NOVEMBER 4TH 2020

ELEMENTARY PROGRAMMING

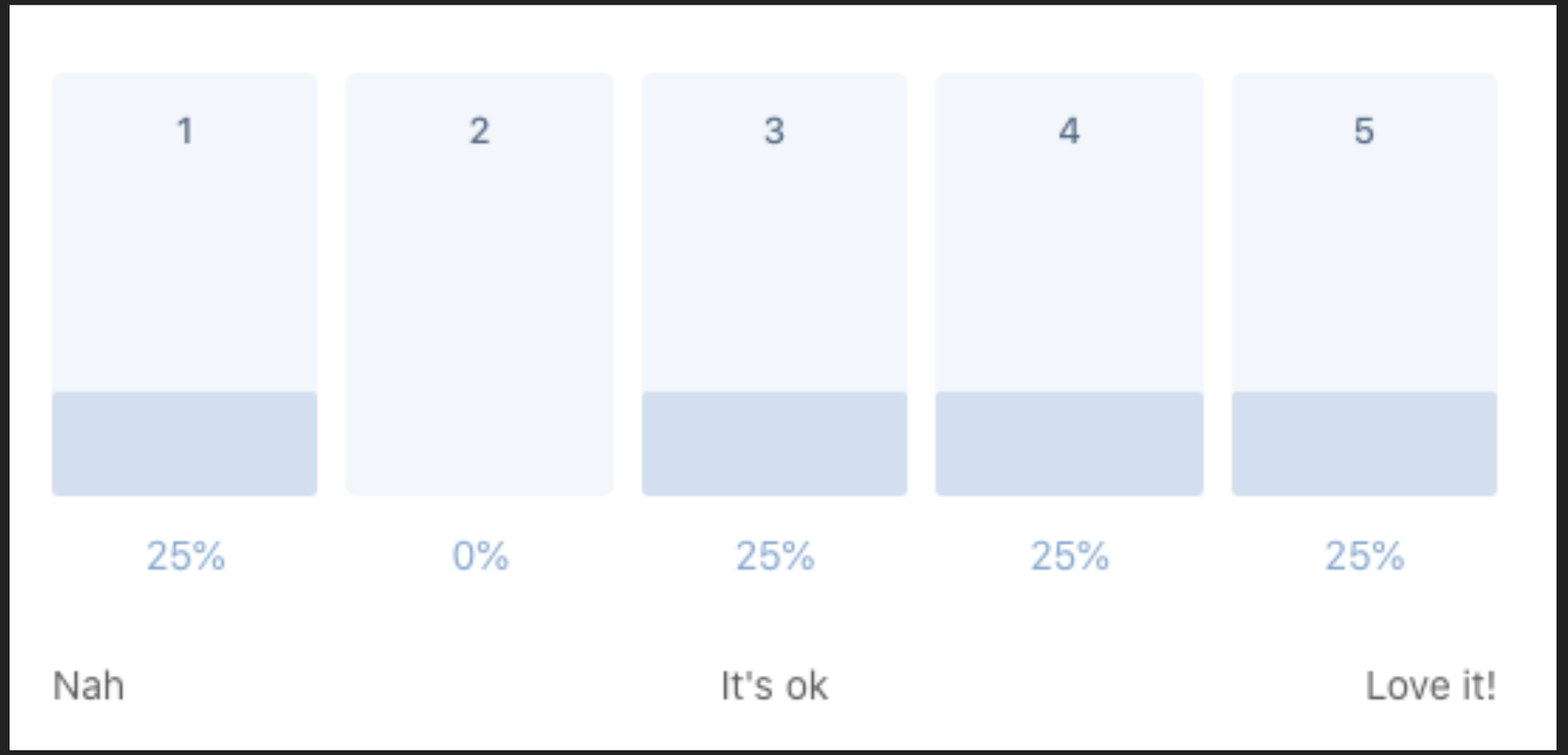
SOME COVID BEST PRACTICES BEFORE WE START

- If you fill ill, go home
- Keep your distance to others
- Wash or sanitise your hands
- Disinfect table and chair
- Respect guidelines and restrictions

REMEMBER TO BOOK YOUR SPOT TO DISCUSS THE ASSIGNMENT

- If you didn't receive my email please tell me I will resend the links
- You need to book an appointment otherwise no evaluation
  - Emanuele: **https://calendly.com/dierre/10min**
  - Alland: **https://calendly.com/a-kareem1991/02318_evaluering_1**
  - Patrick: **https://calendly.com/02318_opgave_eval_ph/10-min-eval**
  - Freja: **https://calendly.com/s200544/10-mins-samtale-om-aflevering**

FEEDBACK CHECK



NEW FEEDBACK

- I would really like for you to take a survey at the end of the session
- Feedback is important, please take the time to do it
- Pretty please <3
- Type this in your browser http://bit.ly/elemprog9

STRUCT

- A struct is composite data type that allows me to do pretty cool thing like model reality
- Being a composite data type means that I can use it to create new types

EXAMPLE OF A STRUCT (1)

```
typedef struct {
    char * name;
    int code;
    int    maxStudents;
} Course;
```

EXAMPLE OF A STRUCT (2)

```
struct CourseStruct {
    char *name;
    int   code;
    int   maxStudents;
};
typedef struct CourseStruct Course;
```

HOW TO INITIALIZE A STRUCT ON THE CALL STACK

```
Course c;
c.name       = "Something";
c.code       = 1234;
c.maxStudents = 4;
```

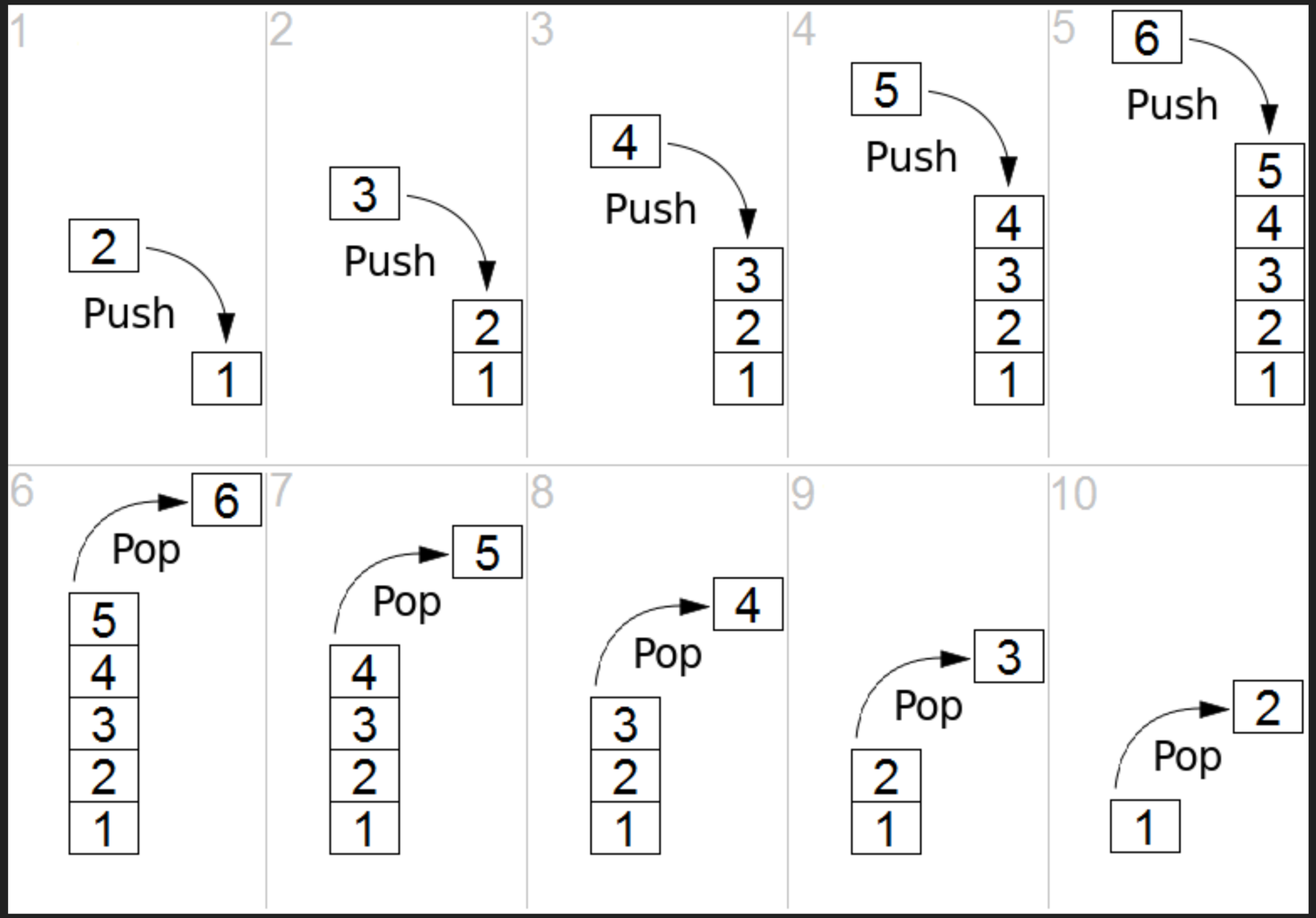
HOW TO INITIALIZE A STRUCT ON THE HEAP

```
Course* c = malloc(sizeof(Course));
    c→name        = “Something";
    c→code        = 1234;
    c→maxStudents = 4;
```

LET'S IMPLEMENT A STACK DATA STRUCTURE IN C

- It's a data structure
- It's a collection of elements
- It follows a Last In First Out policy (LIFO)
- It has two functions:
  - Push => add an element in the stack
  - Pop => remove an element from the top of the stack

STACK


1
2
Push
1
2
3
Push
2
1
3
4
Push
3
2
1
4
5
Push
4
3
2
1
5
6
Push
5
4
3
2
1
6
6
Pop
5
4
3
2
1
7
5
Pop
4
3
2
1
8
4
Pop
3
2
1
9
3
Pop
2
1
10
2
Pop
1

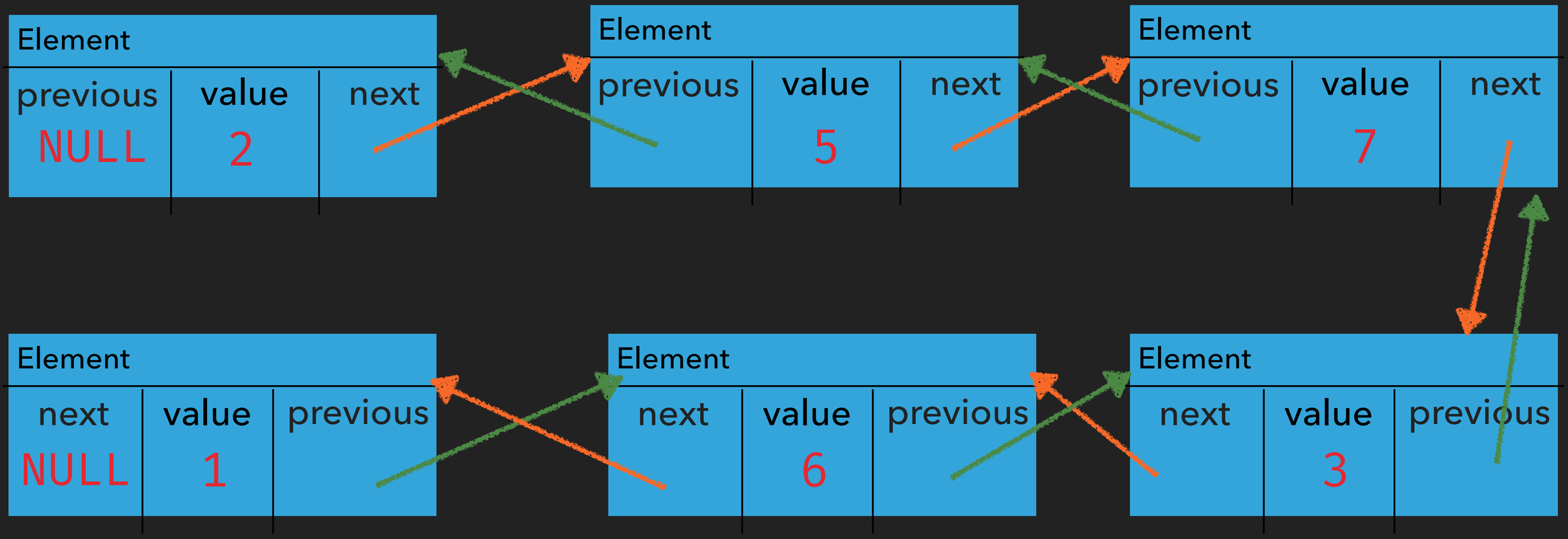
LET'S DISCUSS WHAT WE NEED

- We would like for the data structure to grow dynamically
- We would like to avoid to decide the size before
- We want to have push function
- We want to have a pop function

WE WOULD LIKE FOR THE DATA STRUCTURE TO GROW DYNAMICALLY

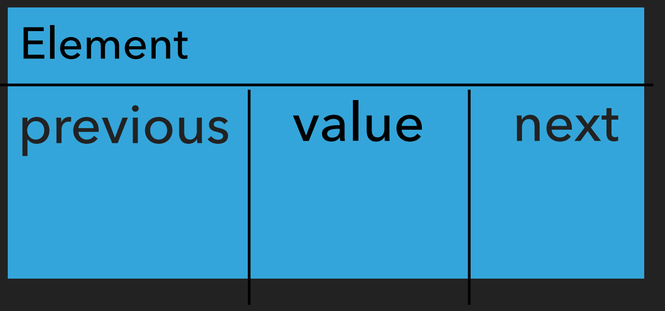
- We can use pointers so that we can use the heap memory
- Every element of the stack has a pointer to next and previous element
- We can represent an element as a struct

WE WOULD LIKE FOR THE DATA STRUCTURE TO GROW DYNAMICALLY



WE WOULD LIKE FOR THE DATA STRUCTURE TO GROW DYNAMICALLY

```
struct Element {
    int             value;
    struct Element *next;
    struct Element *previous;
};
typedef struct Element Element;
```



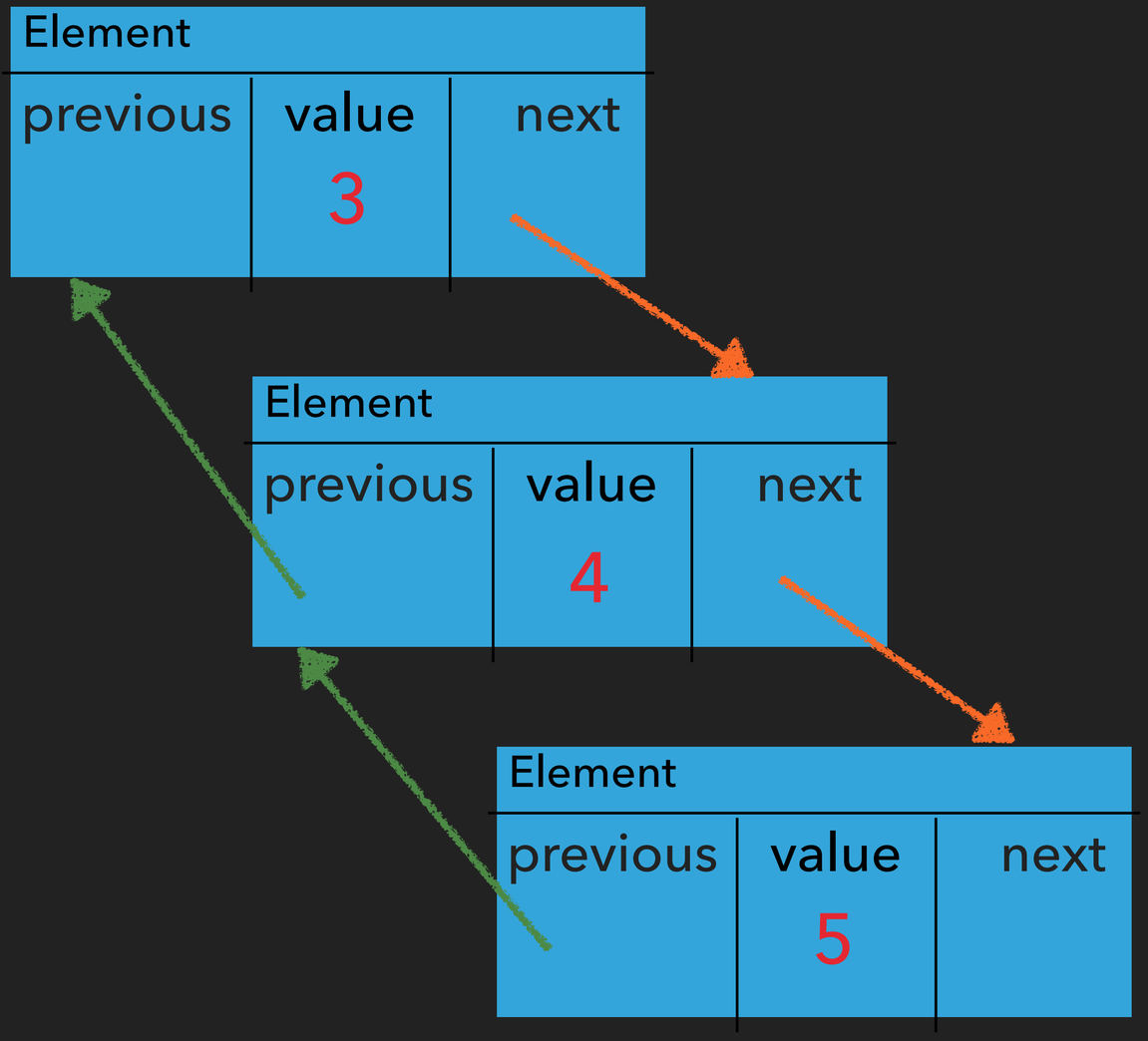
This is weird but if you want to use the struct inside itself you need to use struct keyword. It’s a limitation of the language

WE WOULD LIKE FOR THE DATA STRUCTURE TO GROW DYNAMICALLY

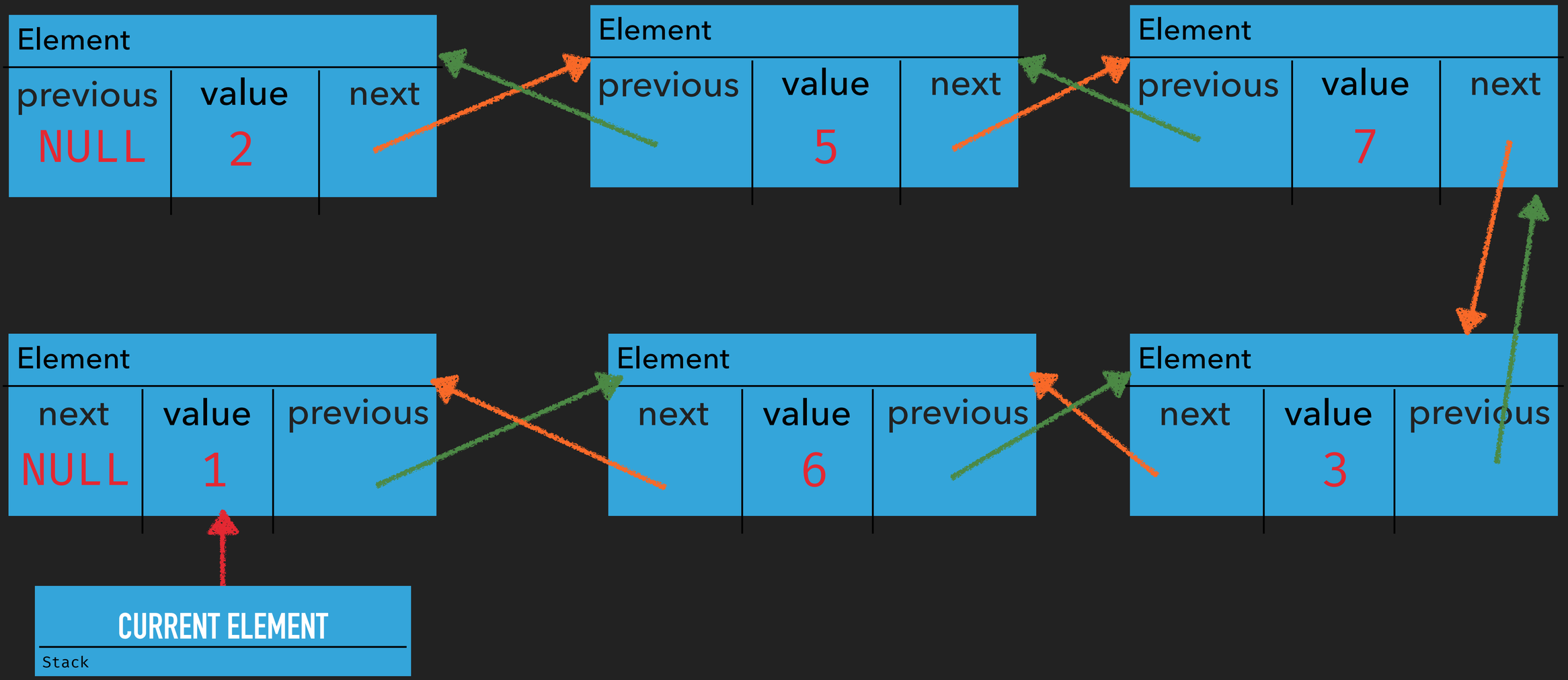
```
Element *createElement(int n) {
    Element *Element = malloc(sizeof(Element));
    if (Element == NULL) {
        printf("Something wrong while creating a new Element\n");
        exit(EXIT_FAILURE);
    }
    Element→value = n; // same as (*element).value = n;
    return Element;
}
```

WE WOULD LIKE FOR THE DATA STRUCTURE TO GROW DYNAMICALLY

```
Element* a = createElement(4);
Element* b = createElement(5);
Element* c = createElement(3);
a→next = b;
b→previous = a;
a→previous = c;
c→next = a;
```



WE WANT TO HAVE PUSH AND POP FUNCTION



WE WANT TO HAVE PUSH AND POP FUNCTION

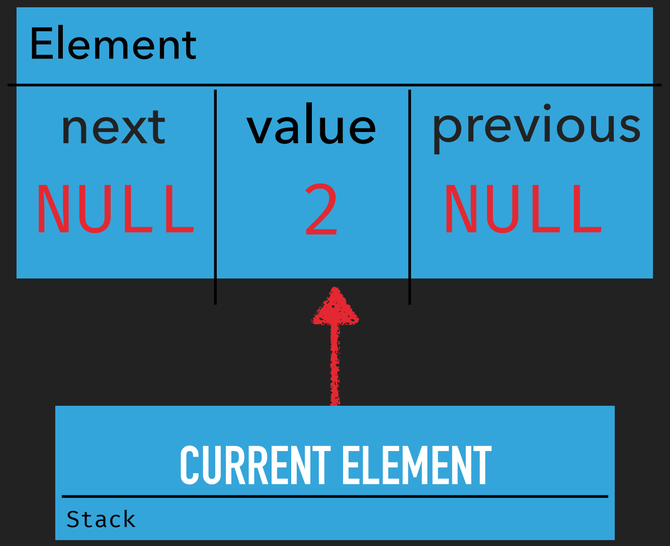
```
typedef struct {
    struct Element *currentElement;
} Stack;
```

WE WANT TO HAVE PUSH AND POP FUNCTION

```
Stack *createStack(int n) {
    Element * Element  = createElement(n);
    Stack *stack = malloc(sizeof(Stack));
    if (stack == NULL) {
        printf("Something wrong while creating a new stack\n");
        exit(EXIT_FAILURE);
    }
   stack->currentElement = Element;
   return stack;
}
```

WE WANT TO HAVE PUSH AND POP FUNCTION

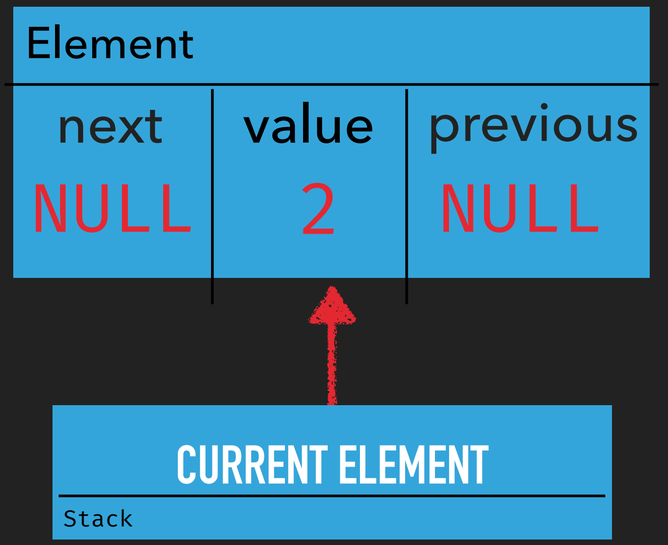
```
Stack *stack = createStack(2);
```



THE PUSH FUNCTION

```
Element *push(Stack *stack, int n) {.
    Element *newElement              = createElement(n);
    stack→currentElement→next = newElement;
    newElement→previous         = stack→currentElement;
    stack→currentElement        = newElement;
    return newElement;
}
```

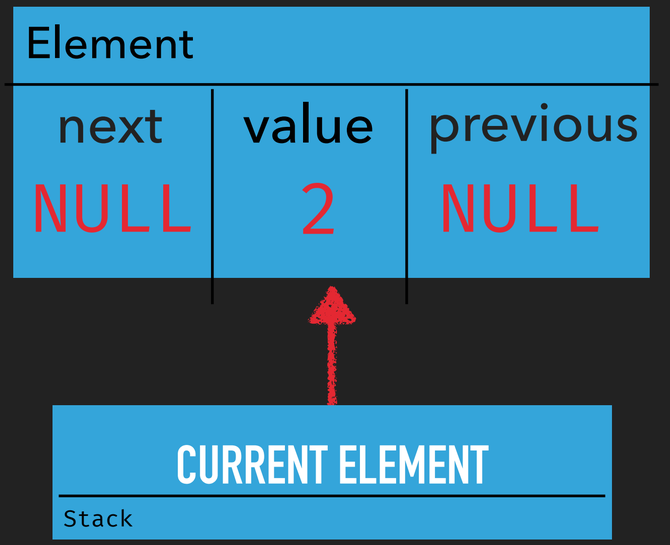
Current State



THE PUSH FUNCTION

```
push(stack, 5);
```

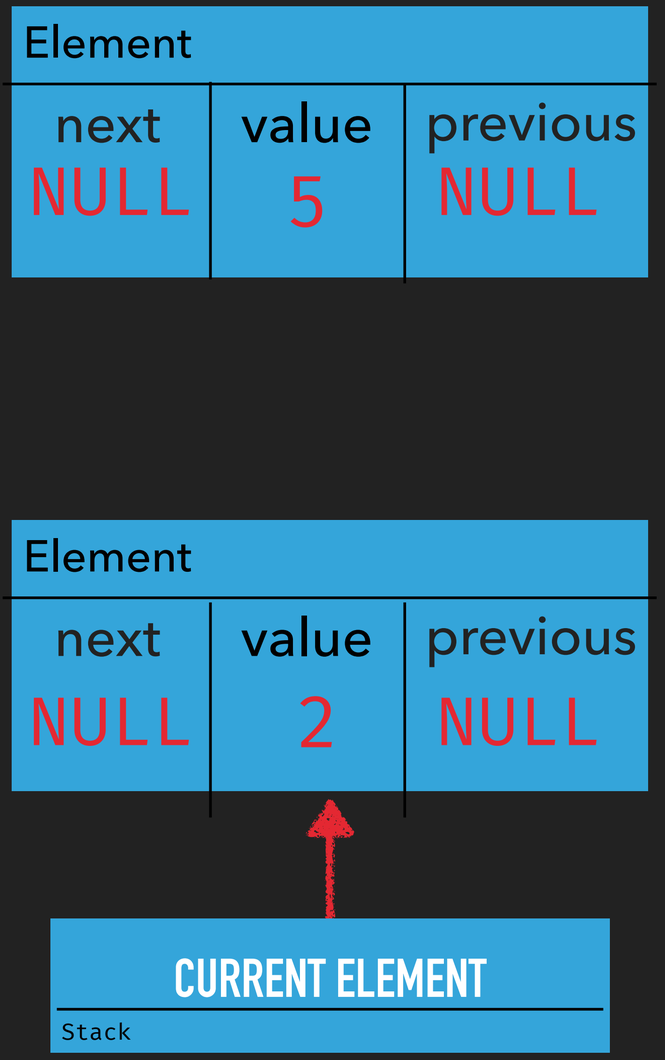
Current State



INSIDE THE PUSH FUNCTION

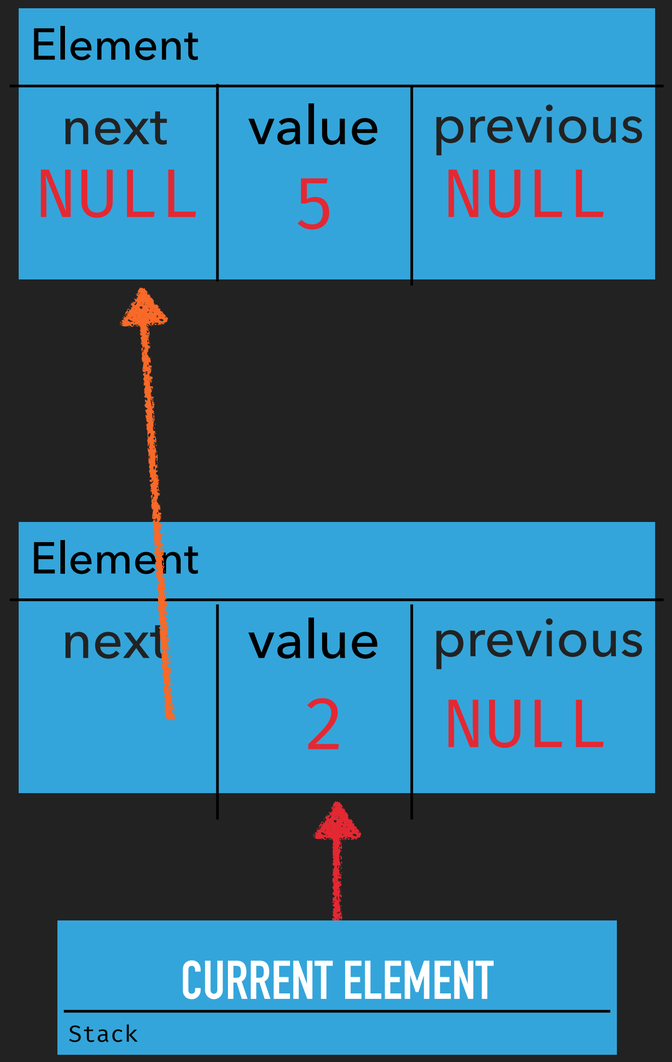
```
Element *push(Stack *stack, int n) {.
    Element *newElement              = createElement(n);
    stack→currentElement→next = newElement;
    newElement→previous         = stack→currentElement;
    stack→currentElement        = newElement;
    return newElement;
}
```

Current State



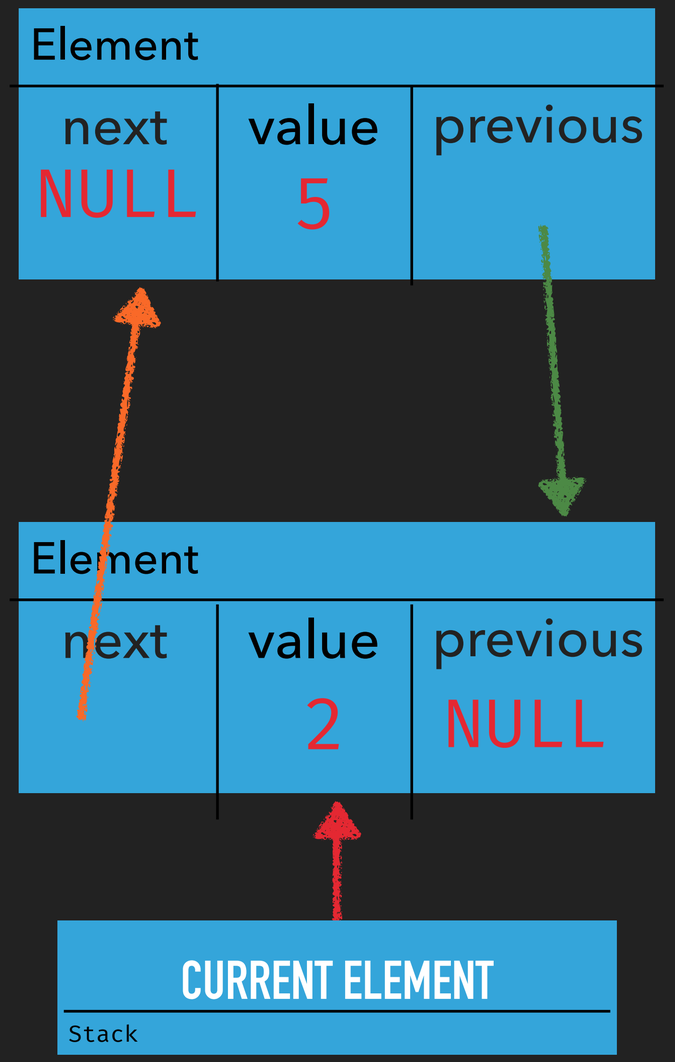
INSIDE THE PUSH FUNCTION

```
Element *push(Stack *stack, int n) {.
    Element *newElement              = createElement(n);
    stack→currentElement→next = newElement;
    newElement→previous         = stack→currentElement;
    stack→currentElement        = newElement;
    return newElement;
}
```



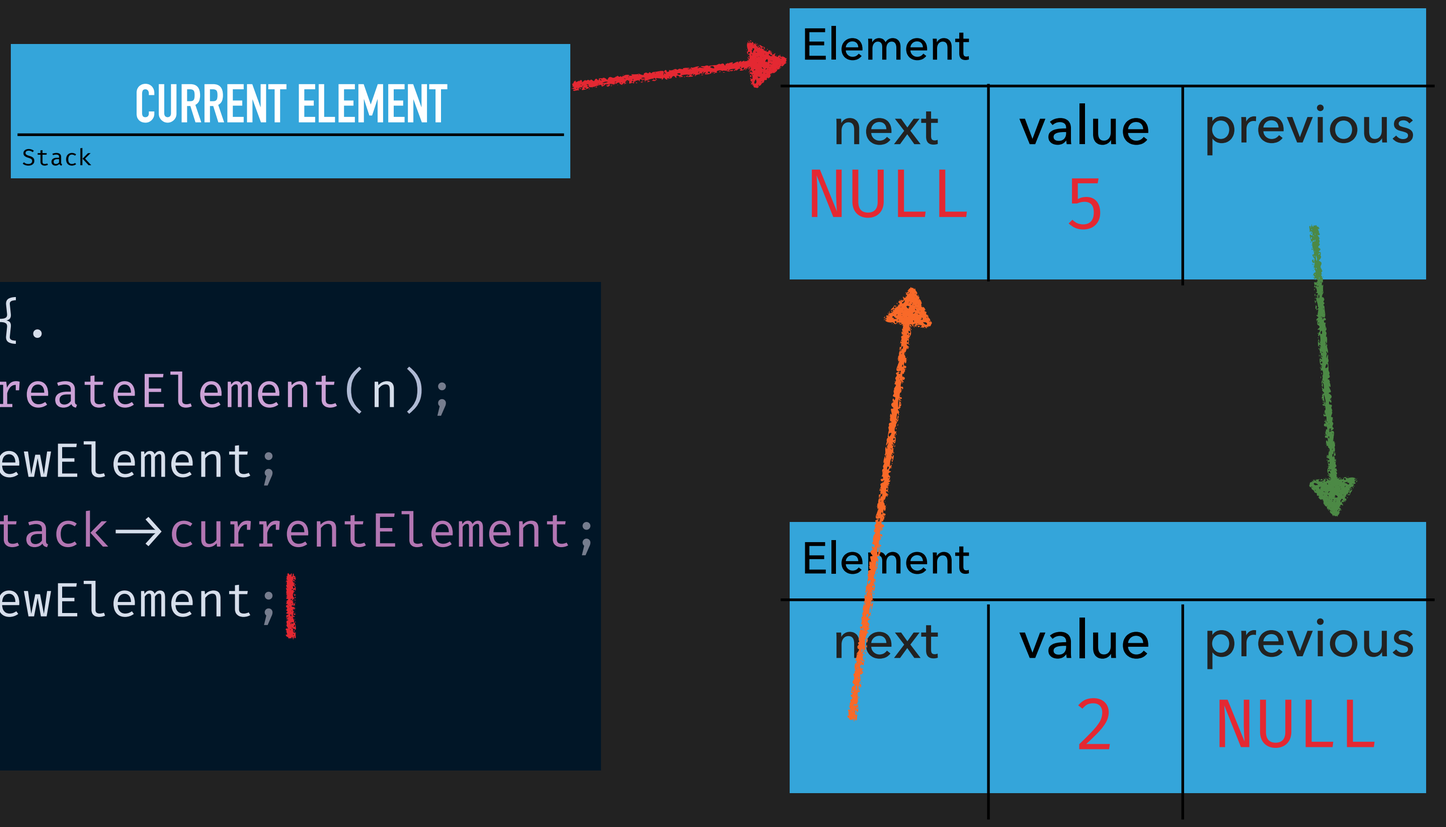
INSIDE THE PUSH FUNCTION

```
Element *push(Stack *stack, int n) {.
    Element *newElement              = createElement(n);
    stack→currentElement→next = newElement;
    newElement→previous         = stack→currentElement;
    stack→currentElement        = newElement;
    return newElement;
}
```



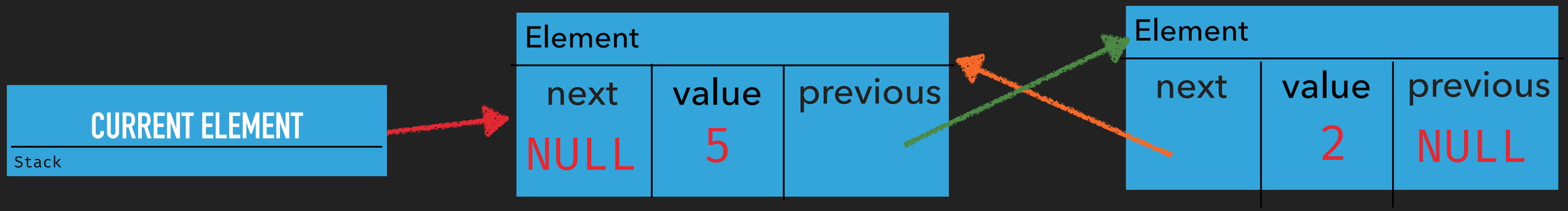
INSIDE THE PUSH FUNCTION

```
Element *push(Stack *stack, int n) {.
    Element *newElement           = createElement(n);
    stack→currentElement→next = newElement;
    newElement→previous           = stack→currentElement;
    stack→currentElement          = newElement;
    return newElement;
}
```



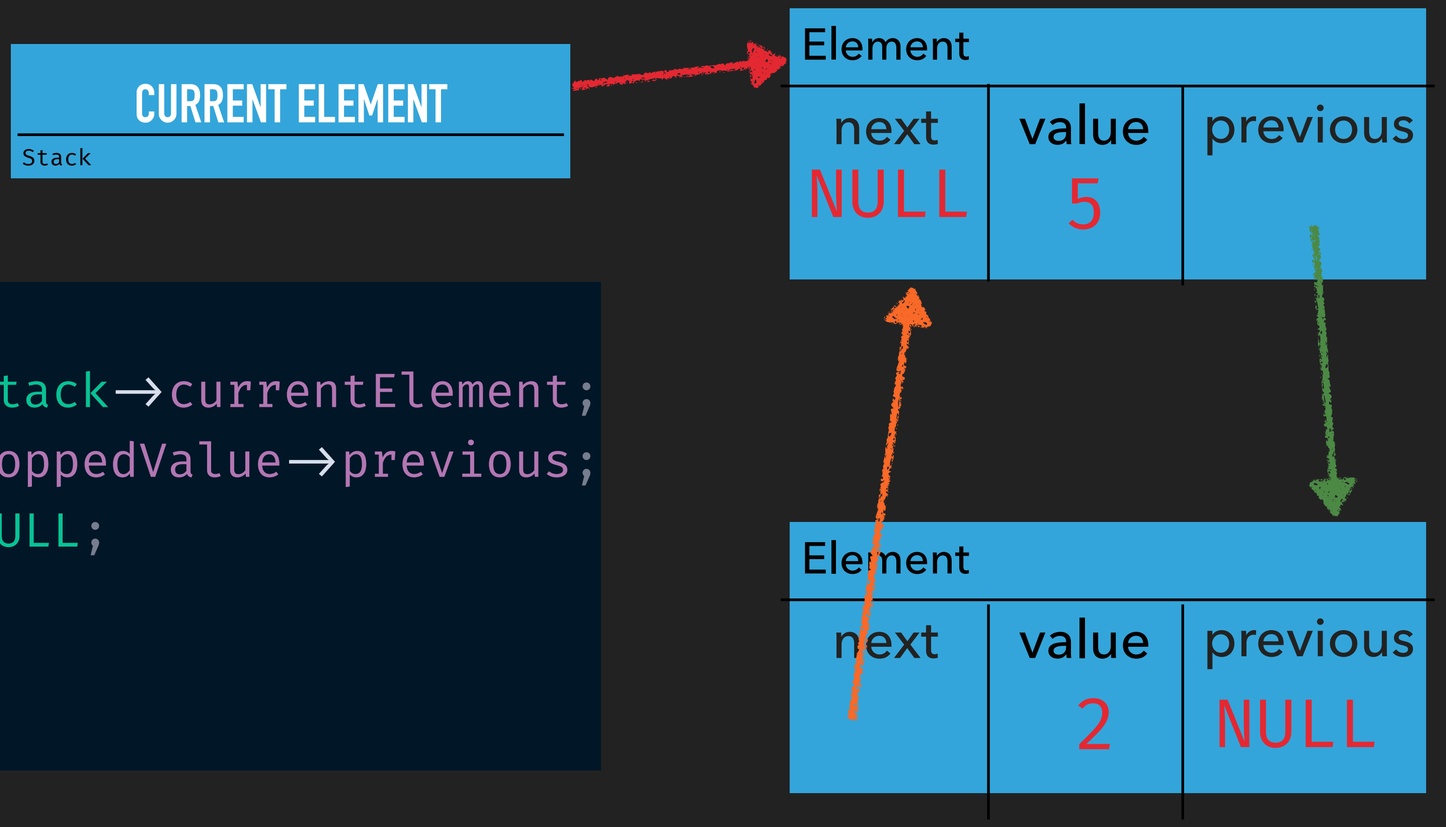
CURRENT STATE

Current State

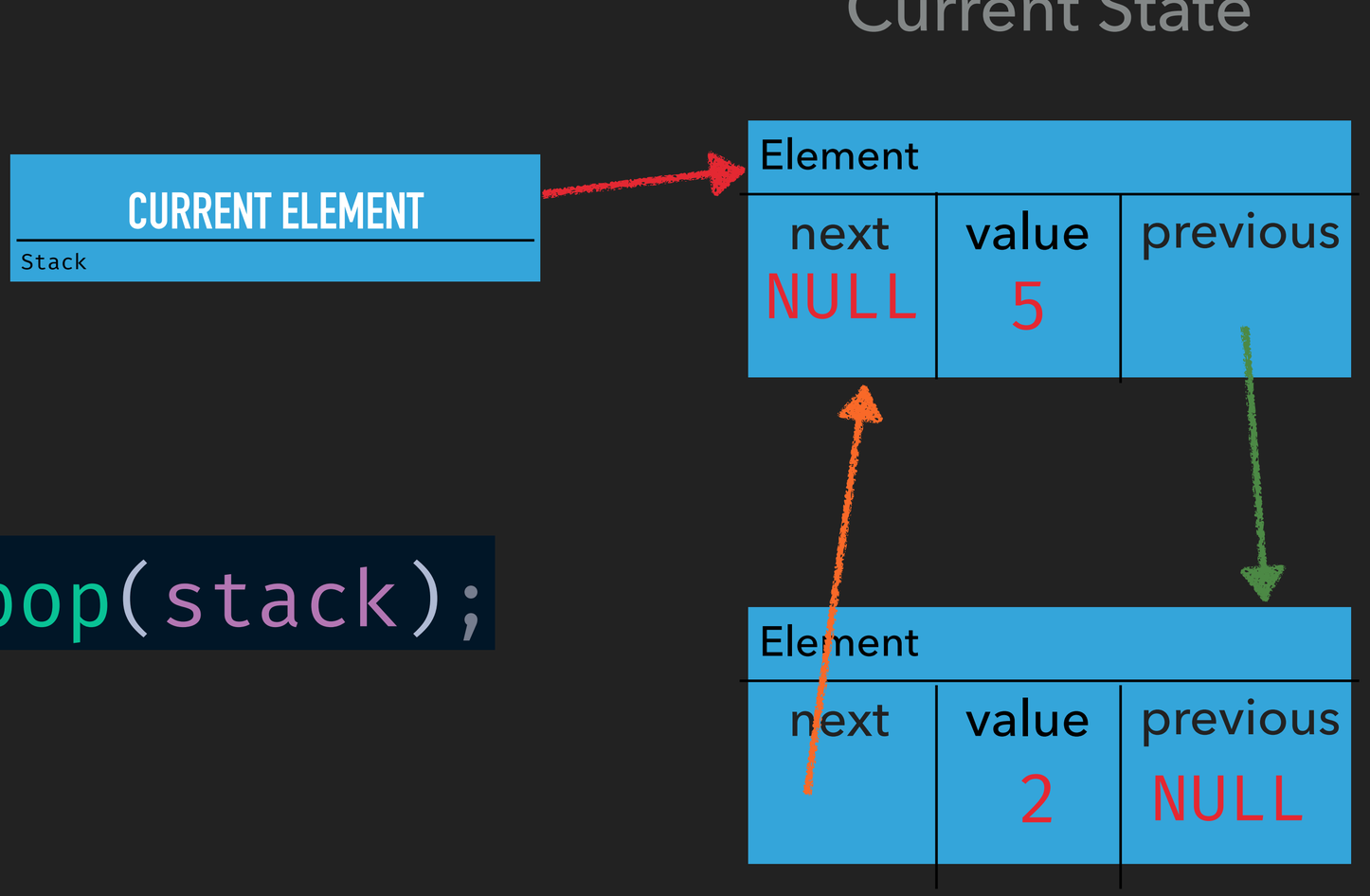


THE POP FUNCTION

```
Element *pop(Stack *stack) {
    Element *poppedValue          = stack→currentElement;
    stack→currentElement          = poppedValue→previous;
    stack→currentElement→next = NULL;
    poppedValue→previous = NULL;
    return poppedValue;
}
```



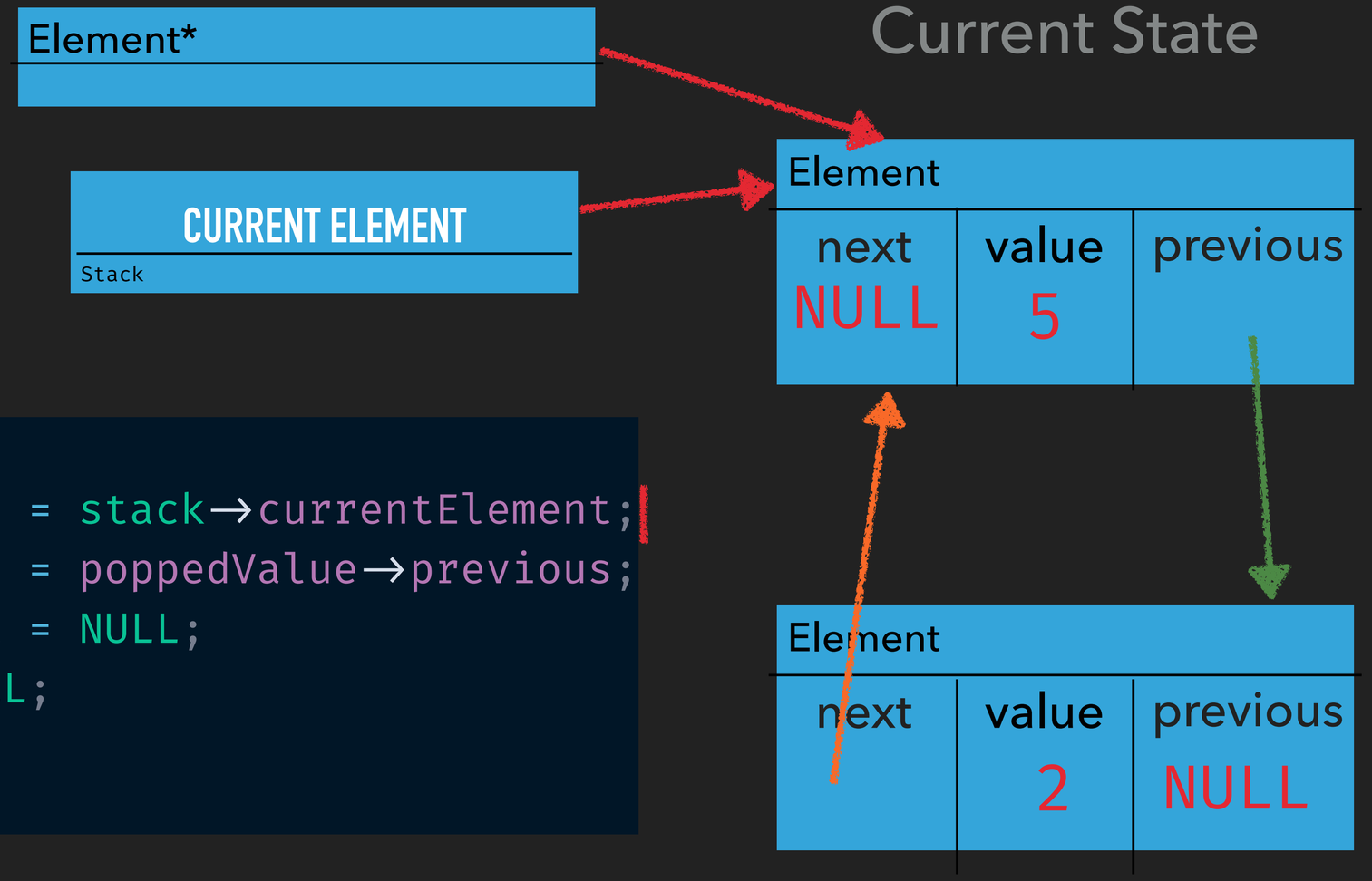
THE POP FUNCTION




```
Element* Element = pop(stack);
```

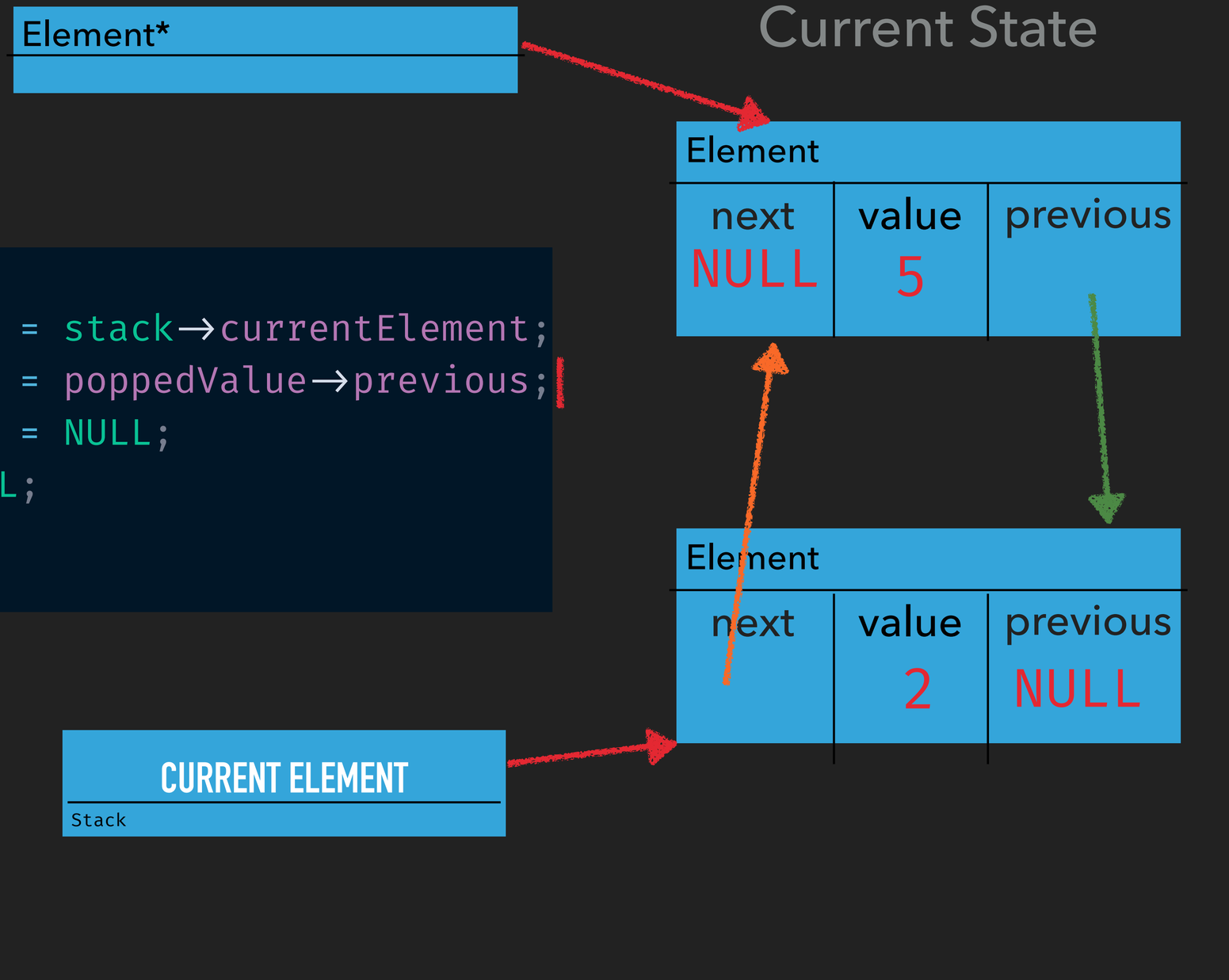
INSIDE THE POP FUNCTION

```
Element *pop(Stack *stack) {
    Element *poppedValue         = stack->currentElement;
    stack->currentElement        = poppedValue->previous;
    stack->currentElement->next = NULL;
    poppedValue->previous = NULL;
    return poppedValue;
}
```



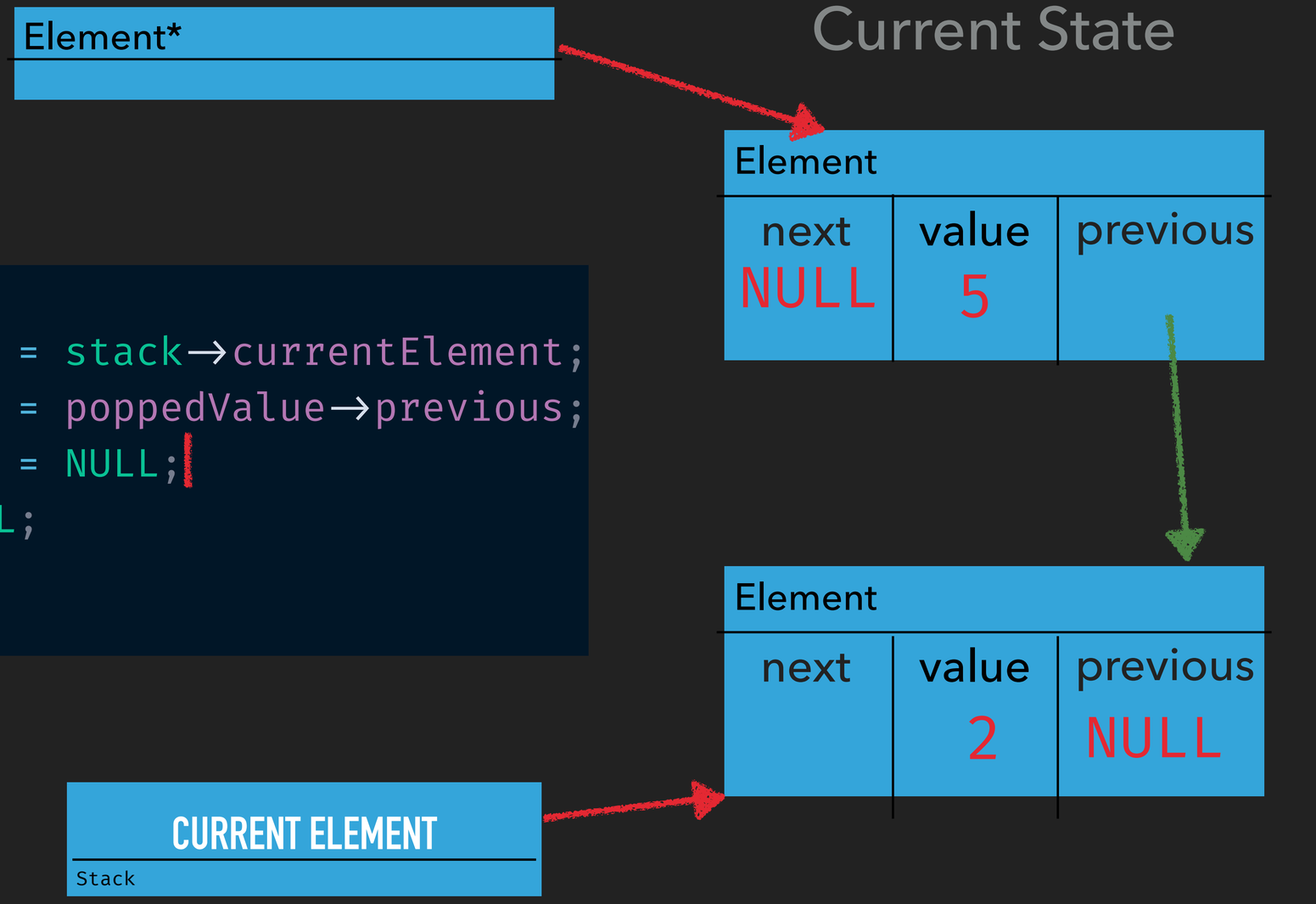
INSIDE THE POP FUNCTION

```
Element *pop(Stack *stack) {
    Element *poppedValue          = stack→currentElement;
    stack→currentElement          = poppedValue→previous;
    stack→currentElement→next = NULL;
    poppedValue→previous = NULL;
    return poppedValue;
}
```



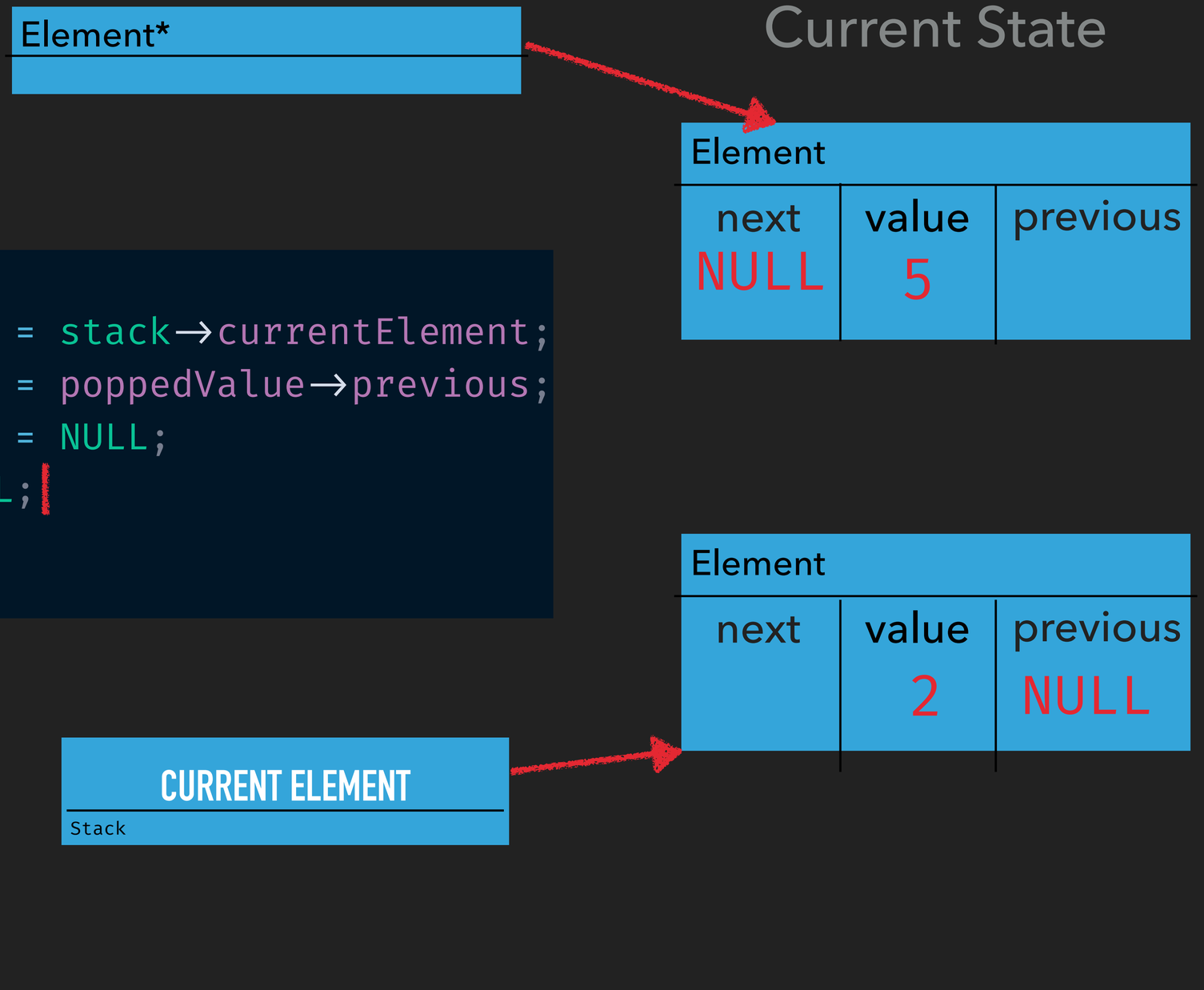
INSIDE THE POP FUNCTION

```
Element *pop(Stack *stack) {
    Element *poppedValue          = stack->currentElement;
    stack->currentElement         = poppedValue->previous;
    stack->currentElement->next = NULL;
    poppedValue->previous = NULL;
    return poppedValue;
}
```



INSIDE THE POP FUNCTION

```
Element *pop(Stack *stack) {
    Element *poppedValue          = stack->currentElement;
    stack->currentElement         = poppedValue->previous;
    stack->currentElement->next = NULL;
    poppedValue->previous = NULL;
    return poppedValue;
}
```



EDGE CASE FOR POP FUNCTION (WHAT IF STACK IS EMPTY?)

```
Element *pop(Stack *stack) {
    if(stack->currentElement == NULL) {
      return NULL;
    }
    Element *poppedValue          = stack->currentElement;
    stack->currentElement         = poppedValue->previous;
    stack->currentElement->next = NULL;
    poppedValue->previous = NULL;
    return poppedValue;
}
```

SECOND CODING ASSIGNMENT

- Opens at 10AM, November 4th 2020
- Closes at 8AM, November 18th 2020
- Here's the link: https://github.com/invasionofsmallcubes/elementary-programming-dtu/blob/master/assignments/assignment02/ASSIGNMENT.MD

NEW FEEDBACK

- I would really like for you to take a survey at the end of the session
- Feedback is important, please take the time to do it
- Pretty please <3
- Type this in your browser http://bit.ly/elemprog9

SOME COVID BEST PRACTICES BEFORE WE LEAVE

- Disinfect table and chair
- Maintain your distance to others
- Wash or sanitise your hands
- Respect guidelines and restrictions outside