Arrays and Functions

Recap

Properties of Arrays:

- Arrays are just pointers (dereferenced by []).
- They have a finite size.
- They have constant size (after memory is allocated).
- Stored in contiguous memory.

Allocation and Deallocation in C vs. C++

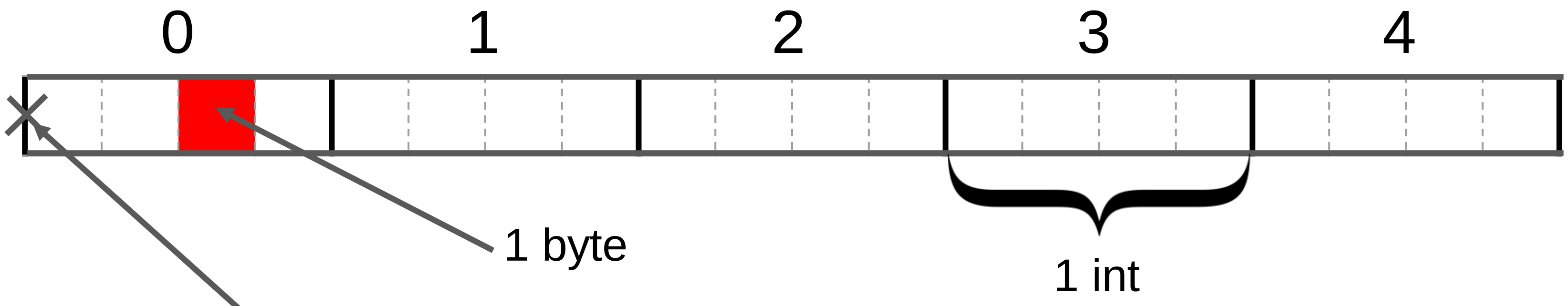
Creating dynamic array of size 5 in:

```c
int* arr = (int*) malloc(sizeof(int) * 5);
free(arr);
```

```cpp
int* arr = new int[5];
delete[] arr;
```

```c
int arr[5];
```

In Memory...



```c
== arr
```

The `sizeof` operator

- Sizeof behaves quite a bit like a function in C++
- It gives the size in **bytes** of a data type or variable

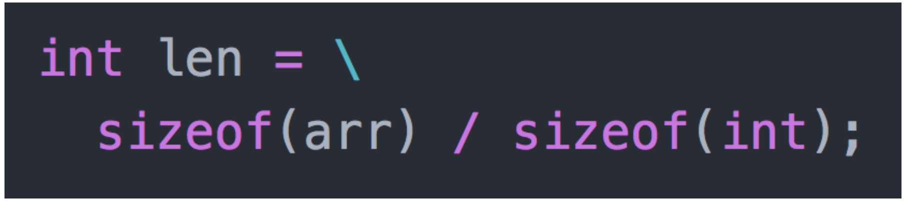
```
sizeof(char) == 1

int n;
sizeof(n) == 4

int arr[5];
sizeof(arr) == ?
```

Determining the ***length*** of an array.

Important distinction:

- *Size* of an array: number of bytes associated with that array.
- *Length* of an array: number of valid indices in the array.

$$\text{Length of array} = \frac{\text{size of array}}{\text{size of data type}}$$



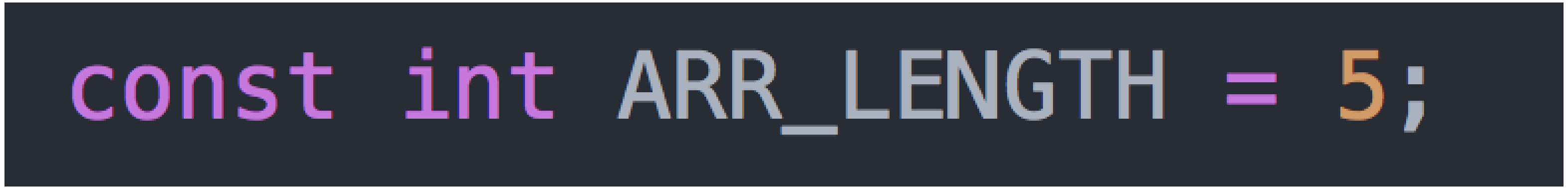
```c
int len = \
  sizeof(arr) / sizeof(int);
```

This does not work when....

- The array is dynamically allocated.
- You pass the array into a function (the array “decays” to a pointer).

So how can we get around these limitations without having to store the length and pass it into all of our functions along with the array?

Global Constants



```
const int ARR_LENGTH = 5;
```

- Should be declared in global scope above main.
- Name should contain only capitals and underscores
- The initial value must be a literal (i.e. a hardcoded value)