

EECS2040 Data Structure Hw #3 (Chapter 4 Linked List)

due date 4/25/2021

by 107061123 孫元駿

Part 2 Coding (due 5/13)

You should submit:

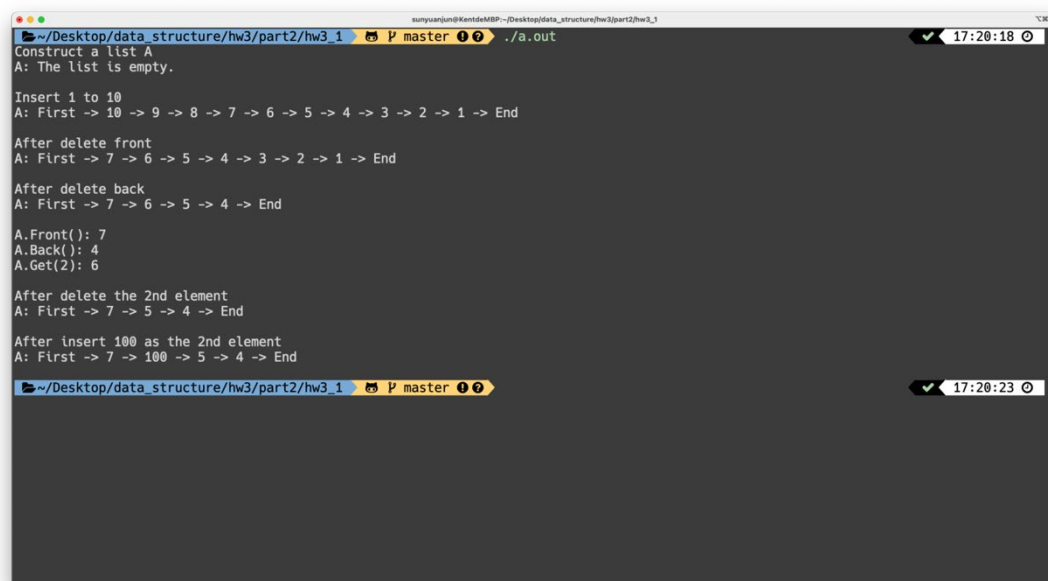
- (a) All your source codes (C++ file).
- (b) Show the execution trace of your program.

1. (30%) Fully code and test the C++ template class List<T> shown in Part 1

Problem 3 above. You must include:

- a. A constructor which constructs an initially empty list.
- b. A destructor which deletes all nodes in the list.
- c. InsertFront() function to insert at the front of the list.
- d. DeleteFront() and DeleteBack() to delete from either end.
- e. Front() and Back() functions to return the first and last elements of the list, respectively.
- f. A function Get(int i) that returns the ith element in the list.
- g. Delete(int i) to delete the ith element
- h. Insert(int i, T e) to insert as the ith element
- i. Overload the output operator << to out put the List object.
- j. As well as functions and forward iterator as shown above.

Write a client program (main()) to **demonstrate** those functions you developed.



```
~/Desktop/data_structure/hw3/part2/hw3_1 master 17:20:18
Construct a list A
A: The list is empty.

Insert 1 to 10
A: First -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1 -> End

After delete front
A: First -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1 -> End

After delete back
A: First -> 7 -> 6 -> 5 -> 4 -> End

A.Front(): 7
A.Back(): 4
A.Get(2): 6

After delete the 2nd element
A: First -> 7 -> 5 -> 4 -> End

After insert 100 as the 2nd element
A: First -> 7 -> 100 -> 5 -> 4 -> End

~/Desktop/data_structure/hw3/part2/hw3_1 master 17:20:23
```

How to use my code:

Just compile it and run it.

First it will create a empty list A and construct it by constructor, and print out the list (show it is empty), then insert 1 to 10 to A and print out A(show all the elements). Then we will delete 3 elements from the front, and 3 elements from the back, and print out the list. After that, we will show the element in the front and show the element in the end of the list and also show the ith (we choose i as 2) element in the list, then delete the ith (we choose i as 2) element. The last, we insert 100 as the ith (we choose i as 2) element, and print out the list.

2. (35%) Develop a C++ class Polynomial to represent and manipulate univariate polynomials with double coefficients (use circular linked list with header nodes). Each term of the polynomial will be represented as a node. Thus a node in this system will have three data members as below.

coef	exp	link
------	-----	------

Each polynomial is to be represented as a circular list with header node. To delete polynomials efficiently, we need to use an **available-space list** and associated functions GetNode() and RetNode() described in Section 4.5. The external (i.e., for input and output) representation of a univariate polynomial will be assumed to be a sequence of integers and doubles of the form: $n, c_1, e_1, c_2, e_2, c_3, e_3, \dots, c_n, e_n$, where e_i represents an integer exponent and c_i a double coefficient; n gives the number of terms in the polynomial. The exponents of the polynomial are in decreasing order.

Write and test the following functions:

- (a) `Istream& operator>>(istream& is, Polynomial& x)`: Read in an input polynomial and convert it to its circular list representation using a header node.
- (b) `Ostream& operator<<(ostream& os, Polynomial& x)`: Convert x from its linked list representation to its external representation and output it.
- (c) `Polynomial::Polynomial(const Polynomial& a)`: copy constructor
- (d) `Const Polynomial& Polynomial::operator=(const Polynomial& a)`
const[assignment operator]: assign polynomial a to *this.
- (e) `Polynomial::~~Polynomial()`: destructor, return all nodes to available-space list
- (f) `Polynomial operator+ (const Polynomial& b) const`: Create and return the

- polynomial *this + b
- (g) Polynomial operator- (const Polynomial& b) const: Create and return the polynomial *this – b
- (h) Polynomial operator* (const Polynomial& b) const: Create and return the polynomial *this * b
- (i) double Polynomial::Evaluate(double x) const: Evaluate the polynomial *this and return the result.

```

~/Desktop/data_structure/hw3/part2/hw3_2 P master 1 1 ./a.out
Input the polynomial(A)
How many element do you want to add: 3
Input by this format "coef exp":
5 5
3 3
1 1
A: 5X^5 + 3X^3 + 1X^1

Input the polynomial(B)
How many element do you want to add: 2
Input by this format "coef exp":
4 4
2 2
B: 4X^4 + 2X^2

Poly C(A): 5X^5 + 3X^3 + 1X^1
D = B: 4X^4 + 2X^2

A + B: 5X^5 + 4X^4 + 3X^3 + 2X^2 + 1X^1
A - B: 5X^5 + -4X^4 + 3X^3 + -2X^2 + 1X^1
A * B: 20X^9 + 22X^7 + 10X^5 + 2X^3

Input a double for Eval: 2
A.Eval(2) = 186

```

How to use my code:

You are going to create two polynomial (A, B) by your own.

First, you construct two polynomial by entering the number of elements and the coeff and exp of the elements.

Then, the program will create a new polynomial C by using copy constructor (copy A), and also create a empty polynomial D and give D value by D = B, then show both polynomial.

After that, the program will show A + B, A – B and A * B by using overload operator.

The last, you can insert a number, and eval it, then the program will tell you the result.

3. (35%) The class definition for sparse matrix in Program 4.29 is shown below.

```

struct Triple{int row, col, value;};
class Matrix; // 前向宣告
class MatrixNode {

```

```

friend class Matrix;
friend istream& operator>>(istream&, Matrix&); // 為了能夠讀進矩陣
private:
    MatrixNode *down , *right;
    bool head;
    union { // 沒有名字的 union
        MatrixNode *next;
        Triple triple;
    };
    MatrixNode(bool, Triple*); // 建構子
}

MatrixNode::MatrixNode(bool b, Triple *t) // 建構子
{
    head = b;
    if (b) {right = down = this;} // 列/行的標頭節點
    else triple = *t; // 標頭節點串列的元素節點或標頭節點
}

class Matrix{
friend istream& operator>>(istream&, Matrix&);
public:
    ~Matrix(); // 解構子
private:
    MatrixNode *headnode;
};

```

Based on this class, do the following tasks.

- (a) Write the C++ function, **operator+(const Matrix& b) const**, which returns the matrix ***this + b**.
- (b) Write the C++ function, **operator*(const Matrix& b) const**, which returns the matrix ***this * b**.
- (c) Write the C++ function, **operator<<()**, which outputs a sparse matrix as triples (i, j, a_{ij}).
- (d) Write the C++ function, Transpose(), which transpose a sparse matrix.
- (e) Write and test a copy constructor for sparse matrices. What is the computing time of your copy constructor?

Write a client program (main()) to **demonstrate** those functions you developed.

```
~/Desktop/data_structure/hw3/part2/hw3_3 P master 23:05:40
Construct matrix A
Numbers of rows: 3
Numbers of cols: 3
Numbers of terms: 6
Input the 1th terms ( row, col, value ): 0 0 1
Input the 2th terms ( row, col, value ): 0 1 2
Input the 3th terms ( row, col, value ): 0 2 3
Input the 4th terms ( row, col, value ): 2 0 4
Input the 5th terms ( row, col, value ): 2 1 5
Input the 6th terms ( row, col, value ): 2 2 6
A: (0, 0, 1) (0, 1, 2) (0, 2, 3) (2, 0, 4) (2, 1, 5) (2, 2, 6)

Construct matrix B
Numbers of rows: 3
Numbers of cols: 3
Numbers of terms: 3
Input the 1th terms ( row, col, value ): 0 0 10
Input the 2th terms ( row, col, value ): 1 1 20
Input the 3th terms ( row, col, value ): 2 2 30
B: (0, 0, 10) (1, 1, 20) (2, 2, 30)

A + B = C: (0, 0, 11) (0, 1, 2) (0, 2, 3) (1, 1, 20) (2, 0, 4) (2, 1, 5) (2, 2, 36)
A * B = D: (0, 0, 10) (0, 1, 40) (0, 2, 90) (2, 0, 40) (2, 1, 100) (2, 2, 180)
Transpose E: (0, 0, 10) (0, 2, 40) (1, 0, 40) (1, 2, 100) (2, 0, 90) (2, 2, 180)

Copy matrix 1 times spend 91/100000 ms
~/Desktop/data_structure/hw3/part2/hw3_3 P master 23:06:33
```

How to use my code:

You are going to create two matrix (A, B) by your own.

First, you will constructor two matrix, by entering the number of rows, the number of column, the number of terms and all the terms (row, col, value).

Then, the program will create matrix C, that $C = A + B$, and create matrix D, that $D = A * B$, and matrix E, that E is the transpose D.

Last, program will computing time of your copy constructor 100000 times and calculate how much time it speed in one time.