НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ “КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп’ютерних систем

Лабораторна робота №6

з дисципліни “Алгоритми та методи обчислень”

Тема: “ ЧИСЕЛЬНІ МЕТОДИ РОЗВ’ЯЗАННЯ ЗАДАЧІ КОШІ”

Варіант 4

Виконала: студентка III курсу

КВ-72 Дорош Карина

Перевірив: Зорін Ю. М.

Київ 2019

Постановка задачі

1. Для ДР з відповідного варіанта скориставшись однокроковим методом Рунге-Кутта 4-го порядку точності отримати знчення інтегральної кривої в точці *b* зпохибкою не гірше за $O(10^2)$. У якості початкового кроку взяти величину $h = (b - a)/10$. Якщо цей крок не дозволяє одержати розв'язок зазначеної точності, необхідно зменшувати його.

2. Для похибки eps, отриманої в п.1, методом подвійного перерахунку визначити крок, що забезпечує ту саму точність, і порівняти його із кроком п.1. За початковий крок взяти величину кореня r-ого степеня з eps, де r - порядок точності методу.

3. Побудувати таблицю значень інтегральної кривої, скориставшись методом Рунге-Кутта 4-го порядку точності на сітці з кроком $h = (b – a)/10$.

4. Звіт повинен містити вихідний текст програми, таблицю із результатами та висновки.

№	ДР	Точний розв'язок	ПУ	$[a, b]$
4	y''+3y' + y - 9cos(x) + 2 = 0	$y = 3\sin x - 2$	$y(0) = -2, y'(0) = 3$	$[0, \frac{\pi}{2}]$

Код програми

main.cpp

```
#include"header.h"

int main() {
	double h = (b - a) / 10;
	cout << "Differential equation: y'' + 3*y' + y - 9 * cos(x) + 2 = 0" << endl <<
endl;
	double eps = 1e-2;
	double delta = 0;
	CurveValue(&h, eps, &delta);
	cout << "Delta " << delta << " for Runge-Kutta " << Runge(h) << " with h: " << h;
	cout << "\n\nH with the same error = " << DoubleConversionMethod(delta) << endl;
	h = (b - a) / 10;
	cout << "\nTable of values of the integral curve: " << endl;
	cout << "\   X\t   Y\n";
	for (double x = a; x <= b; x += h) {
		cout << setprecision(4) << x << "\t" << setprecision(6) << Runge(h, x) <<
endl;
	}
	return 0;
}
```

functions.cpp

```
#include"header.h"

double ExactSolution(double x) {
	return 3 * sin(x) - 2;
}
double SecondDerative(double x, double y, double y1) {
	return (-3) * y1 - y + 9 * cos(x) - 2;
};
double FirstDerative(double x, double y, double y1) {
	return y1;
}
double Runge(double h) {
	double y = -2.0;
	double derative0 = 3.0;
	double k1 = 0, k2 = 0, k3 = 0, k4 = 0, l1 = 0, l2 = 0, l3 = 0, l4 = 0;
	for (double x = a; x < b; x += h) {
		k1 = h * SecondDerative(x, y, derative0);
		l1 = h * FirstDerative(x, y,  derative0);
		k2 = h * SecondDerative(x + h / 2.0, y + l1 / 2.0, derative0 + k1/2.0);
		l2 = h * FirstDerative(x + h / 2.0, y + l1 / 2.0, derative0 + k1 / 2.0);

		k3 = h * SecondDerative(x + h / 2.0, y + l2/ 2.0, derative0 + k2 / 2.0);
		l3 = h * FirstDerative(x + h / 2.0, y + l2 / 2.0, derative0 + k2 / 2.0);

		k4 = h * SecondDerative(x + h, y + l3, derative0 + k3);
		l4 = h * FirstDerative(x + h, y + l3, derative0 + k3);
		y = y + 1 / 6.0 * (l1 + 2 * l2 + 2 * l3 + l4);
		derative0 = derative0 + 1 / 6.0 * (k1 + 2 * k2 + 2 * k3 + k4);
	}
	return y;
```

```
}

void CurveValue(double* h, double eps, double *delta) {
	double exact = ExactSolution(b);
	*delta = fabs(exact - Runge(*h));
	while (*delta > eps) {
		*h /= 2;
		*delta = fabs(exact - Runge(*h));
	}
}

double DoubleConversionMethod(double eps) {
	double h = sqrt(sqrt(eps));
	double r = 4.0;
	double yh = Runge(h);
	h /= 2;
	double yh2 = Runge(h);
	while (fabs(yh - yh2) / (pow(2, r) - 1) >  eps) {
		yh = yh2;
		h /= 2;
		yh2 = Runge(h);
	}
	return h;
}

double Runge(double h, double end) {
	double y = -2.0;
	double derative0 = 3.0;
	double k1 = 0, k2 = 0, k3 = 0, k4 = 0, l1 = 0, l2 = 0, l3 = 0, l4 = 0;
	for (double x = a; x < end; x += h) {
		k1 = h * SecondDerative(x, y, derative0);
		l1 = h * FirstDerative(x, y, derative0);
		k2 = h * SecondDerative(x + h / 2.0, y + l1 / 2.0, derative0 + k1 / 2.0);
		l2 = h * FirstDerative(x + h / 2.0, y + l1 / 2.0, derative0 + k1 / 2.0);

		k3 = h * SecondDerative(x + h / 2.0, y + l2 / 2.0, derative0 + k2 / 2.0);
		l3 = h * FirstDerative(x + h / 2.0, y + l2 / 2.0, derative0 + k2 / 2.0);

		k4 = h * SecondDerative(x + h, y + l3, derative0 + k3);
		l4 = h * FirstDerative(x + h, y + l3, derative0 + k3);
		y = y + 1 / 6.0 * (l1 + 2 * l2 + 2 * l3 + l4);
		derative0 = derative0 + 1 / 6.0 * (k1 + 2 * k2 + 2 * k3 + k4);
	}
	return y;
}
```

header.h

```
#include<iostream>
#include<cmath>
#include <iomanip>

#define a 0
#define M_PI 3.14159265358979323846
#define b M_PI/2

using namespace std;
```

```
double ExactSolution(double x);
double SecondDerative(double x, double y, double first_derative);
double FirstDerative(double x, double y, double y1);
double Runge(double h);
double DoubleConversionMethod(double eps);
double Runge(double h, double end);
void CurveValue(double* h, double eps, double*delta);
```

Вивід програми