User Id : weizhe.goh@digipen.edu | Started : 2020.06.17 09:15:02 | Score : 100%

DigiPen

Member Functions

Assignment

Rules 0 ☑

Read carefully and check all rules you agree with:

- ☐ Your code must represent your own individual work. If something is not clear, ask your instructor for help.
- ☐ Each exercise has description which must be strictly followed.
- ☐ All programs must pass all tests in the main function (when given) to get the final grade. **You are not allowed to make any change in the main function in this case.**
- ☐ Keep the code proper formatted (correct indentation, max line width is 40 characters).

Specs

- ☐ Create a class List that encapsulates structures, data and functions from the previous lab.
- ☐ Make the head as a private member of the class. Initialize the head in the member initialization list.
- ☐ Clean the list in the destructor by calling clear() member function.
- ☐ Implement other member functions that are called in the main function. The first parameter is a zero-based position of the new element, second one (when given) is the element data.
- ☐ Your expected output (without ") is
"1,20,3
"

★ Code 1 ☑

Run

```
#include <iostream>

namespace linkedList
{
  struct Node
  {
    int data;
    Node* next;
  };

  class List
  {
    private:
      Node* head;

    public:
      List():head(nullptr) {}

      int count_Node()
      {
        Node* current = nullptr;
        int count = 0;

        current = head;

        while (current)
        {
          count++;
          current = current->next;
        }
        return count;
      }

      void push_back(int value)
      {
        Node* newNode = new Node;
```

Survey

- What is approximate number of hours you spent implementing this assignment?

5 hours

- Indicate the specific portions of the assignment that gave you the most trouble

Understanding classes, constructor and destructor

```
  Node* current = nullptr;

  newNode->data = value;
  newNode->next = nullptr;

  if (head == nullptr)
  {
    head = newNode;
    return;
  }

  current = head;

  while (current->next)
  {
    current = current->next;
  }
  current->next = newNode;
}

void insertAfter
(int index, int value)
{
  if (index < 0)
  {
    return;
  }

  if(index >= count_Node())
  {
    return;
  }

  Node* newNode = new Node;
  Node* current = nullptr;
  Node* temp = nullptr;

  int count = 0;

  newNode->data = value;
  newNode->next = nullptr;

  current = head;

  while (current)
  {
    if (index == count)
    {
      temp = current->next;
      current->next = newNode;
      newNode->next = temp;
      break;
    }
      current = current->next;
      count++;
  }
}

void remove(int index)
{
  int count = 0;

  if (index < 0)
  {
    return;
  }
```

```
  if(index >= count_Node())
  {
    return;
  }

  Node* deleteNode = nullptr;
  Node* current = nullptr;

  current = head;

  while (current->next)
  {
    if (count == (index-1))
    {
      deleteNode = current->next;

      current->next =
      current->next->next;

      delete deleteNode;
      break;
    }
    current = current->next;
    count++;
  }
}

void print()
{
  Node* current = nullptr;

  if (head == nullptr)
  {
    std::cout << std::endl;
    return;
  }

  current = head;

  while (current)
  {
    if (current->next)
    {
      std::cout << current->data
                << ",";
    }
    else
    {
      std::cout << current->data;
      }
      current = current->next;
    }
    std::cout << std::endl;
}

void clear()
{
  Node* deleteNode = nullptr;
  Node* current = nullptr;

  current = head;

  while (current)
  {
    deleteNode = current;
```

```
          current = current->next;
          delete deleteNode;
        }

        head = nullptr;
      }

      ~List()
      {
        clear();
      }
  };
}

using namespace linkedList;
```

```
int main()
{
  List list;

  list.push_back(1);
  list.push_back(2);
  list.push_back(3);

  // Do not insert when index is
  //   out of range
  list.insertAfter(-100, 20);
  list.insertAfter(100, 20);

  // Insert as a new 3rd element
  list.insertAfter(1, 20);

  // Remove 2nd element
  list.remove(1);

  list.print();
  return 0;
}
```

```
1,20,3
```

By signing this document you fully agree that all information provided therein is complete and true in all respects.	Responder sign: