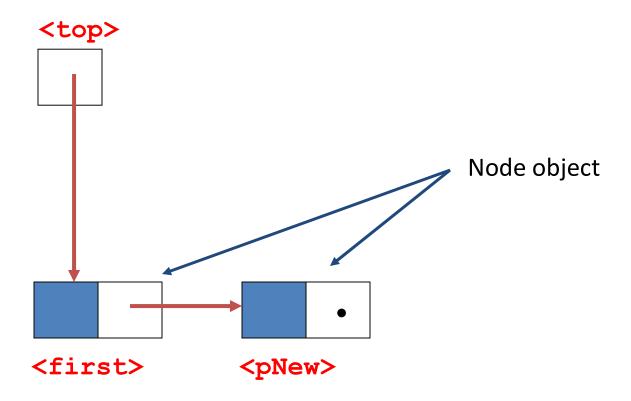
CS 2211 Systems Programming

Part Eleven:

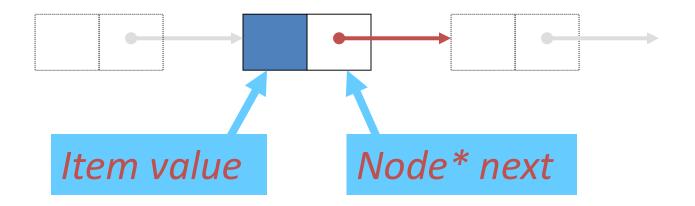
Linked Lists

Linked List

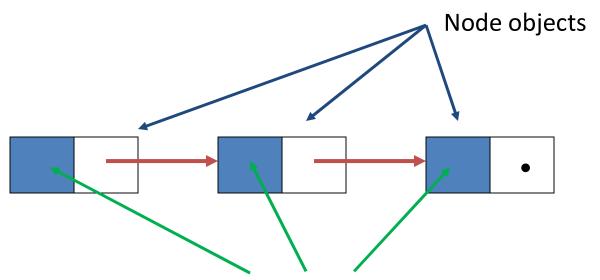


Nodes in Singly Linked Lists

- 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;
```

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
7 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 1
1 403 - 404 - 405 0 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 1 1
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 0 1 0
418 - 419 - 420 0 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0
421 - 422 - 423 -

Address Label	Label	Address	Value

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;
```

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 1 0 0 0 1 1 0 1 1 0 0 1 1 0 0 1 1 0 1 1 1

402 0 1 0 1 0 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 1 1

403 - 404 - 405 0 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 0

406 - 407 - 408 1 1 1 0 0 1 1 0 1 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 1 0 1 0 1 0 1 1 1 0 1 0 0 0 1 1 0

421 - 422 - 423 -

Address Label	Label	Address	Value

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used

```
to DYNAMICALLY allocate memory.
                                         01000110110011100111
typedef struct fraction {
                                        01010111101010110101000110
    int wP;
                                        111001101110011000110010
    int fP;
                                        001000110110111001110111
    struct fraction* link;
                                        10111001111100111001101
     } fract;
                                        111001101110011000110010
fract first *pNew;
                                        01010111101010110101000110
```

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

398 -

399 -

01110011011100110001101

POINTERS and STRUCTURES and LINKS - a pointer variable can also be used 398 -399 -01110011011100110001101 to **DYNAMICALLY** allocate memory. 010001101100111001110111 typedef struct fraction { 01010111101010110101000110 int wP; 111001101110011000110010 int fP; 0010001101100111001110111 struct fraction* link; 101110011011100110001101 } fract; 111001101110011000110010 fract first, *pNew; 01010111101010110101000110 Label Address Address Value first.wp = 7;Label first.fP = 3;first.wP 400 - 403 first.fP 404 - 407 408 - 411 first.link 412 - 415 pNew

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used

```
01110011011100110001101
          to DYNAMICALLY allocate memory.
                                           01000110110011100111
typedef struct fraction {
                                          01010111101010110101000110
    int wP;
                                          111001101110011000110001
    int fP;
                                          0010001101100111001110111
    struct fraction* link;
                                          1011100111011100110001101
     } fract;
                                          111001101110011000110010
fract first, *pNew;
                                          01010111101010110101000110
                                               Address
                                                                Value
first.wp = 7;
first.fP = 3;
                                                   400 - 403
                                                   404 - 407
                                                                 3
pNew = (fract*)malloc(sizeof (fract));
                                                   408 - 411
                                                   412 - 415
                              pNew
                                                                10100
                             {DM}
                                                 10100 - 10111
```

398 -

399 -

POINTERS and STRUCTURES and LINKS

- a pointer variable can also be used

to **DYNAMICALLY** allocate memory.

```
01000110110011100111
typedef struct fraction {
                                              01010111101010110101000110
    int wP;
                                              111001101110011000110010
    int fP;
                                              001000110110111001110111
    struct fraction* link;
                                              10111001111100111001101
     } fract;
                                              111001101110011000110001
fract first, *pNew;
                                              0\;1\;0\;1\;0\;1\;1\;1\;1\;0\;1\;0\;1\;0\;1\;0\;1\;0\;0\;0\;1\;1\;0
```

399 -

01110011011100110001101

			421 -	422 -	423 -	
first.wp = 7;	Address	Label		Address		Value
-	Label					
first.fP = 3;		first.wP		400 -	403	7
pNew = (fract*)mal		first.fP		404 -	407	3
		first.link		408 -	411	
pNew->wP = 56;		pNew		412 -	415	10100
pNew->fP = 92;		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.

```
01000110110011100111
typedef struct fraction {
                                        01010111101010110101000110
    int wP;
                                        111001101110011000110010
    int fP;
                                        001000110110111001110111
    struct fraction* link;
                                        1011100111011100110001101
     } fract;
                                        111001101110011000110010
fract first, *pNew;
                                        01010111101010110101000110
```

399 -

01110011011100110001101

		421 -	422 - 423 -	
<pre>first.wp = 7; first.fP = 3;</pre>	Address Label	Label	Address	Value
11136.11 3,		first.wP	400 - 403	7
pNew = (fract*)mal		first.fP	404 - 407	3
		first.link	408 - 411	10100
pNew->wP = 56;		pNew	412 - 415	10100
pNew->fP = 92;		pNew->wP	10100 - 10103	56
Ginat link -		pNew->fP	10104 - 10107	92
<pre>first.link = pNew;</pre>		{DM}	10108 - 10111	
pnew,				

POINTERS and STRUCTURES and LINKS

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

```
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 1
```

```
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 ) ;
```

POINTERS and STRUCTURES and LINKS

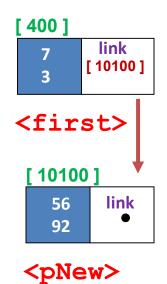
- a pointer variable can also be used

```
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 ) ;
```



		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	

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