UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical & Computer Engineering

ECE 150 *Fundamentals of Programming*

Introduction to linked lists

Douglas Wilhelm Harder, M.Math. LEL
Prof. Hiren Patel, Ph.D., P.Eng.
Prof. Werner Dietl, Ph.D.

1

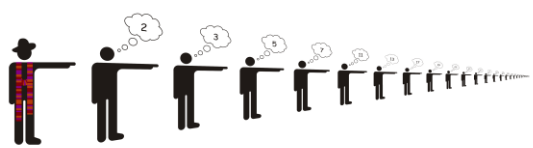
Outline

- In this lesson, we will:
  - Describe the idea of a linked list
  - Describe a simple variation of a linked list that you can implement
  - Add values to this linked list
  - Consider some issues and benefits of linked lists

2

Nodes

- Suppose I want to remember some numbers
  - I could use this class



3

Nodes

- Each student is remembering two pieces of information:
  - A number
  - The next student
- This sound like a class, and we will call this class a *node*:

```
class Node {
    public:
        double value_;
        ???    next_;
};
```

4

Nodes

- Let assume we have an array of these nodes,
  thus, we could identify each node with an *index*

```
class Node {
    public:
        double      value_;
        std::size_t next_index_;
};
```

- Thus, we could have the nodes as follows:

```
std::size_t const list_cap{10};
Node a_nodes[list_cap]{};

for ( std::size_t k{0}; k < list_cap; ++k ) {
    a_nodes[k].next_index_ = list_cap + 1;  // indicates not used
}
```

- All the program need do is remember the first index
  - The *head* of the linked list

```
std::size_t list_head{ list_cap };
```


Nodes

- Thus, we have:

```
int main() {
    std::size_t const list_cap{10};
    Node a_nodes[list_cap]{};

    for ( std::size_t k{0}; k < list_cap; ++k ) {
        a_nodes[k].next_index_ = list_cap + 1;
    }

    std::size_t list_head{ list_cap };

    // Let us use this linked list

    return 0;
}
```


Nodes

- Without coding, let's see what we'd like to do:
  - Here is our initial set-up:

10 list_head

	0	1	2	3	4	5	6	7	8	9
value_	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
next_index_	11	11	11	11	11	11	11	11	11	11

Nodes

- Without coding, let's see what we'd like to do:
  - To add 4.2, we
    - Find an unused node
    - Set the value and its next index to `10` to indicate it is the last node
    - Remember that index

5 list_head

	0	1	2	3	4	5	6	7	8	9
value_	0.0	0.0	0.0	0.0	0.0	4.2	0.0	0.0	0.0	0.0
next_index_	11	11	11	11	11	10	11	11	11	11

Nodes

- Without coding, let's see what we'd like to do:
  - To add 9.1, we
    - Find an unused node
    - Set the value and its next index to 5
    - Remember that index

2	list_head

	0	1	2	3	4	5	6	7	8	9
value_	0.0	0.0	9.1	0.0	0.0	4.2	0.0	0.0	0.0	0.0
next_index_	11	11	5	11	11	10	11	11	11	11

9

Nodes

- Without coding, let's see what we'd like to do:
  - To add 6.3, we
    - Find an unused node
    - Set the value and its next index to 2
    - Remember that index

7	list_head

	0	1	2	3	4	5	6	7	8	9
value_	0.0	0.0	9.1	0.0	0.0	4.2	0.0	6.3	0.0	0.0
next_index_	11	11	5	11	11	10	11	2	11	11

10

Nodes

- We can see why this is called a *linked list*

7	list_head

	0	1	2	3	4	5	6	7	8	9
value_	0.0	0.0	9.1	0.0	0.0	4.2	0.0	6.2	0.0	0.0
next_index_	11	11	5	11	11	10	11	2	11	11

11

Nodes

- Could we program a walk through this linked list?

```
std::size_t index{ list_head };

while ( index != list_cap ) {
    std::cout << a_nodes[index].value_ << std::endl;
    index = a_nodes[index].next_index_;
}
```

7	list_head

	0	1	2	3	4	5	6	7	8	9
value_	0.0	0.0	9.1	0.0	0.0	4.2	0.0	6.2	0.0	0.0
next_index_	11	11	5	11	11	10	11	2	11	11

12

Nodes

- In fact, this could be a for loop:

```
for ( std::size_t index{ list_head }; index != list_cap;
                      index = a_nodes[index].next_index_ ) {
    std::cout << a_nodes[index].value_ << std::endl;
}
```

`7` list_head

	0	1	2	3	4	5	6	7	8	9
value_	0.0	0.0	9.1	0.0	0.0	4.2	0.0	6.2	0.0	0.0
next_index_	11	11	5	11	11	10	11	2	11	11

Questions

- How would you do the following?
  - Add a node to the end of the linked list, not the start
  - Remove the first node from the linked list
    - This assumes there is at least one node in that linked list
  - Remove the last node from the linked list
    - What happens if there is only one node in the linked list?
  - Remove an arbitrary node in the linked list?
  - What would you do to a removed node to ensure it can be reused?

Question

- What happens if we accidentally execute the following?

```
if ( list_head = 0 ) {
    std::cout << "We are at the array[0]" << std::endl;
}
```

`0` list_head

	0	1	2	3	4	5	6	7	8	9
value_	0.0	0.0	9.1	0.0	0.0	4.2	0.0	6.2	0.0	0.0
next_index_	11	11	5	11	11	10	11	2	11	11

Issue

- Once again, we are restricted to the capacity of our array
  - Would it not be better to get each node from the operating system?
  - Benefit: as long as there is memory available,
    we can continue to build our linked list
  - Drawback: asking for new memory is actually sort-of slow... ☹

Summary

- Following this lesson, you now
  - Understand the idea of a node within a linked list
  - Know that it is only necessary to store the first index
  - Know that each node stores the index of the next node
  - Understand how you can add additional nodes to this linked list
  - Have a few questions to work out before your next lecture
  - Know that there are problems with the current design

References

[1] https://en.wikipedia.org/wiki/Linked_list
[2] https://en.wikipedia.org/wiki/Node_(computer_science)

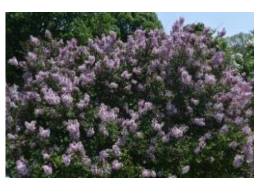
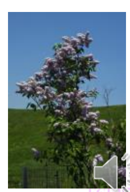
Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.

Disclaimer

These slides are provided for the ECE 150 *Fundamentals of Programming* course taught at the University of Waterloo. The material in it reflects the authors' best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.