Two's Complement, Memory Allocation, and Function Pointers

Lecture 11
Feb 21st 2023 | COMP 211-002 | Kenan Poole

Welcome!

Today:

- Two's Complement
- Assignment 3 Overview
  - Memory allocation
  - Function pointers

Logistics:

- Assignment 3 posted, due before class March 7th
- Our excellent LA, Kenan Poole will be teaching today

Fun fact...

Have an urgent question during the day?

Prof. Bakita has an office phone!
(919) 590-6103

Vote for Student Body President!

Candidates

TJ Edwards
DTH Article

Chris Everett
DTH Article

Theodore Nollert*
Platform: go.unc.edu/theo

Vote

On HeelLife

*Write-in candidate

What is the Student Body President?

- ➔ The representative of all UNC students to university administration

What do they do?

- ➔ Represent students as a full, voting member of the Board of Trustees
- ➔ Regularly meet with the Chancellor, Provost, and Deans to provide student input on key decisions

Read about the candidates and make an informed decision!

Practical Two's Complement

Equivalences

System (8 bits)	Hex	Binary	Converting	Decimal Value
Unsigned	8C	1000 1100	$2^7+2^3+2^2$	140
Signed Magnitude	8C	1000 1100	$-1(2^3+2^2)$	-12
Two's Complement	8C	1000 1100	$-2^7+2^3+2^2$	-116
Unsigned	25	0010 0101	$2^5+2^2+2^0$	37
Signed Magnitude	25	0010 0101	$1(2^5+2^2+2^0)$	37
Two's Complement	25	0010 0101	$2^5+2^2+2^0$	37

Why we don’t use Signed Magnitude?

- Binary Addition

	0111 1000	(120)		0011 1000	(56)		0111 1000	(120)
+	1000 1111	(-15)	+	1000 1111	(-15)	+	0100 1111	(79)
	0000 0111	(7)		1100 0111	(-71)		1100 0111	(-71)
							Overflow!	

- 0? +0? -0?
  - 0000 = +0, 1000 = -0 (for 4 bits)

Binary Addition for Two's Complement

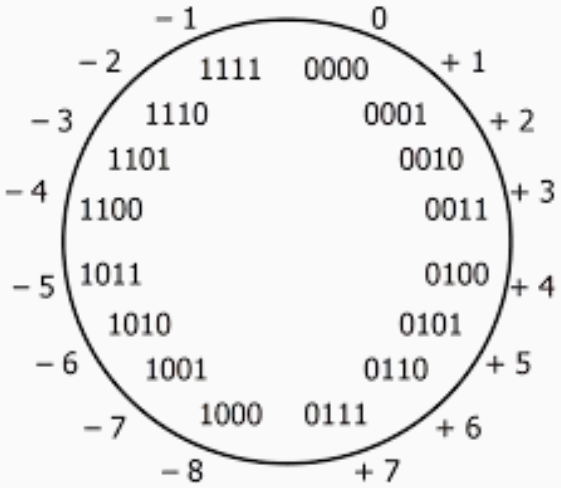
Two's Complement:

	0111 1000	(120)		0011 1000	(56)		0111 1000	(120)
+	1000 1111	(-113)	+	1000 1111	(-113)	+	0100 1111	(79)
	0000 0111	(7)		1100 0111	(-57)		1100 0111	(-57)
							Overflow!	

Overflow for Two's Complement

1) If the sum of two positive numbers yields a negative result, the sum has overflowed.
2) If the sum of two negative numbers yields a positive result, the sum has overflowed.
3) Otherwise, the sum has not overflowed

Overflow



2's Complement

4 bit number circle

```c
#include <stdio.h>

int main(){

        int x = 0b11111111111111110000000000000000;

        printf("x printed as unsigned int: %u\n", x);
        printf("x printed as int: %d\n", x);

        return 0;
}
```

Printing int

Terminal Output

```
x printed as unsigned int: 4294901760
x printed as int: -65536
```

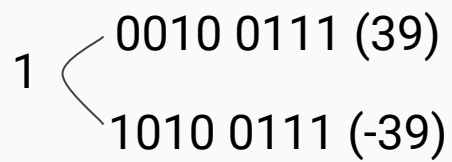
Equivalences

System	Hex	Binary	Converting	Decimal Value
Two's Complement (4 bit)	C	1100	-2^3+2^2	-4
Two's Complement (8 bit)	FC	1111 1100	$-2^7+2^6+2^5+2^4+2^3+2^2$	-4
Two's Complement (8 bit)	0C	0000 1100	2^3+2^2	12
Two's Complement (4 bit)	4	0100	2^2	4
Two's Complement (8 bit)	F4	1111 0100	$-2^7+2^6+2^5+2^4+2^2$	-12
Two's Complement (8 bit)	04	0000 0100	2^2	4

Flipping the Sign

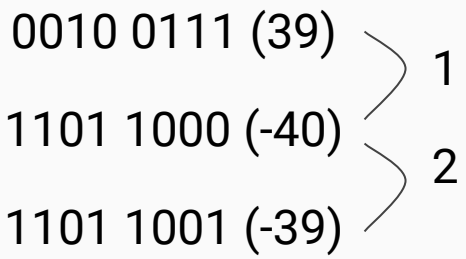
Signed Magnitude:

1. Flip the MSB



Two's Complement:

1. Flip all the bits (1's complement)
2. +1



Assignment 3 Overview

Tetris Tournament

- Assignment 3 is up on class website
- Due Thursday, March 7
- Given a list of quicksaves, sort the list, and output the sorted list
  - Variable number of players (ie quicksaves)
  - Variable method of sorting
- Autograder is not released yet, but not informative where the code may go wrong, so you must depend on your own testing of use and edge cases

The *malloc* Function

```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```

main

Stack

Heap

```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```

main

n | 5

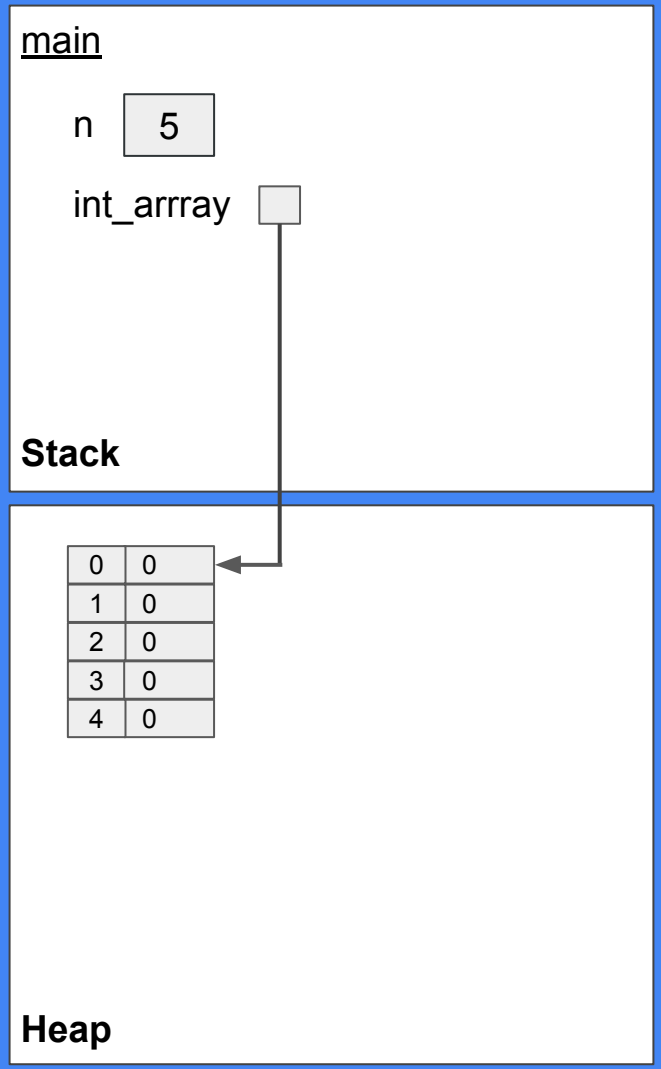
Stack

Heap

```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

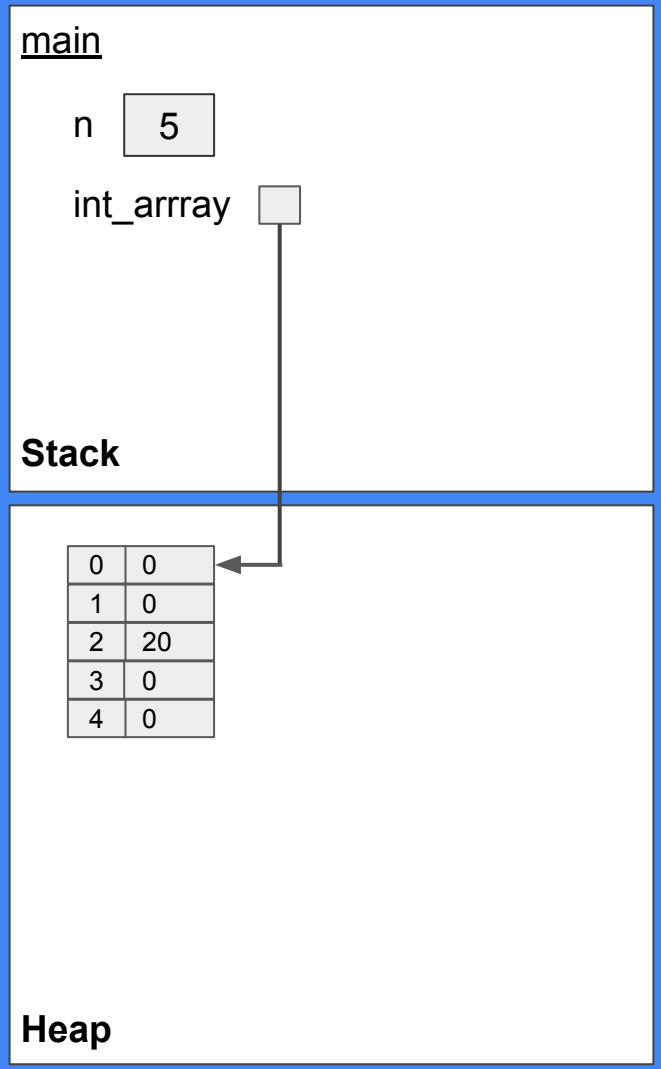
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

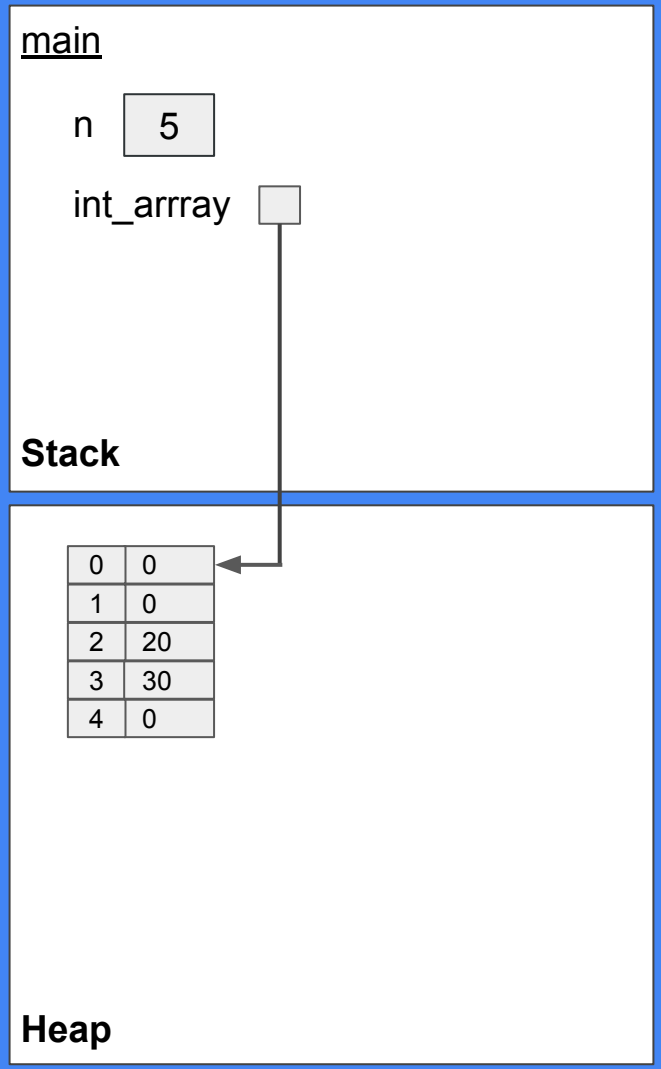
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

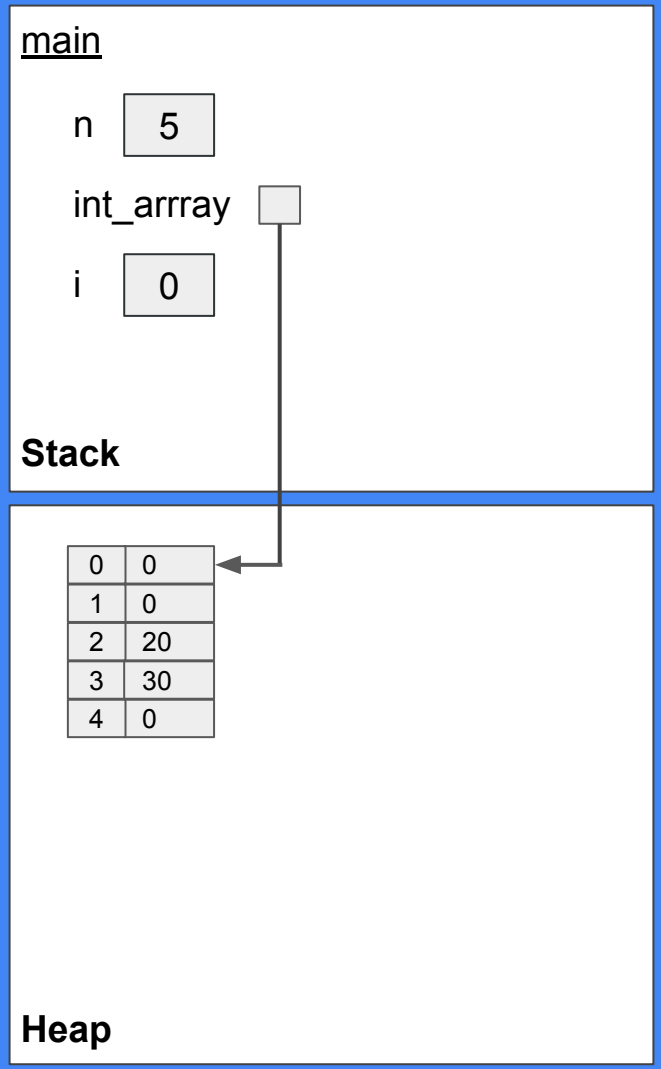
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

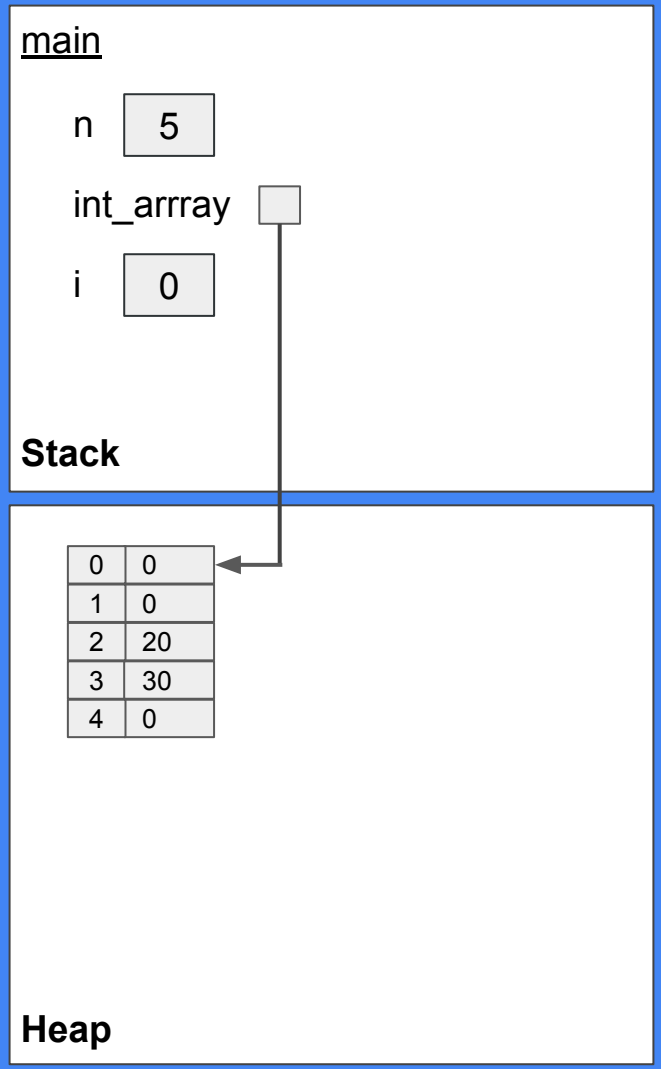
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

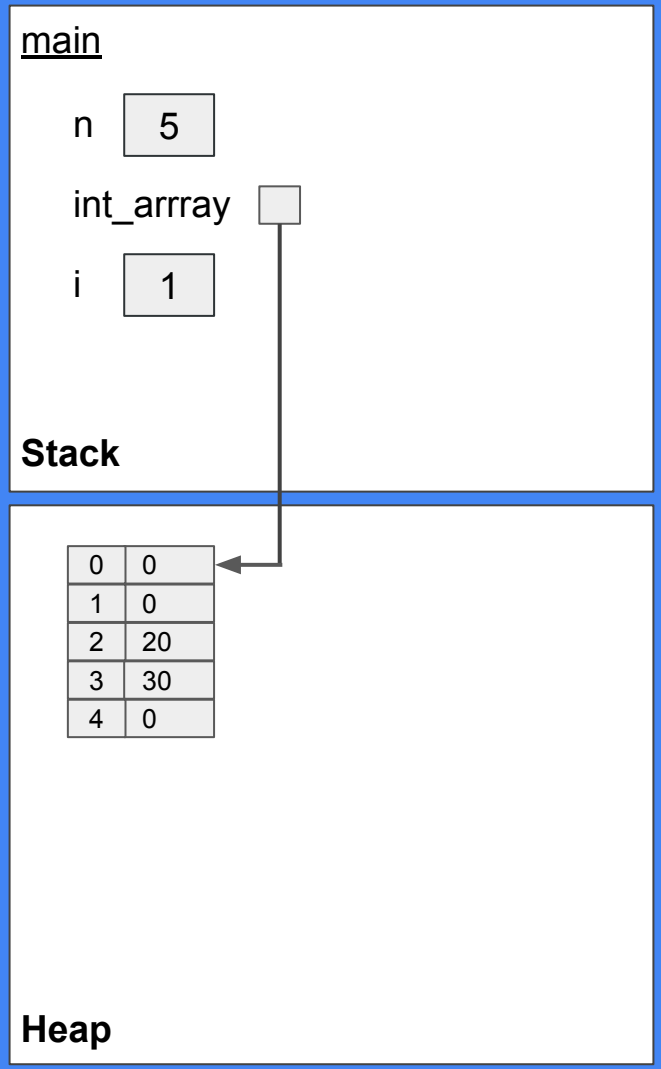
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

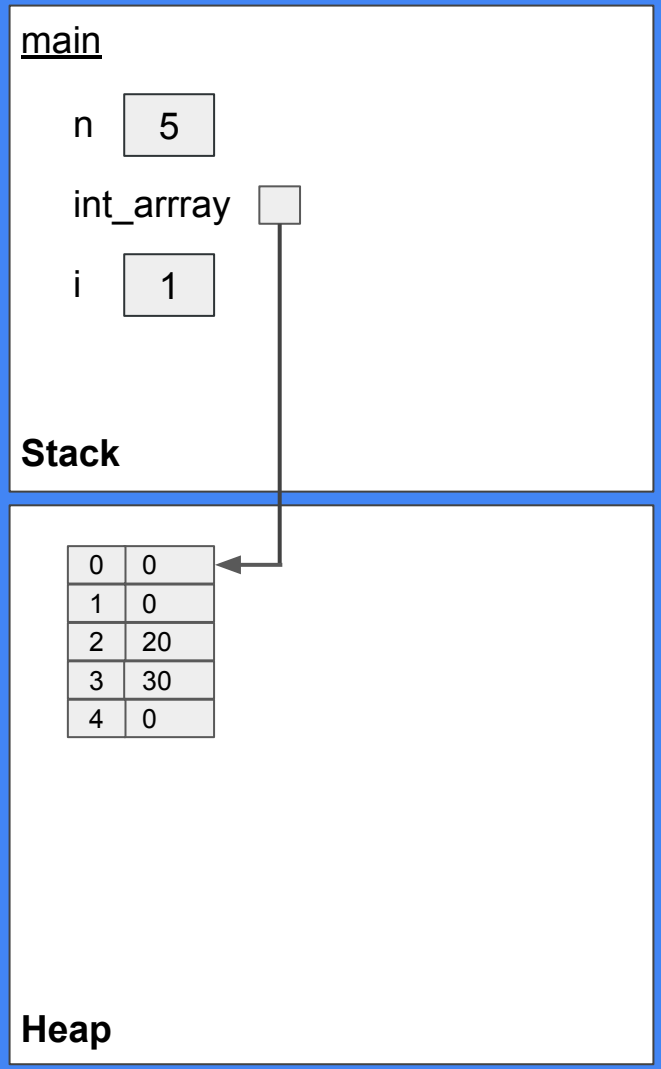
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

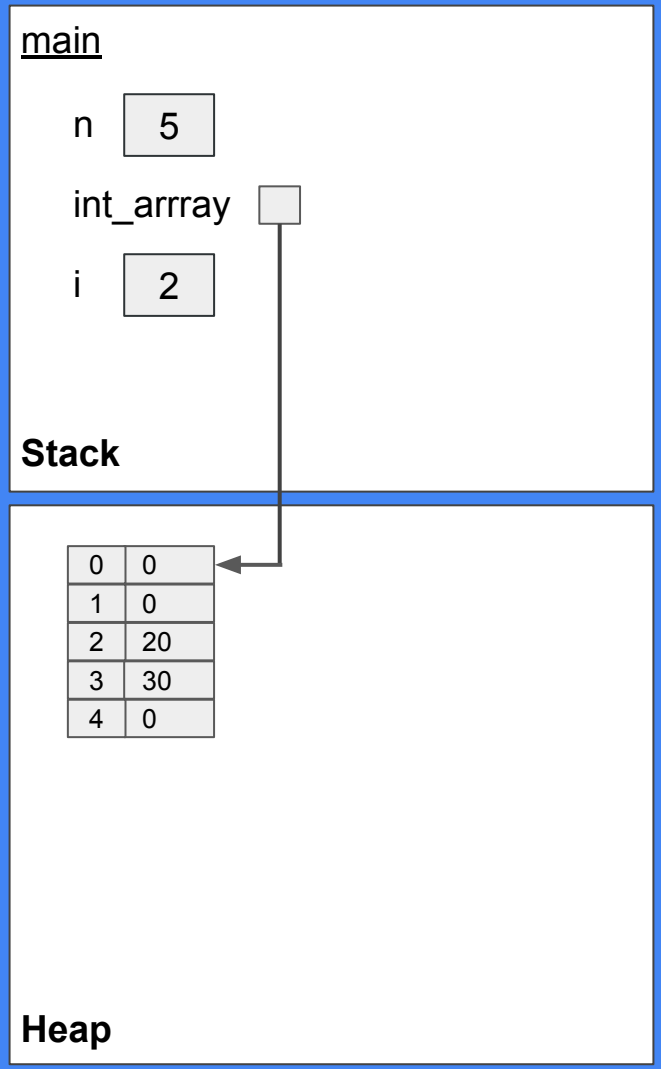
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

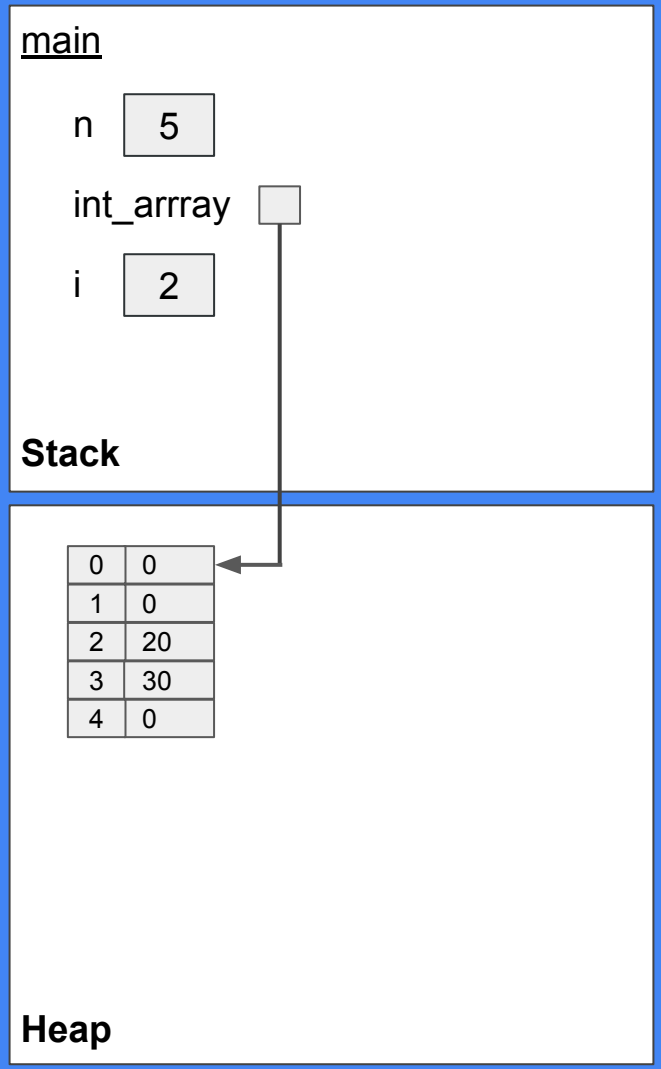
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

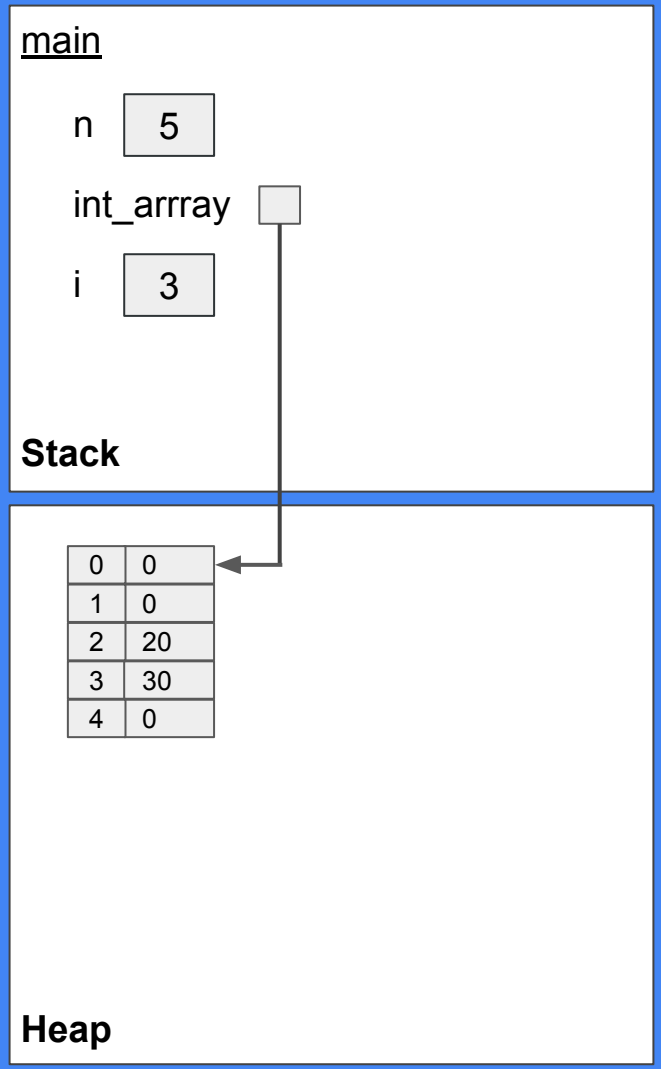
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

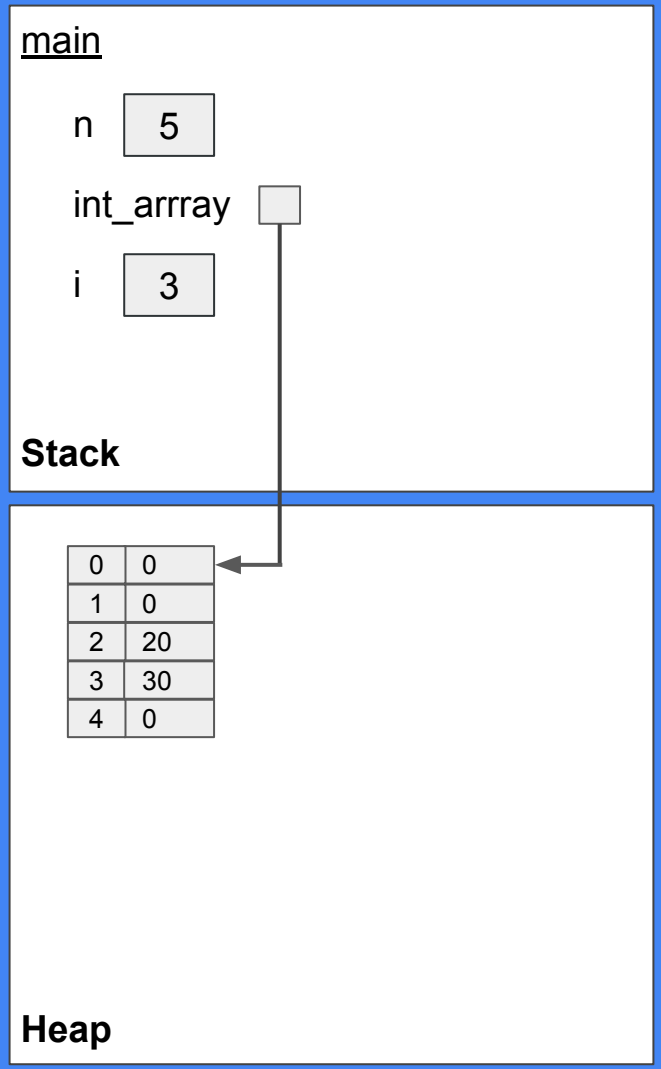
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

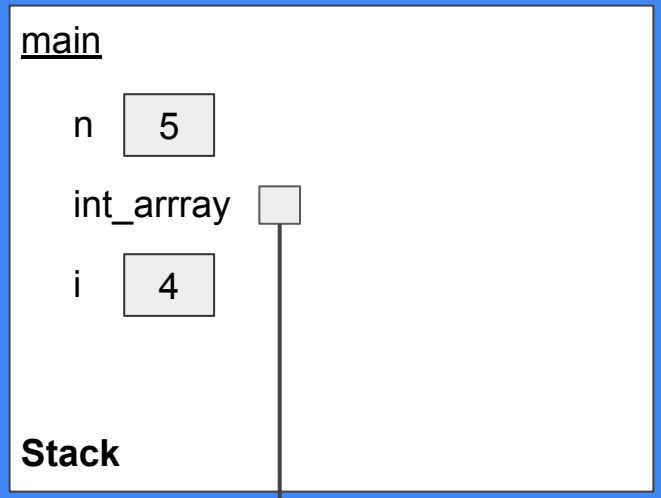
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```

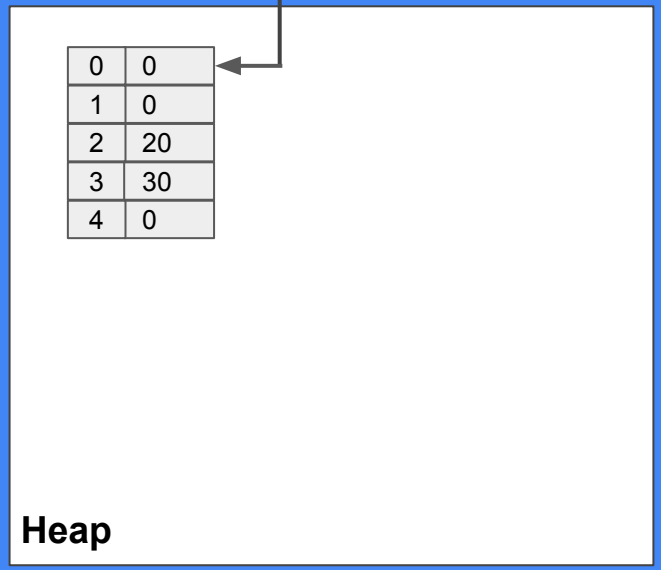



```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```

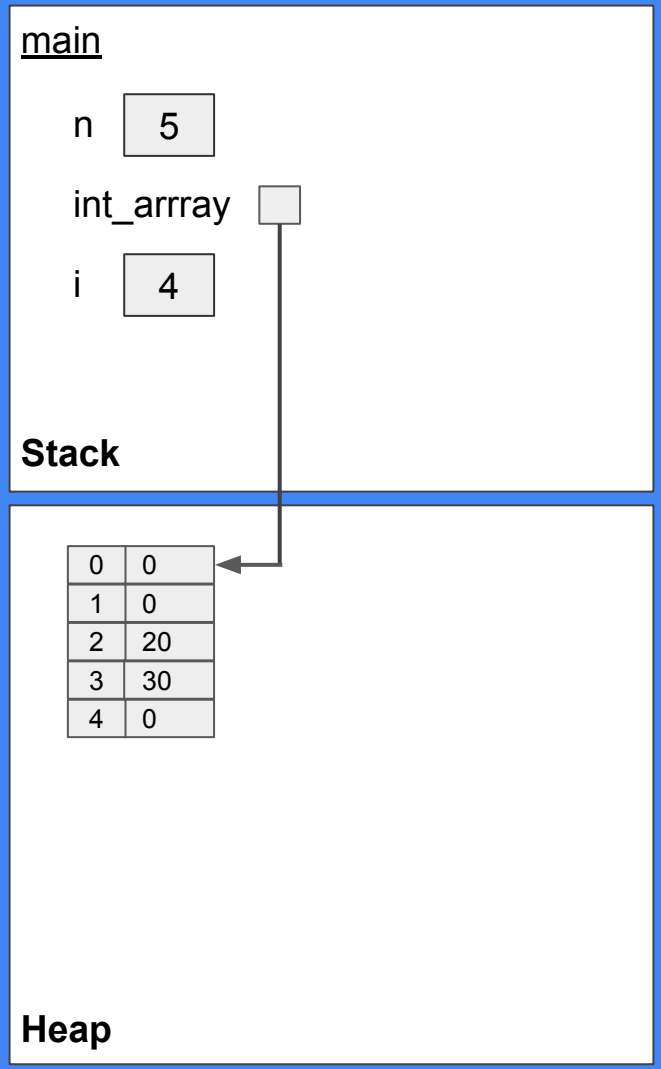





```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

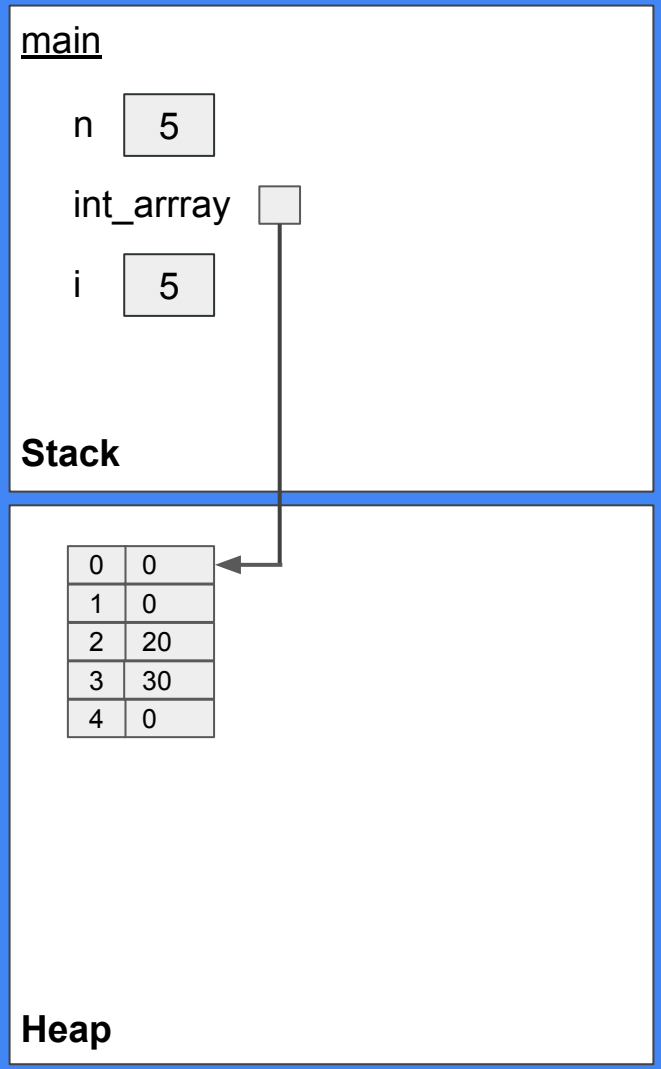
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

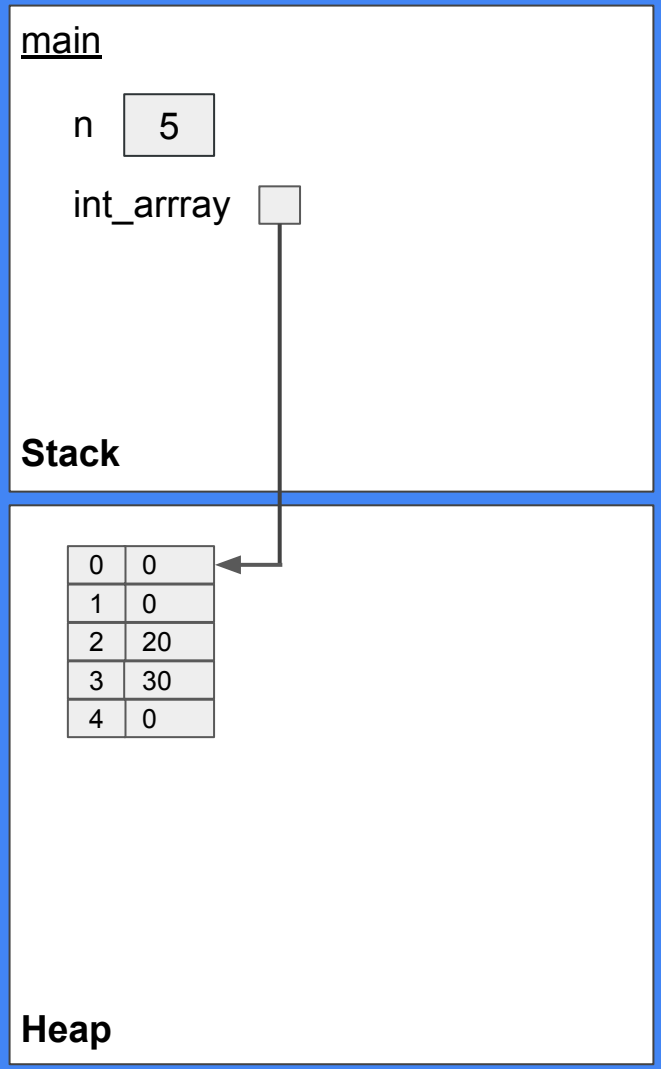
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

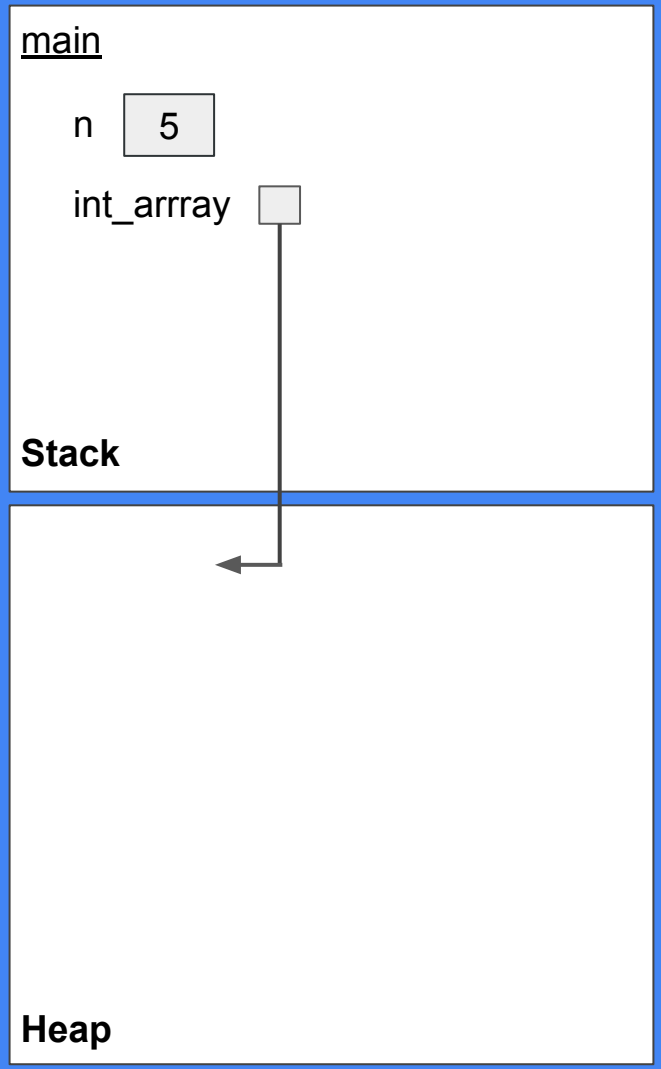
```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```




```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```

main

n 5

int_arrray

Stack

Heap

```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```

Stack

Heap

```c
#include <stdio.h>
#include <stdlib.h>

int main(){

        int n = 5;
        int* int_array = (int*)malloc(sizeof(int) * n);

        int_array[2] = 20;
        *(int_array+3) = 30;

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

        free(int_array);
        int_array = NULL;

        return 0;
}
```

Terminal Output

```
int_array[0] = 0
int_array[1] = 0
int_array[2] = 20
int_array[3] = 30
int_array[4] = 0
```

Functions as Params

```c
#include <stdio.h>


int triple(int x){ return x*3; };

void map(int *source, int *dest, int (*operator)(int x)){
        *dest = operator(*source);
}

int main(){

        int x = 4;
        int* xp = &x;

        int* ip;

        map(xp, ip, triple);
        printf("xp value: %d\n", *xp);
        printf("ip value: %d\n", *ip);
}
```

Terminal Output

```
xp value: 4
ip value: 12
```

Questions?

See office hour calendar on the website for availability.

Contact:
Email: hacker@unc.edu
Twitter: @JJBakita
Web: https://cs.unc.edu/~jbakita

Old Well, University of North Carolina at Chapel Hill, Winter 2017