資料結構-hw2

張帟淇

41243129

November 30, 2024

內容

解題說明

實作多項式類別，類別內容如下:

```
class Polynomial{
    private:
        Term *termArray;  //陣列本體
        int capacity;     //陣列大小
        int terms;        //陣列內非零數字
    public:
        Polynomial():capacity(2),terms(0){
            termArray = new Term[capacity];
        }

        Polynomial Add(Polynomial b);

        Polynomial Mult(Polynomial b);

        float Eval(float x);

        void newTerm(const float newcoef,const int newexp);

        friend istream& operator>>(istream &input,Polynomial &Poly);
        friend ostream& operator<<(ostream &output,const Polynomial &Poly);
};
```

Term 類別:

```
class Term{
    friend Polynomial;
    friend ostream& operator<<(ostream &output,const Polynomial &Poly);
    private:
        float coef;  //係數
        int exp;     //指數
};
```

程式實作

Add 函式

```
Polynomial Polynomial::Add(Polynomial b){
    Polynomial c;
    int aPos=0,bPos=0;
    while((aPos<terms)&&(bPos<b.terms))
        if(termArray[aPos].exp==b.termArray[bPos].exp){            //如果指數相等就相加
            float t=termArray[aPos].coef+b.termArray[bPos].coef;    //係數相加
            if(t) c.newTerm(t,termArray[aPos].exp);
            aPos++; bPos++;                                        //往下一個數字移動
        }
        else if(termArray[aPos].exp<b.termArray[bPos].exp){        //如果b的指數較大
            c.newTerm(b.termArray[bPos].coef,b.termArray[bPos].exp);//就把b的係數與指數加入到結果多項式
            bPos++;
        }
        else{
            c.newTerm(termArray[aPos].coef,termArray[aPos].exp);    //把a的係數與指數加入到結果多項式
            aPos++;
        }

    for(;aPos<terms;aPos++)
        c.newTerm(termArray[aPos].coef,termArray[aPos].exp);        //把剩餘的項加進去
    for(;bPos<b.terms;bPos++)
        c.newTerm(b.termArray[bPos].coef,b.termArray[bPos].exp);    //把剩餘的項加進去
    return c;                                                       //返回結果多項式
}
```

Mult 函式

```
Polynomial Polynomial::Mult(Polynomial b){
    Polynomial c;
    for(int aPos=0;aPos < terms;aPos++){
        for(int bPos=0;bPos<b.terms;bPos++){
            float newcoef=termArray[aPos].coef*b.termArray[bPos].coef;  //係數相乘
            int newexp=termArray[aPos].exp+b.termArray[bPos].exp;       //指數相加

            //檢查結果多項式中是否已有相同的指數
            bool found=false;
            for(int cPos=0;cPos<c.terms;cPos++){
                if(c.termArray[cPos].exp == newexp){  //如果已有相同的指數
                    c.termArray[cPos].coef+=newcoef;  //係數相加
                    found=true;
                    break;
                }
            }

            if(!found){                                 //如果沒有找到相同的指數,就新增
                c.newTerm(newcoef,newexp);
            }
        }
    }

    //排序,確保按指數順序排列
    for(int i=0;i<c.terms-1;i++){
        for(int j=i+1;j<c.terms;j++){
            if(c.termArray[i].exp<c.termArray[j].exp){  //按指數從大到小排序
                //開始交換
                Term temp=c.termArray[i];
                c.termArray[i]=c.termArray[j];
                c.termArray[j]=temp;
            }
        }
    }

    return c;
}
```

程式實作

Eval 函式

```
float Polynomial::Eval(float x){
    float sum=0;
    for(int aPos=0;aPos<terms;aPos++){
        sum=sum+termArray[aPos].coef*pow(x,termArray[aPos].exp);  //把每個數都計算出來並相加
    }
    return sum;                                                   //回傳結果
}
```

newTerm 函式

```
void Polynomial::newTerm(const float theCoef,const int theExp){
    if (theCoef == 0) return;
    if(terms==capacity){
        capacity*=2;
        Term *temp=new Term[capacity];
        copy(termArray,termArray+terms,temp);
        delete []termArray;
        termArray=temp;
    }
    termArray[terms].coef=theCoef;
    termArray[terms++].exp=theExp;
}
```

運算子多載　輸出

```
ostream& operator<<(ostream &output,const Polynomial &Poly){
    for(int aPos=0;aPos<Poly.terms;aPos++){
        if(Poly.termArray[aPos].coef>0 && aPos!=0){
            output<<"+";                                  //只有第1項後+才會顯示
        }

        if(Poly.termArray[aPos].coef<0){
            output<<"-";                                  //如果係數為負就加負號
        }
        if(abs(Poly.termArray[aPos].coef)!=1 || Poly.termArray[aPos].exp==0){
            output<<abs(Poly.termArray[aPos].coef);
        }

        if(Poly.termArray[aPos].exp!=0){          //只要指數不為零就輸出x
            output<<"X";
            if(Poly.termArray[aPos].exp!=1){      //如果次方為1就不輸出^這個符號
                output<<"^"<<Poly.termArray[aPos].exp;
            }
        }
    }
    return output;
}
```

程式實作

運算子多載　輸入

```
istream& operator>>(istream &input,Polynomial &Poly) {
    string line;
    getline(input,line);

    istringstream stream(line);
    char ch;
    int coef=0, exp=0;
    bool isNegative=false;
    bool coefSet=false;  //用來標記係數是否設定

    while(stream>>ch){
        if(isdigit(ch)){                    //檢查是否為數字
            stream.putback(ch);             //放回去後續讀取完整數字
            stream>>coef;
            if(isNegative)coef=-coef;
            isNegative=false;
            coefSet=true;                   //標記係數已設定
        }else if(ch=='X'){                  //處理變數x
            if(!coefSet){                   //如果沒有設定係數
                coef=isNegative ? -1 : 1;   //默認係數為正負1
                isNegative = false;
            }
            if(stream.peek()=='^') {
                stream.get();               //跳過'^'
                stream>>exp;                //讀取次方
            }else{
                exp=1;                      //如果沒寫次方,當作是X^1
            }
            coefSet=false;                  //處理完後重置
        }else if(ch=='+' || ch=='-'){       //判斷運算子
            if(coef!=0 || exp!=0){          //確保不儲存多餘項
                Poly.newTerm(coef,exp);
            }
            coef=0;
            exp=0;
            isNegative=(ch=='-');
        }
    }

    //加入最後一項   如果有的話
    if(coef!=0 || exp!=0){
        Poly.newTerm(coef,exp);
    }

    return input;
}
```

效能分析

時間複雜度

Add:O(aPos+bPos)

Mult:O(aPosXbPos+i^2)

Eval:O(aPos)

空間複雜度

Add: O(aPos+bPos)

Mult:O(aPosXbPos)

Eval:O(1)

測試與驗證

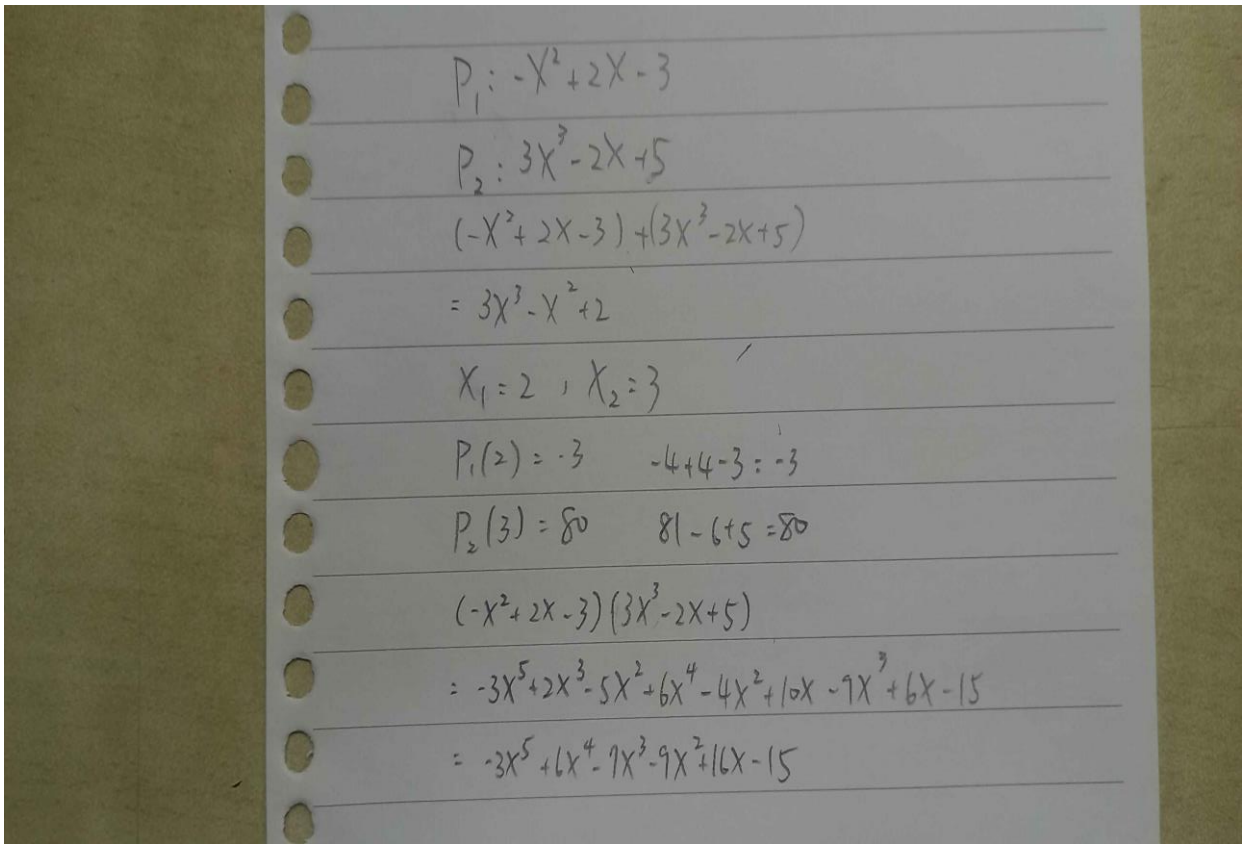
主程式:

```
int main(){
    Polynomial p1;  //第一個多項式
    Polynomial p2;  //第二個多項式
    float x1,x2;
    cout<<"請以這樣的格式輸入5X^4+X^2+1"<<endl;
    cout<<"請輸入多項式"<<endl<<"p1:";
    cin>>p1;        //輸入第一個多項式
    cout<<"p2:";
    cin>>p2;        //輸入第二個多項式
    cout<<"請輸入欲代入多項式的x值"<<endl<<"x1:";
    cin>>x1;        //輸入x1
    cout<<"x2:";
    cin>>x2;        //輸入x2
    //計算p1代入x1的值並印出
    cout<<"p1:"<<p1<<"代入"<<x1<<"的值為:"<<p1.Eval(x1)<<endl;
    //計算p2代入x2的值並印出
    cout<<"p2:"<<p2<<"代入"<<x2<<"的值為:"<<p2.Eval(x2)<<endl;
    //計算p1+p2的值並印出
    Polynomial p3=p1.Add(p2);
    cout<<p1<<" 和 "<<p2<<" 相加為:"<<p3<<endl;
    //計算p1xp2的值並印出
    Polynomial p4=p1.Mult(p2);
    cout<<p1<<" 和 "<<p2<<" 相乘為:"<<p4<<endl;
    return 0;
}
```

輸出結果:

```
請以這樣的格式輸入5X^4+X^2+1
請輸入多項式
p1:-X^2+2X-3
p2:3X^3-2X+5
請輸入欲代入多項式的X值
x1:2
x2:3
p1:-X^2+2X-3代入2的值為:-3
p2:3X^3-2X+5代入3的值為:80
-X^2+2X-3 和 3X^3-2X+5 相加為:3X^3-X^2+2
-X^2+2X-3 和 3X^3-2X+5 相乘為:-3X^5+6X^4-7X^3-9X^2+16X-15
```

測試與驗證


$P_1: -X^2+2X-3$
$P_2: 3X^3-2X+5$
$(-X^2+2X-3)+(3X^3-2X+5)$
$=3X^3-X^2+2$
$X_1=2, X_2=3$
$P_1(2)=-3$ $-4+4-3=-3$
$P_2(3)=80$ $81-6+5=80$
$(-X^2+2X-3)(3X^3-2X+5)$
$=-3X^5+2X^3-5X^2+6X^4-4X^2+10X-9X^3+6X-15$
$=-3X^5+6X^4-7X^3-9X^2+16X-15$

申論及開發報告

Add 函式

主要是看指數是否相等或是哪邊比較大，大的會先新增到陣列裡，如果一樣就相加再加入陣列。

Mult 函式

相乘的函示比較麻煩一點，首先把兩個多項式的每一項都相乘送到 Polynomial C 裡面，再把同指數的數字相加，然後排序後再新增到陣列裡面。

Eval 函式

把每一項帶入 x 的值算出來，然後加總。

運算子多載 輸入

輸入多項式後，檢查是否為數字，如果是的話就把係數輸入進去，如果是 X 的話就先檢查是正 X 還是負 X 再把^跳過然後輸入指數，如果不是 X 的話就檢查是否為+或-且係數和指數不為零就把這一項加到陣列裡，再把最後一項加入。

運算子多載 輸出

如果係數為正就輸出'+'符號但第一項不輸出，為負就輸出'-'符號，只要係數不為 1 或指數不為 0 就會把係數項印出，且只要指數不為 0 就會輸出'X'如果指數大於 1 就會再輸出'^'次方符號和指數項數字。

這隻程式最麻煩的部分就是多載輸入那邊，因為要考慮的情況太多了，要使用很多判斷式，像是他是不是為負，如果這一項只有一個 X 的話要先檢查他是否為正或是負，如果有地方弄錯的話最後輸出結果就會變成輸出 0 像是:5X^3-4X+1 會變成 5X^304X+1，這部分就用了很久。