Додаток 1

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної

техніки Кафедра інформатики та програмної

інженерії

Звіт

з лабораторної роботи № 3 з дисципліни
«Основи програмування 2. Модульне програмування»

«Перевантаження операторів»

Варіант 15

Виконав студент ІП-13 Карамян Вартан Суренович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Лабораторна робота 3

Перевантаження операторів

Варіант 15

Визначити клас “Коло”,членами якого є радіус кола та координати його центру. Реалізувати для нього декілька конструкторів, геттери, метод обчислення довжини кола. Перевантажити оператори: префіксий “++”/ постфіксний “++” – для інкрементування x-координати і y-координати центру кола відповідно, “*” – для збільшення радіусу кола у вказану кількість разів (ціле число). Створити три кола (C1, C2, C3), використовуючи різні конструктори. Інкрементувати x-координату кола C1 і y-координату кола C2. Збільшити радіус кола C3 у 3 рази. Серед кіл C1, C2, C3 визначити коло найбільшої довжини.

Код програми на мові C++:

lab3 C++.cpp

```
#include "Circle.h"
#include "Functions.h"

int main()
{
	vector<Circle> circles = createCircle();
	cout << "================================================\n";

	printCircles(circles);
	cout << "================================================\n";

	++circles[0];
	circles[1]++;
	circles[2] = circles[2] * 3;
	printCircles(circles);
	cout << "================================================\n";

	Circle maxCircle = maxCircleLength(circles);
	cout << "Circle with a max length:\n";
	maxCircle.print();
}
```

Circle.h

```
#pragma once
#define _USE_MATH_DEFINES
#include <iostream>
#include <math.h>
#include <iomanip>
#include <vector>
#include <string>

using namespace std;

class Circle
{
	double x;
	double y;
	double radius;
public:
	Circle(double, double, double);
	Circle(double, double);
	Circle(double = 1);

	double getX();
	double getY();
	double getRadius();
	double circleLength();
	void print();

	Circle operator++();
	Circle operator++(int);
	Circle operator*(int);
};
```

Circle.cpp

```
#include "Circle.h"

Circle:: Circle(double x, double y, double r)
{
	this->x = x;
	this->y = y;
	this->radius = r;
}

Circle::Circle(double x, double y)
{
	this->x = x;
	this->y = y;
	this->radius = 1;
}

Circle::Circle(double r)
{
	this->x = 0;
	this->y = 0;
	this->radius = r;
}

double Circle::getX()
{
```

```
	return x;
}
double Circle::getY()
{
	return y;
}
double Circle::getRadius()
{
	return radius;
}

double  Circle::circleLength()
{
	return 2 * radius * M_PI;
}

void Circle::print()
{
	cout << ">> Circle <<\n";
	cout << "Coordinates of center:\tx = " << x << "\ty = " << y;
	cout << "\nRadius = " << radius;
	cout << "\nCircle length: " << setprecision(4) << circleLength() << "\n\n";
}

Circle Circle::operator++()
{
	++this->x;
	return *this;
}

Circle Circle::operator++(int unused)
{
	++this->y;
	return *this;
}

Circle Circle::operator*(int num)
{
	this->radius *= num;
	return *this;
}
```

Functions.h

```
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include "Circle.h"
using namespace std;

double inputX();
double inputY();
double inputR();
vector<Circle> createCircle();
void printCircles(vector<Circle>);
Circle maxCircleLength(vector <Circle> circles);
```

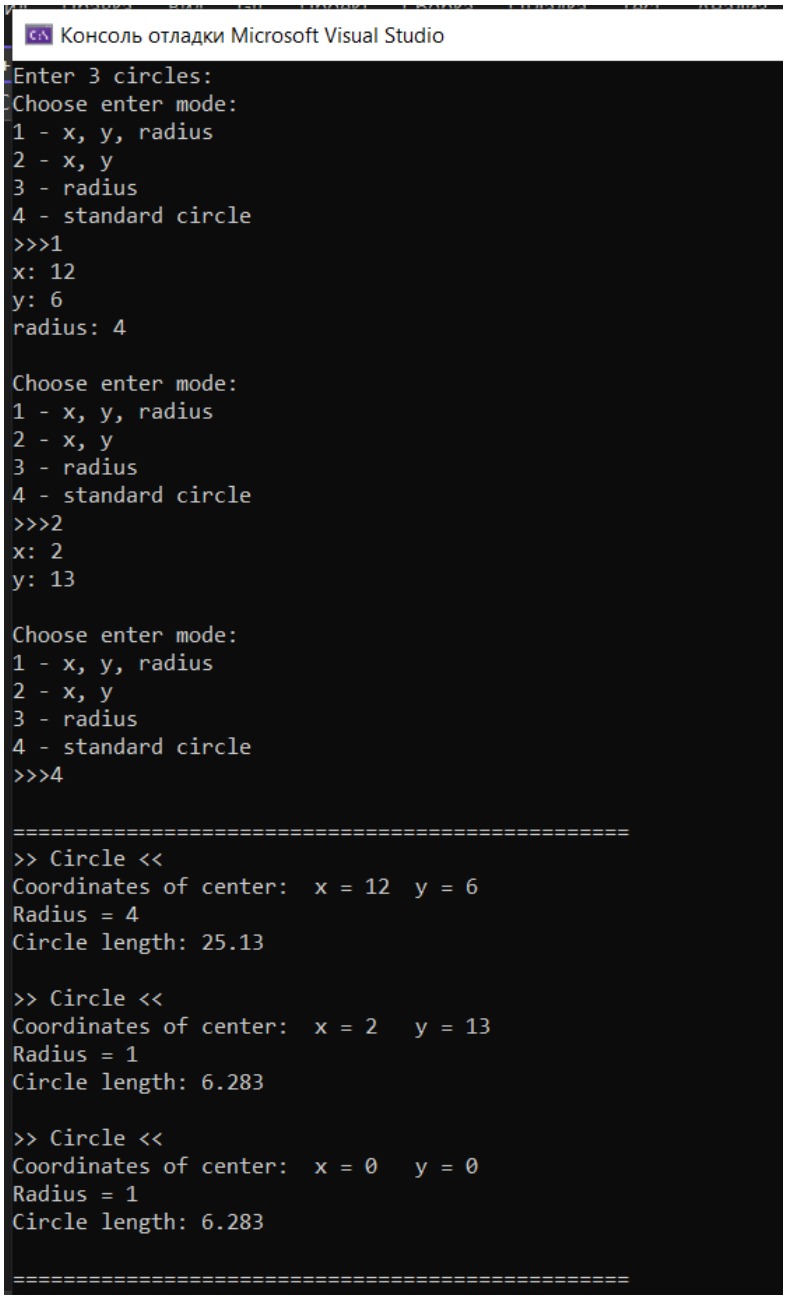
Functions.cpp

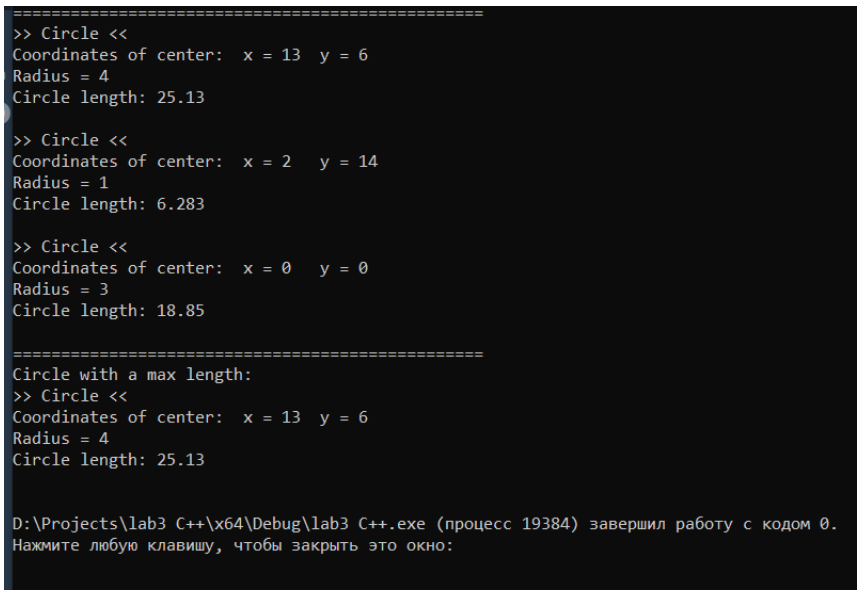
```cpp
#include "Functions.h"

double inputX()
{
	string x;
	double xFloat;
	while (true)
	{
		cout << "x: "; cin >> x;
		try
		{
			xFloat = stof(x);
			break;
		}
		catch (const std::exception&)
		{
			cout << "Enter float number!\n";
		}

	}
	return xFloat;
}

double inputY()
{
	string y;
	double yFloat;
	while (true)
	{
		cout << "y: "; cin >> y;
		try
		{
			yFloat = stof(y);
			break;
		}
		catch (const std::exception&)
		{
			cout << "Enter float number!\n";
		}

	}
	return yFloat;
}

double inputR()
{
	string r;
	double rFloat;
	while (true)
	{
		cout << "radius: "; cin >> r;
		try
		{
			rFloat = stof(r);
			break;
		}
		catch (const std::exception&)
		{
			cout << "Enter float number!\n";
		}

	}
	return rFloat;
}
```

```
vector<Circle> createCircle()
{
	vector<Circle> circles;
	Circle pushCircle;
	int size = 3;
	string mode;
	double x, y, r;

	cout << "Enter 3 circles:\n";

	for (size_t i = 0; i < size; i++)
	{
		while (true)
		{
			cout << "Choose enter mode:\n1 - x, y, radius\n2 - x, y\n3 - radius\n4 - standard circle\n>>>";
			cin >> mode;

			if (mode == "1")
			{
				x = inputX();
				y = inputY();
				r = inputR();
				Circle circle(x, y, r);
				pushCircle = circle;
				break;
			}
			else if (mode == "2")
			{
				x = inputX();
				y = inputY();
				Circle circle(x, y);
				pushCircle = circle;
				break;
			}
			else if (mode == "3")
			{
				r = inputR();
				Circle circle(r);
				pushCircle = circle;
				break;
			}
			else if (mode == "4")
			{
				Circle circle;
				pushCircle = circle;
				break;
			}
		}
		circles.push_back(pushCircle);
		cout << endl;
	}
	return circles;
}

void printCircles(vector<Circle> list)
{
	for (size_t i = 0; i < list.size(); i++)
	{
		list[i].print();
	}
}

Circle maxCircleLength(vector <Circle> circles)
{
	Circle maxCircle = circles[0];
```

```
	double maxLength = circles[0].circleLength();
	for (size_t i = 1; i < circles.size(); i++)
	{
		if (circles[i].circleLength() > maxLength)
		{
			maxCircle = circles[i];
			maxLength = circles[i].circleLength();
		}
	}
	return maxCircle;
}
```

Результати роботи:





Висновок:

У результаті виконання лабораторної роботи ми познайомились з таким механізм ООП як перевантаження операторів.