МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

Отчет по лабораторной работе №4.

По ОБЪЕКТО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ

«Параметризованные классы»

Выполнила:

студентка гр. ПМ-21-2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бутусова В.М.

Проверил:

доц., к.п.н. кафедры АСУ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кургасов В. В.

Липецк 2022

Цель работы:

Изучить механизм параметрического полиморфизма на основе создания и использования параметризованных классов.

Задание кафедры:

Реализовать на языке C++ параметризованный класс «Матрица», типы элементов которого могут быть заданы в соответствии с вариантом.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Тип 1 | Тип 2 | Возможность класса |
| 3 | Комплексное число. | Вектор в полярных координатах. | Умножение матрицы на вещественное число и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A\*b, где b - вещественное число. |

Код программы:

#include <iostream>

#define \_USE\_MATH\_DEFINES

#include <math.h>

using namespace std;

class Complex // класс для матрицы комплексного типа

{

double a, b;

public:

static Complex load() // статическая функция ввода элементов матрицы

{

Complex val;

double a1, b1;

cin >> a1 >> b1;

val.a = a1;

val.b = b1;

return val;

}

Complex operator \*(double b) // перегрузка оператора умножения

{

Complex val;

val.a = this->a \* b;

val.b = this->b \* b;

return val;

}

void print() // функция вывода комплексного числа

{

cout <<"(" <<a << "+" << b << "\*i) ";

}

};

class Polar // класс для матрицы полярного типа

{

double r, p;

double x, y;

public:

static Polar load() // статическая функция ввода элементов матрицы

{

Polar val;

double a1, b1;

cin >> a1 >> b1;

val.r = a1;

val.p = b1;

return val;

}

Polar operator \*(double b) // перегрузка оператора умножения

{

Polar val;

x = this->r \* cos(this->p\*M\_PI/180) \* b; // перевод в декартову систему координат

y = this->r \* sin(this->p\*M\_PI/180) \* b; // и умножение на вещественное число b

val.p = atan((x / y) \* M\_PI / 180); // обратный перевод в полярную систему координат

val.r = x / cos(val.p \* M\_PI / 180);

return val;

}

void print() // функция вывода радиуса и угла

{

cout << "(" << r << ", " << p << ") ";

}

};

template<class T> //создание шаблона класса, хранящего матрицу из объектов типа T

class Matrix

{

T m[3][3]; // создание массива элементов типа T

public:

Matrix() = default;

void Load() // функция записи элементов матрицы

{

int cnt = 1;

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

cout << cnt << ") ";

m[i][j] = T::load(); // вызов статической функции записи элементов из конкретного типа данных

cnt++;

}

}

}

Matrix operator \*(double b) // перегрузка операции умножения

{

Matrix<T> val;

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

val.m[i][j] = this->m[i][j] \* b;

}

}

return val;

}

void print() // функция вывода матрицы на экран

{

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

m[i][j].print(); // вызов функции вывода элементов из конкретного типа данных

}

cout << "\n";

}

}

};

int main()

{

setlocale(LC\_ALL, "");

int n;

cout << "Введите 1, если матрица размерности 3 будет состоять из комплексных чисел" << endl;

cout << "Введите 2, если матрица размерности 3 будет состоять из полярных координат" << endl;

cin >> n;

double b;

if (n == 1)

{

Matrix<Complex> A;

cout << "Введите значения a и b комплексной формы (a + b\*i):" << endl;

A.Load();

cout << "Введите число для умножения матрицы: " << endl;

cin >> b;

Matrix<Complex> C = A \* b;

cout << "Исходная матрица:\n";

A.print();

cout << "Производная матрица:\n";

C.print();

}

if (n == 2)

{

Matrix<Polar> A;

cout << "Введите параметры длины r и угла phi:" << endl;

A.Load();

cout << "Введите число для умножения матрицы: " << endl;

cin >> b;

Matrix<Polar> C = A \* b;

cout << "Исходная матрица в полярной системе (r, phi):\n";

A.print();

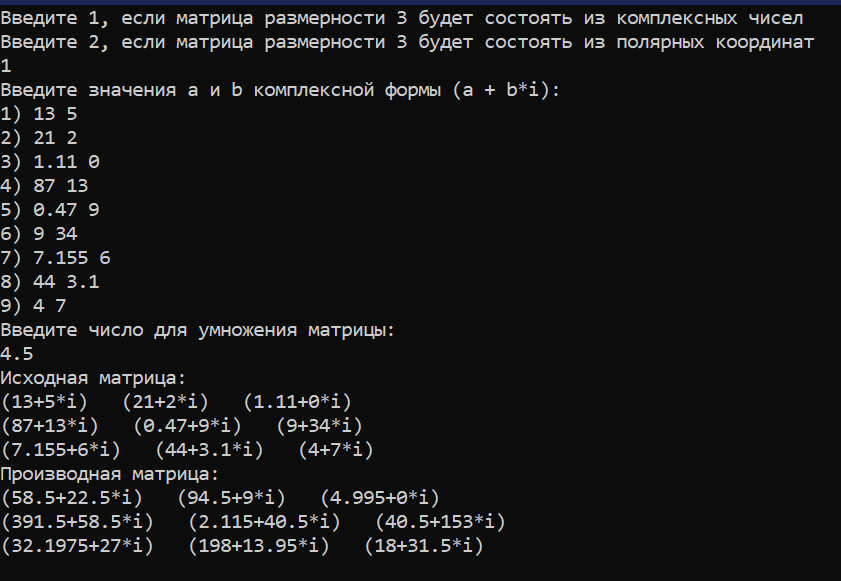
cout << "Производная матрица в полярной системе (r, phi):\n";

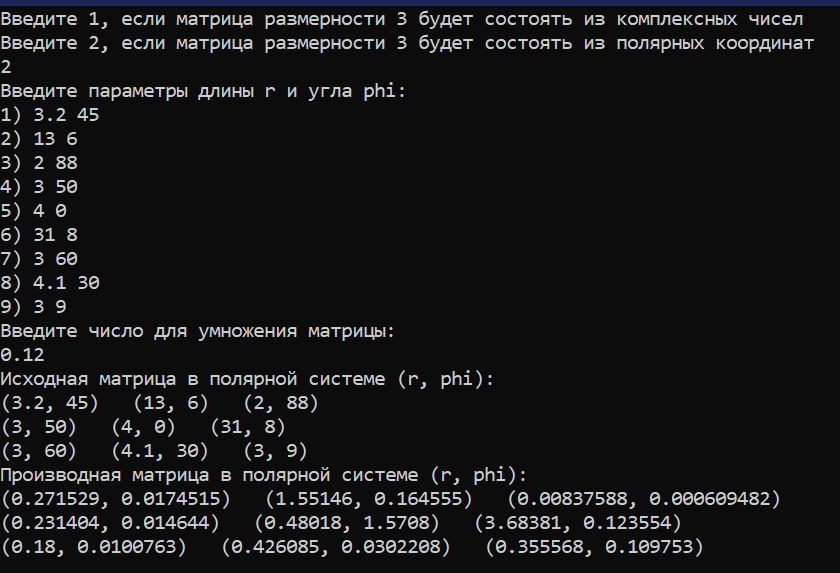
C.print();

}

}

Пример работы программы:





Вывод:

В данной лабораторной работе я изучила механизм параметрического полиморфизма на основе создания и использования параметризованных классов, и реализовала на языке C++ параметризованный класс «Матрица», типы элементов которого могут быть заданы в комплексной форме или в форме вектора в полярных координатах.

Ответы на контрольные вопросы:

**1. Зачем нужны параметризованные классы?**

Параметризованные классы используются для создания семейства родственных функций и классов. Это позволяет автоматизировать создание функций и классов, способных обрабатывать данные различного типа.

**2. Какой из механизмов ООП реализуется с использованием параметризованных классов?**

С использованием параметризованных классов реализуется такой механизм ООП, как параметризованный полиморфизм.

**3. Что такое шаблон класса?**

Средство создания параметризованных классов, обобщённое описание алгоритма без привязки к некоторым конкретным параметрам, которые не могут быть известны заранее.

Для применения шаблонов перед классом указывается ключевое слово template, после которого идут угловые скобки, а в угловых скобках после слова typename идет параметр шаблона (template <typename T>).

Параметр шаблона представляет произвольный идентификатор. То есть в данном случае параметр T будет представлять некоторый тип, который становится известным во время компиляции. Это может быть любой другой тип данных. И идентификатор счета будет представлять тип, который передается через параметр T.

**4. Во сколько раз сокращается исполняемый машинный код программы с использованием параметризованных классов?**

По идее, шаблоны предоставляют краткую форму записи участка кода, но на самом деле их использование не сокращает исполняемый код, так как для каждого набора параметров компилятор создаёт отдельный экземпляр функции или класса.

**5. Какого вида могут быть параметры при создании шаблона класса?**

Параметром при задании шаблона класса может выступать как некоторый класс, обозначаемый ключевым словом class, так и некоторый неизвестный заранее тип данных, который описывается ключевым словом typename.