Class CS\_UC211 Number 202115030121 Name Hao Yang Data 2022/11/22

**Experiment topic：**

Design and implementation of Univariate Polynomial Calculator

**1. Problem analysis**

1.1 Statement

In scientific research, there are two different computational problems, namely numerical computation and symbolic computation. Symbolic computation is widely used and challenging. Real coefficient univariate polynomial computation is one of the most basic symbolic computation tasks. After learning the linked list of data structures, you can perform simple operations on univariate polynomials. At the same time, it can also enhance personal proficiency in the linked list.

1.2 Abstraction

Polynomial is an algebraic expression composed of several monomials. Each monomial is composed of coefficients and exponential powers. For a polynomial, each of its coefficients is not related to the exponential power.

1.3 ADT design

Consider that each coefficient of the polynomial has no correlation with the exponential power and that the possible difference between adjacent exponential powers is huge. Therefore, chain storage is considered for the storage structure.

Start with a class name named "Po", which stores the coefficients and exponential powers of the mononomial, and points to the next mononomial element with the next pointer.

The polynomial is then stored in a class named Polynomial, where the Prime Minister defines the first node of the polynomial, traverses it with the tail pointer, and then joins the class named Po with a chain table to complete the storage of the polynomial.

**2.** **Experimental scheme**

2.1 Storage Scheme

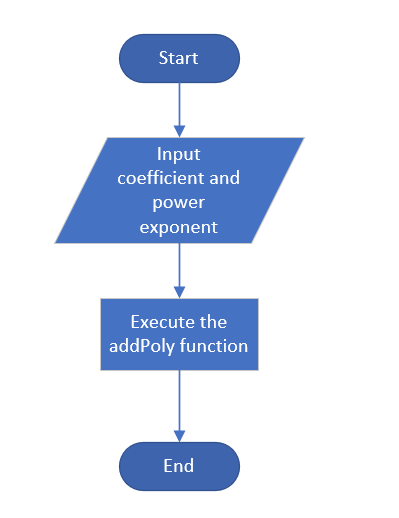
Sequential storage is a convenient structure for storing polynomials, but it wastes space when the exponential power of an item varies greatly, and it is at the O(n) level for adding and deleting elements of a polynomial. The chain storage structure uses the form of a chain table, which does not waste space. At the same time, the size of the polynomial can be unlimited. It is more convenient to add and delete elements of the polynomial.

2.2 Algorithm design

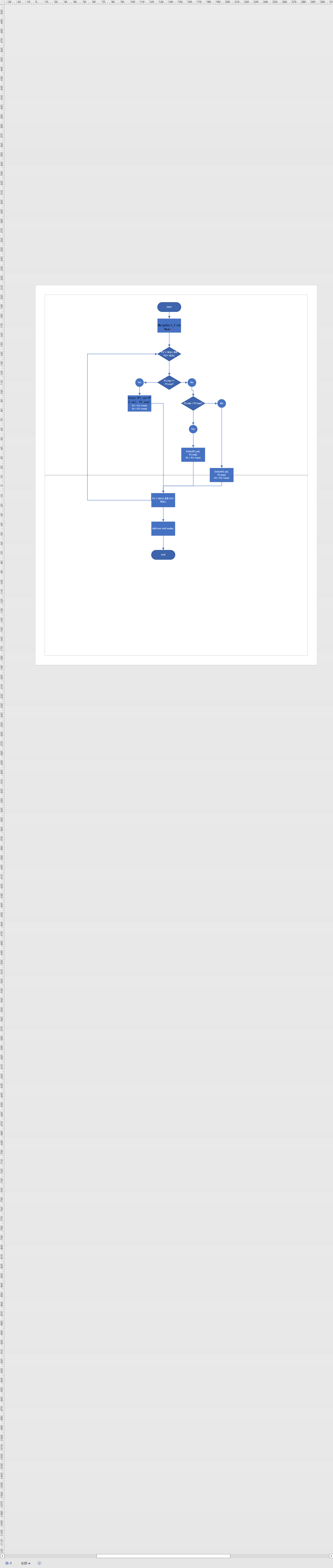
*  Adds a single term to the polynomial.



*  Adds a single term to the polynomial.



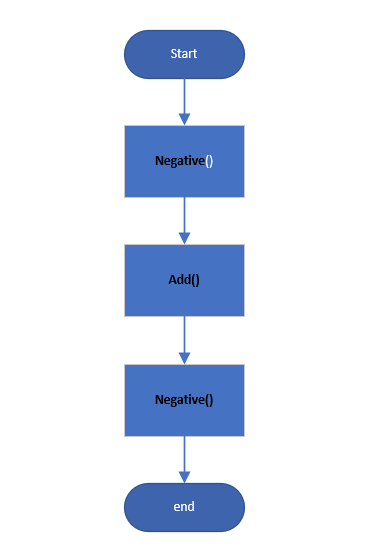
* Merge and sort the linked lists. Divide the linked list into two parts with the fast or slow pointer until there is only one node in the linked list. Then add a head or tail pointer to the two linked lists, combine the two linked lists into a sequential linked list, and then repeat this step all the time. Finally, return the chain header pointer.
* void add(const Polynomial& P1, Polynomial& P2)



* negative( Polynomial& P)

Go through the polynomial, multiply the coefficient by - 1, and convert the tail to the opposite number.

* subtract(const Polynomial& P1, Polynomial& P2)



* multiply(const Polynomial& P1, const Polynomial& P2)

Two polynomials, with the first polynomial as the benchmark. Each time it points to the next node of the first polynomial, it is multiplied with the second polynomial in turn. Then we use the sorting algorithm defined once again to sort and de-duplicate the polynomials obtained.

2.3 Test scheme

Use the random number library in C++ to generate the coefficient or exponential power of a polynomial every second and store it in a new text file named "Text File". Then define a polynomial array and store the polynomial data in the polynomial array.

Add, subtract and multiply different polynomial arrays, and store the results in a new text file named "resultText.txt". At the same time, add, subtract and multiply for another 1000 cycles, and save this time to the text file named "hasadd.txt".

**3. Task solutions**

**Analyzes the expanding task in the task book, and gives the key points of implement.**

**Complete at least 2 questions and list references.**

1. How to realize the division of univariate polynomial？

For simple polynomial division, consider the operation similar to multiplication. For the division between complex polynomials, because of the remainder, it will lead to the loss of precision in the division, so we have not come up with a better solution.

1. How to determine whether a univariate polynomial can be factorized and how to implement it?

Factorize polynomials with undetermined coefficients or determinants.

1. How to solve the problem of sign operation of binary polynomials?

Declare another variable in the class. Each time you input a polynomial, you will judge which variable is being input. Other operations are the same as those for univariate polynomials.

1. If univariate polynomials are armored by array, how to implement the multiplication, and how about its efficient

Using an array to simulate a linked class array to simulate linked lists is faster than using linked lists because opening up space wastes a lot of time when the data is large. But for arrays, it is a fixed size, so it is difficult to increase the polynomial length.

1. 5. The index of the accumulated items generated in the multiplication operation is usually piecewise increasing. Can the multiplication algorithm be optimized by reducing the time consumption of inserting the add item?

I think it's very difficult. Maintain one more coefficient and exponential power.

1. How to deal with the univariate polynomial with the standard format?

For a polynomial whose exponential power is unordered, I will sort the polynomial and convert it into a standard polynomial. But for polynomials with brackets, consider using stacks and queues of data structures to complete the transformation to standard polynomials.