Stack based array

int arr[10];

If we know the size of the array before handed, we can use this.

What if we don’t know the size ?

- We allocate memory **dynamically in run time.**

Dynamic Memory - Stored in the heap

```
int *p = new int[10];
```

- We use the **new** keyword.
- Memory is allocated during run time.
- The size of the array can be a variable.
- If we are reading values from a file which can have any number of values, our array size has to be changed accordingly.
- For such a use case, we use **dynamic** memory allocation.

In the assignment, instead of using data type int, we use a struct

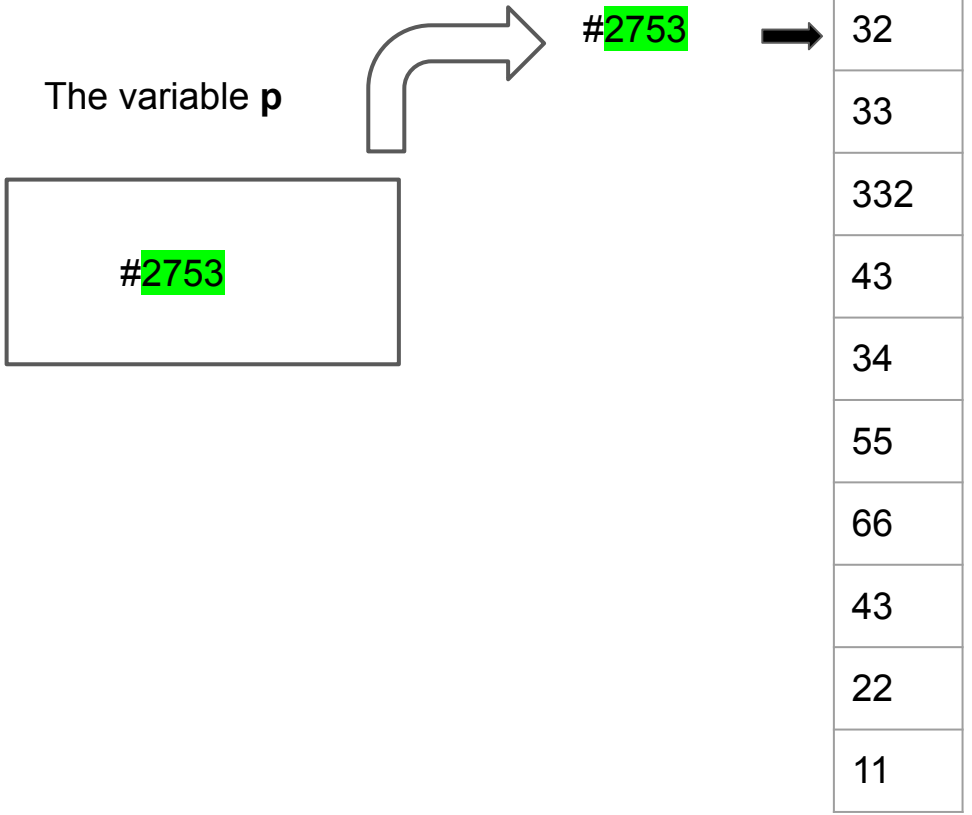
Dynamic Memory - Stored in the heap

int *p = **new** int[**capacity**];

- The size of the array can be a variable.
- If we are reading values from a file which can have any number of values, our array size has to be changed accordingly.
- **capacity -** can be any number depending on file size.

Pointers and arrays

int *p = new int[10];



Exercise:

Task: Read integers from an input file

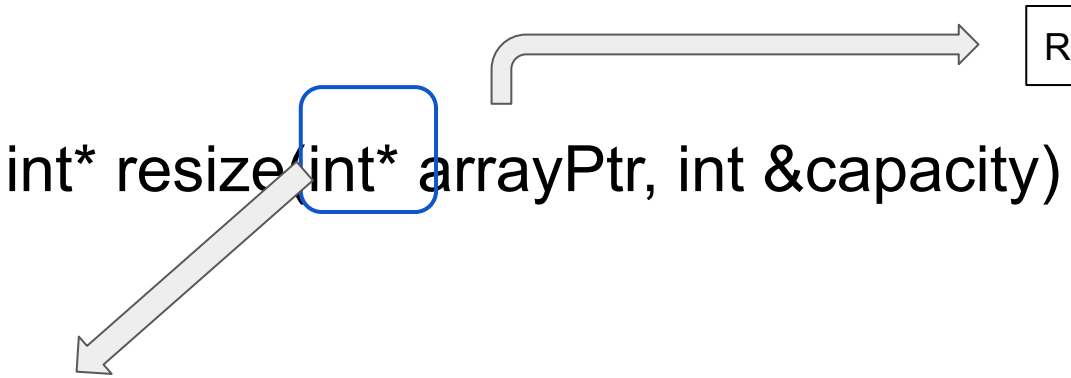
Things to do:

- Create an array of a some initial size using dynamic memory.
- If array overflows, call **resize** function.

Resize function - Understanding the function signature :

Arguments:

- Reference to the array (int * **arrayPtr**)
- Reference to capacity variable(int &**capacity**) - pass by reference
- Returns the pointer to new array



Resize

int *arrayPtr;

Initially, **arrayPtr = x1000**

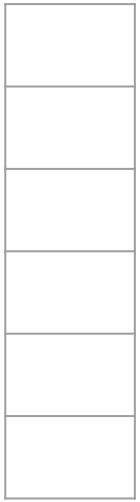
x1000 ⇨

Old array

But we created a new array of double size.

So, **newArray = x2000**

x2000 ⇨

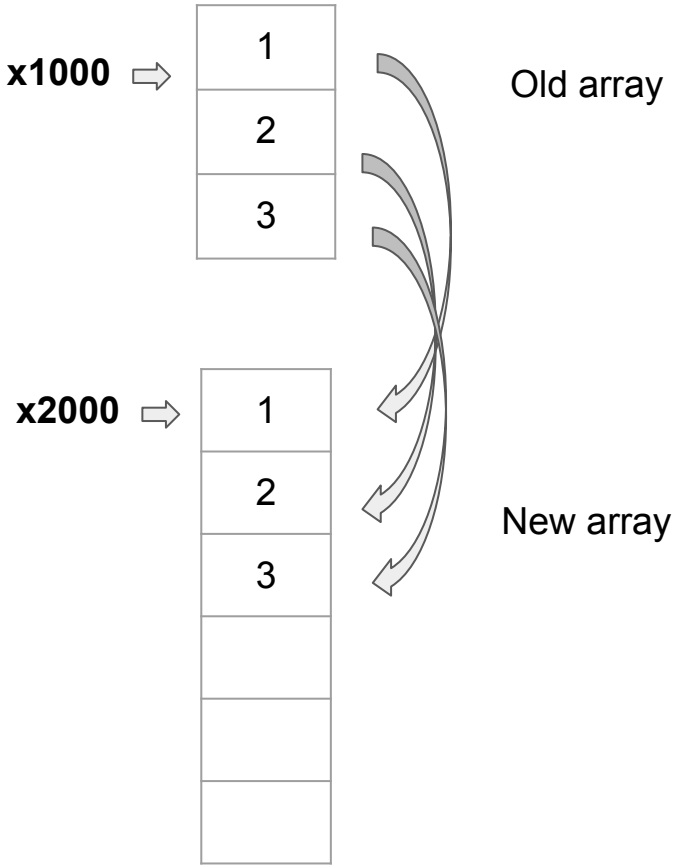


New array

Resize

Copy values from **arrayPtr** to **newArray**

.

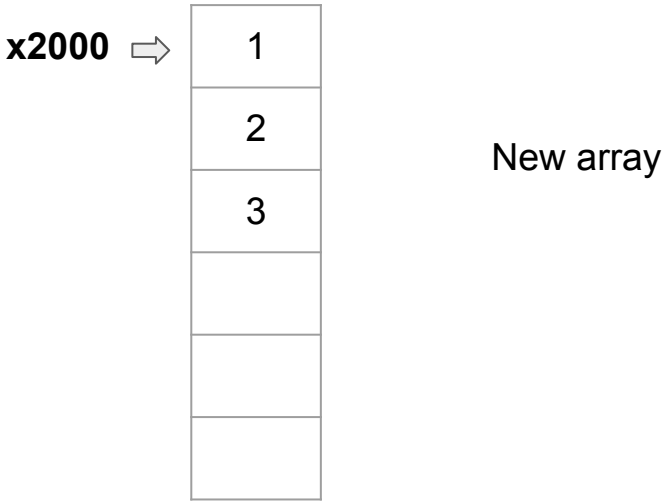


Resize

Deallocate memory of **arrayPtr.**

return the **newArray address (x2000)**

.



Resize function:

Arguments:

- Pointer to the array (int * **arrayPtr**)
- Reference to capacity variable(int **&capacity**) - pass by reference

Functionality:

- Create a new array of double the size of the original array.
- Copy elements from original to new array
- Deallocate memory of the old array

Delete

After you are done, you need to deallocate the memory assigned to the initial array.

For a dynamic array defined like below, deallocation is done as follows:

```
int *ptr2 = new int[10];
delete [] ptr2;
```

Double pointers

```
int *p = new int[10];
int **q = &p;
```