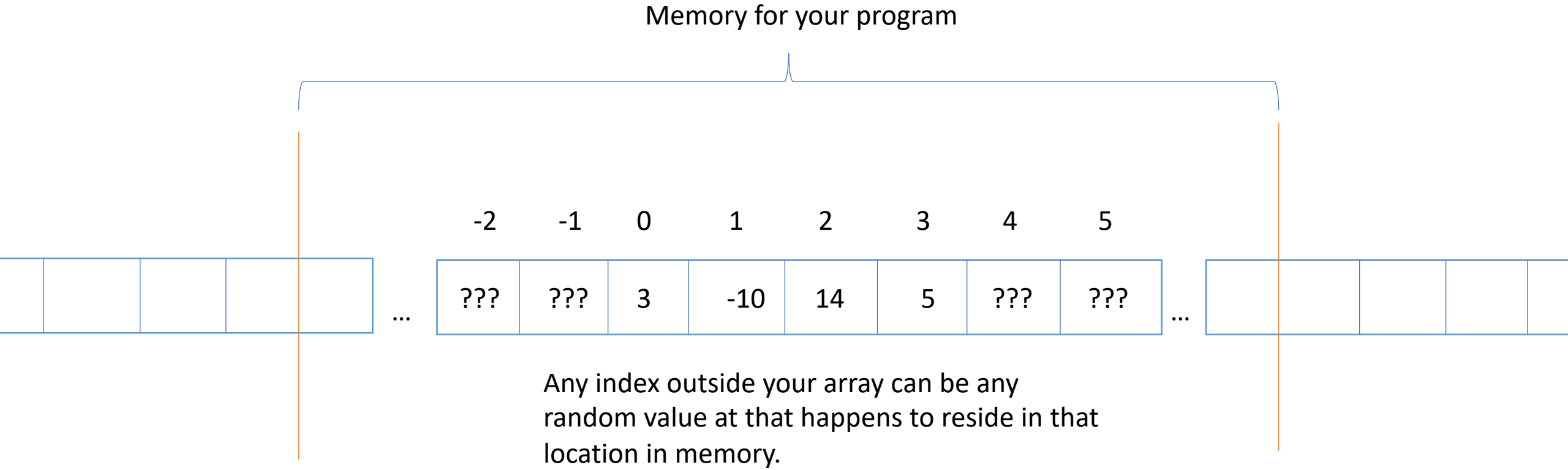
```
int array[4] = { 3, -10, 14, 4};
```



Out of Bounds

```
int array[4] = { 3, -10, 14, 4};
```

C does not do **ANY** checking to make sure an index is valid

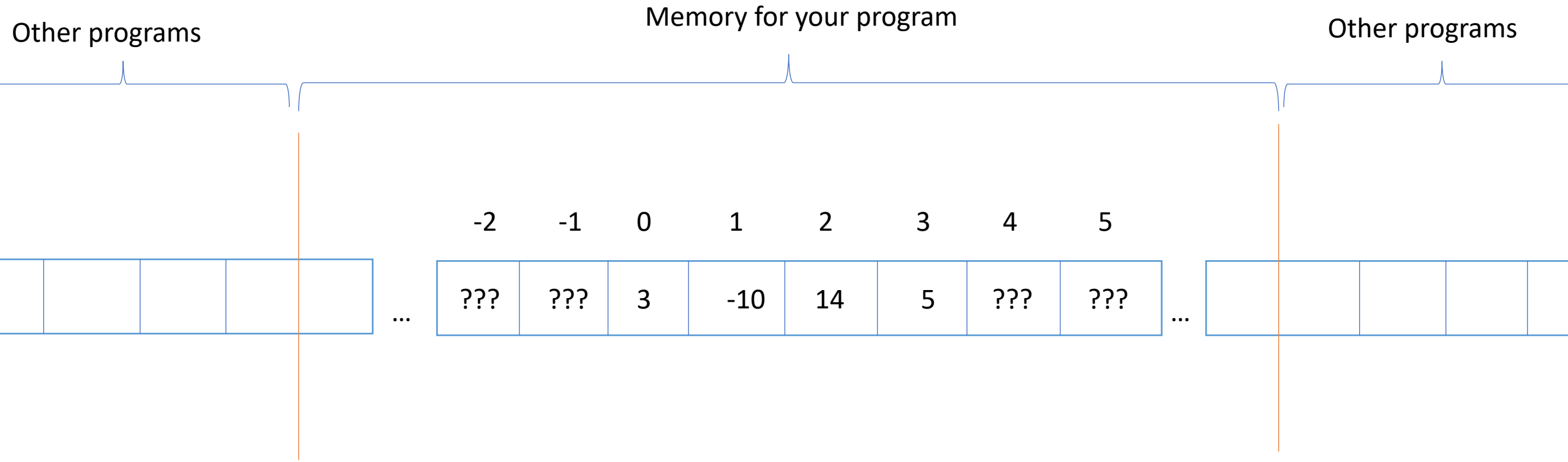


Any index outside your array can be any random value at that happens to reside in that location in memory.

Out of Bounds

```
int array[4] = { 3, -10, 14, 4};
```

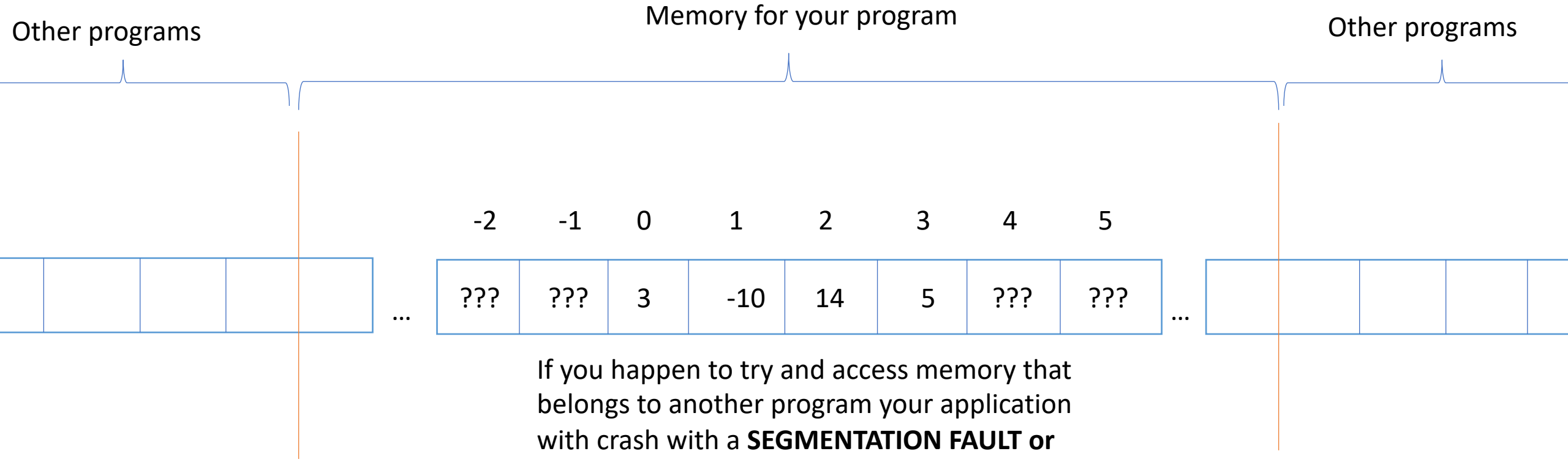
C does not do **ANY** checking to make sure an index is valid



Out of Bounds

```
int array[4] = { 3, -10, 14, 4};
```

C does not do **ANY** checking to make sure an index is valid



If you happen to try and access memory that belongs to another program your application with crash with a **SEGMENTATION FAULT or SEGFAULT**!