Міністерство освіти і науки України

Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатика та програмної інженерії

Звіт

З лабораторної роботи № 5 з дисципліни “Основи програмування-2” “Успадкування та поліморфізм”

Варіант 2

Виконав: ІП-13 Бабашев Олексій

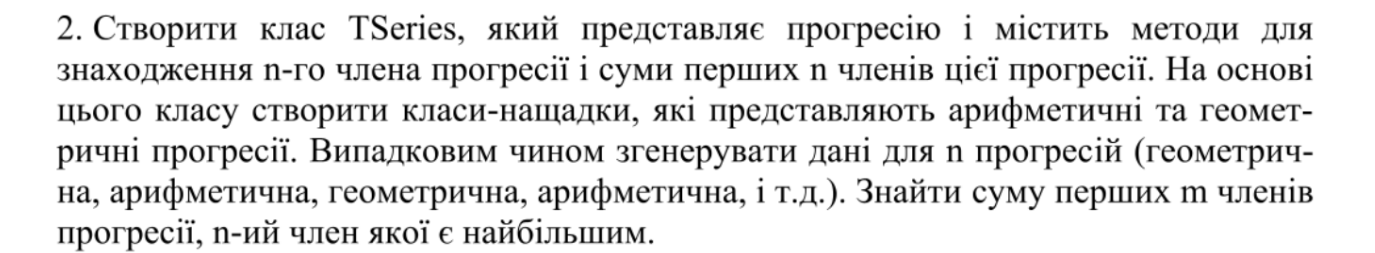
Перевірила: Вєчерковська А. С.

Київ - 2022

**Лабораторна робота 4**

**Класи та об’єкти**

**Індивідуальне завдання:**



**Виконання програми в терміналі та код на С++:**

**Вихідний код:**

**lab5.cpp**

#include "ASeries.h"

#include "GSeries.h"

#include <iostream>

using namespace std;

int main()

{

srand(time(nullptr));

long int n, m;

do

{

cout << "Enter POSITIVE n and m " << endl; cin >> n; cin >> m; cout << endl;

} while (m<=0||n<=0);

long int max\_n = 0, max\_sum = 0;

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

{

long int firstMember = rand() % 11+1;

long int step = rand() % 11+1;

cout << i + 1 << ") first number is " << firstMember << " step is " << step << endl;

if (i % 2 == 0)

{

GSeries gSeries(firstMember, step);

cout << "n number is " << gSeries.FindNElement(n) << " sum is " << gSeries.Sum(m) << endl << endl;

if (gSeries.FindNElement(n) > max\_n)

{

max\_n = gSeries.FindNElement(n);

max\_sum = gSeries.Sum(m);

}

}

else

{

ASeries aSeries(firstMember, step);

cout << "n number is " << aSeries.FindNElement(n) << " sum is " << aSeries.Sum(m) << endl << endl;

if (aSeries.FindNElement(n) > max\_n)

{

max\_n = aSeries.FindNElement(n);

max\_sum = aSeries.Sum(m);

}

}

}

cout <<endl << "max n number is " << max\_n << " with sum of first m numbers " << max\_sum << endl;

}

**TSeries.cpp**

#include "TSeries.h"

TSeries::TSeries(int firstNumber = 1, int step = 1)

{

this->firstNumber = firstNumber;

this->step = step;

}

int TSeries::GetStep() {

return step;

}

int TSeries::GetFirstNumber() {

return firstNumber;

}

void TSeries::SetStep(int step)

{

this->step = step;

}

void TSeries::SetFirstNumber(int firstNumber)

{

this->firstNumber = firstNumber;

}**TSeries.h**

#pragma once

class TSeries

{

protected:

int firstNumber;

int step;

TSeries(int firstMember, int step);

public:

void SetFirstNumber(int firstNumber);

void SetStep(int step);

int GetStep();

int GetFirstNumber();

virtual int FindNElement(int n) = 0;

virtual int Sum(int n) = 0;

};

**ASeries.cpp**

#include "ASeries.h"

ASeries::ASeries(int firstNumber, int step)

:TSeries(firstNumber, step)

{

}

int ASeries::Sum(int n)

{

return (2 \* firstNumber + step \* (n - 1) \* n) / 2;

}

int ASeries::FindNElement(int n) {

return firstNumber + step \* (n - 1);

}

**ASeries.h**

#pragma once

#include "TSeries.h"

class ASeries: protected TSeries {

public:

ASeries(int firstNumber, int step);

int FindNElement(int n);

int Sum(int n);

};

**GSeries.h**

#pragma once

#include "TSeries.h"

#include <cmath>

class GSeries: protected TSeries {

public:

GSeries(int firstNumber, int step);

int FindNElement(int n);

int Sum(int n);

};

**GSeries.cpp**

#include "GSeries.h"

GSeries::GSeries(int firstNumber, int step)

:TSeries(firstNumber, step)

{

}

int GSeries::FindNElement(int n)

{

return firstNumber \* (int)pow(step, n - 1);

}

int GSeries::Sum(int n)

{

return firstNumber \* (int)(pow(step, n) - 1) / (step - 1);

}

**Виконання програми в терміналі та код на python:**

**Main.py**

*import* random  
*import* time  
*from* ASeries *import* ASeries  
*from* GSeries *import* GSeries  
  
  
n = *int*(*input*("Enter 'n': "))  
m = *int*(*input*("Enter 'm': "))  
*print*()  
random.seed(time.time())  
gSeries = GSeries()  
aSeries = ASeries()  
  
max = 0  
  
max\_sum = 0  
  
*for* i *in range*(n):  
 first\_member = random.randint(1, 11)  
 step = random.randint(1, 11)  
  
 *if*(i % 2 == 0):  
 gSeries.set\_first(first\_member)  
 gSeries.set\_parameter(step)  
  
 *print*(f"{i + 1}) first number is {first\_member} step is {step}")  
 *print*(f" n number is {gSeries.calculate\_n(n)} and sum is {gSeries.calculate\_sum(m)}\n")  
  
 *if*(max < gSeries.calculate\_n(n)):  
 max = gSeries.calculate\_n(n)  
  
 max\_sum = gSeries.calculate\_sum(m)  
  
 *else*:  
 *print*(f"{i + 1}) first number is {first\_member} step is {step}")  
 *print*(f"n number is {aSeries.calculate\_n(n)} and sum is {aSeries.calculate\_sum(m)}\n")  
  
 aSeries.set\_first(first\_member)  
 aSeries.set\_parameter(step)  
 *if*(max < aSeries.calculate\_n(n)):  
 max = aSeries.calculate\_n(n)  
  
 max\_sum = aSeries.calculate\_sum(m)  
  
*print*(f"\nMax n number is: {max} with sum of first m numbers is {max\_sum}\n\n")

**TSeries.py**

*class* TSeries():  
 first = *None* parameter = *None  
  
 def \_\_init\_\_*(*self*, first=0, parameter=0):  
 *self*.set\_first(first)  
 *self*.set\_parameter(parameter)  
  
 *def* set\_first(*self*, first):  
 *self*.first = first  
  
 *def* set\_parameter(*self*, parameter):  
 *self*.parameter = parameter

**ASeries.py**

*from* TSeries *import* TSeries  
  
  
*class* ASeries(TSeries):  
 *def* calculate\_n(*self*, index):  
 result = *self*.first + *self*.parameter\*(index-1)  
 *return* result  
  
 *def* calculate\_sum(*self*, index):  
 last = *self*.calculate\_n(index)  
 *return int*(((*self*.first + last)/2)\*index)

**GSeries.py**

*from* TSeries *import* TSeries  
  
  
*class* GSeries(TSeries):  
 *def* calculate\_n(*self*, index):  
 result = *self*.first \* *self*.parameter\*\*(index-1)  
 *return* result  
  
 *def* calculate\_sum(*self*, index):  
 result = (*self*.first\*(*self*.parameter\*\*index-1))/(*self*.parameter-1)  
 *return int*(result)

**Висновок:** Вивчив особливості успадкування та поліморфі́зм класів. Застосував отриманні знання на практиці.