Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и

информатики»

кафедра ПМ иК

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование»

Тема: «Текстовый калькулятор»

Выполнил: студент группы ИП-213

Дмитриев Егор Александрович

Проверил: преподаватель кафедры ПМ и К

Дементьева Кристина Игоревна

Новосибирск – 2023

Содержание

1. Постановка задач 3
2. Технологии ООП 4
3. Структура классов 5
4. Программная реализация 7
5. Результаты работы 8
6. Заключение 10
7. Приложение 11
8. ПОСТАНОВКА ЗАДАЧ

Реализовать текстовый калькулятор, используя объектно-ориентированный подход. Иерархия классов должна включать минимум четыре класса, один из которых – абстрактный.

Язык и среда программирования – С++.

Режимы калькулятора должны включать в себя сложение, вычитание, умножение, деление, возведение в степень и элементарные функции.

1. ТЕХНОЛОГИИ ООП

Для реализации программного продукта использовались такие технологии ООП как:

* Абстрактные классы
* Чисто виртуальные методы
* Конструкторы
* Перегрузка конструкторов
* Деструкторы
* Наследование
* Инкапсуляция
* Списки инициализации

1. СТРУКТУРА КЛАССОВ

*Operation (Операция):*

Абстрактный класс, представляющий базовую операцию. Содержит чисто виртуальный метод perform, который должен быть реализован в производных классах.

Виртуальный деструктор для возможности корректного освобождения ресурсов при удалении объектов.

*Addition (Сложение):*

Класс, наследующий от Operation, представляющий операцию сложения (+).

Реализует виртуальный метод perform, который выполняет сложение двух чисел.

*Subtraction (Вычитание):*

Класс, наследующий от Operation, представляющий операцию вычитания (-).

Реализует виртуальный метод perform, который выполняет вычитание одного числа из другого.

*Multiplication (Умножение):*

Класс, наследующий от Operation, представляющий операцию умножения (\*).

Реализует виртуальный метод perform, который выполняет умножение двух чисел.

*Division (Деление):*

Класс, наследующий от Operation, представляющий операцию деления (/).

Реализует виртуальный метод perform, который выполняет деление одного числа на другое, при этом обрабатывая исключение в случае деления на ноль.

*Power (Возведение в степень):*

Класс, наследующий от Operation, представляющий операцию возведения в степень (^).

Реализует виртуальный метод perform, который возводит число в указанную степень.

*ElementaryFunction (Элементарная функция):*

Класс, представляющий элементарные функции (sin, cos, tan).

Хранит указатель на функцию (double (\*function)(double)), принимающую один аргумент типа double и возвращающую double.

Реализует метод perform, который принимает аргумент и возвращает результат применения указанной элементарной функции.

*Calculator (Калькулятор):*

Класс, содержащий функционал для вычислений.

Метод calculate принимает два операнда и объект операции, выполняет операцию и возвращает результат.

Метод calculateElementaryFunction принимает объект элементарной функции и аргумент, выполняет функцию и возвращает результат.

Эта программа предоставляет пользователю выбор между обычными выражениями и элементарными функциями, а затем выполняет вычисления в соответствии с выбором пользователя.

1. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

*int main(){}*

Создается объект класса Calculator.

Пользователю предлагается выбрать режим: обычные выражения или элементарные функции.

В зависимости от выбора:

Если выбран режим обычных выражений:

Вводятся операнды и операция.

Создается соответствующий объект операции (Addition,

Subtraction и т. д.).

Выводится результат выполнения операции.

Освобождается память, выделенная для объекта операции.

Если выбран режим элементарных функций:

Вводится номер функции (sin, cos, tan) и аргумент.

Создается соответствующий объект элементарной функции.

Выводится результат выполнения элементарной функции.

Программа использует объектно-ориентированный подход для обработки операций и функций, обеспечивая гибкость и расширяемость кода.

1. РЕЗУЛЬТАТЫ РАБОТЫ



**Рисунок 1** - Начальное меню выбора режима работы

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 2** – Меню режима работы для «Обычных выражений»

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 3** – Результат выполнения операции умножения

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

**Рисунок 4** – Меню режима работы для «Элементарных функций»

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

**Рисунок 5** – Результат выполнения расчёта cos(3.14)

1. ЗАКЛЮЧЕНИЕ

В результате выполнения работы я получил опыт разработки приложения с использованием технологий объектно-ориентированного программирования, включающих в себя классы, полиморфизм, наследование, инкапсуляцию, конструкторы, деструкторы и другое.

1. ПРИЛОЖЕНИЕ

#include <iostream>

#include <cmath>

#include <stdexcept>

using namespace std;

class Operation {

public:

virtual double perform(double left, double right) const = 0;

virtual ~Operation() {}

};

class Addition : public Operation {

public:

double perform(double left, double right) const override {

return left + right;

}

};

class Subtraction : public Operation {

public:

double perform(double left, double right) const override {

return left - right;

}

};

class Multiplication : public Operation {

public:

double perform(double left, double right) const override {

return left \* right;

}

};

class Division : public Operation {

public:

double perform(double left, double right) const override {

if (right == 0) {

throw runtime\_error("Деление на ноль!");

}

return left / right;

}

};

class Power : public Operation {

public:

double perform(double left, double right) const override {

return pow(left, right);

}

};

class ElementaryFunction {

private:

double (\*function)(double);

public:

ElementaryFunction(double (\*func)(double)) : function(func) {}

~ElementaryFunction(){}

double perform(double arg) const {

return function(arg);

}

};

class Calculator {

public:

double calculate(double left, const Operation& op, double right) const {

return op.perform(left, right);

}

double calculateElementaryFunction(const ElementaryFunction& func, double arg) const {

return func.perform(arg);

}

};

int main() {

system("clear");

Calculator calculator;

cout << "Выберите режим:\n1. Обычные выражения\n2. Элементарные функции\n";

int mode;

cin >> mode;

if (mode == 1) {

double operand1, operand2;

char operation;

cout << "Операции:\n\"+\" - сложение\n\"-\" - вычитание\n\"/\" - деление\n\"\*\" - умножение\n\"^\" - возведение в степень" << endl;

cout << "Введите выражение (например: 2 + 3): ";

cin >> operand1 >> operation >> operand2;

Operation\* op;

switch (operation) {

case '+':

op = new Addition();

break;

case '-':

op = new Subtraction();

break;

case '\*':

op = new Multiplication();

break;

case '/':

op = new Division();

break;

case '^':

op = new Power();

break;

default:

cerr << "Неподдерживаемая операция!" << endl;

return 1;

}

try {

double result = calculator.calculate(operand1, \*op, operand2);

cout << "Результат: " << result << endl;

} catch (const runtime\_error& e) {

cerr << "Ошибка: " << e.what() << endl;

}

delete op;

} else if (mode == 2) {

cout << "Выберите функцию:\n1. sin\n2. cos\n3. tan\n";

int functionChoice;

cin >> functionChoice;

double argument;

cout << "Введите аргумент: ";

cin >> argument;

switch (functionChoice) {

case 1: {

ElementaryFunction sinFunction(sin);

double sinResult = calculator.calculateElementaryFunction(sinFunction, argument);

cout << "Результат sin: " << sinResult << endl;

break;

}

case 2: {

ElementaryFunction cosFunction(cos);

double cosResult = calculator.calculateElementaryFunction(cosFunction, argument);

cout << "Результат cos: " << cosResult << endl;

break;

}

case 3: {

ElementaryFunction tanFunction(tan);

double tanResult = calculator.calculateElementaryFunction(tanFunction, argument);

cout << "Результат tan: " << tanResult << endl;

break;

}

default:

cerr << "Неподдерживаемая функция!" << endl;

return 1;

}

} else {

cerr << "Неправильный выбор режима!" << endl;

return 1;

}

return 0;

}