600.120 Intermediate Programming, Spring 2017*

Misha Kazhdan

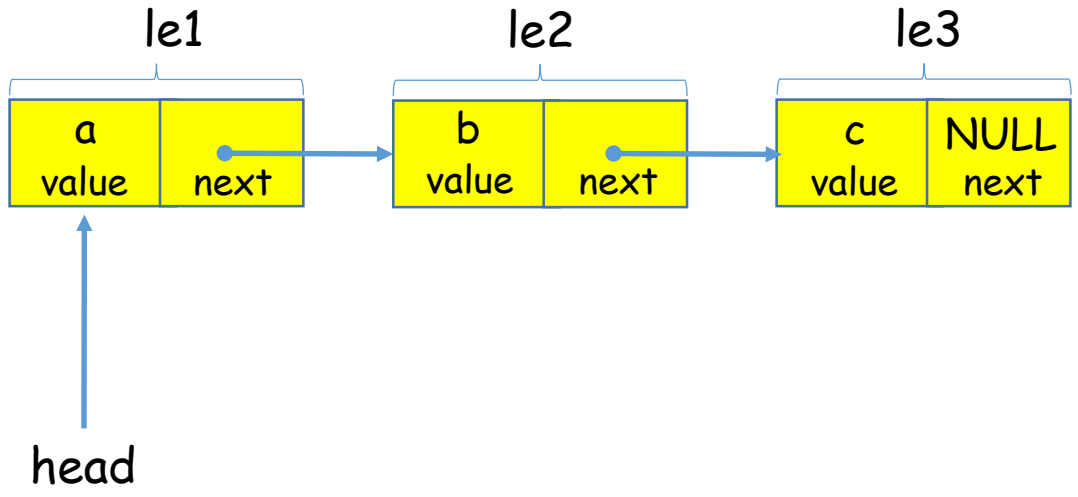
**Much of the code in these examples is not commented because it would otherwise not fit on the slides. This is bad coding practice in general and you should not follow my lead on this.*

Outline

- Linked lists
- Header guards

Linked lists

- Represent the linked list by a set of elements, with each element storing:
  - Its own value
  - A pointer to the next element



charList.h

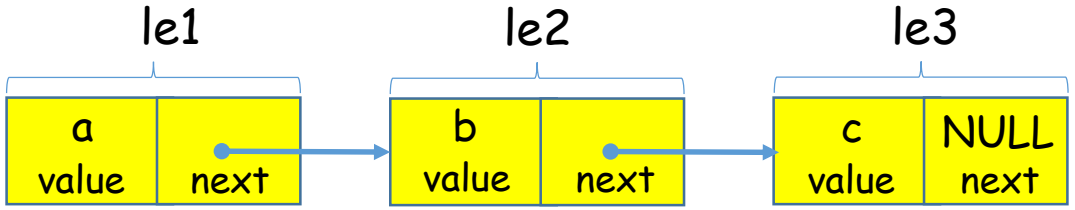
```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;
```

main.c

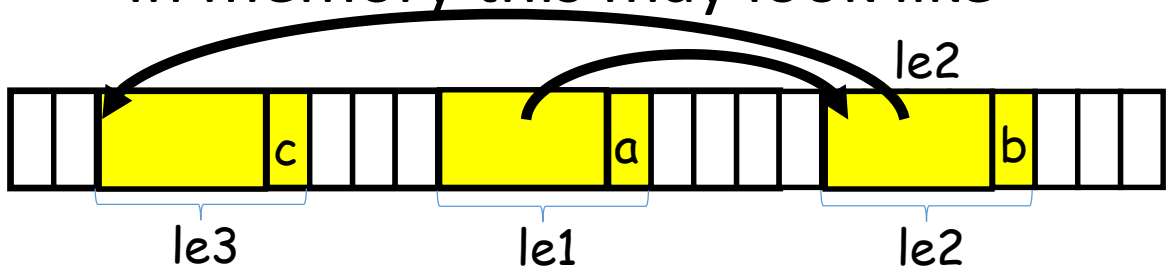
```
#include <stdio.h>
#include "charList.h"
void main( void )
{
    Elem * le1 = malloc( sizeof( Elem ) );
    Elem * le2 = malloc( sizeof( Elem ) );
    Elem * le3 = malloc( sizeof( Elem ) );

    Elem* head = le1;
    le1->value = 'a' ; le1->next = le2;
    le2->value = 'b' ; le2->next = le3;
    le3->value = 'c' ; le3->next = NULL;
    …
}
```

Linked lists

- Represent the linked list by a set of elements, with each element storing:
  - Its own value
  - A pointer to the next element



- In memory this may look like



charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;
```

main.c

```
#include <stdio.h>
#include "charList.h"
void main( void )
{
    Elem * le1 = malloc( sizeof( Elem ) );
    Elem * le2 = malloc( sizeof( Elem ) );
    Elem * le3 = malloc( sizeof( Elem ) );

    le1->value = 'a' ; le1->next = le2;
    le2->value = 'b' ; le2->next = le3;
    le3->value = 'c' ; le3->next = NULL;
    …
}
```

Linked lists

- Creation
  - Allocate the linked-list element
  - Set its members

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"

Elem* Create( char c )
{
    Elem *e = malloc( sizeof( Elem ) );
    assert( e );
    e->next = NULL ; e->value = c;
    return e;
}
```

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

Linked lists

- Insertion
  - Create the linked-list element
  - Update the pointers

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
…
void InsertAfter( Elem *e , char c )
{
    Elem *newE = Create( c );
    newE->next = e->next;
    e->next = newE;
}
```

Linked lists

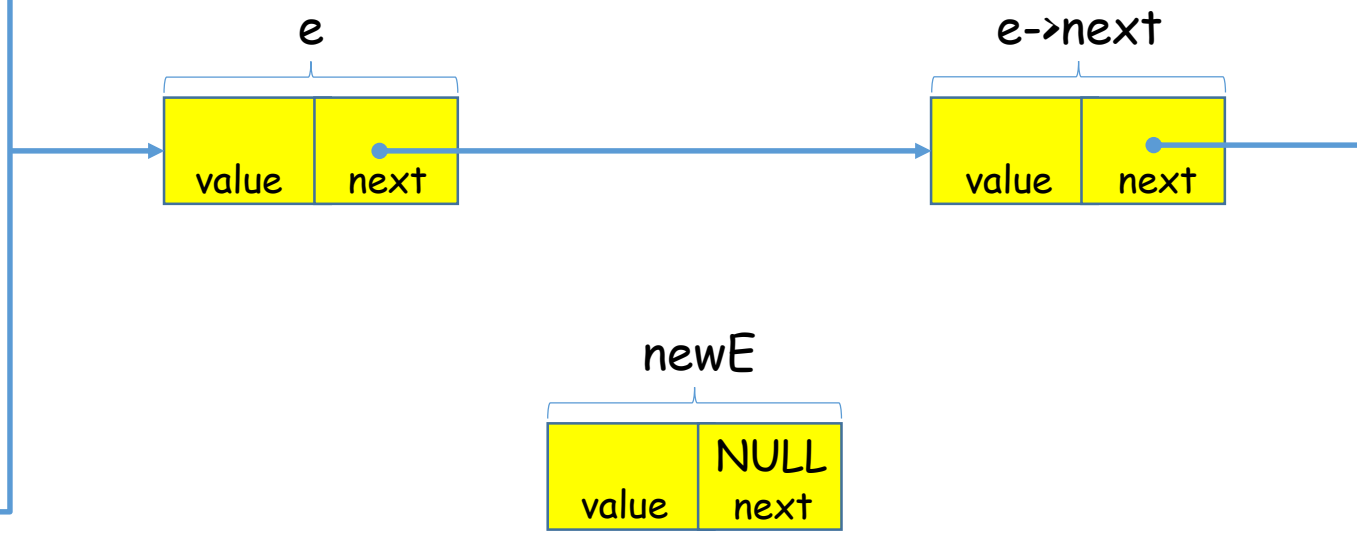
- Insertion
  - Create the linked-list element
  - Update the pointers

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
...
void InsertAfter( Elem *e , char c )
{
    Elem *newE = Create( c );
    newE->next = e->next;
    e->next = newE;
}
```



Linked lists

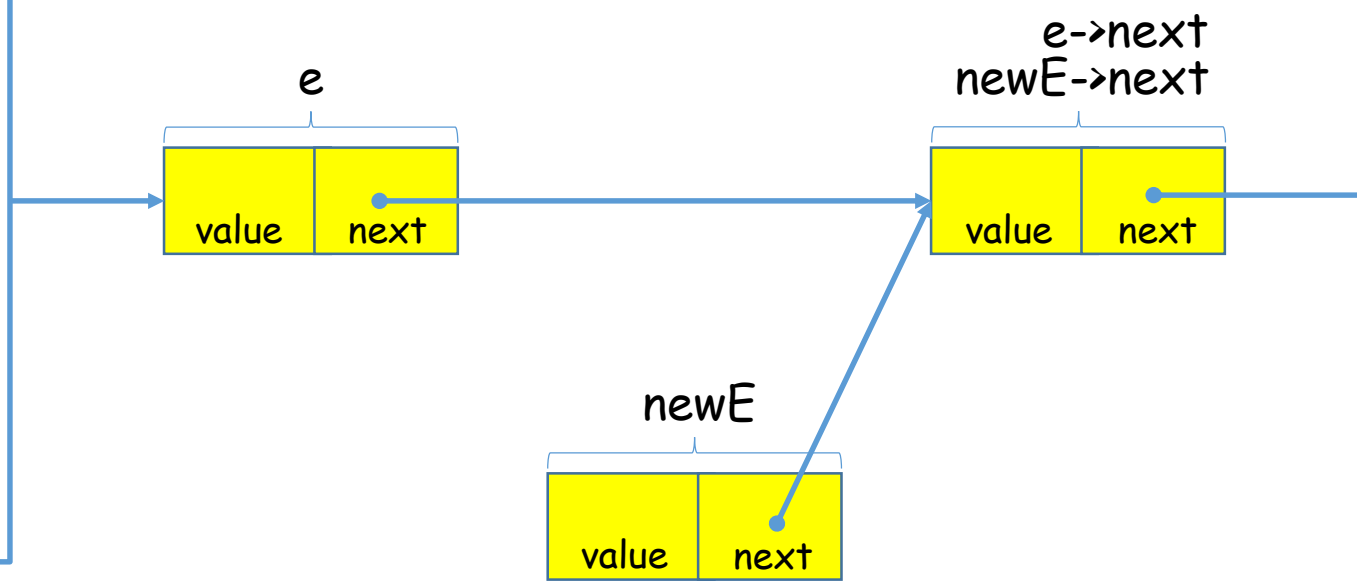
- Insertion
  - Create the linked-list element
  - Update the pointers

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
…
void InsertAfter( Elem *e , char c )
{
    Elem *newE = Create( c );
    newE->next = e->next;
    e->next = newE;
}
```



Linked lists

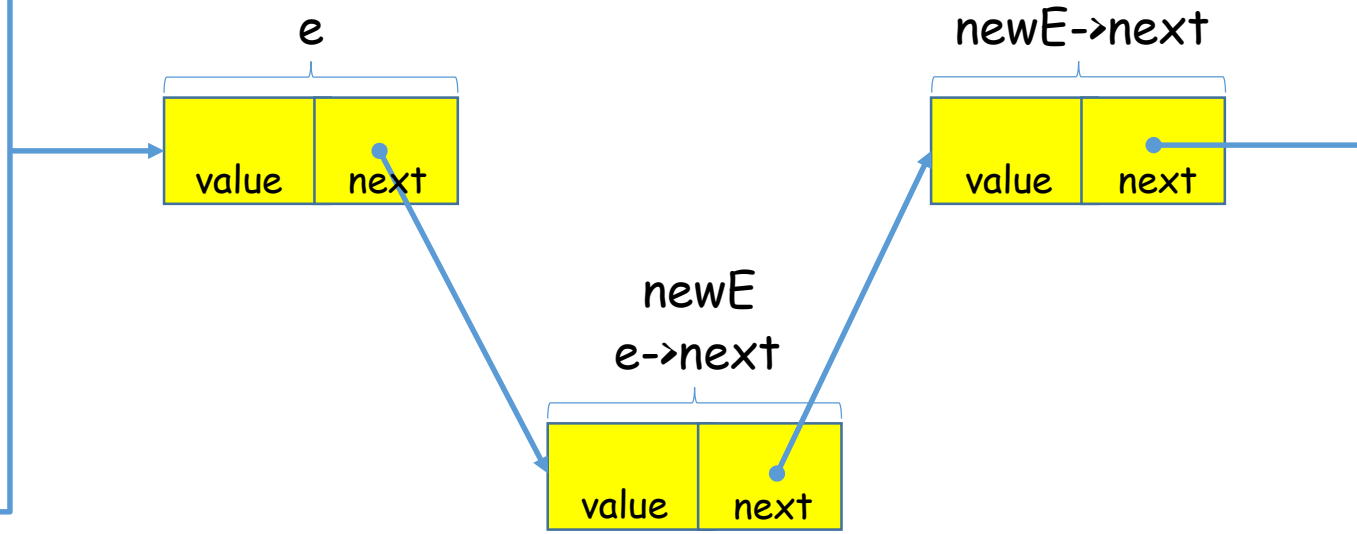
- Insertion
  - Create the linked-list element
  - Update the pointers

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
...
void InsertAfter( Elem *e , char c )
{
    Elem *newE = Create( c );
    newE->next = e->next;
    e->next = newE;
}
```



Linked lists

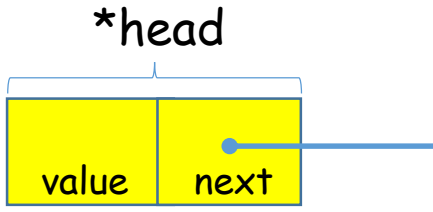
- Insertion
  - Create the linked-list element
  - Update the pointers

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
…
void InsertHead( Elem **head , char c )
{
    Elem *e = Create( c );
    e->next = *head;
    *head = e;
}
```



Linked lists

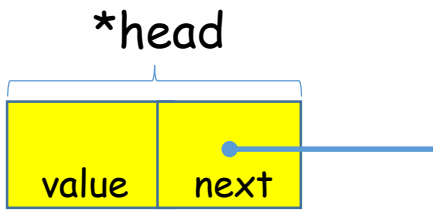
- Insertion
  - Create the linked-list element
  - Update the pointers

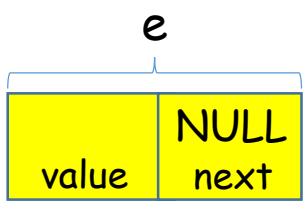
charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
…
void InsertHead( Elem **head , char c )
{
      Elem *e = Create( c );
      e->next = *head;
      *head = e;
}
```

charList.h

```
typedef struct _Elem
{
      struct _Elem *next;
      char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```





Linked lists

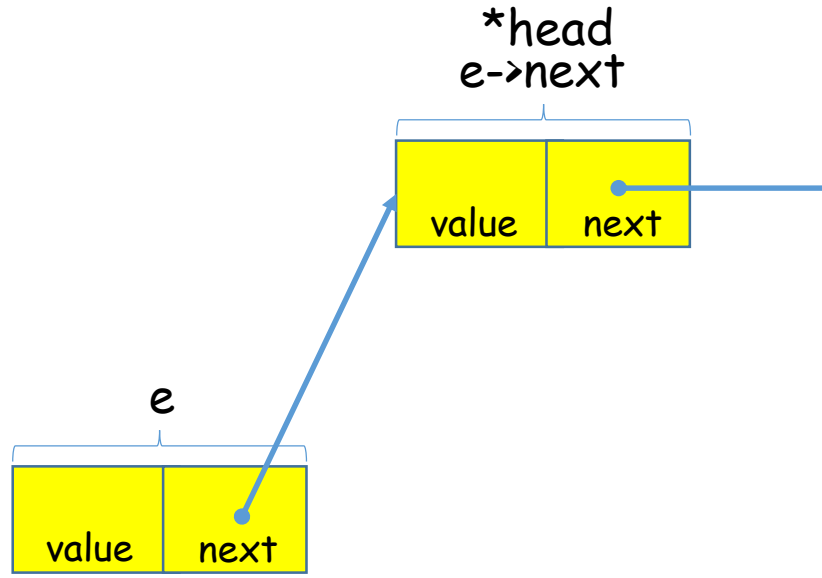
- Insertion
  - Create the linked-list element
  - Update the pointers

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
…
void InsertHead( Elem **head , char c )
{
      Elem *e = Create( c );
      e->next = *head;
      *head = e;
}
```

charList.h

```
typedef struct _Elem
{
      struct _Elem *next;
      char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```



Linked lists

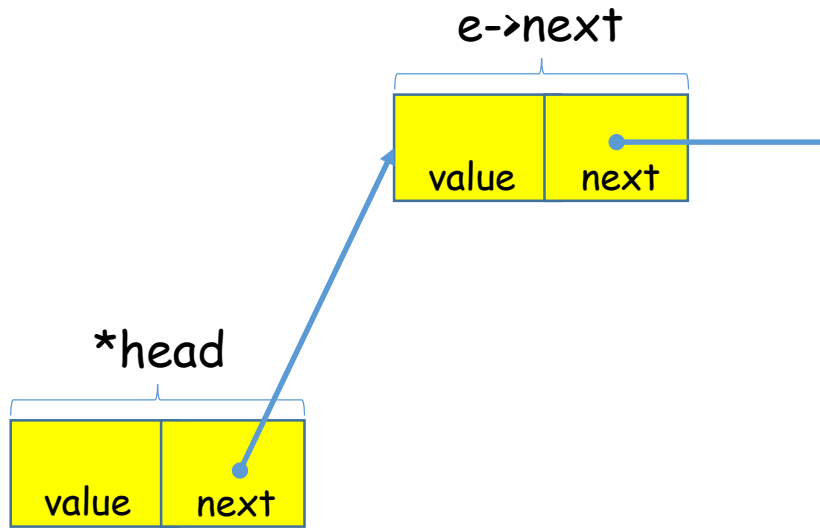
- Insertion
  - Create the linked-list element
  - Update the pointers

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
…
void InsertHead( Elem **head , char c )
{
      Elem *e = Create( c );
      e->next = *head;
      *head = e;
}
```

charList.h

```
typedef struct _Elem
{
      struct _Elem *next;
      char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```



Linked lists

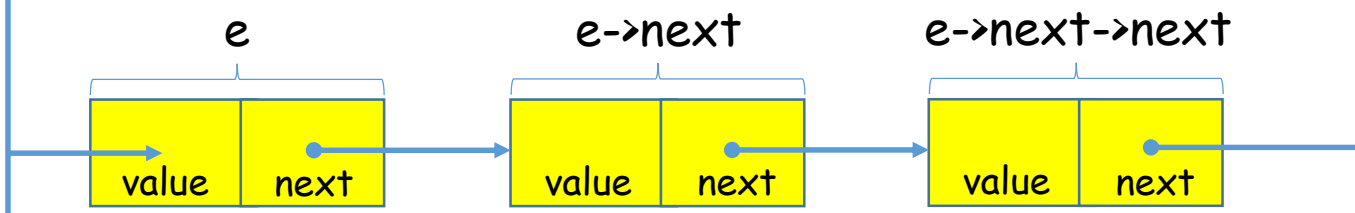
- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
...
void DeleteAfter( Elem *e )
{
    Elem* eNext = e->next;
    if( !eNext ) return;
    e->next = e->next->next;
    free( eNext );
}
```



Linked lists

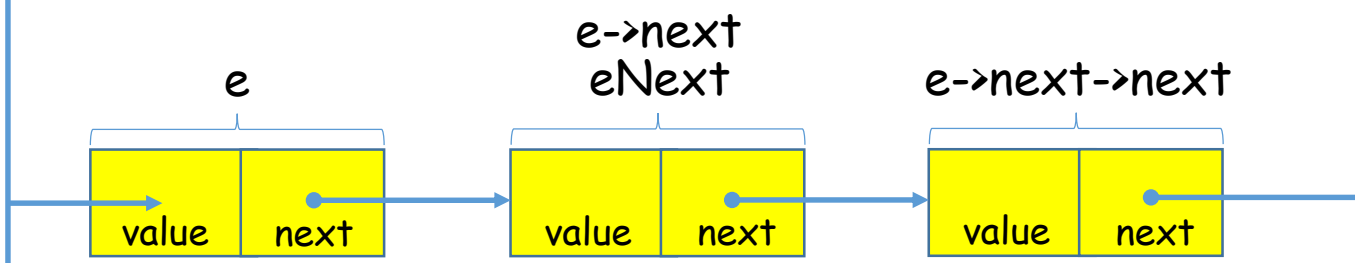
- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
…
void DeleteAfter( Elem *e )
{
    Elem* eNext = e->next;
    if( !eNext ) return;
    e->next = e->next->next;
    free( eNext );
}
```



Linked lists

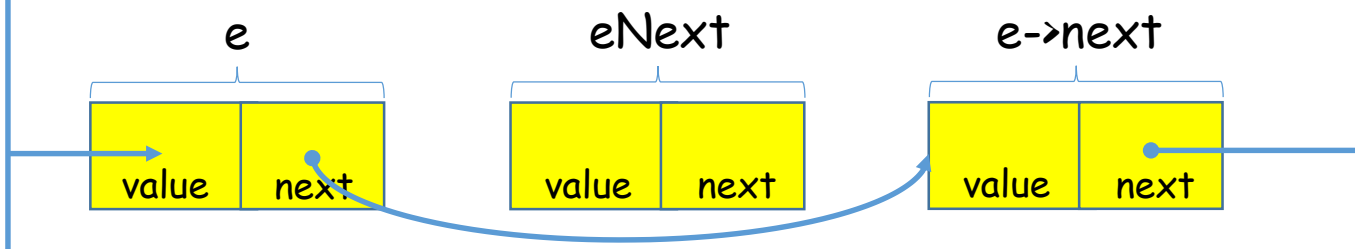
- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
...
void DeleteAfter( Elem *e )
{
    Elem* eNext = e->next;
    if( !eNext ) return;
    e->next = e->next->next;
    free( eNext );
}
```



Linked lists

- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
...
void DeleteAfter( Elem *e )
{
    Elem* eNext = e->next;
    if( !eNext ) return;
    e->next = e->next->next;
    free( eNext );
}
```

Linked lists

- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
...
void DeleteHead( Elem **head )
{
    Elem* e = (*head);
    *head = e->next;
    free( e );
}
```

Linked lists

- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
...
void DeleteHead( Elem **head )
{
    Elem* e = (*head);
    *head = e->next;
    free( e );
}
```

Linked lists

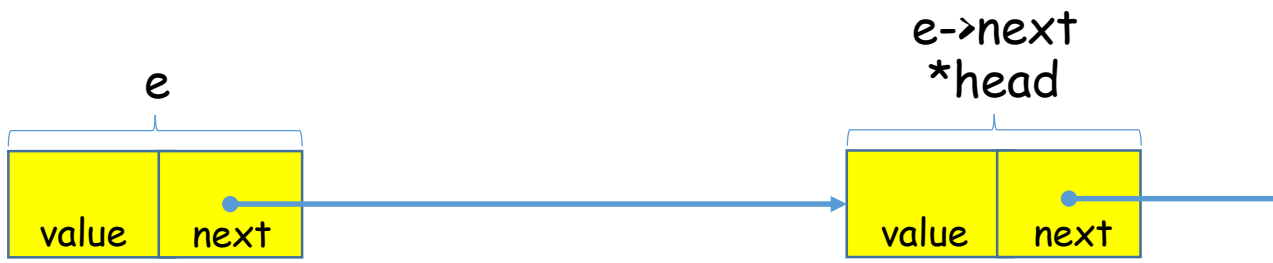
- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
...
void DeleteHead( Elem **head )
{
    Elem* e = (*head);
    *head = e->next;
    free( e );
}
```



Linked lists

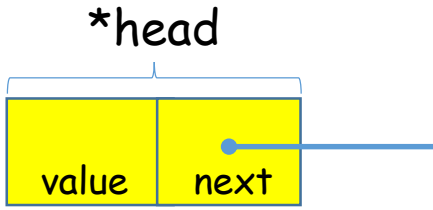
- Deletion
  - Update the pointers
  - Delete the linked-list element

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

charList.c

```
#include <stdlib.h>
#include "charList.h"
#include "assert.h"
...
void DeleteHead( Elem **head )
{
    Elem* e = (*head);
    *head = e->next;
    free( e );
}
```



Linked lists

- Printing
  - Print out the value in the current element
  - Advance to the next element (if it isn't NULL)

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
```

charListIO.h

```
#include "charList.h"
void PrintList( const Elem* head );
```

charListIO.c

```
#include "charList.h"
#include "stdio.h"

void PrintList( const Elem *head )
{
    for( const Elem* e=head ; e!=NULL ; e=e->next )
        printf( "%c" , e->value );
    printf( "\n" );
}
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
int main( void )
{
    Elem *head = NULL , *e;
    char c;
    while( ( c = getc( stdin ) )!=EOF && !isspace(c) )
    {
        if( !head ) head  = Create( c );
        else if( c<head->value ) InsertHead( &head , c );
        else
        {
            for( e=head ; e->next!=NULL && c>=e->next->value ; e=e->next ) ;
            InsertAfter( e , c );
        }
    }
    PrintList( head );
    while( head ) DeleteHead( &head );
    return 0;
}
```

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
```

charListIO.h

```
#include "charList.h"
void PrintList( const Elem* head );
```

```
>> gcc -std=c99 -Wall -Wextra -g main.c charList.c charListIO.c
In file included from charListIO.h:1:0,
                 from main.c:5:
charList.h:3:16: error: redefinition of struct _Elem
 typedef struct _Elem
                ^~~~~

...
>>
```

Outline

- Linked lists
- Header guards

Header guards

- Each time a header file is included inside a `.c` file, the contents of the header file are placed inside the `.c` file by the preprocessor

⇒ If the header file is included multiple times, all the function / type declarations are included multiple times

⇒ The same function / type will be declared multiple times

```
>> gcc -std=c99 -Wall -Wextra -g main.c charList.c charListIO.c
In file included from charListIO.h:1:0,
                 from main.c:5:
charList.h:3:16: error: redefinition of struct _Elem
 typedef struct _Elem
                ^~~~~
...
>>
```

Header guards

Q: Where does the second definition come from?

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
…
```

charListIO.h

```
#include "charList.h"
void PrintList( const Elem* head );
```

Header guards

Q: Where does the second definition come from?

A. The header `charList.h` is included in the header `charListIO.h`

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
…
```

charListIO.h

```
#include "charList.h"
void PrintList( const Elem* head );
```

Header guards

Q: Where does the second definition come from?

A. The header `charList.h` is included in the header `charListIO.h`

But both are included in `main.c`

charList.h

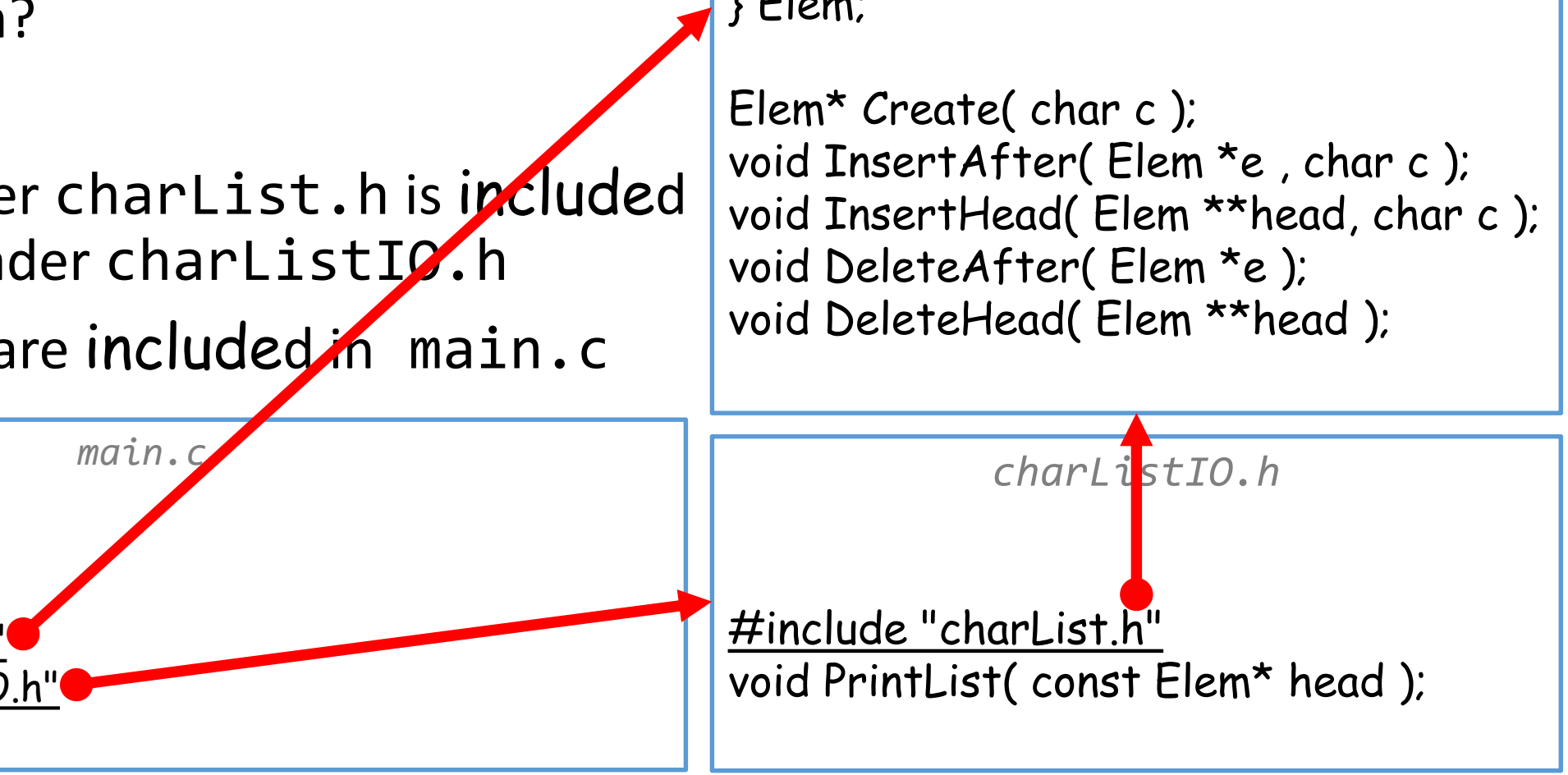
```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
...
```

charListIO.h

```
#include "charList.h"
void PrintList( const Elem* head );
```



Header guards

- The headers are included in the pre-processing phase
- We can use the preprocessor **#define** and **#ifdef / #endif** commands to guard against multiple inclusions

charList.h

```
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
…
```

charListIO.h

```
#include "charList.h"
void PrintList( const Elem* head );
```

Header guards

- The headers are included in the pre-processing phase
- We can use the preprocessor **#define** and **#ifdef / #endif** commands to guard against multiple inclusions

charList.h

```
#ifndef CHAR_LIST_INCLUDED
#define CHAR_LIST_INCLUDED
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
#endif // CHAR_LIST_INCLUDED
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
…
```

charListIO.h

```
#ifndef CHAR_LIST_IO_INCLUDED
#define CHAR_LIST_IO_INCLUDED
#include "charList.h"
void PrintList( const Elem* head );
#endif // CHAR_LIST_IO_INCLUDED
```

Header guards

CHAR_LIST_INCLUDED has not been defined yet:

⇒ define it and expose the contents of the header file `charList.h`

charList.h

```
#ifndef CHAR_LIST_INCLUDED
#define CHAR_LIST_INCLUDED
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
#endif // CHAR_LIST_INCLUDED
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
...
```

charListIO.h

```
#ifndef CHAR_LIST_IO_INCLUDED
#define CHAR_LIST_IO_INCLUDED
#include "charList.h"
void PrintList( const Elem* head );
#endif // CHAR_LIST_IO_INCLUDED
```

Header guards

CHAR_LIST_IO_INCLUDED has not been defined yet:

⇒ define it and expose the contents of the header file `charListIO.h`

charList.h

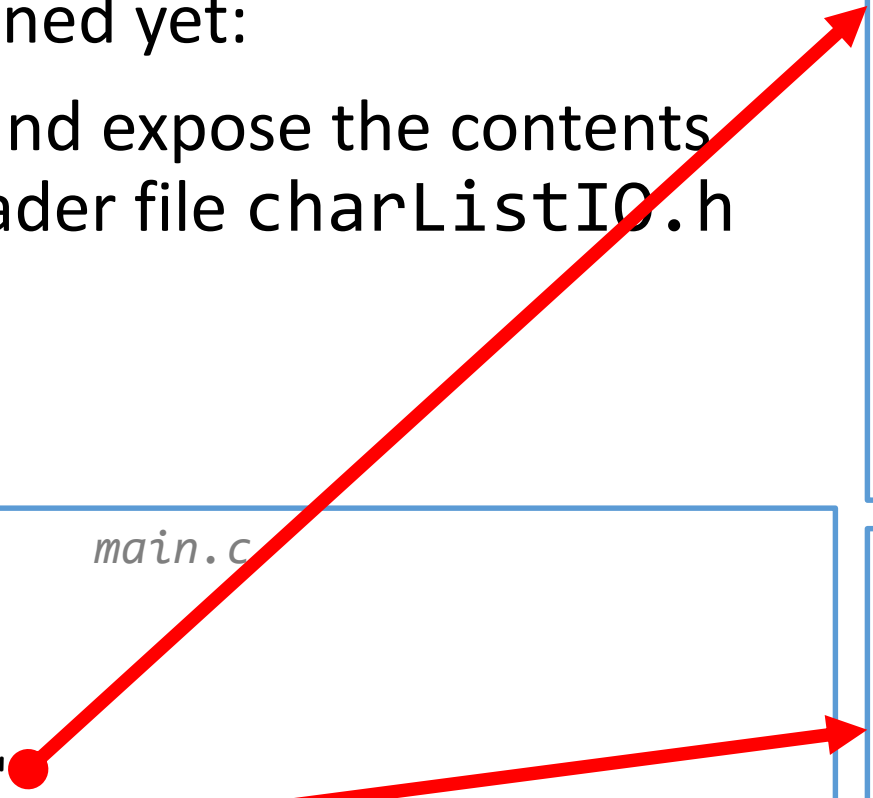
```
#ifndef CHAR_LIST_INCLUDED
#define CHAR_LIST_INCLUDED
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
#endif // CHAR_LIST_INCLUDED
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
...
```

charListIO.h

```
#ifndef CHAR_LIST_IO_INCLUDED
#define CHAR_LIST_IO_INCLUDED
#include "charList.h"
void PrintList( const Elem* head );
#endif // CHAR_LIST_IO_INCLUDED
```



Header guards

CHAR_LIST_INCLUDED has already been defined:

⇒ hide the contents of the header file `charList.h`

charList.h

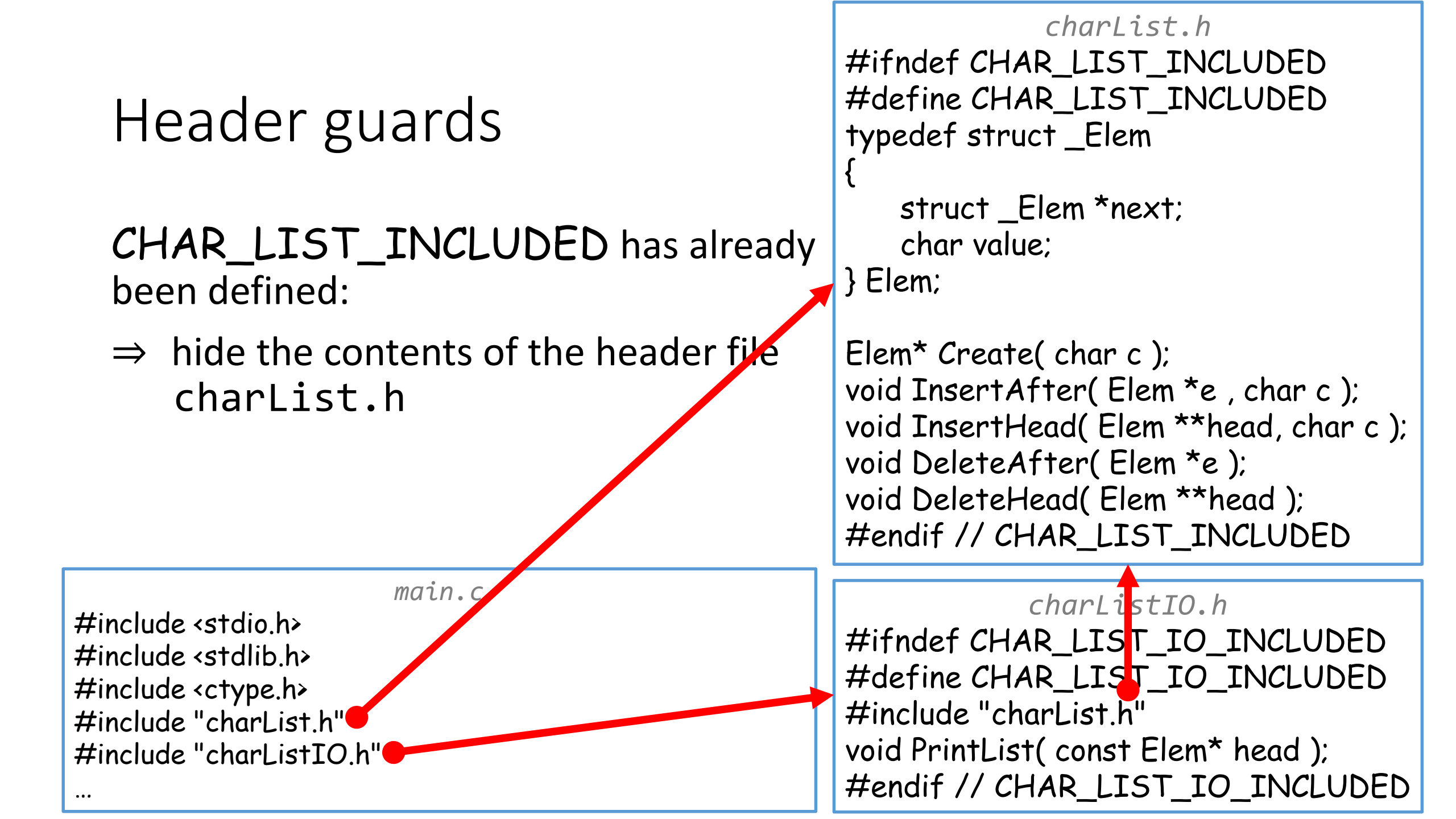
```
#ifndef CHAR_LIST_INCLUDED
#define CHAR_LIST_INCLUDED
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
#endif // CHAR_LIST_INCLUDED
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
...
```

charListIO.h

```
#ifndef CHAR_LIST_IO_INCLUDED
#define CHAR_LIST_IO_INCLUDED
#include "charList.h"
void PrintList( const Elem* head );
#endif // CHAR_LIST_IO_INCLUDED
```



Header guards

[WARNING]

- Be sure that your header guards have unique names
- Otherwise you may inadvertently hide files that need to be exposed

charList.h

```
#ifndef CHAR_LIST_INCLUDED
#define CHAR_LIST_INCLUDED
typedef struct _Elem
{
    struct _Elem *next;
    char value;
} Elem;

Elem* Create( char c );
void InsertAfter( Elem *e , char c );
void InsertHead( Elem **head, char c );
void DeleteAfter( Elem *e );
void DeleteHead( Elem **head );
#endif // CHAR_LIST_INCLUDED
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
…
```

charListIO.h

```
#ifndef CHAR_LIST_IO_INCLUDED
#define CHAR_LIST_IO_INCLUDED
#include "charList.h"
void PrintList( const Elem* head );
#endif // CHAR_LIST_IO_INCLUDED
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "charList.h"
#include "charListIO.h"
int main( void )
{
    Elem *head = NULL , *e;
    char c;
    while( ( c = getc( stdin ) )!=EOF && !isspace(c) )
    {
        if( !head ) head  = Create( c );
        else if( c<head->value ) InsertHead( &head , c );
        else
        {
            for( e=head ; e->next!=NULL && c>=e->next->value ; e=e->next ) ;
            InsertAfter( e , c );
        }
    }
    PrintList( head );
    while( head ) DeleteHead( &head );
    return 0;
}
```

```
>> gcc -std=c99 -Wall -Wextra -g main.c charList.c charListIO.c
>> ./a.out
misha
ahims
>>
```

Piazza → Resources section → Resources tab → Exercise 6-2

Piazza → Resources section → Resources tab → Exercise 6-3