Array

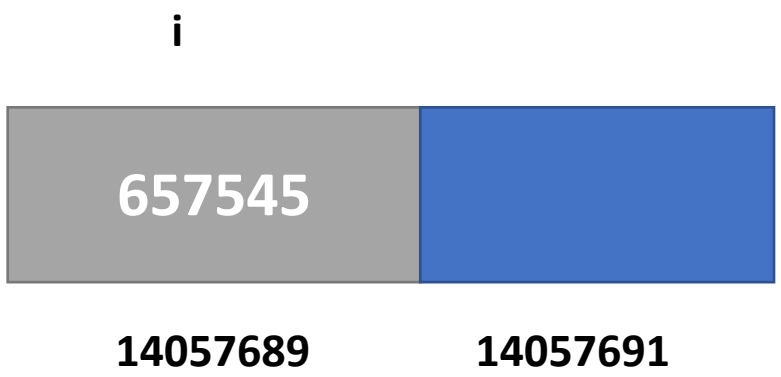
Review

- What is meant by declaration and initialization?
- int i; //declaration

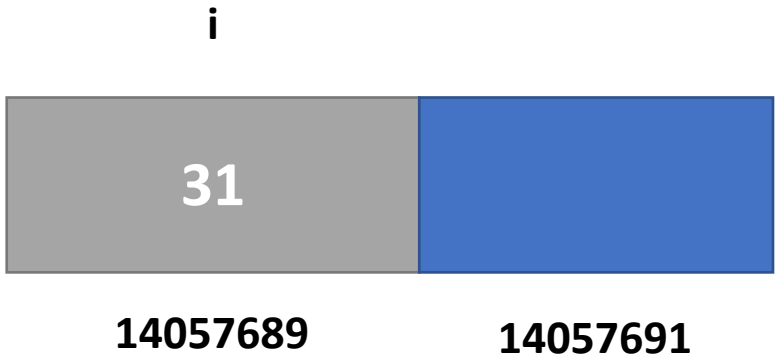
- int i =31; //declaration and initialization

Declaration and Initialization

int i;



int i = 31;



Review

```
int count = 0;
for (int x = 0; x < 6; x++){
        for (int y = 1; y <= 5; y++) {
                count += 1;
        }
}

printf("%i\n", count);
```

- Answer: 30

Review

```
int n = -90;

if(n){
    printf("The value of n is considered True\n");
}
else{
    printf("The value of n is considered False\n");
}
```

- The value of n is considered True.

Arrays

- C provides a functionality that lets users design a set of similar data types
- An array is a collection of similar element
- Any given array must be of the same type

Array Declaration

- An array needs to be declared like any other variable
- Compiler needs to know the type of the array and the size of the array
  - int marks[30];
  - int specifies the type of the variable
  - [30] tells how many elements of type int will be in the array

Accessing Array Elements

- Done with subscript
  - the number in the brackets following the array name
- The number specifies the element’s position in the array
- All array elements are numbered
  - starting with 0

Accessing Array Elements

- 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?

Accessing Array Elements

```c
#include <stdio.h>

int main() {
    int even_integers[5];

    printf("%i\n", even_integers[0]);
    printf("%i\n", even_integers[1]);
    printf("%i\n", even_integers[2]);
    printf("%i\n", even_integers[3]);
    printf("%i\n", even_integers[4]);

    return 0;
}
```

```
0
0
0
0
-427866360
```

Array Initialization

```c
#include <stdio.h>

int main() {
    int even_integers[5] = {10, 20, 30, 40, 50};

    printf("%i\n", even_integers[0]);
    printf("%i\n", even_integers[1]);
    printf("%i\n", even_integers[2]);
    printf("%i\n", even_integers[3]);
    printf("%i\n", even_integers[4]);

    return 0;
}
```

```
10
20
30
40
50
```

Array Initialization

```c
#include <stdio.h>

int main() {
    int even_integers[5] = {10};

    printf("%i\n", even_integers[0]);
    printf("%i\n", even_integers[1]);
    printf("%i\n", even_integers[2]);
    printf("%i\n", even_integers[3]);
    printf("%i\n", even_integers[4]);

    return 0;
}
```

```
10
0
0
0
0
```

Array Initialization

```c
#include <stdio.h>

int main() {
    int even_integers[5] = {0};

    printf("%i\n", even_integers[0]);
    printf("%i\n", even_integers[1]);
    printf("%i\n", even_integers[2]);
    printf("%i\n", even_integers[3]);
    printf("%i\n", even_integers[4]);

    return 0;
}
```

```
0
0
0
0
0
```

Initializing Arrays with Variable Size

```c
#include <stdio.h>

int main() {
    int size = 5;

    int even_integers[size] = {0};

    printf("%i\n", even_integers[0]);
    printf("%i\n", even_integers[1]);
    printf("%i\n", even_integers[2]);
    printf("%i\n", even_integers[3]);
    printf("%i\n", even_integers[4]);

    return 0;
}
```

```
int even_integers[size]
```

variable "even_integers" may not be initialized C/C++(145)

Initializing Arrays with Variable Size

```c
#include <stdio.h>

int main() {
    int size = 5;

    int even_integers[size];

    for(int i = 0; i < size; i++){
        even_integers[i] = 0;
    }

    return 0;
}
```

Printing Array Elements

```c
#include <stdio.h>

int main() {
    int size = 5;

    int even_integers[size];

    for(int i = 0; i < size; i++){
        even_integers[i] = 2 * i;
    }

    for(int i = 0; i < size; i++){
        printf("%i\n", even_integers[i]);
    }

    return 0;
}
```

```
0
2
4
6
8
```

Array Elements in Memory

- What happens in memory when an array is declared?
  - For example: int arr[8];
- Memory is immediately reserved for storing 8 integers (16/ 32 bytes)
- Since array is not initialized, the eight values present in the memory are garbage values
- Array elements always stored in contiguous memory locations.

Array Elements in Memory

Element	12	34	66	-45	23	346	77	90
Address	508	510	512	514	516	518	520	522
Index	0	1	2	3	4	5	6	7

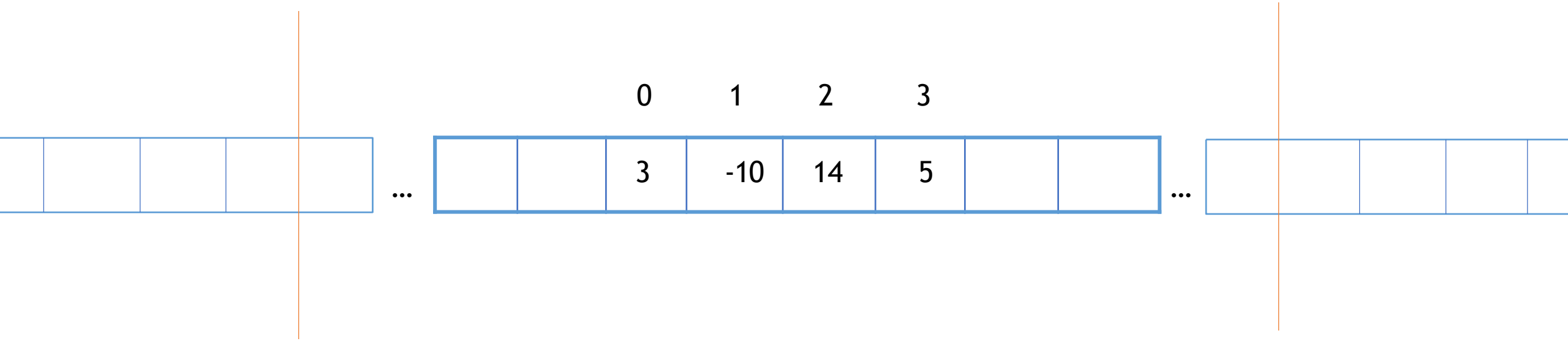
Array Elements in Memory

```c
#include <stdio.h>

int main() {
    int even_integers[5] = {2, 4, 6, 8, 10};

    printf("Element: %i Address: %zu\n", even_integers[0], &even_integers[0]);
    printf("Element: %i Address: %zu\n", even_integers[1], &even_integers[1]);
    printf("Element: %i Address: %zu\n", even_integers[2], &even_integers[2]);
    printf("Element: %i Address: %zu\n", even_integers[3], &even_integers[3]);
    printf("Element: %i Address: %zu\n", even_integers[4], &even_integers[4]);


    return 0;
}
```

```
Element: 2 Address: 140732827272912
Element: 4 Address: 140732827272916
Element: 6 Address: 140732827272920
Element: 8 Address: 140732827272924
Element: 10 Address: 140732827272928
```

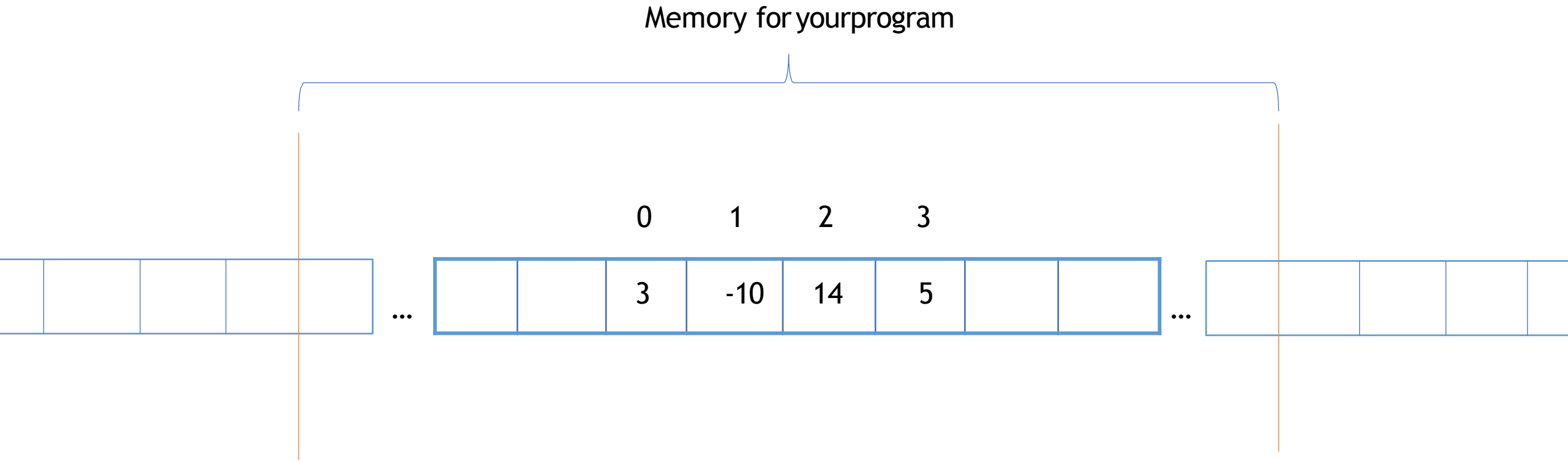
An Array in Memory

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



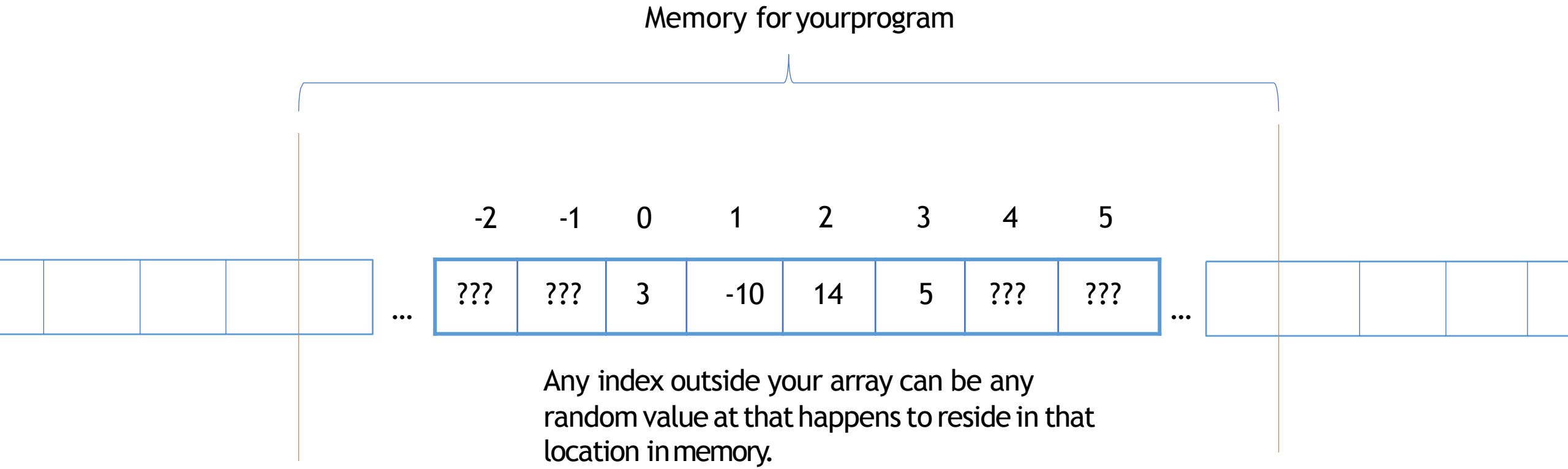
An Array in Memory

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



Out of Bounds

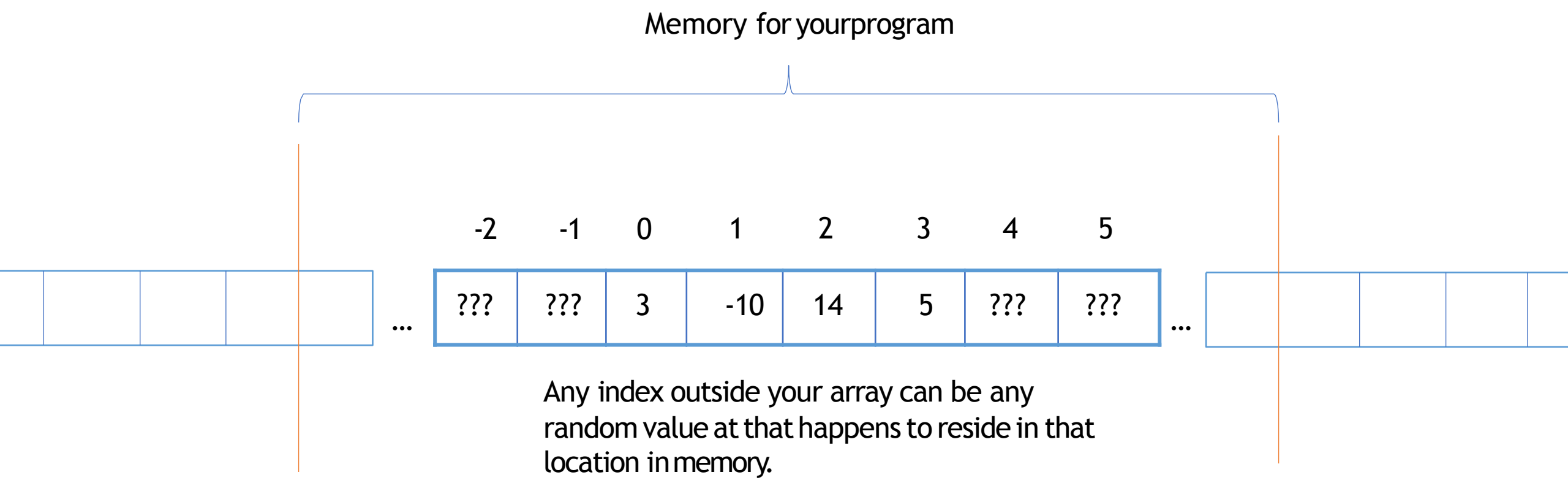
```
int array[4] = { 3, -10, 14, 4};
```



Out of Bounds

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

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

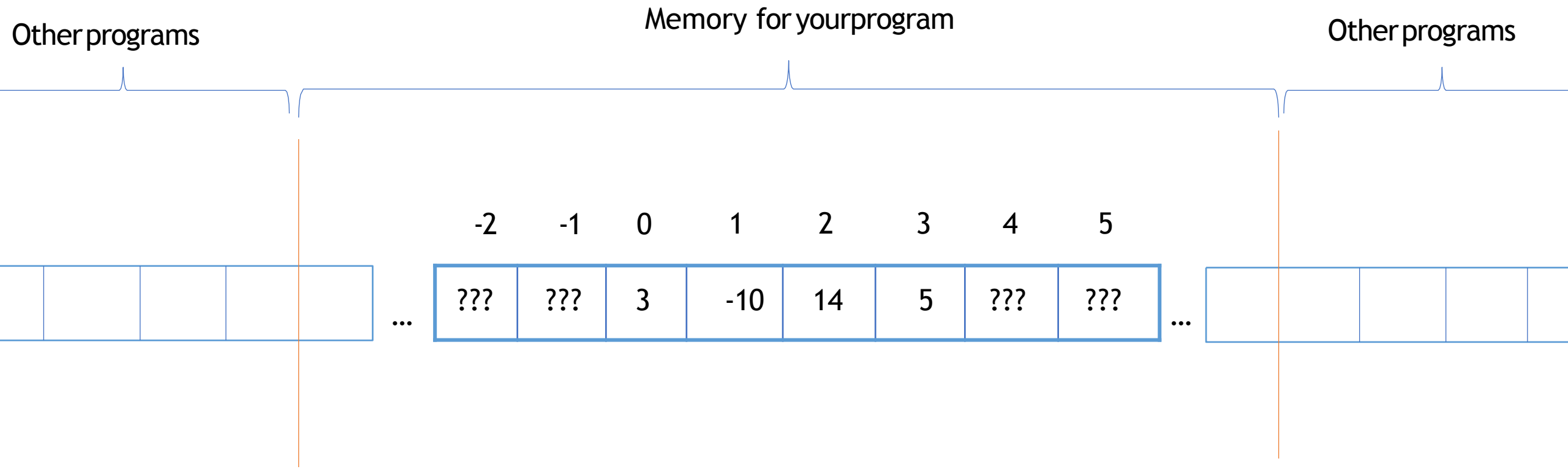


Any index outside your array can be any random value at that happens to reside in that location in memory.

Out of Bounds

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

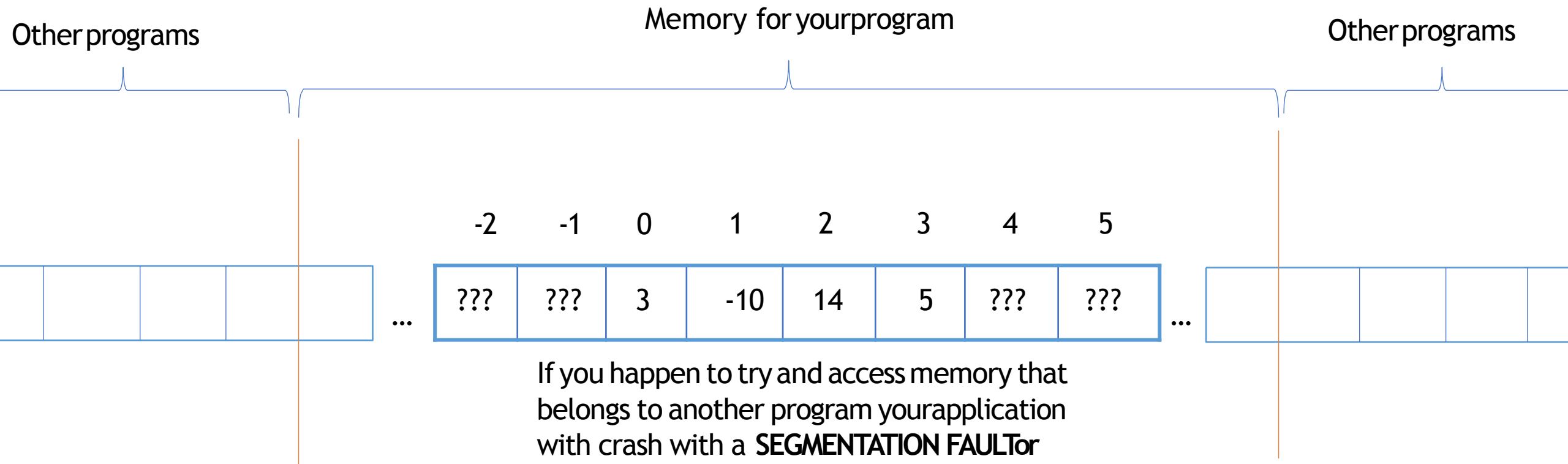
C does not do **ANY** checking to makesure an index is valid



Out of Bounds

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

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



Out of Bounds

```c
#include <stdio.h>

int main() {
    int even_integers[5] = {2, 4, 6, 8, 10};

    for (int i = 0; i <=5; ++i){
        printf("%i\n", even_integers[i]);
    }

    return 0;
}
```

```
2
4
6
8
10
32766
```