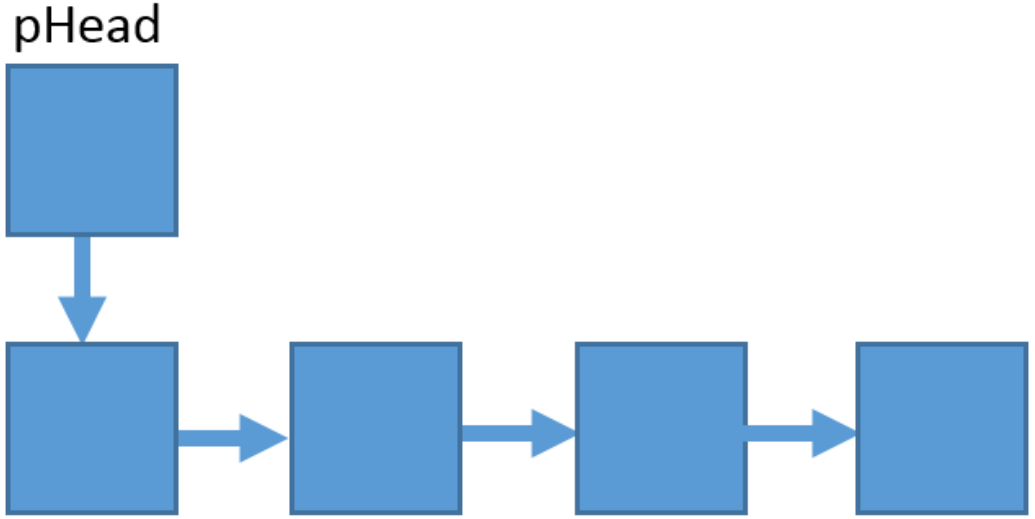
Linked list

Basics

- Each node is allocated separately as needed
- Each node is independent and only linked to others using pointers
- Typically the node that has no pointer to the next element is considered to be the end of the list (the pNext pointer on that node is valued NULL)



Compared: data creation, allocation

- Fixed size array
  - Size is determined with variable declaration
  - Elements are sequential in memory
  - Fixed size per record
- Dynamic memory allocation
  - Size can be modified during runtime
  - Elements are sequential in memory
  - Fixed size per record
- Linked list
  - Each node is given memory independently
  - Elements can be located anywhere in the memory
  - Record size is not fixed and can contain different data

Compared: removing a record

- Fixed size array
  - Memory cannot be freed
  - Removing an element is inconvenient and demanding
- Dynamic memory allocation
  - Excess memory can be freed at any time
  - Removing an element is inconvenient and demanding
- Linked list
  - Nodes can be removed at will
  - Removing an element is fast and simple

- *There's also a time factor to find the node that you need removed*

Data structures

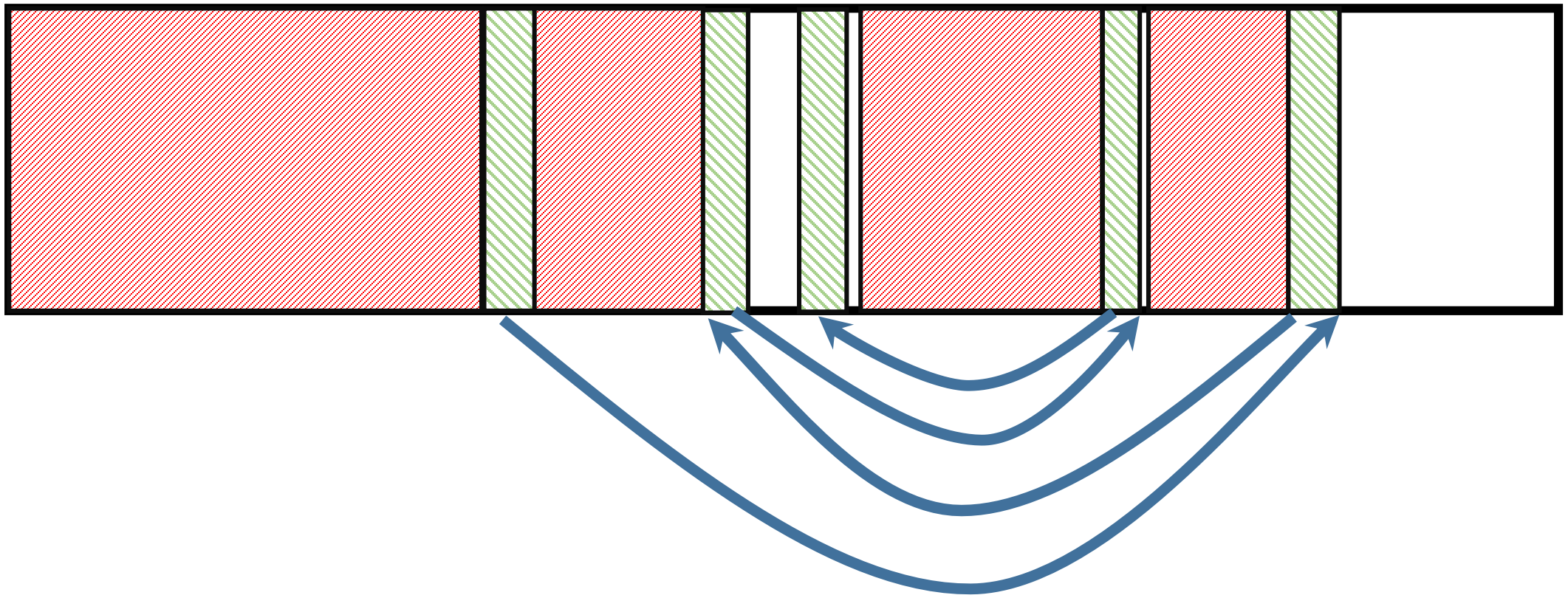
- Linked list based stack

```
typedef struct node {
    int num;
    struct node *pNext;
} stack;
```

- Linked list

```
typedef struct node {
    int employeeCode;
    float hourlyPay;
    char *name;
    DATE *dateBorn;
    struct node *pNext;
} list;
```

Memory



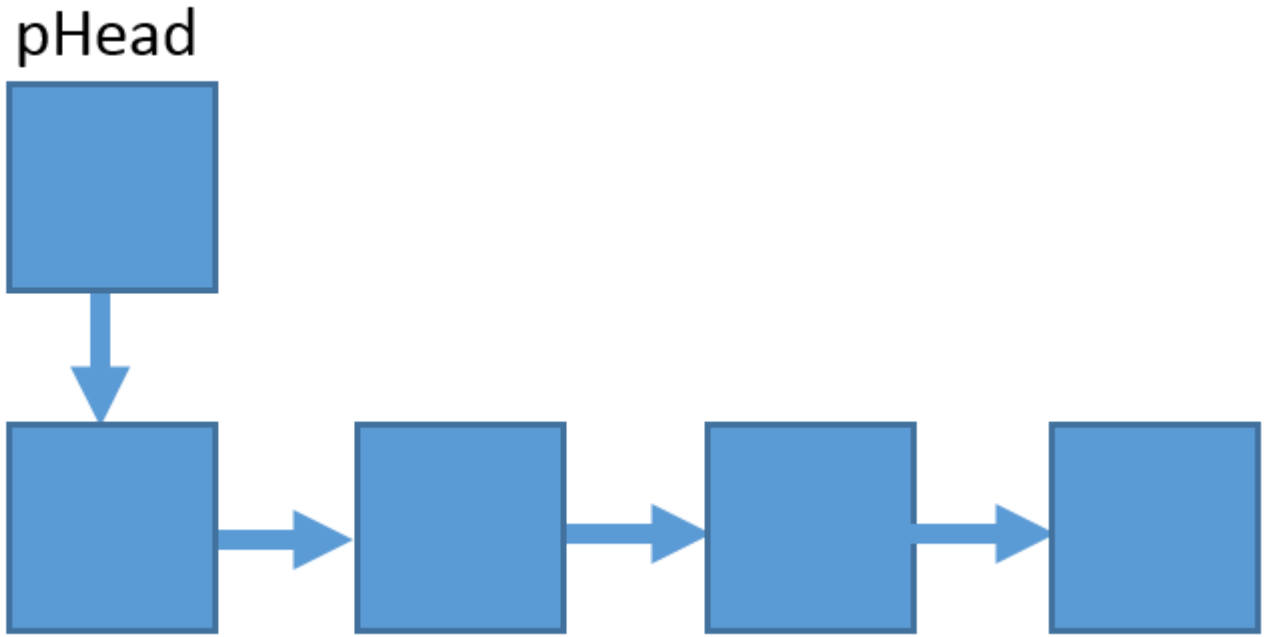
Advantages of linked list

- Memory fragmentation won’t cause issues on large datasets
- Optimal use of memory – we can easily allocate and free nodes at runtime
- We can create more advanced data structures using additional pointers to make processes much faster
- Many data structures are created based on this kind of linking (stack, circular buffers, queues, trees e.t.c.)

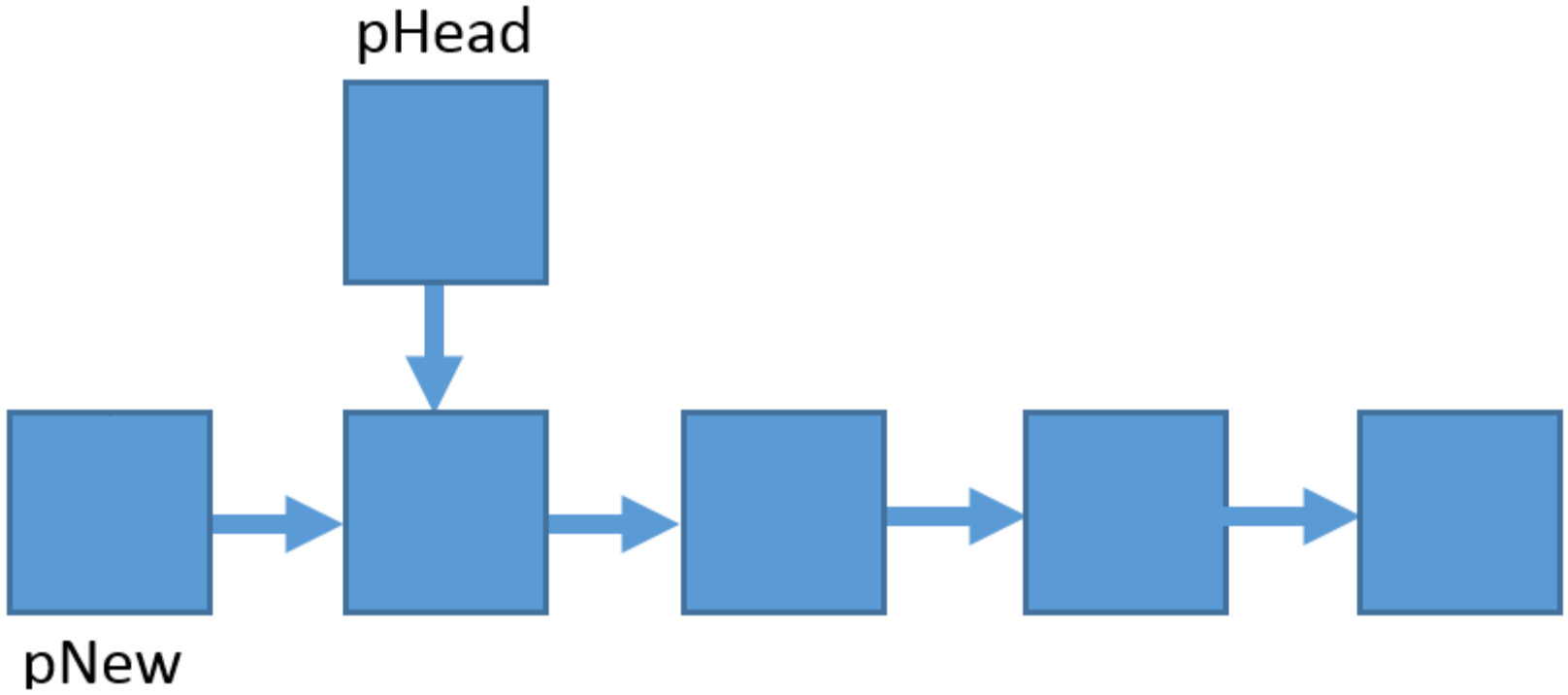
Disadvantages of singly linked list

- Linear access – finding an element in a singly linked list is slow
- Larger overhead – the pointers need additional memory. In many cases there isn't just one pointer
- Even moving through the list backwards needs additional pointers
- Not all devices support dynamic memory allocation and linked lists
- No random access

Adding to the front of the list (1) initial state



Adding to the front of the list (2) new element



Adding to the front of the list (3) make the header point to the new node