Linked Lists

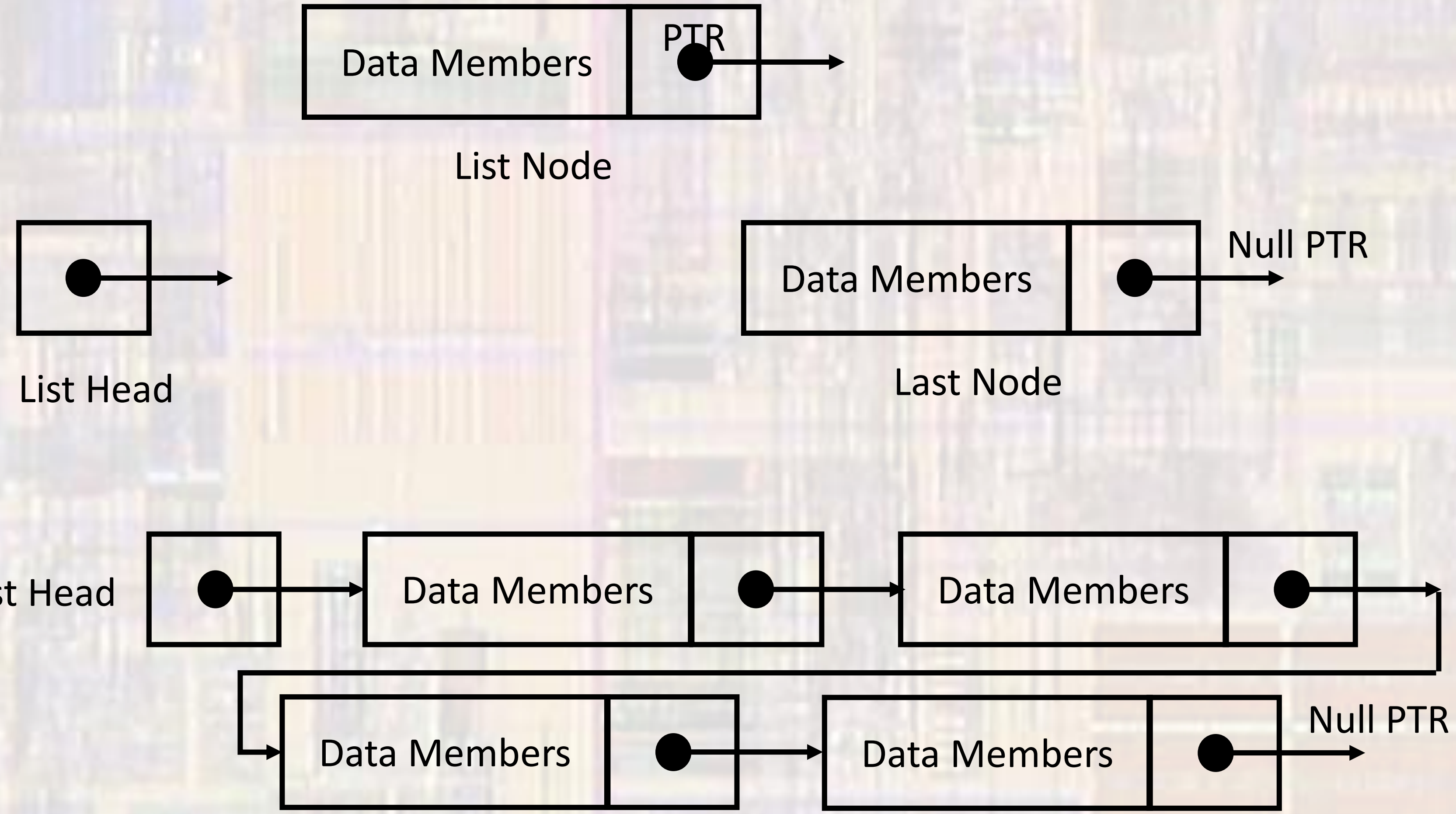
Last updated 4/27/20

Linked Lists

- Motivation
  - Vectors and Arrays have distinct advantages and disadvantages
    - Array – easy to use, but fixed in size
    - Vectors – variable size, but costly to insert or delete elements
  - Desire a similar structure but without the disadvantages
  - Linked List
    - Variable size
    - Efficient addition or deletion of elements
    - More difficult to code

Linked Lists

- Basic Structure



Linked Lists

- Implementation

```
struct ListNode{
   int id;
   double gpa;
   ListNode* next;
};
```

Linked Lists

- Implementation
  - Starting a new list

```
ListNode * head;            // create ptr
head = new ListNode;        // allocate memory
head->id = 1234;            // assign values
head->gpa = 3.45;
head->next = nullptr;       // end of list (so far)
```

Linked Lists

- Implementation
  - Adding a new node

```
ListNode * newnode = new ListNode;        // create new node
newnode>id = 5678                         // assign values
newnode->gpa = 2.89;
newnode->next = nullptr;                  // new end of list
head->next = newnode;                     // previous node
                                             (head) now points
                                             to newnode
```

Linked Lists

- Implementation

```
/*
 * GPAList.h
 *
 *  Created on: Apr 27, 2020
 *      Author: johnsontimoj
 */

#ifndef GPALIST_H_
#define GPALIST_H_

using namespace std;
#include <iostream>

class GPAList {
protected:
    struct ListNode{
        int id;
        double gpa;
        ListNode* next;

        ListNode(int idval, double gpaval, ListNode* ptrval){
            id = idval;
            gpa = gpaval;
            next = ptrval;
        }
    };

    ListNode* head;
```

Structures can have constructors

When a function is defined inside the class declaration it is called an inline fn

```
public:
    GPAList();
    ~GPAList(void);
    void add(int idval, double gpaval);
    void remove(int idval);
    void displayList(void) const;
};

#endif /* GPALIST_H_ */
```

Linked Lists

- Implementation

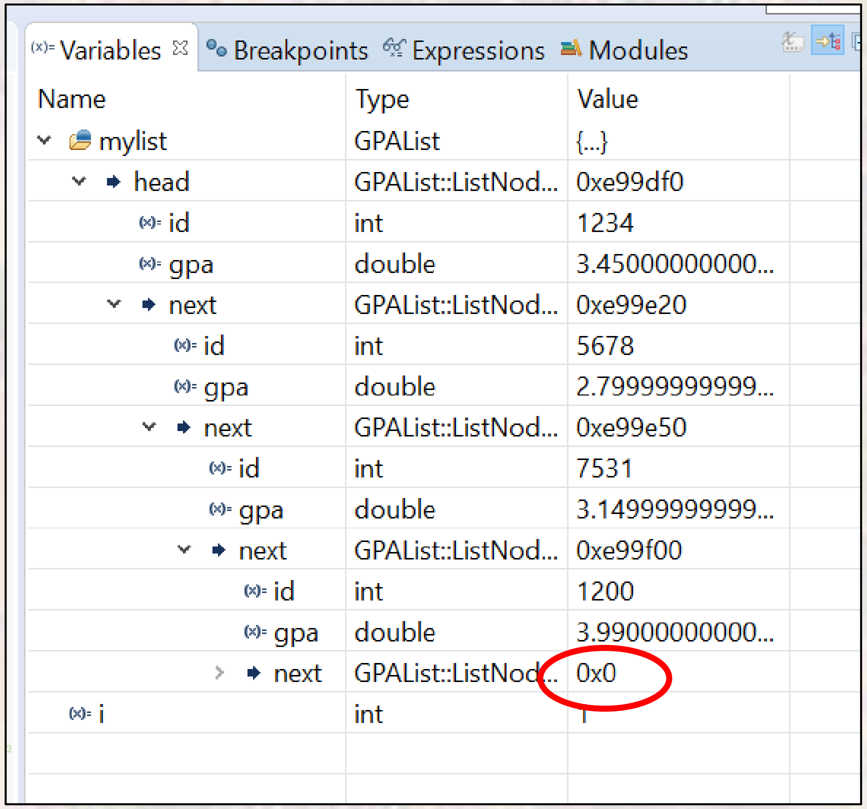
```cpp
/*
 * GPAList.cpp
 *
 *  Created on: Apr 27, 2020
 *      Author: johnsontimoj
 */

#include "GPAList.h"

using namespace std;

GPAList::GPAList(){
    head = nullptr;
}

void GPAList::add(int idval, double gpaval){
    if(head == nullptr)
        // empty list
        head = new ListNode(idval, gpaval, nullptr);
    else{
        // non-empty list
        // go to the end using node_ptr to traverse
        ListNode* node_ptr = head;   // start at head
        while(node_ptr->next != nullptr)
            node_ptr = node_ptr->next;

        //create the new node as end of list
        // current node_ptr points to last item
        // redirect it to a new ListNode
        node_ptr->next = new ListNode(idval, gpaval, nullptr);
    } // end if
} // end add
```

```cpp
void GPAList::displayList(void) const{
    // use node_ptr to traverse list
    // start at head
    ListNode* node_ptr = head;
    while(node_ptr){
        cout << "id: " << node_ptr->id << ", gpa: " << node_ptr->gpa << endl;
        node_ptr = node_ptr->next;
    }
}

GPAList::~GPAList(){
    // use node_ptr to traverse list
    // start at head
    ListNode* node_ptr = head;
    while(node_ptr){
        // collect garbage
        ListNode* garbage = node_ptr;
        // move to next node so you don't delete yourself
        node_ptr = node_ptr->next;
        // empty garbage
        delete garbage;
    }
}
```

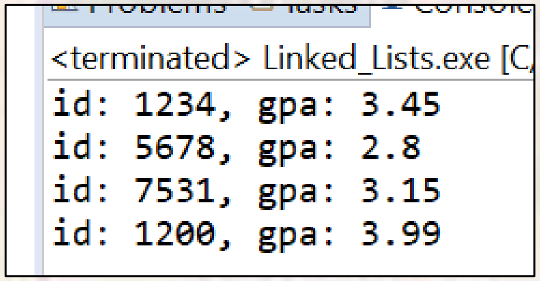
Linked Lists

- Implementation

```
/*
 * linked_list_test.cpp
 *
 *  Created on: Apr 27, 2020
 *      Author: johnsontimoj
 */

#include "GPAList.h"
using namespace std;

int main(void){
    // create scope to watch destructor
    int i;

    for(i=1; i<2; i++){
        // create empty list
        GPAList mylist;

        // add nodes
        mylist.add(1234, 3.45);
        mylist.add(5678, 2.80);
        mylist.add(7531, 3.15);
        mylist.add(1200, 3.99);

        // display list
        mylist.displayList();
    }

    return 0;
}// end main
```



Name	Type	Value
mylist	GPAList	{...}
head	GPAList::ListNod...	0xe99df0
id	int	1234
gpa	double	3.45000000000...
next	GPAList::ListNod...	0xe99e20
id	int	5678
gpa	double	2.79999999999...
next	GPAList::ListNod...	0xe99e50
id	int	7531
gpa	double	3.14999999999...
next	GPAList::ListNod...	0xe99f00
id	int	1200
gpa	double	3.99000000000...
next	GPAList::ListNod...	0x0
i	int	1



```
<terminated> Linked_Lists.exe [C,
id: 1234, gpa: 3.45
id: 5678, gpa: 2.8
id: 7531, gpa: 3.15
id: 1200, gpa: 3.99
```

Linked Lists

- Sorted list

```
/*
 * SortedGPAList.h
 *
 *  Created on: Apr 28, 2020
 *      Author: johnsontimoj
 */

#ifndef SORTEDGPALIST_H_
#define SORTEDGPALIST_H_

#include "GPAList.h"

class SortedGPAList : public GPAList {
public:
    // overloading add
    void add(int idval, double gpaval); // overloading add
};

#endif /* SORTEDGPALIST_H_ */
```

Linked Lists

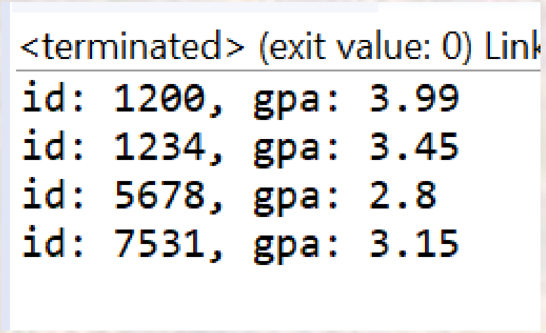
- Sorted list

```
/*
 * SortedGPAList.cpp
 *
 *  Created on: Apr 28, 2020
 *      Author: johnsontimoj
 */

#include "SortedGPAList.h"

void SortedGPAList::add(int idval, double gpaval){
    ListNode * node_ptr;
    ListNode * prev_node_ptr;

    if((head == nullptr) || (head->id >= idval)){
        // new node goes to beginning of list
        head = new ListNode(idval, gpaval, head);
    } else {
        // traverse past the head
        prev_node_ptr = head;
        node_ptr = head->next;

        // find the correct insertion location
        while((node_ptr != nullptr) && (node_ptr->id < idval)){
            // continue to traverse
            prev_node_ptr = node_ptr;
            node_ptr = node_ptr->next;
        }// end while

        // at correct location, but overshot by 1, so use prev
        prev_node_ptr->next = new ListNode(idval, gpaval, node_ptr);
    }// end if
}// end add
```

Linked Lists

- Sorted list

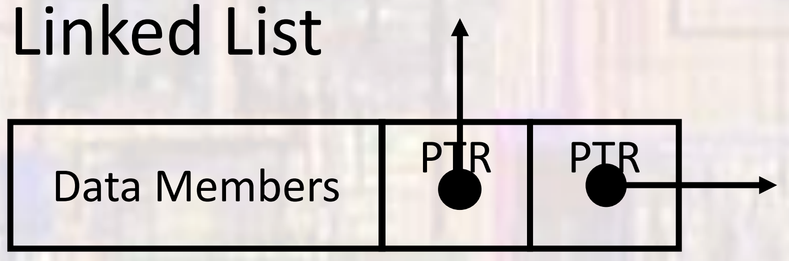
```
/*
 * linked_list_test.cpp
 *
 *  Created on: Apr 27, 2020
 *      Author: johnsontimoj
 */

#include "SortedGPAList.h"
using namespace std;

int main(void){
    // create scope to watch destructor
    int i;

    for(i=1; i<2; i++){
        // create empty list
        SortedGPAList mysortedlist;

        // add nodes
        mysortedlist.add(1234, 3.45);
        mysortedlist.add(7531, 3.15);
        mysortedlist.add(5678, 2.80);
        mysortedlist.add(1200, 3.99);

        // display list
        mysortedlist.displayList();
    }

    return 0;
}// end main
```



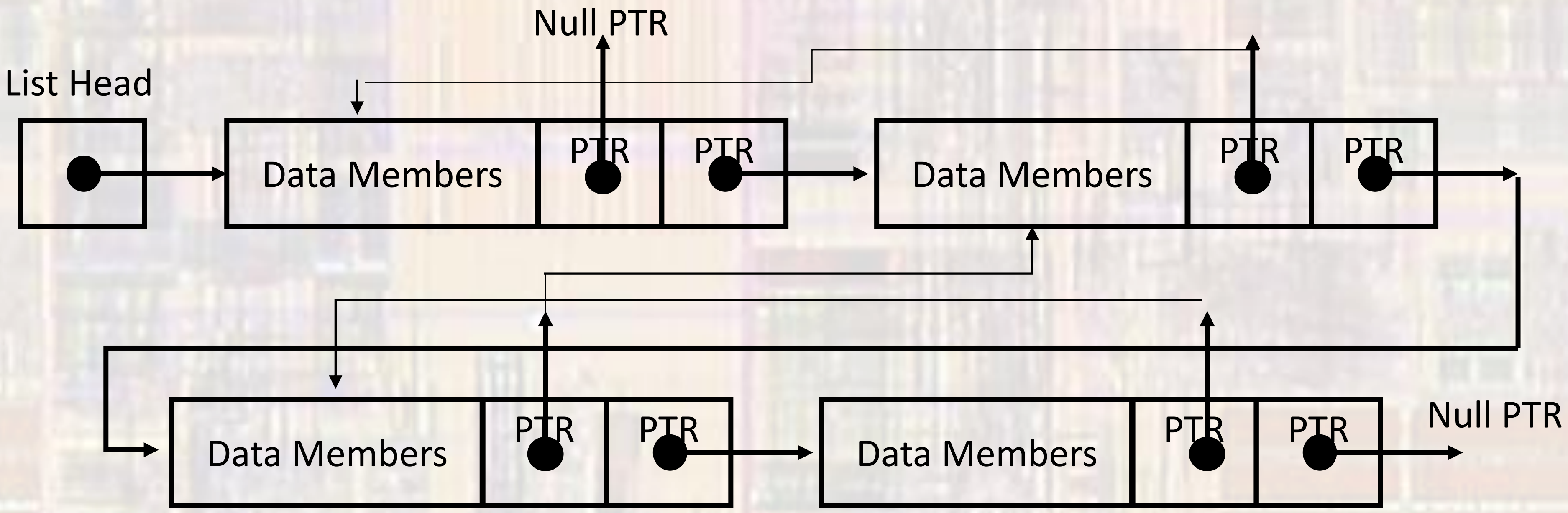
```
<terminated> (exit value: 0) Link
id: 1200, gpa: 3.99
id: 1234, gpa: 3.45
id: 5678, gpa: 2.8
id: 7531, gpa: 3.15
```

Linked Lists

- Doubly Linked List



List Node


Null PTR
List Head
Data Members
PTR
PTR
Data Members
PTR
PTR
Data Members
PTR
PTR
Data Members
PTR
PTR
Null PTR