

по дисциплине Практикум по программированию

Выполнил студент группы 23533/2

Руководитель

ассистент

В.Э. Ковалевский

ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого» Институт компьютерных наук и технологий

Высшая школа киберфизических систем и управления

Санкт-Петербург 2019

30.05.2019

А.А. Тарасов

Отчет по лабораторной работе

Друзья класса

Друзья класса – это классы или функции, имеющие доступ к private членам другого класса. Часто требуется создавать дружественные классу функции для реализации перегрузки оператора. Хороший пример – перегрузка оператора «+» для класса, представляющего полином. Легко написать метод

Polynomial operator + (const Polynomial<Coef>& P) const;

Если А и Б – полиномы, то А+Б вернёт В – также полином. Если Вы хотите сложить полином и число, то достаточно либо написать шаблонный метод

//Сложение полинома с любым типом

template<typename T>

Polynomial<Coef> operator + (const T& op) const;

либо определить функцию неявного преобразования встроенного типа в тип полинома

//Для инициализации единственным числом

template<typename T>

Polynomial(const T& T\_object);

чтобы для преобразованного числа вызовется обычная перегрузка сложения для двух полиномов. Это выглядит наиболее приемлемым способом. Будьте осторожны. Пусть будет перегруженная функция function(Polynomial input) / function(int input). В этом случае возникнет неоднозначность, так как обе функции могут быть вызваны. Если Вы хотите отменить неявные преобразования, обозначьте преобразовывающий конструктор класса explicit.

В моей работе – разработке шаблонного класса полином, пришлось воспользоваться дружественными функциями для удобства.

Реализация сложения полиномов, или сложения полинома и какого-то типа

//Сложение полиномов, или полинома + тип A+B //Тут происходит неявное преобразование к типу Polynomial<Coef>

template<typename Coef>

Polynomial<Coef> Polynomial<Coef>::operator + (const Polynomial<Coef>& P) const

{

int i = this->power; // "указатель" на члены 1го многочлена, начиная с конца

int j = P.power; // "указатель" на члены 2го многочлена, начиная с конца

Polynomial<Coef> Temp\_Poly; //Его будем возвращать по значению

int counter = Temp\_Poly.power = i > j ? i : j; //Счетчик показывает мах порядок

delete[] Temp\_Poly.coef\_arr; //Освободить память, выделенную под 0;

Temp\_Poly.coef\_arr = new Coef[Temp\_Poly.power+1]; //Выделим память под все элементы нового полинома

while (i >= 0 && j >= 0) //Покуда не дойдём до старшего члена многочлена

{

Temp\_Poly.coef\_arr[counter] = this->coef\_arr[i] + P.coef\_arr[j]; //Сложить почленно

counter--; i--; j--;

}

//Теперь довнесём оставшиеся члены.(сложение с 0)

if(i>j)

while (counter >= 0)

Temp\_Poly.coef\_arr[counter--] = this->coef\_arr[i--];

else

{

while (counter >= 0)

Temp\_Poly.coef\_arr[counter--] = P.coef\_arr[j--];

}

Temp\_Poly.Normalize();

return Temp\_Poly;

}

Реализация сложения какого-то типа и полинома потребовала бы перегрузки метода сложения для типа-слагаемого, что сложно осуществить, если Вы не разработчик того типа, а если и возможно, то это практически бесполезно. Задача состоит в том, чтобы научить чужой класс взаимодействовать с своим классом

Для этого используют дружественные функции. Код элементарно прост, если помнить свойства сложения. Функции-друзья не являются членами класса, но имеют доступ к private полям класса.

//Сложение любого типа с полиномом

template<typename T>

friend Polynomial<Coef> operator + (const T& op, const Polynomial<Coef>& P);

//Сложение любого типа и полинома type+A

template<typename Coef, typename T>

Polynomial<Coef> operator+(const T& op, const Polynomial<Coef>& P)

{

return P+op;

//Тут вызовется Polynomial<Coef>::operator + (const Polynomial<Coef>& P)

}

Вывод: Друзья класса – мощное средство, необходимое для объектно-ориентированного программирования. Оно позволяет, как минимум, сократить объём кода, а также позволяет модифицировать свой класс для работы с различными типами.