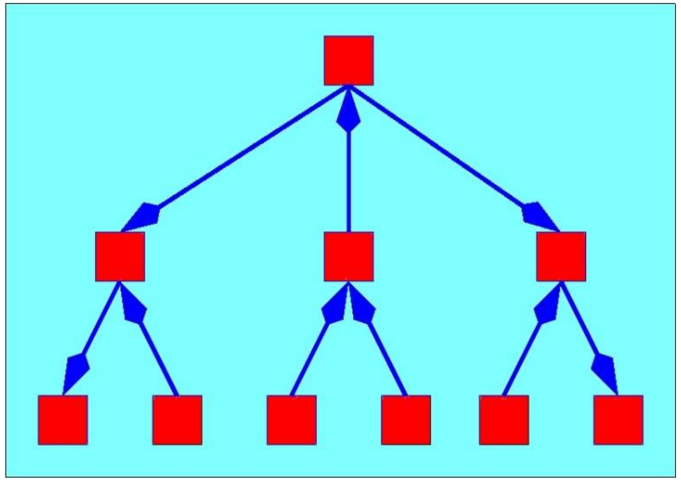
Data Structures

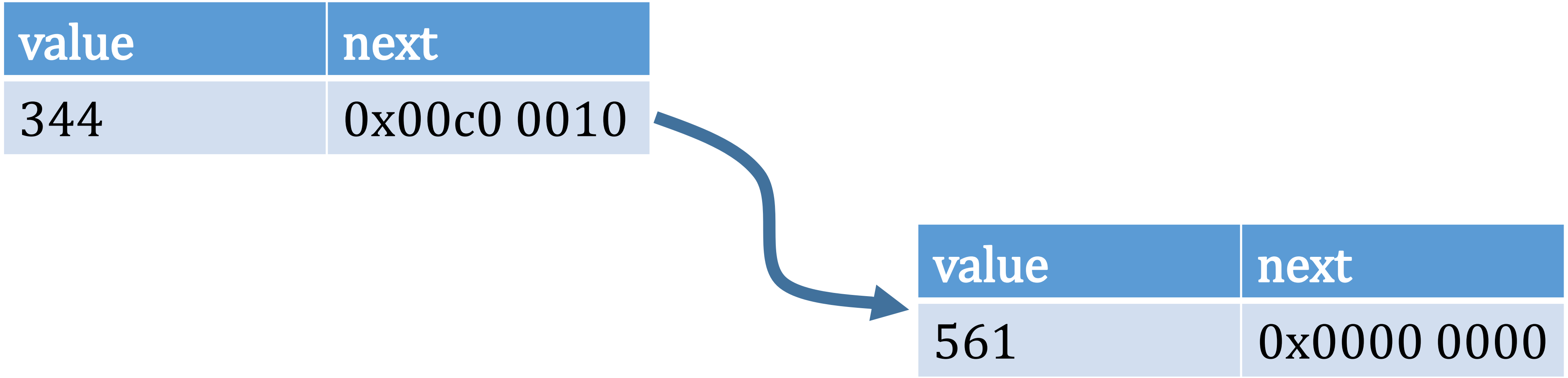


Using Pointers to Link Structures

- There is an exception to the “define before used” rule.
- You may reference a structure in its own definition.
- It is common to use pointers to other instances of the same structure

Example of a linked list Node structure

```
struct node {
        int value;
        struct node *next;
};
```



value	next
344	0x00c0 0010

value	next
561	0x0000 0000

Linked List

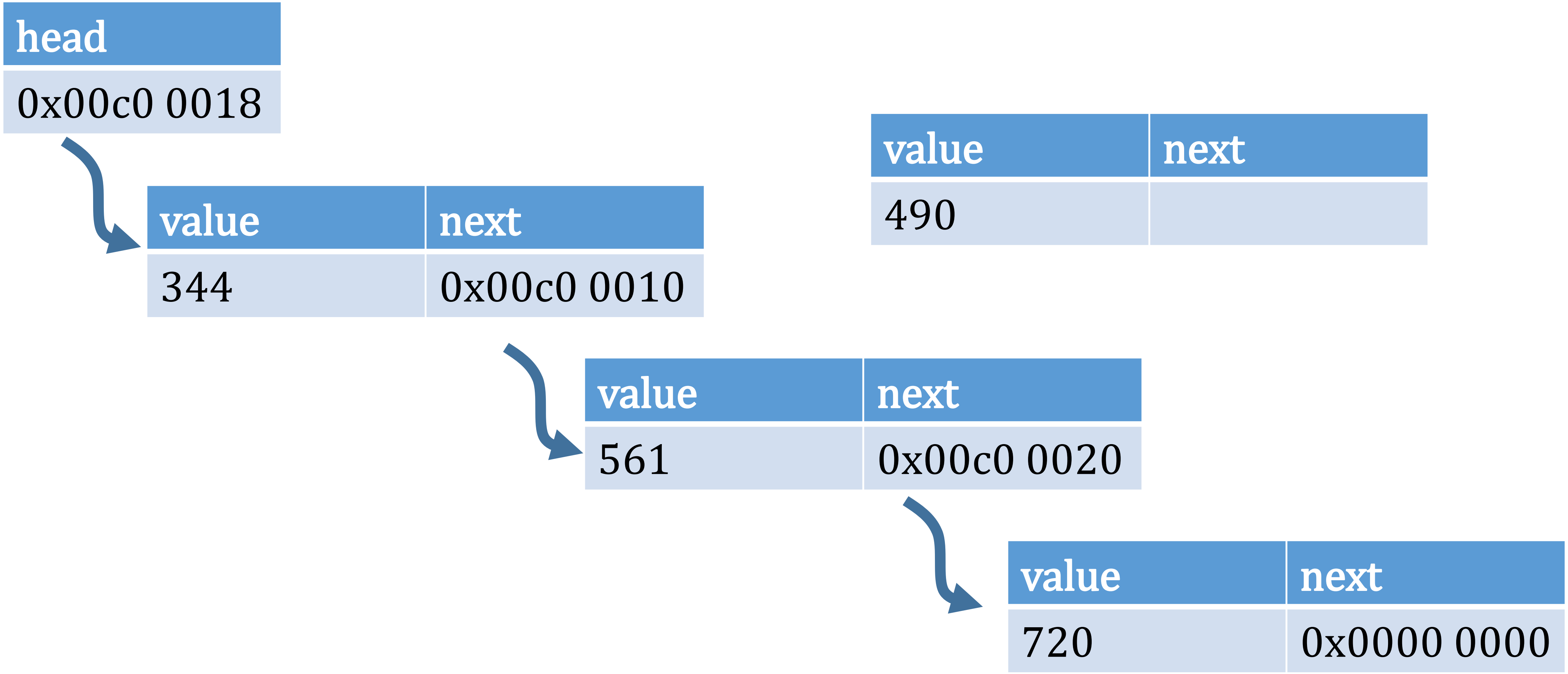
- List of nodes
- Each node has a value or payload
- Each node has a "next" pointer
- First node in the list is the **head** node
  - Special variable "head" points to the first node

  ```
  struct node *head;
  ```
- Last node is the **tail** node
  - Tail node "next" pointer is NULL (0x0000 0000)
- Empty list when head==NULL

Why a Linked List?

- It’s easy to insert in a linked list
- For example, suppose I want to insert a node with the value 490 between 344 and 561…

Example Linked List



head
0x00c0 0018

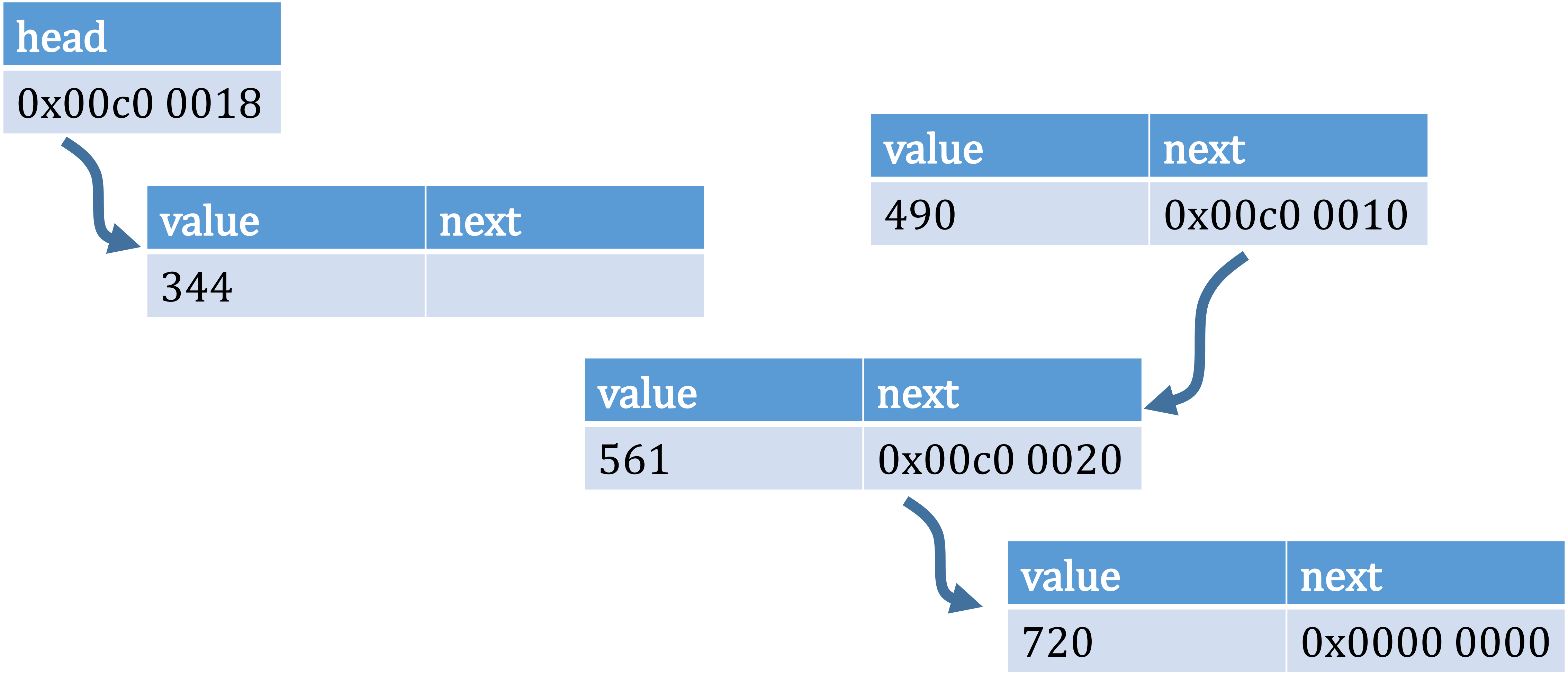
value	next
344	0x00c0 0010

value	next
490	

value	next
561	0x00c0 0020

value	next
720	0x0000 0000

Example Linked List



head

0x00c0 0018

value | next

344 |

value | next

490 | 0x00c0 0010

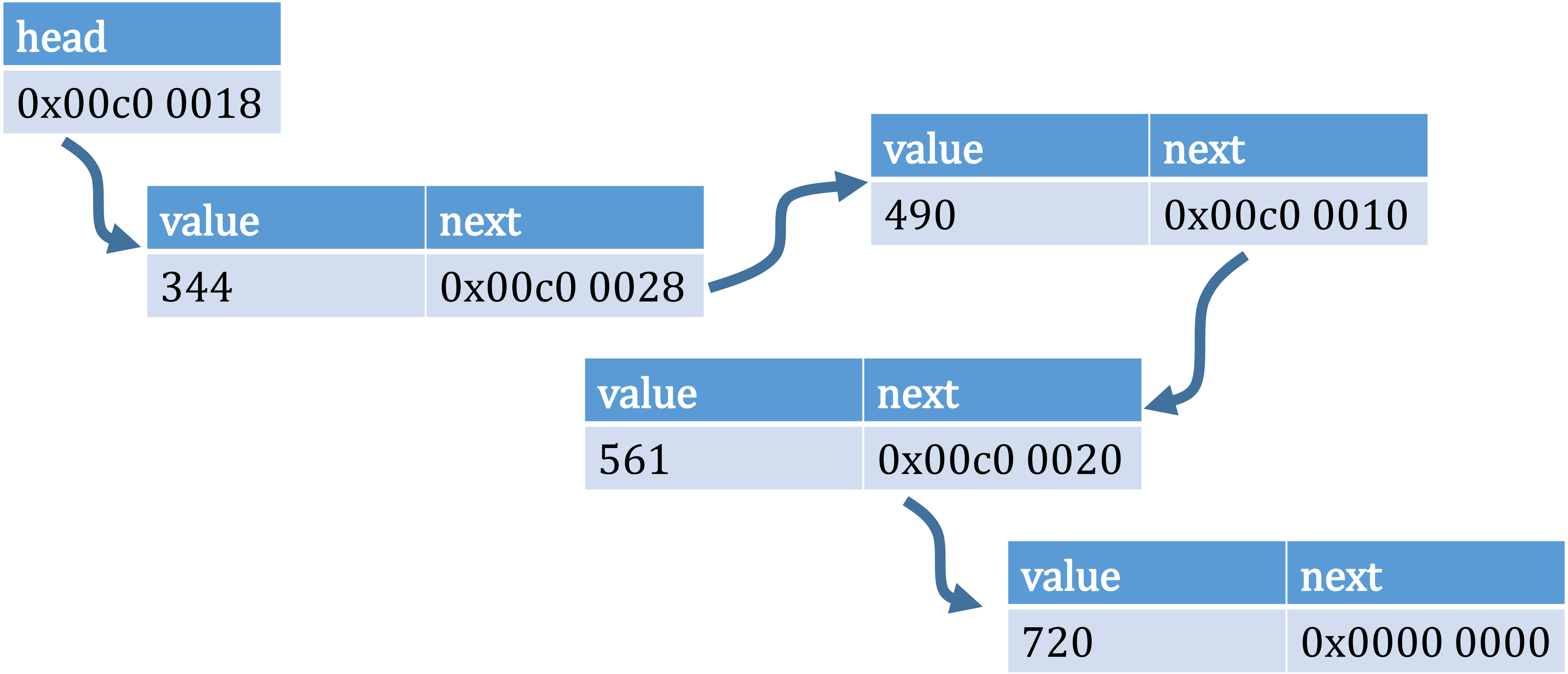
value | next

561 | 0x00c0 0020

value | next

720 | 0x0000 0000

Example Linked List



head
0x00c0 0018

value	next
344	0x00c0 0028

value	next
490	0x00c0 0010

value	next
561	0x00c0 0020

value	next
720	0x0000 0000

Example Insertion Function

```
void insertNode(struct node *after, struct node*new) {
        new->next=after->next;
        after->next=new;
}
```

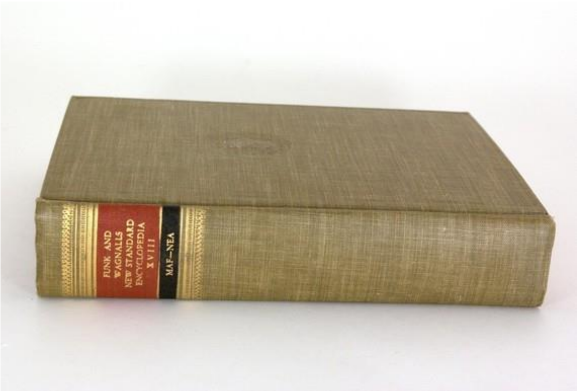
Example of Insertion to Vector

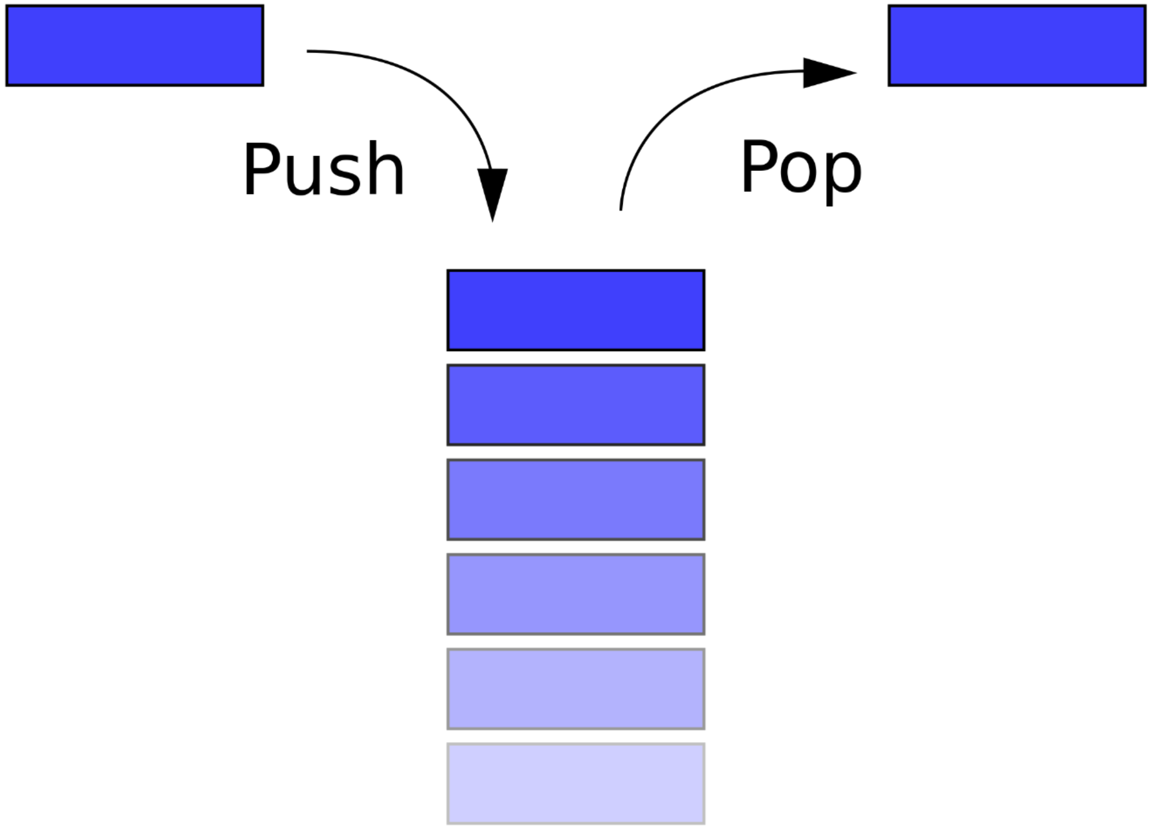
```
void insertVector(int vec[],int new,int position, int num) {
        int j;
        for(j=num; j>pos; j−−) {
                vec[j]=vec[j−1]; // move everything over 1
        }
        vec[pos]=new;
}
```

Dynamic Node Allocation/Free

```
struct node * makeNode(int value) {
        struct node * np=
                (struct node *)malloc(sizeof(struct node));
        np->value=value;
        np->next=NULL;
        return np;
}
void freeNode(struct node * np) { free(np); }
```

The Stack


FUNK AND WAGNALLS NEW STANDARD ENCYCLOPEDIA
XVIII
MAF–NEA


Push
Pop

Implementing a Stack with a Linked List

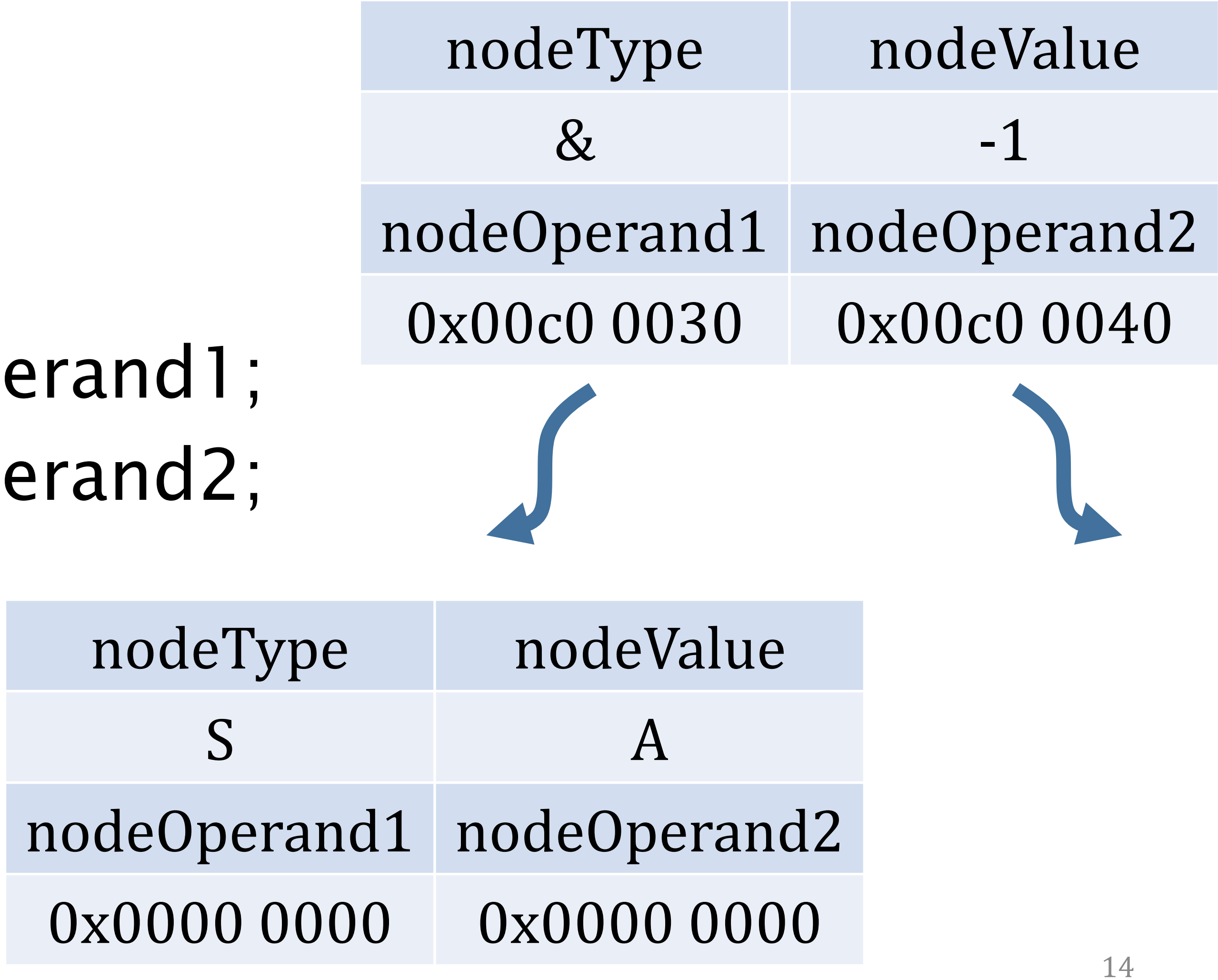
```
struct node *head=NULL;

void push(int value) {
        struct node *np=
                makeNode(value);
        np->next=head;
        head=np;
}
```

```
int pop() {
        assert(head);
        struct node *np=head;
        head=np->next;
        int val=np->value;
        freeNode(np);
        return val;
}
```

Tree Data Structure

```
struct tnode {
        char nodeType;
        char nodeValue;
        struct tnode *nodeOperand1;
        struct tnode *nodeOperand2;
};
```



nodeType	nodeValue
&	-1
nodeOperand1	nodeOperand2
0x00c0 0030	0x00c0 0040

nodeType	nodeValue
S	A
nodeOperand1	nodeOperand2
0x0000 0000	0x0000 0000

And so on…

- The possibilities are almost endless
  - Doubly linked lists
  - Circularly linked lists
  - Directed Graphs with Nodes/Vertices
  - Trees with “n” branches (multi-way trees)

- All possible because of self-referential pointers!

Resources

- Wikipedia Linked List https://en.wikipedia.org/wiki/Linked_list
- Wikipedial Data Structure https://en.wikipedia.org/wiki/Data_structure
- Linked List Tutorial http://www.learn-c.org/en/Linked_lists

Pop Quiz 3

1. The C expression “int *xyz;” results in an xyz that can be thought of as (choose all that apply)…
   a) A pointer to a string
   b) A pointer to a single character
   c) A pointer to a floating point value
   d) A pointer to a vector of characters
   e) None of the above
2. After the expression “int nums[4]={10,11,12,13};”, what is the value of nums[2]?