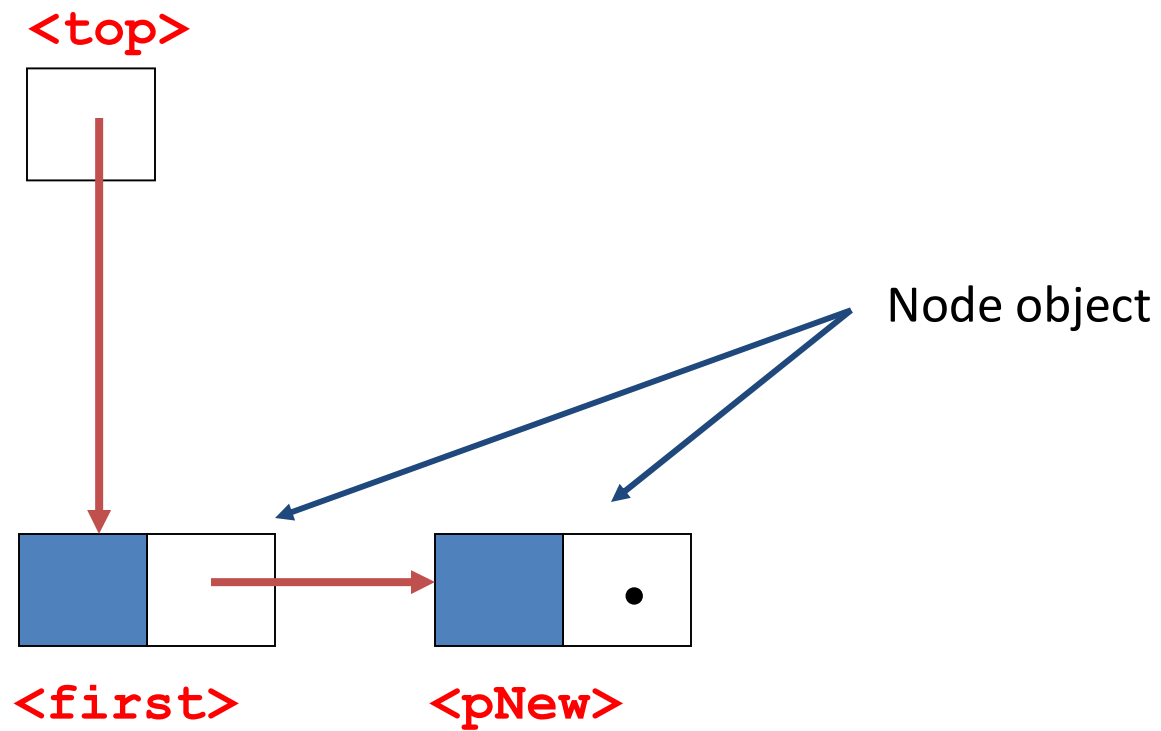


CS 2211

Systems Programming

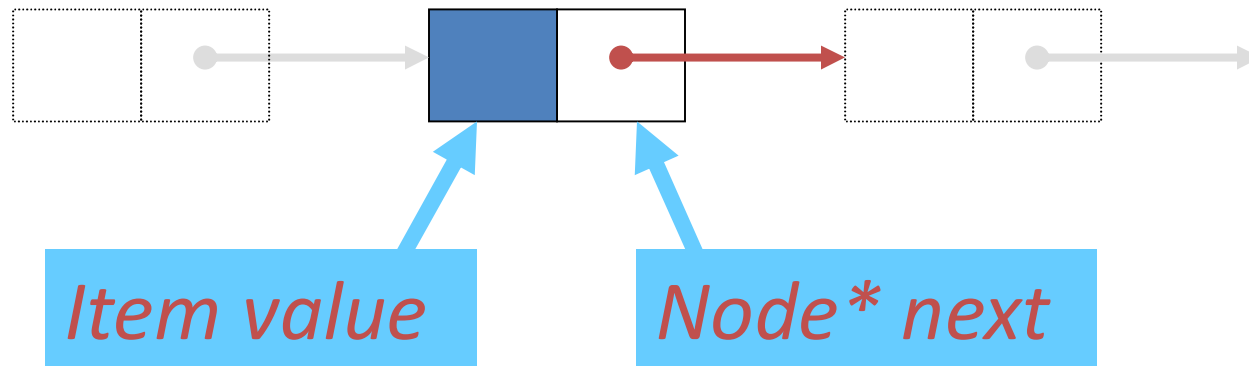
Part Eleven: Linked Lists

Linked List

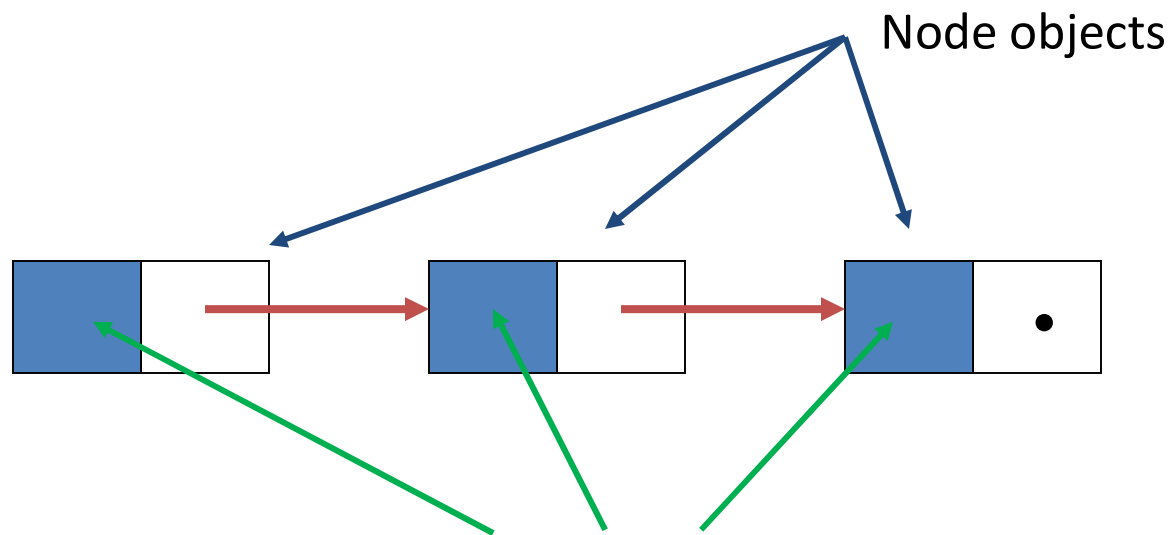


Nodes in Singly Linked Lists

- Nodes for our linked lists will be objects dynamically created or deleted in the HEAP
- Each *Node* object in a singly linked list will contain two member variables:



Linked List



Data objects can be simple data (*int*, *double*, etc) or complex data (arrays, structures or unions) or pointers to data outside each individual nodes

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int    wP;
    int    fP;

} fract;
```

| Address | Value |
|---|-------|
| 7 - | 398 - |
| 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | 399 - |
| 0 - | 401 - |
| 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | 402 - |
| 403 - | 404 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | 405 - |
| 406 - | 407 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | 408 - |
| 409 - | 410 - |
| 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | 411 - |
| 412 - | 413 - |
| 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | 414 - |
| 415 - | 416 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | 417 - |
| 418 - | 419 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | 420 - |
| 421 - | 422 - |
| | 423 - |

[illegible]

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int    wP;
    int    fP;
    struct fraction* link;
} fract;
```

| Address | Value |
|---|-------|
| 7 - | 398 - |
| 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | 399 - |
| 0 - | 401 - |
| 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | 402 - |
| 403 - | 404 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | 405 - |
| 406 - | 407 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | 408 - |
| 409 - | 410 - |
| 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | 411 - |
| 412 - | 413 - |
| 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | 414 - |
| 415 - | 416 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | 417 - |
| 418 - | 419 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | 420 - |
| 421 - | 422 - |
| | 423 - |

[illegible]

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {  
    int  wP;  
    int  fP;  
    struct fraction* link;  
} fract;  
  
fract first *pNew;
```

| | | |
|---|-------|-------|
| 397 - | 398 - | 399 - |
| 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | | |
| 400 - | 401 - | 402 - |
| 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | | |
| 403 - | 404 - | 405 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | | |
| 406 - | 407 - | 408 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | | |
| 409 - | 410 - | 411 - |
| 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | | |
| 412 - | 413 - | 414 - |
| 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | | |
| 415 - | 416 - | 417 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | | |
| 418 - | 419 - | 420 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | | |
| 421 - | 422 - | 423 - |

| Address Label | Label | Address | Value |
|---------------|------------|-----------|-------|
| | first.wP | 400 - 403 | |
| | first.fP | 404 - 407 | |
| | first.link | 408 - 411 | |
| | pNew | 412 - 415 | |
| | | | |
| | | | |
| | | | |
| | | | |

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {  
    int  wP;  
    int  fP;  
    struct fraction* link;  
} fract;
```

```
fract first, *pNew;
```

```
first.wP = 7;  
first.fP = 3;
```

| | | |
|---|-------|-------|
| 397 - | 398 - | 399 - |
| 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | | |
| 400 - | 401 - | 402 - |
| 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | | |
| 403 - | 404 - | 405 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | | |
| 406 - | 407 - | 408 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | | |
| 409 - | 410 - | 411 - |
| 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | | |
| 412 - | 413 - | 414 - |
| 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | | |
| 415 - | 416 - | 417 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | | |
| 418 - | 419 - | 420 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | | |
| 421 - | 422 - | 423 - |

| Address Label | Label | Address | Value |
|---------------|------------|-----------|-------|
| | first.wP | 400 - 403 | 7 |
| | first.fP | 404 - 407 | 3 |
| | first.link | 408 - 411 | |
| | pNew | 412 - 415 | |
| | | | |
| | | | |
| | | | |
| | | | |

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
} fract;

fract first, *pNew;

first.wP = 7;
first.fP = 3;

pNew = (fract*)malloc(sizeof (fract));
```

| | | |
|---|-------|-------|
| 7 - | 398 - | 399 - |
| 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | | |
| 0 - | 401 - | 402 - |
| 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | | |
| 403 - | 404 - | 405 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | | |
| 406 - | 407 - | 408 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | | |
| 409 - | 410 - | 411 - |
| 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 | | |
| 412 - | 413 - | 414 - |
| 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 | | |
| 415 - | 416 - | 417 - |
| 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 | | |
| 418 - | 419 - | 420 - |
| 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 | | |
| 421 - | 422 - | 423 - |

| | Address | Value |
|--|---------------|-------|
| | 400 - 403 | 7 |
| | 404 - 407 | 3 |
| | 408 - 411 | |
| | 412 - 415 | 10100 |
| | 10100 - 10111 | |
| | | |
| | | |
| | | |

| | |
|--|------|
| | pNew |
| | {DM} |
| | |
| | |
| | |

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {  
    int  wP;  
    int  fP;  
    struct fraction* link;  
} fract;
```

```
fract first, *pNew;
```

```
first.wP = 7;
```

```
first.fP = 3;
```

```
pNew = (fract*)mal
```

```
pNew->wP = 56;
```

```
pNew->fP = 92;
```

```
7 - 398 - 399 -  
0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1  
0 - 401 - 402 -  
0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1  
403 - 404 - 405 -  
0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0  
406 - 407 - 408 -  
1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0  
409 - 410 - 411 -  
0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1  
412 - 413 - 414 -  
1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1  
415 - 416 - 417 -  
1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0  
418 - 419 - 420 -  
0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0  
421 - 422 - 423 -
```

| Address Label | Label | Address | Value |
|---------------|------------|---------------|-------|
| | first.wP | 400 - 403 | 7 |
| | first.fP | 404 - 407 | 3 |
| | first.link | 408 - 411 | |
| | pNew | 412 - 415 | 10100 |
| | pNew->wP | 10100 - 10103 | 56 |
| | pNew->fP | 10104 - 10107 | 92 |
| | {DM} | 10108 - 10111 | |
| | | | |

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used
to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {  
    int  wP;  
    int  fP;  
    struct fraction* link;  
} fract;
```

```
fract first, *pNew;
```

```
first.wP = 7;
```

```
first.fP = 3;
```

```
pNew = (fract*)mal
```

```
pNew->wP = 56;
```

```
pNew->fP = 92;
```

```
first.link =  
pNew;
```

```
7 - 398 - 399 -  
0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1  
0 - 401 - 402 -  
0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1  
403 - 404 - 405 -  
0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0  
406 - 407 - 408 -  
1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0  
409 - 410 - 411 -  
0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1  
412 - 413 - 414 -  
1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1  
415 - 416 - 417 -  
1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0  
418 - 419 - 420 -  
0 1 0 1 0 1 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0  
421 - 422 - 423 -
```

| Address Label | Label | Address | Value |
|---------------|------------|---------------|-------|
| | first.wP | 400 - 403 | 7 |
| | first.fP | 404 - 407 | 3 |
| | first.link | 408 - 411 | 10100 |
| | pNew | 412 - 415 | 10100 |
| | pNew->wP | 10100 - 10103 | 56 |
| | pNew->fP | 10104 - 10107 | 92 |
| | {DM} | 10108 - 10111 | |
| | | | |

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used to **DYNAMICALLY** allocate memory.

- a pointer variable can also be used to **DYNAMICALLY** allocate memory.

```
typedef struct fraction {
    int    wP;
    int    fP;
    struct fraction* link;
} fract;

fract first, *pNew;

first.wp = 7;
first.fP = 3;

pNew = (fract*)malloc(sizeof (fract));

pNew->wP = 56;
pNew->fP = 92;

first.link = pNew;
printf( "\nwP: %d, fP: %d\n" , first.link->wP , first.link->fP ) ;
```

MEMORY MAPPING - STRUCTURE

POINTERS and STRUCTURES and LINKS

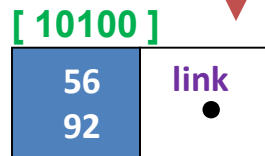
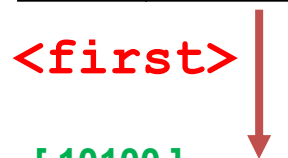
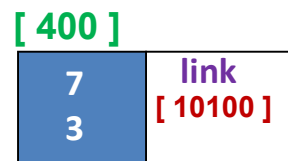
- a pointer variable can also be used

7 - 398 - 399 -
0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1

```
...
pNew = (fract*)malloc(sizeof (fract));

pNew->wP = 56;
pNew->fP = 92;

first.link = pNew;
printf( "\nwP: %d, fP: %d\n" , first.link->wP , first.link->fP ) ;
```



<pNew>

| Address Label | Label | Address | Value |
|---------------|------------|---------------|-------|
| | first.wP | 400 - 403 | 7 |
| | first.fP | 404 - 407 | 3 |
| | first.link | 408 - 411 | 10100 |
| | pNew | 412 - 415 | 10100 |
| | pNew->wP | 10100 - 10103 | 56 |
| | pNew->fP | 10104 - 10107 | 92 |
| | {DM} | 10108 - 10111 | |
| | | | |

MEMORY MAPPING - STRUCTURE

better use of pointer variables in a link

397 -

398 -

399 -

```
typedef struct fraction {
    int  wP;
    int  fP;
    struct fraction* link;
} fract;

fract first, *pTop, *pNew;

first.wP = 7;
first.fP = 3;

pTop = &first;

pNew = (fract*)malloc(sizeof (fract));

pNew->wP = 56;
pNew->fP = 92;

pTop->link = pNew;
printf( "\nwP: %d, fP: %d\n", pTop->link->wP , pTop->link->fP ) ;
```

Linked Lists in C

END OF PART 1