Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 3

з дисципліни

“ООП”

Тема: «Поліморфізм С++»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Головченко  Максим  Миколайович |  | студент 2-го курсу  гр. ІП-51 ФІОТ  Субцельний Олександр Володимирович |

Київ – 2016

**ЗМІСТ**

[1 Мета роботи 4](#_Toc463208490)

[2 Постановка задачі (варіант 20, рівень Б) 5](#_Toc463208491)

[3 Покроковий алгоритм 6](#_Toc463208492)

[4 Діаграма класів 7](#_Toc463208493)

[5 Код програми 8](#_Toc463208494)

[6 Приклади виконання програми 12](#_Toc463208495)

[7 Висновок 13](#_Toc463208496)

# Мета роботи

Цель работы – изучить основные концепции объектно-ориентированного программирования. Изучить особенности виртуальных функций, абстрактных классов и полиморфизма

# Постановка задачі (варіант 20, рівень В)

. Спроектировать иерархию классов: класс функция от одной переменной и его наследники: экспоненциальная и ее производная, гиперболический синус и его производная, гиперболический косинус и его производная. Определить в базовом классе и переопределить в наследниках методы вычисления значения функции для заданного значения переменной. Элементы-данные объявляются в базовом классе, а инициализируются в наследниках (элементы данные: значение переменной). C условием, работы с наследниками через объект базового абстрактного класса. Определяемые в базовых классах методы должны быть чисто виртуальными.

# Покроковий алгоритм

* Знаходження експотенціальної функції і похідної

1. ПОЧАТОК
2. Знайти експоненту
3. Знайти похідну експоненти
4. КІНЕЦЬ

* Знаходження гіперболічний косинус і похідну

1. ПОЧАТОК
2. Знайти гіперболіческий косинус
3. Знайти похідну гіперболіченского косинуса
4. КІНЕЦЬ

* Знаходження гіперболіческий синус і похідної

1. ПОЧАТОК
2. Знайти гіперболіческий синус
3. Знайти похідну гіперболіческого синуса
4. КІНЕЦЬ

# Діаграма класів

Діаграма класів наведена на рисунку 4.1:

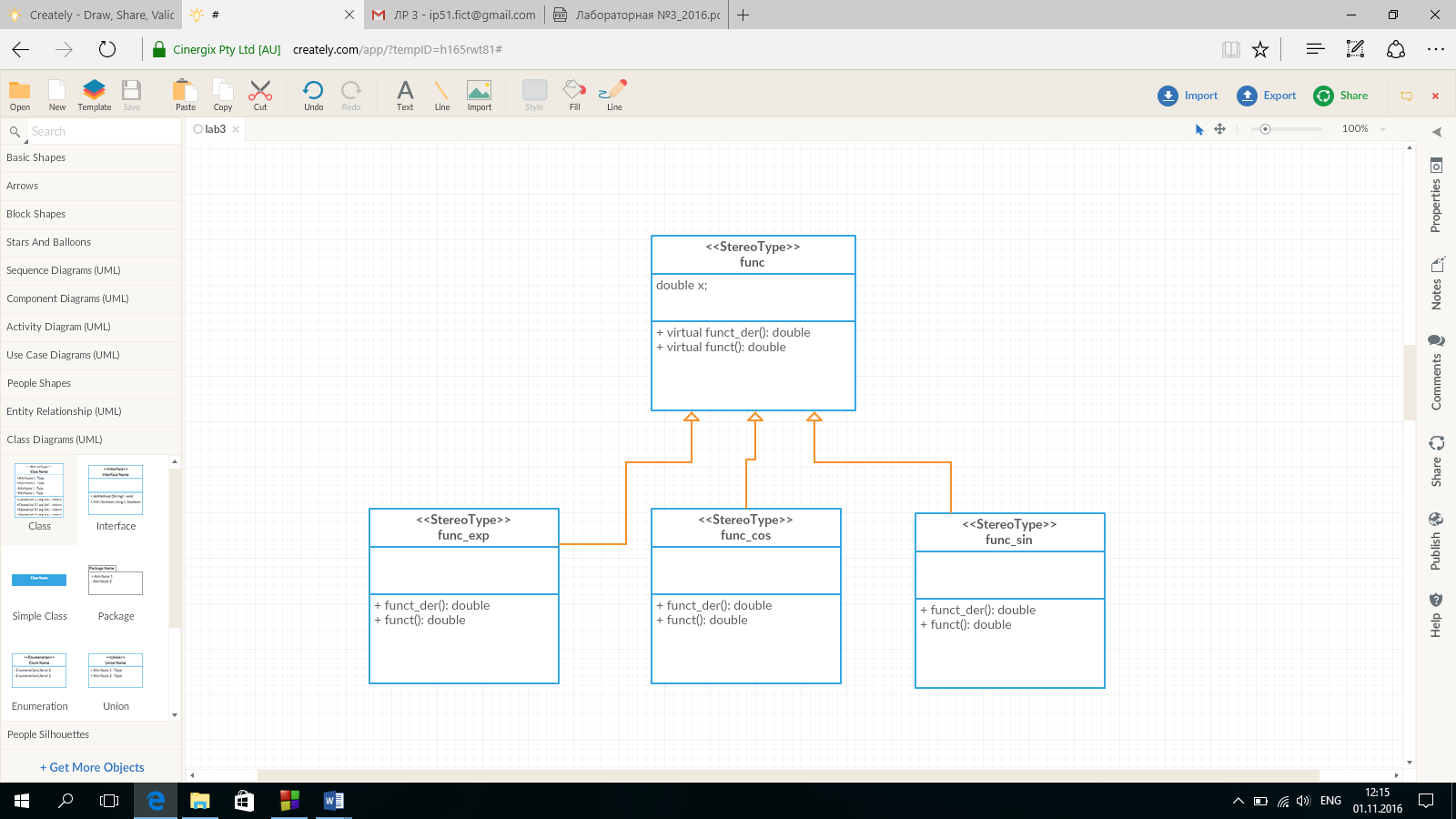


Рисунок 4.1 – Діаграма класів

# Код програми

“main.cpp”

#include <iostream>

#include <stdio.h>

#include "func.h"

#include "func\_exp.h"

#include "func\_cos.h"

#include "func\_sin.h"

using namespace std;

int main()

{

printf("OOP lab work #3\nVariant - 20, Level - B\nBy: Alexander Subtselnyi\n\n");

int t=1;

while(t!=2){

printf("Menu:\n1. Start program\n2. Exit\n ");

scanf("%d", &t);

switch (t) {

case 1: {

cout<<"Enter the argument"<<endl;

double f;//ведення аргументу

cin>>f;

func\* object[3]; //створення масиву обєктів

object[0]=new func\_exp(f);//обєкт класу експотенціальної функції

object[1]=new func\_cos(f+1); //косинуса

object[2]=new func\_sin(f+2); //синуса

cout<<"Core functions"<<endl;

for (int i = 0; i < 3; i++) printf("%d - %.3Lf\n", i + 1, object[i]->funct());//виклик методыв обэктыв

cout<<"Derivated functions"<<endl;

for (int i = 0; i < 3; i++) printf("%d - %.3Lf\n", i + 1, object[i]->funct\_der());//виклик методів лбєктів

printf("Where:\n1 - exp\n2 - cosh\n3 - sinh\n\n");

for (int i = 0; i < 3; i++) delete object[i];//видалення обєктів

break;

}

case 2: return 0;

}

}

}

“func.h”

#pragma once

#ifndef FUNC\_H

#define FUNC\_H

#include "Math.h"

class func

{

public:

double x; //аргумент для функцій

func();

virtual ~func();

virtual double funct()=0;//функія

virtual double funct\_der()=0; //похідна функції

protected:

private:

};

#endif // FUNC\_H

“func\_exp.h”

#pragma once

#ifndef FUNC\_EXP\_H

#define FUNC\_EXP\_H

#include "func.h"

class func\_exp : public func

{

public:

func\_exp(double);

virtual ~func\_exp();

double funct();

double funct\_der();

protected:

private:

};

#endif // FUNC\_EXP\_H

“func\_cos.h”

#pragma once

#ifndef FUNC\_COS\_H

#define FUNC\_COS\_H

#include "func.h"

class func\_cos : public func

{

public:

func\_cos(double);

virtual ~func\_cos();

double funct();

double funct\_der();

protected:

private:

};

#endif // FUNC\_COS\_H

“func\_sin.h”

#pragma once

#ifndef FUNC\_SIN\_H

#define FUNC\_SIN\_H

#include "func.h"

class func\_sin : public func

{

public:

func\_sin(double);

virtual ~func\_sin();

double funct();

double funct\_der();

protected:

private:

};

#endif // FUNC\_SIN\_H

“func\_exp.cpp”

#include "func\_exp.h"

func\_exp::func\_exp(double x)

{

func::x=x;

}

func\_exp::~func\_exp()

{

//dtor

}

double func\_exp::funct() //функція експотенціальна

{

return exp(x);

}

double func\_exp::funct\_der()//похадна експотенціальної ьфункції

{

return 2\*exp(x);

}

“func\_cos.cpp”

#include "func\_cos.h"

func\_cos::func\_cos(double x)

{

func::x=x;

}

func\_cos::~func\_cos()

{

//dtor

}

double func\_cos::funct() //функція гіперболічного косинусу

{

return acosh(x);

}

double func\_cos::funct\_der() // похідна функція гіперболічного косинусу

{

return asinh(x);

}

“func\_sin.cpp”

#include "func\_sin.h"

func\_sin::func\_sin(double x)

{

func::x=x;

}

func\_sin::~func\_sin()

{

//dtor

}

double func\_sin::funct() //фуекція гіперболічного синуссу

{

return asinh(x);

}

double func\_sin::funct\_der() //похідна фуекція гіперболічного синуссу

{

return acosh(x);

}

# Приклади виконання програми

Приклад виконання програми наведений на рисунку 6.1:

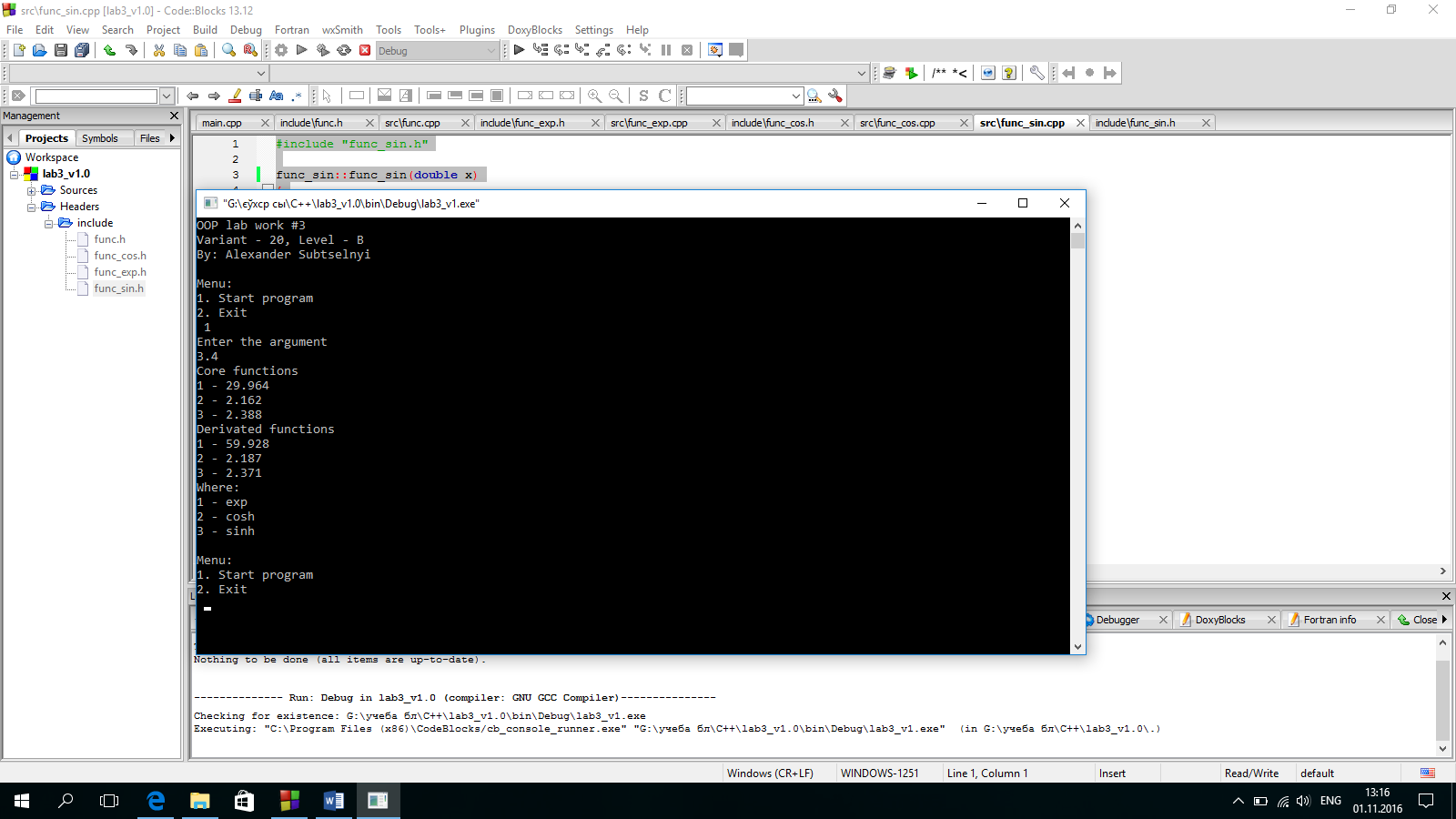


Рисунок 6.1 – Приклад виконання програми

# Висновок

Отже, дана програма створює масив обєктів func, кожен елемент якого обєкт класу, який наслідує класс func та перевизначає віртуальні функіїї базового абстрактоного класу. Оскільки дана програма реалізована за модульним принципом і кожен модуль програми створювався й налагоджувався автономно, то дана програма буде видавати очікуваний результат на всіх наборах вхідних даних.

Розглянемо на прикладі х=3.4, тоді експонента має дорівнювати 29,9, похідна 59,9  
Для даного прикладу в функцію косинуса передається х=4.4, тоді гіперкосинус =2.16, а похідна 2.18

В синус передається х=5.4 , тоді гіперсинус=2.38, а похідна 2.37  
Порівняємо результати обчисленні вручну з програмою. Результати співпадають, отже програма працює вірно.