Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки
інформації та управління

ЗВІТ

до лабораторної роботи № 3

з дисципліни ООП

Виконав студент *ІП-61 Кушка Михайло Олександрович*

(№ групи, прізвище, ім’я, по батькові)

Прийняв *Головченко М.М.*

(посада, прізвище, ім’я, по батькові)

Київ 2017

ЗМІСТ

1. МЕТА РОБОТИ

Мета роботи - вивчити основні концепції об'єктно-орієнтованого програмування. Вивчити особливості віртуальних функцій, абстрактних класів і поліморфізму.

2. ПОСТАНОВКА ЗАДАЧІ

Спроектувати ієрархію класів: клас функції і його спадкоємці: експонента, натуральний логарифм. Визначити в базовому класі і перевизначити в спадкоємців методи обчислення значення функції ex і ln(x) для заданого значення змінної, використовуючи розкладання в ряд Маклорена. Елементи-дані оголошуються в базовому класі, а не започатковано в спадкоємців (елементи дані: значення змінної).

Виконати завдання з умовою роботи зі спадкоємцями через об'єкт базового абстрактного класу. Обумовлені в базових класах методи повинні бути чисто віртуальними.

3. АНАЛІТИЧНІ ВИКЛАДКИ

Поліморфізм - властивість коду С ++ поводитися по-різному в залежності від ситуації, що виникає в момент виконання; це можливість об'єктів різних класів, пов'язаних спадкуванням, реагувати по-різному при зверненні до однієї і тієї ж функції-елементу з однаковим ім'ям і різним визначенням. Виникає проблема розпізнавання функцій, яке вирішується за допомогою раннього (статичного) зв'язування або пізнього (динамічного).

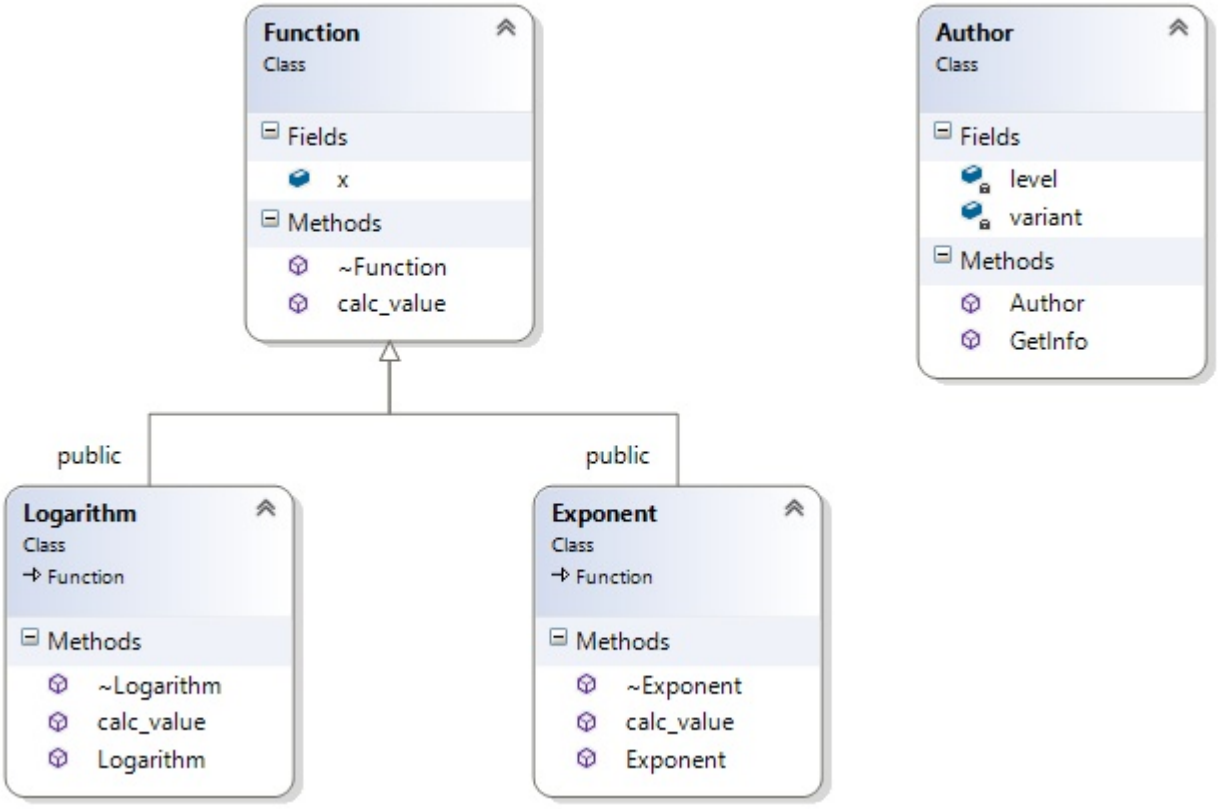
Раннє зв'язування відбувається на етапі компіляції, шляхом виклику фіксованих ідентифікаторів функцій компілятором, заміни ідентифікаторів даними (фізичними адресами) компоновщиком завдань.

Пізніше зв'язування - при виконанні програми, вибір потрібної функції - самою програмою. Можливо для об'єктів пов'язаних ієрархією і реалізується через механізм віртуальних функцій. Перевага пізнього зв'язування - гнучкість коду, простота зміни програми; недолік - зменшення швидкості обробки (побудова віртуальних таблиць, визначення віртуальних функцій).

Віртуальна функція - функція, оголошується в базовому класі і перевизначати в похідних. Похідні класи можуть мати свою власну реалізацію віртуальної функції. При цьому функція, переобумовленої в похідному класі, повинна мати той же список аргументів і тип значення, що повертається, що і віртуальна функція базового класу. Віртуальна функція оголошується і визначається за допомогою специфікатор virtual. Формат визначення віртуальної функції:

virtual тип_значення_що_повертається імя_функції (параметри) {тіло}

4. UML-ДІАГРАМА КЛАСІВ



5. ВИХІДНИЙ КОД ПРОГРАМИ

class_prototypes.cpp

```
//
//  class_prototypes.cpp
//  Lab3
//
//  Created by Kushka Misha on 9/30/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

#include "class_prototypes.hpp"

void Exponent::calc_value(int n) {
    long double result = 1;
    for (int i = 1; i < n; ++i) {
        result += pow(x, i) / factorial(i);
    }

    cout << "e^(" << x << ") = " << result << endl;
}

void Logarithm::calc_value(int n) {
    if (x <= 0 || x > 2) {
        cout << "Sorry, but x must be in range (0, 2]." << endl;
    } else {
        long double result = 0;
        for (int i = 1; i < n; ++i) {
            if (i % 2 == 0) {
                result -= pow(x-1, i) / i;
            } else {
                result +=  pow(x-1, i) / i;
            }
        }

        cout << "log(" << x << ") = " << result << endl;
    }
}

Function::~Function() {
    cout << "Function class destructor." << endl;
}

Exponent::~Exponent() {
    cout << "Exponent destructor." << endl;
}

Logarithm::~Logarithm() {
    cout << "Logarithm destructor." << endl;
}

unsigned long int factorial(int x) {
    return x == 0 ? 1 : x * factorial(x-1);
}

void Author::GetInfo() {
    // Displays author info.
    cout << "\
```

```
    ------------------------\n\
    | Kushka Misha, IP-61 |\n\
    | Level: " << level << "             |\n\
    | Variant: " << variant << "          |\n\
    ------------------------\n\n";
}
```

class_prototypes.hpp

```
//
//  class_prototypes.hpp
//  Lab3
//
//  Created by Kushka Misha on 9/30/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

#ifndef class_prototypes_hpp
#define class_prototypes_hpp

#include "stdafx.hpp"

// Abstract parent class.
class Function {
public:
    double x;
    virtual void calc_value(int n=25)=0;
    virtual ~Function();
};

// Exponent class (parent: Function).
class Exponent : public Function {
public:
    Exponent(double x_) {
        x = x_;
    };
    void calc_value(int n=25);
    ~Exponent();
};

// Logarithm class (parent: Function).
class Logarithm : public Function{
public:
    Logarithm(double x_) {
        x = x_;
    }
    void calc_value(int n=25);
    ~Logarithm();
};

// Class to display some useful info about author of the program.
class Author {
    int level, variant;
public:
    Author(int level=3, int variant=15) : level(level), variant(variant) {}
    void GetInfo();
};

// Calculate factorial.
unsigned long int factorial(int);
```

```
#endif /* class_prototypes_hpp */
```

stdafx.hpp

```
//
//  stdafx.hpp
//  Lab3
//
//  Created by Kushka Misha on 9/30/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

#ifndef stdafx_hpp
#define stdafx_hpp

#include <iostream>
#include <cmath>

using namespace std;

#endif /* stdafx_hpp */
```

main.cpp

```
#include "class_prototypes.hpp"

int main() {
    // Display some usefull info.
    Author *auth = new Author(2);
    auth->GetInfo();

    Function *log_;
    Function *exp_;

    double x;
    cout << "Enter x\n> ";
    cin >> x;

    exp_ = new Exponent(x);
    log_ = new Logarithm(x);

    exp_->calc_value();
    log_->calc_value();

    char cont = '\0';

    // Try again cycle.
    while(true) {
        cout << "\nContinue? (y/n)\n> ";
        cin >> cont;
        if (cont == 'n')
            break;

        // Try again.
        cout << "Enter x\n> ";
        cin >> x;

        exp_ = new Exponent(x);
        log_ = new Logarithm(x);
```

```
        exp_->calc_value();
        log_->calc_value();
    }

    delete exp_;
    delete log_;

    return 0;
}
```

6. ПРИКЛАДИ РОБОТИ ПРОГРАМИ

```
    ----------------------
    | Kushka Misha, IP-61 |
    | Level: 2            |
    | Variant: 15         |
    ----------------------

Enter x
> 5
e^(5) = 148.42
Sorry, but x must be in range (0, 2].

Continue? (y/n)
> y
Enter x
> 1.2
e^(1.2) = 3.32012
log(1.2) = 0.182322

Continue? (y/n)
> n
Exponent destructor.
Function class destructor.
Logarithm destructor.
Function class destructor.
Program ended with exit code: 0
```

7. ВИСНОВКИ

У даній лабораторній роботі я познайомився з такими поняттями, як поліморфізм та віртуальна функція, особливостями їх реалізації мовою C++. Розробив програму з використанням цих понять.