Минобрнауки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тульский государственный университет»

Институт прикладной математики и компьютерных наук

Кафедра информационной безопасности

Языки программирования

Отчет по выполнению лабораторной работы № \_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Вариант №\_\_

Выполнил

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тула 202\_

**Цель лабораторной работы:**

Изучение принципов создания и использования иерархии классов в программах на С++.

**Задание на работу:**

Разработайте иерархию классов по своему варианту (или по собственному выбору). Кроме указанных в варианте задания свойств и методов, можно добавить свои, необходимые по смыслу предметной области, свойства и методы классов.

Минимальные требования:

* не менее двух виртуальных функций,
* не менее трех свойств у классов-потомков;
* не менее трех методов;
* наличие конструкторов у всех классов.

Составьте диаграмму классов и покажите ее для согласования преподавателю. После этого реализуйте составленную иерархию классов на языке С++ (в виде подключаемых .h или .cpp файла или в виде DLL).

Алгебраическая функция, линейная функция, квадратичная функция, еще какая-нибудь функция.

**Ход выполнения работы**

**Текст файла «funcs.h»:**

#include <stdio.h>

#include <cmath>

using namespace std;

class Function {

public:

virtual double calculate(double x) = 0;

virtual void printF() = 0;

virtual void calculateF(double x) = 0; };

class Powered {

protected: double x; int power;

public:

Powered(double xX = 1, int pow = 1) : x(xX), power(pow) {}

double calculate() const {

double ret = 1;

for (int i = 0; i < power; i++) ret \*= x;

return ret; }

void printF(){printf("Powered function: %f ^ %d\n", x, power);}

~Powered(){}

Powered operator=(Powered a){ x = a.x; power = a.power;

return \*this; }

void calculateF() {

printf("Function: "); printF();

printf("F(%f) = %f\n", x, calculate()); }

};

class Algebraic : public Function {

protected: double arg, adds; int power;

public:

Algebraic(double Farg = 1, int pow = 1, double add = 0) :

arg(Farg), adds(add), power(pow) {};

double calculate(double x) override {

Powered p(x, power);

return p.calculate() \* arg + adds; }

void printF() override {

printf("Algebraic function: ");

if (!(abs(arg - 1) < 0.001))printf("%f \*", arg);

printf("x");

if (power != 1)printf(" ^ %d", power);

if (!(adds > -0.001 && adds < 0.001)) {

if (adds > 0)printf(" +");

printf("%f", adds); }

printf("\n");

}

~Algebraic(){}

Algebraic operator=(Algebraic a) {

arg = a.arg;

adds = a.adds;

power = a.power;

return \*this; }

void calculateF(double x) {

printf("Function: "); printF();

printf("F(%f) = %f\n", x, calculate(x)); }

};

class Trigonometry : public Function {

protected: double prearg = 1, inarg = 1; int xpow = 1, fpow = 1;

public:

enum type {sinus = 1, cosinus, tangens, cotangens } Ft;

Trigonometry(double FunctionArgument = 1, double XArgument = 1, int XPower = 1, int FunctionPower = 1, type FunctionType = (type)1) :

prearg(FunctionArgument), inarg(XArgument), xpow(XPower), Ft(FunctionType), fpow(FunctionPower) {};

double calculate(double x) override {

Powered XUncut(x, xpow);

double FUncut = 1;

switch (Ft) {

case 1: FUncut = sin(XUncut.calculate()); break;

case 2: FUncut = cos(XUncut.calculate()); break;

case 3: FUncut = tan(XUncut.calculate()); break;

case 4: FUncut = 1 / tan(XUncut.calculate()); break;

default: printf("\nError with Function-type\n"); break;}

Powered Func(FUncut, fpow);

return Func.calculate();

}

void printF() override {

if (!(abs(prearg - 1) < 0.001)) printf("%f \* ", prearg);

switch (Ft) {

case 1: printf("sin("); break;

case 2: printf("cos("); break;

case 3: printf("tan("); break;

case 4: printf("ctg("); break;

default: printf("\nError with Function-type\n"); break;}

if (!(abs(inarg - 1) < 0.001)) printf("%f \* x", inarg);

if (xpow != 1)printf(" ^ %d", xpow);

printf(")");

if (fpow != 1)printf(" ^ %d", fpow);

printf("\n"); }

~Trigonometry(){}

Trigonometry operator=(Trigonometry a) {

fpow = a.fpow; xpow = a.xpow;

Ft = a.Ft; prearg = a.prearg;

inarg = a.inarg; return \*this; }

void calculateF(double x) {

printf("Function: "); printF();

printf("F(%f) = %f\n", x, calculate(x)); }

};

class Logar2: public Function {

protected: double arg;

public: Logar2 (int a = 2) : arg(a) {}

double calculate(double x = 2.0) {

arg = (abs((arg - 2)) < 0.001 ? x : arg); return log2(arg);}

~Logar2(){}

void printF() override {

printf("Log2 function: log2(%f) = %f\n", arg, this->calculate());}

Logar2 operator=(Logar2 a) {arg = a.arg; return \*this; }

void calculateF(double x) {

if (x <= 0) {

printf("Unsupported argument in function call. Argument = 2.\n");

x = 2; }

arg = x; printf("Function: "); printF();

printf("F(%f) = %f\n", arg, calculate(arg)); }

};

**Текст файла «s03\_04.cpp»:**

#include <iostream>

#include "funcs.h"

using namespace std;

int main() {

Algebraic A1(3, 2, 0), A2(9.5, 10, 6.9);

Trigonometry

T1(4, 6, 7, 2, Trigonometry::type(1)),

T2(2, 5, 1, 1, Trigonometry::type(2));

Logar2 L1(3), L2(12);

Powered P1(6, 7), P2(1, 100);

A1.calculateF(12.5); A2.calculateF(4.5);

T1.calculateF(7.3); T2.calculateF(-3.3);

L1.calculateF(5); L2.calculateF(4);

P1.calculateF(); P2.calculateF();

system("Pause");

return 0;

}

**Описание классов.**

**Виртуальный класс Function**

|  |  |  |  |
| --- | --- | --- | --- |
| Элемент класса | Принимаемое значение (функции) / Тип (элементы) | Возвращаемое значение (функции) | Роль |
| Доступ : public | | | |
| calculate | double x | double 0 | Для всех производных классов обязательно наличие функции calculate (возвращает значение функции при ) |
| printF |  | void 0 | Для всех производных классов обязательно наличие функции printF (выводит функцию) |
| calculateF | double x | void 0 | Для всех производных классов обязательно наличие функции calculate (выводит функцию и её значение при ) |

**Класс Powered**

Представляет функцию

|  |  |  |  |
| --- | --- | --- | --- |
| Элемент класса | Принимаемое значение (функции) / Тип (элементы) | Возвращаемое значение (функции) | Роль |
| Доступ : protected | | | |
| x | double |  | Аргумент “x” функции |
| power | int |  | Степень аргумента “x” |
| Доступ: public | | | |
| Powered |  | Powered | Конструктор класса |
| calculate |  | double | Возвращает значение функции |
| printF |  | void | Выводит функцию |
| ~Powered |  |  | Деструктор класса |
| operator= | Powered a | Powered | Оператор, присваивающий этой функции значение после знака “=” |
| calculateF |  | void | Выводит функцию и её значение |

**Класс Algebraic, производный от класса Function и имеющий доступ public**

Представляет функцию

|  |  |  |  |
| --- | --- | --- | --- |
| Элемент класса | Принимаемое значение (функции) / Тип (элементы) | Возвращаемое значение (функции) | Роль |
| Доступ : protected | | | |
| arg | double |  | Аргумент “a” функции |
| adds | double |  | Аргумент “b” функции |
| power | int |  | Аргумент “p” функции |
| Доступ : public | | | |
| Algebraic | double Farg (умолч.=1), int pow (умолч.=1), double add (умолч.=0) | Algebraic | Конструктор класса |
| printF |  | void | Выводит функцию |
| ~Algebraic |  |  | Деструктор класса |
| calculate | double x | double | Возвращает значение функции |
| operator= | Algebraic a | Algebraic | Оператор, присваивающий этой функции значение после знака “=” |
| calculateF | double x | void | Выводит функцию и её значение |

**Класс Trigonometry, производный от класса Function и имеющий доступ public**

Представляет функцию или

**Описание перечисления (enum) type**

|  |  |
| --- | --- |
| Элемент | Значение |
| sinus | 1 |
| cosinus | 2 |
| tangens | 3 |
| cotangens | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| Элемент класса | Принимаемое значение (функции) / Тип (элементы) | Возвращаемое значение (функции) | Роль |
| Доступ : protected | | | |
| prearg | double (умолч. = 1) |  | Аргумент “a” функции |
| inarg | double (умолч. = 1) |  | Аргумент “b” функции |
| xpow | int  (умолч. = 1) |  | Аргумент “p” функции |
| fpow | int  (умолч. = 1) |  | Аргумент “d” функции |
| Ft | enum type |  |  |
| Доступ : public | | | |
| Trigonometry | double FunctionArgument  (умолч. = 1), double XArgument (умолч. = 1),  int XPower  (умолч. = 1),  int FunctionPower  (умолч. = 1), type FunctionType (умолч. = (type)1 = sinus) | Trigonometry | Конструктор класса |
| calculate | double x | double | Возвращает значение функции |
| printF |  | void | Выводит функцию |
| ~Trigonometry |  |  | Деструктор класса |
| operator= | Trigonometry a | Trigonometry | Оператор, присваивающий этой функции значение после знака “=” |
| calculateF | double x | void | Выводит функцию и её значение |

**Класс Logar2, производный от класса Function и имеющий доступ public**

Представляет функцию

|  |  |  |  |
| --- | --- | --- | --- |
| Элемент класса | Принимаемое значение (функции) / Тип (элементы) | Возвращаемое значение (функции) | Роль |
| Доступ : protected | | | |
| arg | double |  | Аргумент “x” функции |
| Доступ : public | | | |
| Logar2 | int a  (умолч. = 2) | Logar2 | Конструктор класса |
| ~Logar2 |  |  | Деструктор класса |
| calculate | double x | double | Возвращает значение функции |
| printF |  | void | Выводит функцию |
| operator= | Logar2 a | Logar2 | Оператор, присваивающий этой функции значение после знака “=” |
| calculateF | double x | void | Выводит функцию и её значение |

**Результаты работы программы.**

Function: Algebraic function: 3.000000 \*x ^ 2

F(12.500000) = 468.750000

Function: Algebraic function: 9.500000 \*x ^ 10 +6.900000

F(4.500000) = 32348104.370215

Function: 4.000000 \* sin(6.000000 \* x ^ 7) ^ 2

F(7.300000) = 0.871872

Function: 2.000000 \* cos(5.000000 \* x)

F(-3.300000) = -0.987480

Function: Log2 function: log2(5.000000) = 2.321928

F(5.000000) = 2.321928

Function: Log2 function: log2(4.000000) = 2.000000

F(4.000000) = 2.000000

Function: Powered function: 6.000000 ^ 7

F(6.000000) = 279936.000000

Function: Powered function: 1.000000 ^ 100

F(1.000000) = 1.000000

Press any key to continue . . .