COMP1511 PROGRAMMING FUNDAMENTALS

LECTURE 6

Arrays

LAST LECTURE...

ON TUESDAY...

- Talked about good style/bad style
- Functions - what/how/why?

THIS LECTURE...

TODAY...

- look at arrays

WHERE IS THE CODE?

Live lecture code can be found here:

HTTPS://CGI.CSE.UNSW.EDU.AU/~CS1511/22T2/LIVE/WEEK03/

FUNCTIONS RECAP

WHAT?

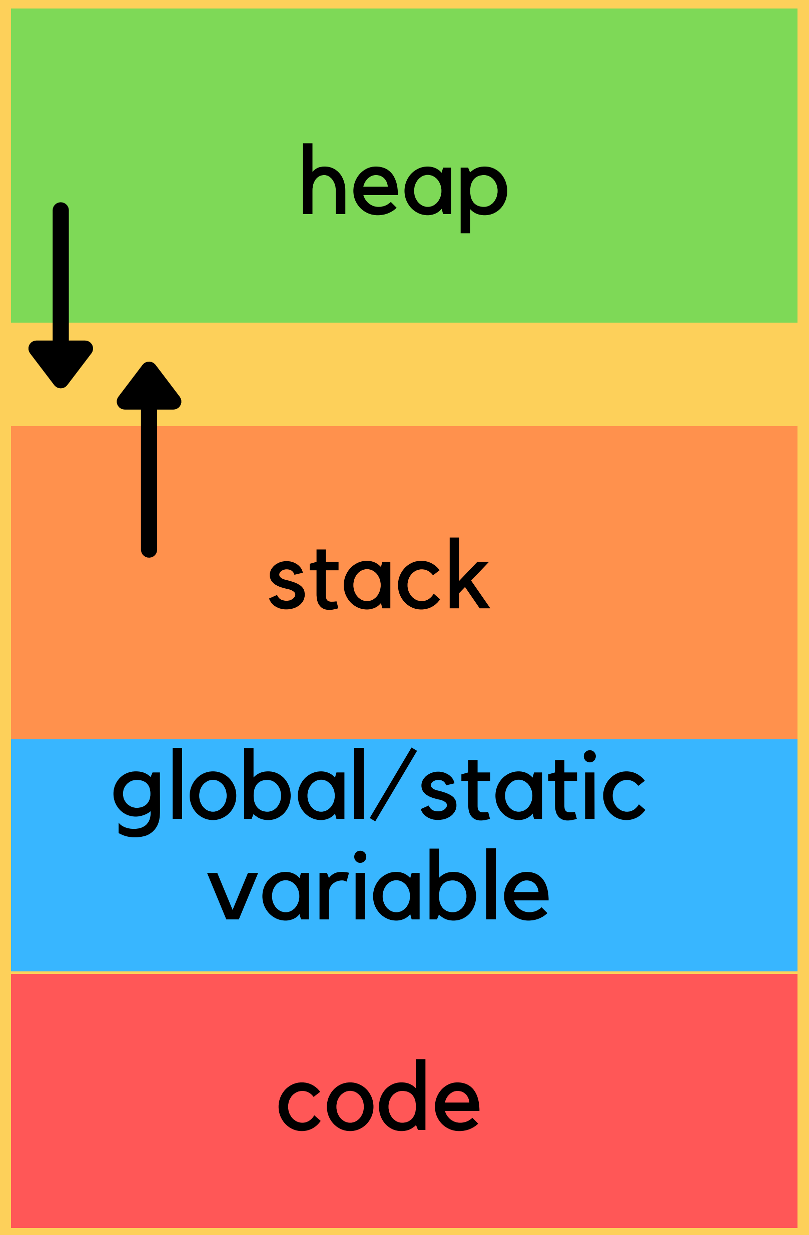
- A function is a block of statements that performs a specific task
- There are two classifications:
    - procedures don't return anything, but have a side effect
    - functions which return a result (but shouldn't have a side effect)
    - in C, these are all called functions

FUNCTIONS RECAP

WHY?

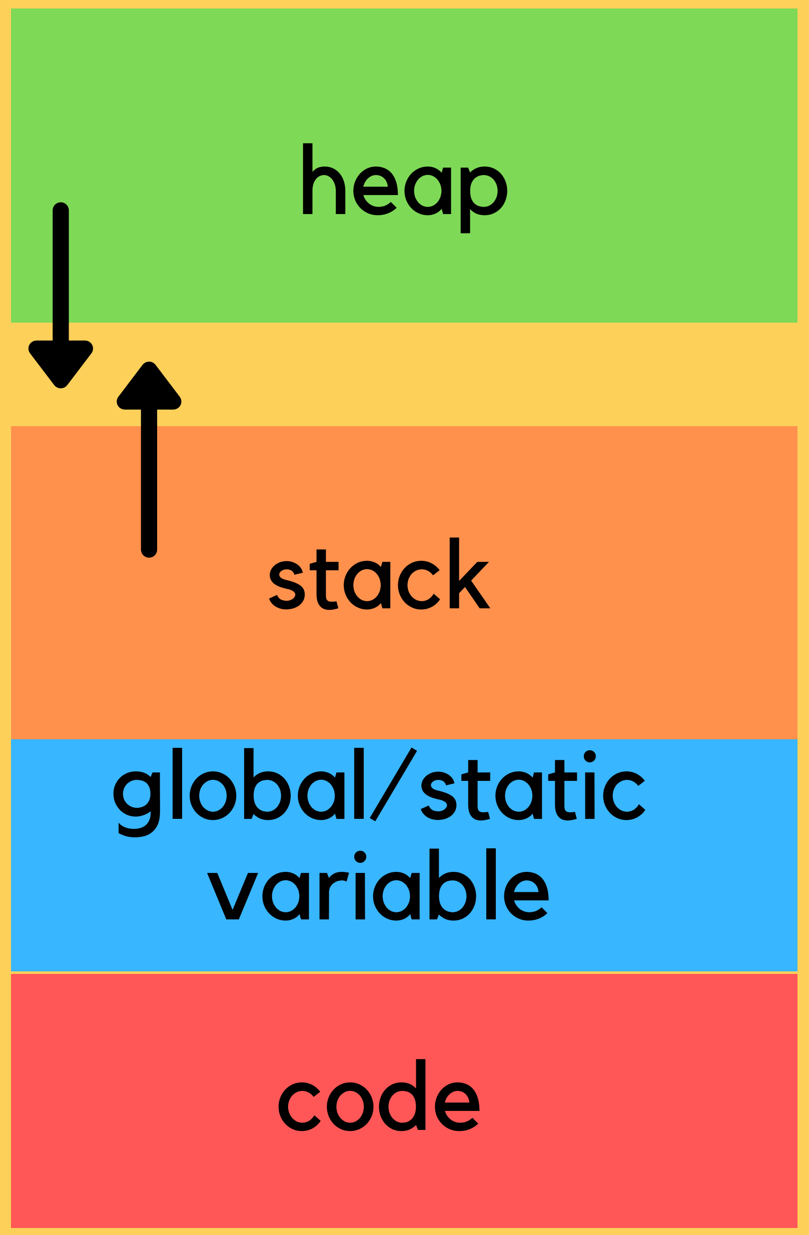
- Improve readability of the code
- Improve reusability of the code
- Debugging is easier (you can narrow down which function is causing issues)
- Reduces size of code (you can reuse the functions as needed, wherever needed)

REVISITING MEMORY



- Our C file is stored on the hard drive
- Our Compiler compiles the code into another file that the computer can read
- When we execute code, the CPU will actually process the instructions and perform basic arithmetic, but the RAM will keep track of all the data needed in those instructions and operations, such as our variables.
- Reading and writing to variables will change the numbers in RAM
- Memory is divided into the stack and the heap
- The stack is an ordered stack and the heap is a random free for all - insert something where you can find space for it.

REVISITING MEMORY



- Stack memory is where relevant information about your program goes:
  - which functions are called,
  - what variables you created,
- Once your block of code finishes running {}, the function calls and variables will be removed from the stack (it's alive!)
- It means at compile time we can allocate stack memory space (not at run time)
- The stack is controlled by the program NOT BY THE developer
- The heap is controlled by the developer (more on this in a few weeks) and can be changed at run time

MEMORY IS IMPORTANT

WITHOUT MEMORY, WE CAN'T REALLY RUN ANYTHING

- Think of your own memory and what it allows you to do.
- Computer memory is important to consider when you are writing your code (we don't focus on this in 1511, but you will in later courses)
- The more you waste memory, the slower your program gets… you will learn all about this in later computing courses! In 1511 we don't mind the wastage :)

SO FAR...

- We have been able to declare variables which store a single item of a single type (an int, a double, a char)
- We can use enumerations to define a set of possible values
- We can also define structs to store a record of related data

SO FAR...

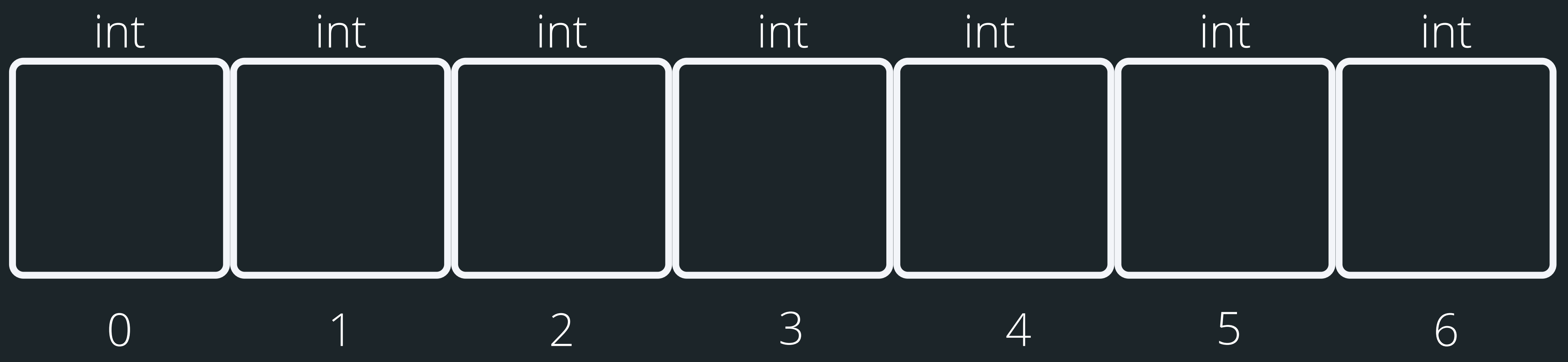
- We can create variables of a particular type, so our program knows how much data is required to store

SO FAR...

- But what if we want to store a lot of data of the same type?

ARRAYS

- Arrays allow us to store multiple elements of the same type in a single variable



WHY DO WE NEED AN ARRAY?

LET'S LOOK AT AN EXAMPLE PROBLEM

- Let's say I am tracking my ice cream consumption over a week (without arrays)

```
int mon = 2;
int tues = 3;
int wedn = 3;
int thur = 5;
int fri = 7;
int sat = 1;
int sun = 3;
// Any day with 3 or more scoops is too
much!
if (mon >= 3){
    printf("Too much ice cream\n");
}
if (tue >= 3) {.......
```

WHY DO WE NEED AN ARRAY?

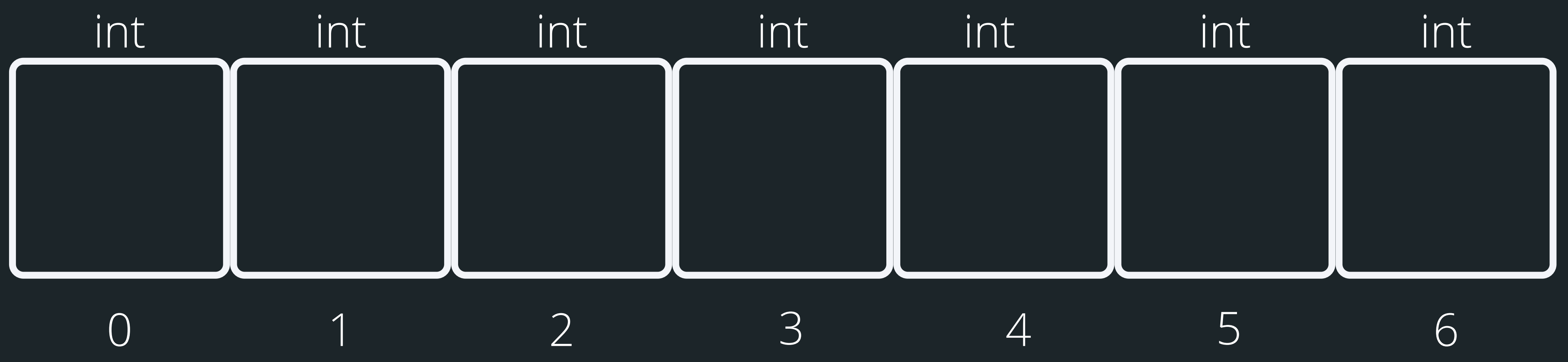
LET'S LOOK AT AN EXAMPLE PROBLEM

- What if I am tracking this over the month or over a year?
  - Will I need 30 variables/365 variables?

ARRAY (VISUALLY)

NOTE: ALL ELEMENTS OF AN ARRAY MUST BE OF THE SAME DATA TYPE (HOMOGENOUS)

- If we group our data type as a collection, for example a collection of integers:
- We can access them as a group(collection)
- We can loop through and access each individual element of that collection



this array holds 7 integers

You can access elements of an array by referring to their index

THIS IS A GREAT PLACE TO USE AN ARRAY...

HOW DO WE DECLARE AN ARRAY

Type of data stored in array

Name of the array

Number of items in the array

```
// 1. Declaring an array
int ice_cream_consum[7];


// 2. Declaring and Initialise the array
// Note that once you declare an array,
// you can't initialise it in this way
int ice_cream_consum[7] = {3, 2, 1, ...};
```

To initialise, open curly bracket and separate values by comma. If you have empty {}, it means to intialise the whole array to 0

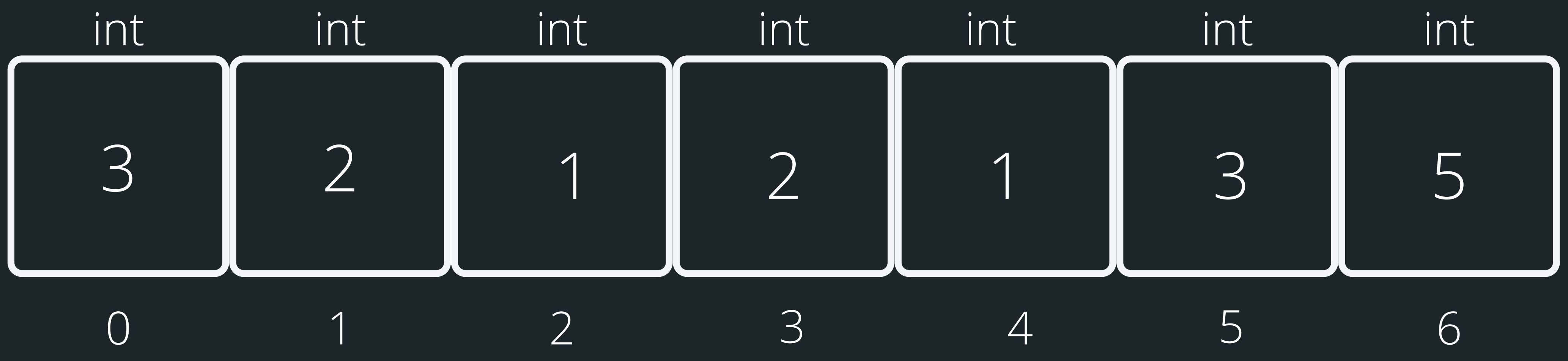
ARRAY (VISUALLY)

DECLARING AND INITIALISING AN ARRAY

- So let's say we have this declared and initialised:

```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};
```

- This is what it looks like visually:



this array holds 7 integers

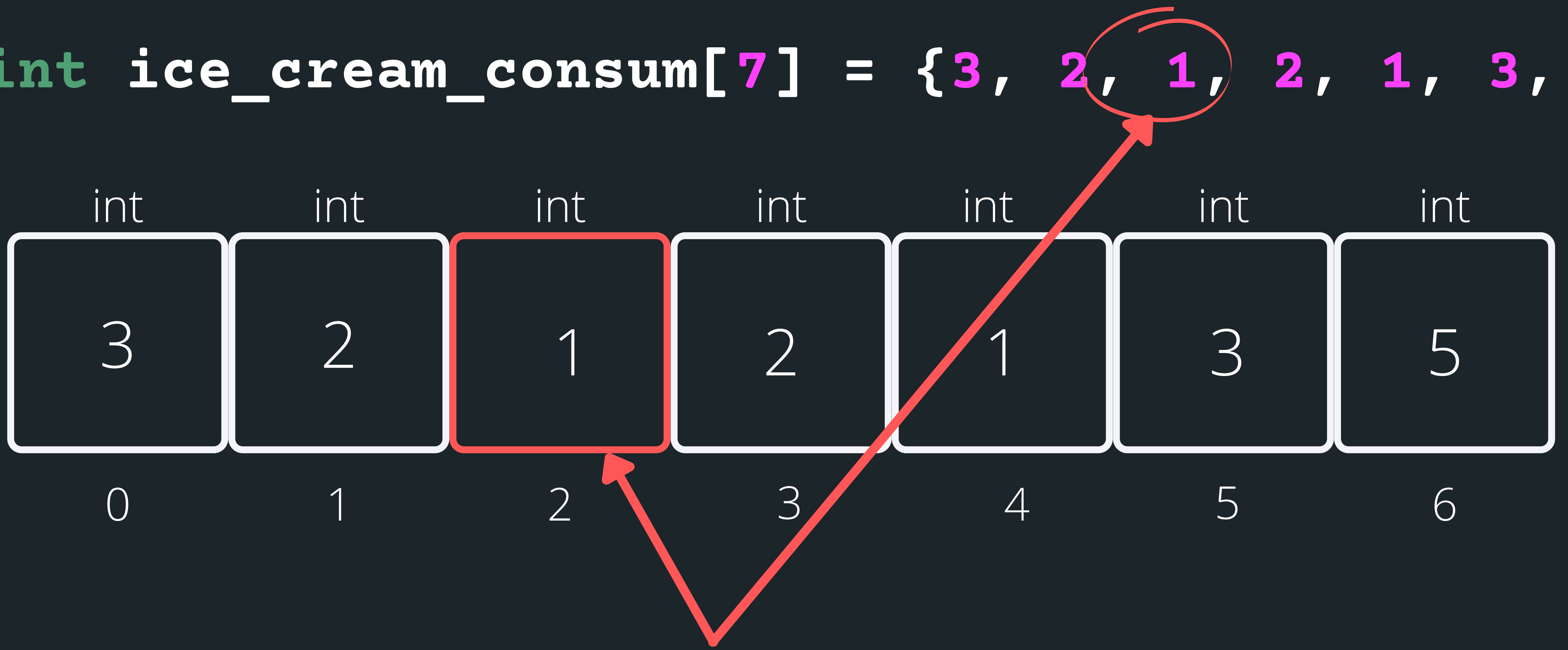
Note that indexing starts at 0

ARRAY (VISUALLY)

ACCESSING ARRAY ELEMENTS

- You can access any element of the array by referencing its index
- Note, that indexes start from 0
- Trying to access an index that does not exist, will result in an error

```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};
```



If I wanted the third element of the array

The index would be 2, so to access it:

ice_cream_consum[2]

USING ARRAYS

CLOSER LOOK

- You can’t printf() a whole array, but you can print individual elements (consider how you could go through the array to print out every element…)
- You can’t scanf() a whole array, i.e. a line of user input test into an array, but you can can scanf() individual elements (think how to do every element in an array…)

USING ARRAYS

CLOSER LOOK

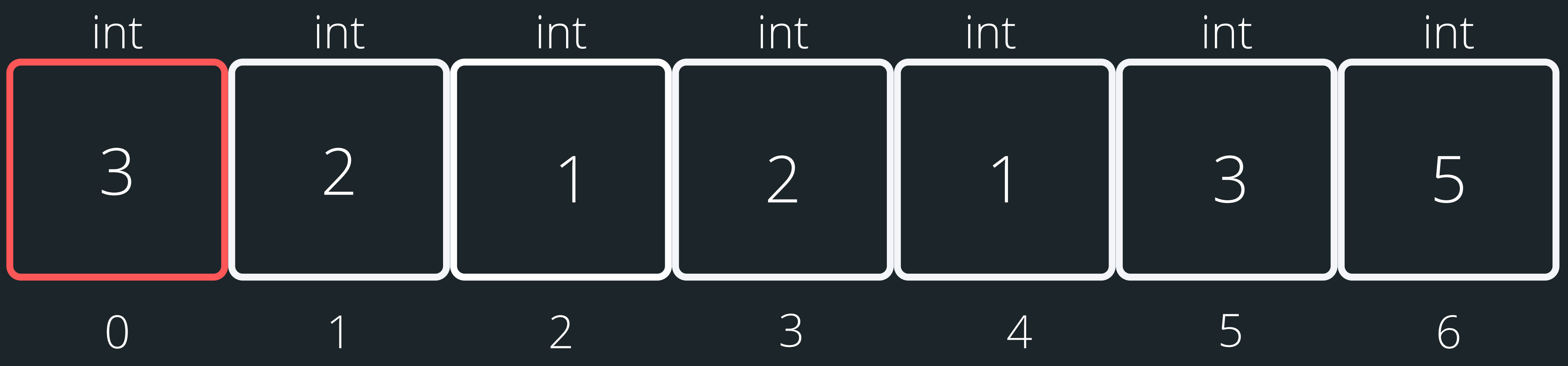
```c
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};

int i = 0;
while (i < 7){
    printf("%d ", ice cream_consum[i]);
    i++;
}
```

Start at index 0 (first entry into while loop)

ice_cream_consum[0]

print what is inside index 0



USING ARRAYS

CLOSER LOOK

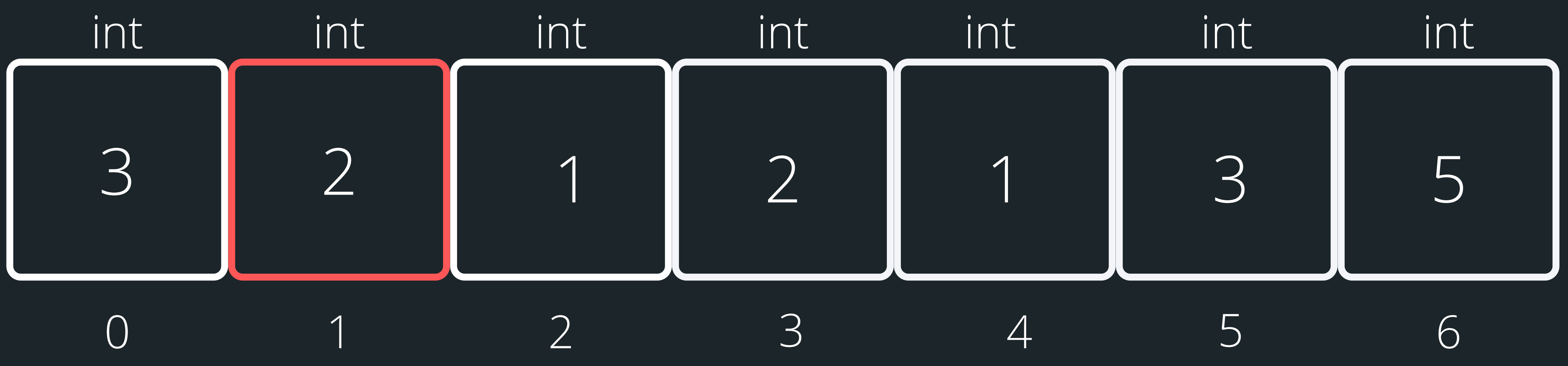
```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};

int i = 0;
while (i < 7){
    printf("%d ", ice cream_consum[i]);
    i++;
}
```

increase index by 1

ice_cream_consum[1]

print what is inside index 1



USING ARRAYS

CLOSER LOOK

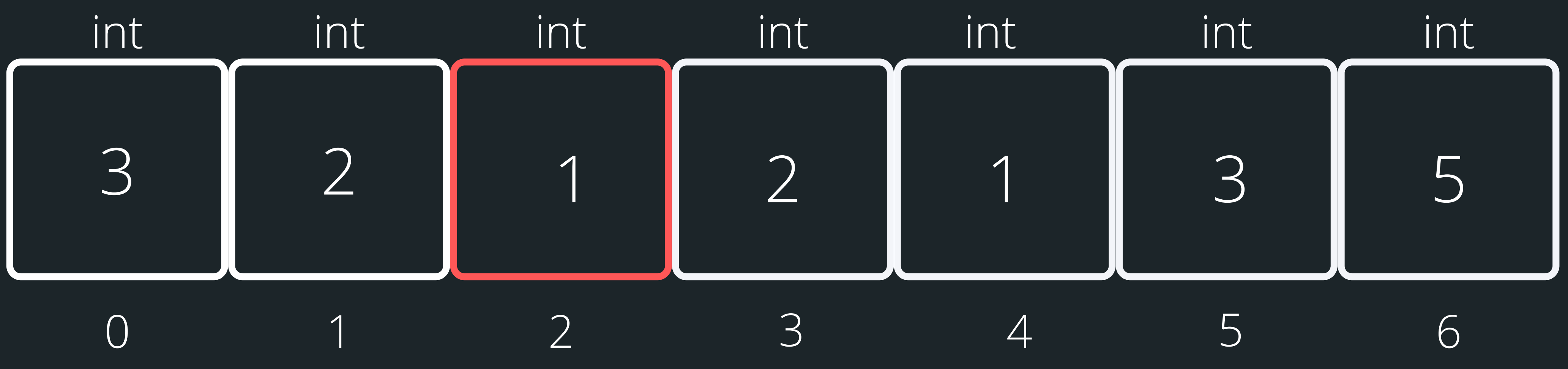
```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};

int i = 0;
while (i < 7){
    printf("%d ", ice cream_consum[i]);
    i++;
}
```

increase index by 1

ice_cream_consum[2]

print what is inside index 2



USING ARRAYS

CLOSER LOOK

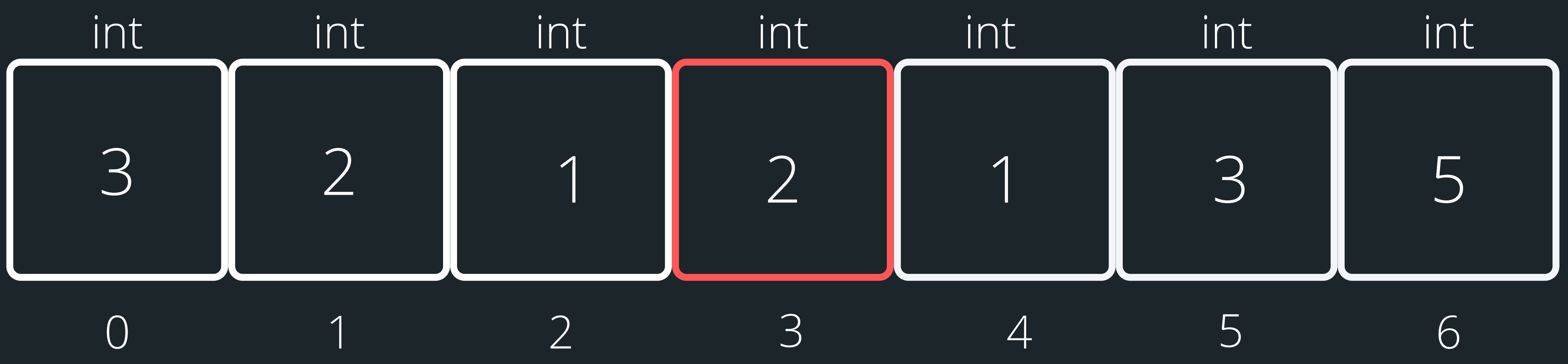
```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};

int i = 0;
while (i < 7){
    printf("%d ", ice cream_consum[i]);
    i++;
}
```

increase index by 1

ice_cream_consum[3]

print what is inside index 3



USING ARRAYS

CLOSER LOOK

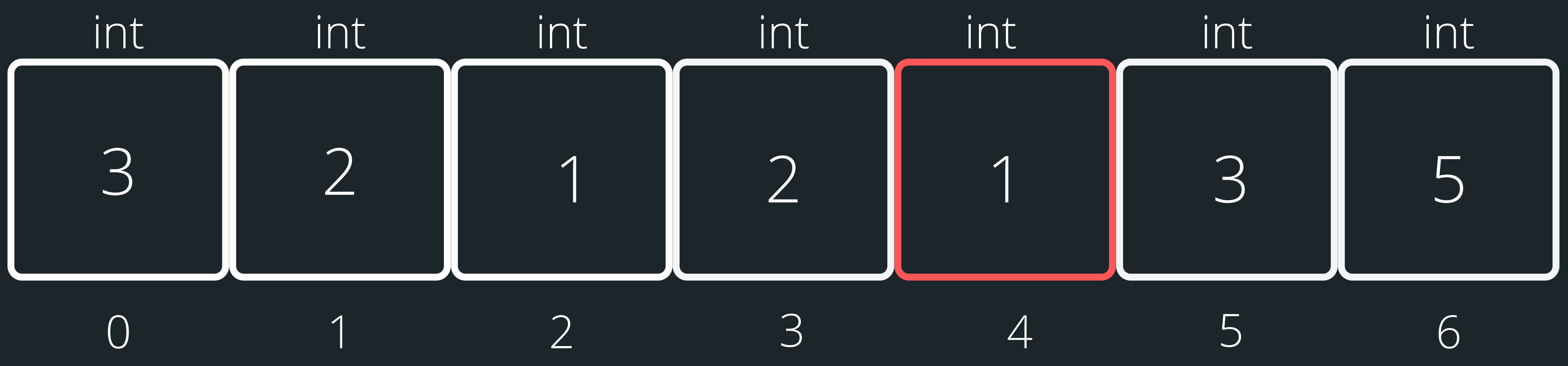
```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};

int i = 0;
while (i < 7){
    printf("%d ", ice cream_consum[i]);
    i++;
}
```

increase index by 1

ice_cream_consum[4]

print what is inside index 4



USING ARRAYS

CLOSER LOOK

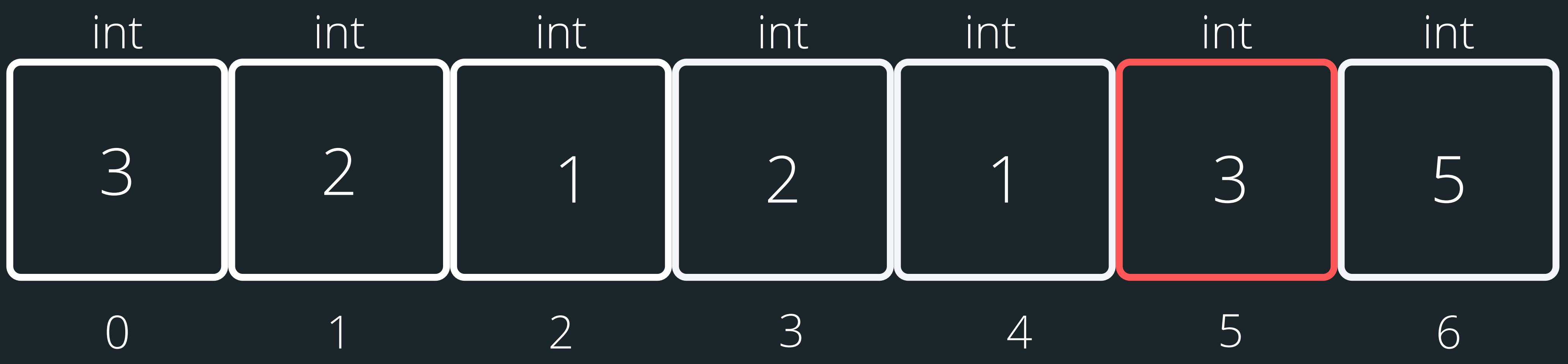
```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};

int i = 0;
while (i < 7){
    printf("%d ", ice cream_consum[i]);
    i++;
}
```

increase index by 1

ice_cream_consum[5]

print what is inside index 5



USING ARRAYS

CLOSER LOOK

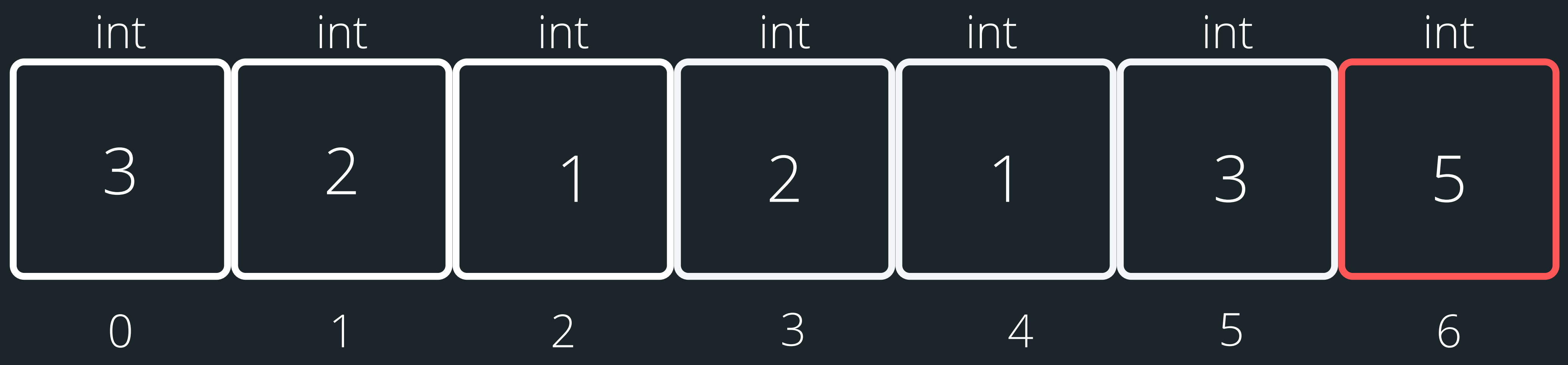
```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};

int i = 0;
while (i < 7){
    printf("%d ", ice cream_consum[i]);
    i++;
}
```

increase index by 1

ice_cream_consum[6]

print what is inside index 6



BREAK TIME

TIME TO STRETCH

You have two eggs in a 100-story building. You want to find out what floor the egg will break on, using the least number of drops.

PROBLEM SOLVING TIME

HOORAY!

- I meet my friend for ice cream every day for a week (I don't drink coffee)... We want to be able to track how many ice creams in total we all consumed in a week, and also who ate the most ice cream in that week!

`ice_cream_total.c`

Feedback please!

I value your feedback and use it to pace the lectures and improve your overall learning experience. If you have any feedback from today's lecture, please follow the link below. Please remember to keep your feedback constructive, so I can action it and improve the learning experience.

https://forms.microsoft.com/r/dKssTn3AU4

WHAT DID WE LEARN TODAY?

FUNCTIONS RECAP

functions_recap.c

EXPLORING ARRAYS

ice_cream.c

ice_cream_total.c

REACH OUT

CONTENT RELATED QUESTIONS

Check out the forum

ADMIN QUESTIONS

cs1511@cse.unsw.edu.au