Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №3

Вариант 10

Студент: Крикунов Олег Евгеньевич

P3267

Преподаватель: Машина Екатерина Алексеевна

Оценка: __________

Подпись преподавателя: __________

1. Цели работы

Изучить численные методы нахождения определенных интегралов, выполнить программную реализацию методов.

2. Описание метода, расчётные формулы

Формула Ньютона - Котеса

Простой прием построения квадратурных формул состоит в том, что подынтегральная функция $f(x)$ заменяется на отрезке $[a, b]$ **интерполяционным многочленом Лагранжа** $L_n(x)$, совпадающий с $f(x)$ в узлах интерполяции $x_0, x_{1,} \ldots, x_n \in [a, b]$. Полином Лагранжа имеет вид:

$$L_n(x) = \sum_{i=0}^{n} f(x_i) L_n^i(x), \qquad n = 0, 1, 2 \ldots$$

где $L_n^i(x)$ - коэффициенты Лагранжа (полиномы степени *n*):

$$L_n^i(x) = \frac{(x - x_0)(x - x_1) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_n)}{(x_i - x_0)(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)}$$

Если полином Лагранжа «близок» к $f(x)$, то интегралы от них тоже должны быть «близки»:

$$\int_a^b f(x)dx \approx \int_a^b L_n(x)dx = \sum_{i=0}^{n} f(x_i) \int_a^b L_n^i(x)dx$$

Коэффициенты Котеса: $c_n^i = \int_a^b L_n^i(x)dx$

Формула Ньютона-Котеса порядка n:

$$\int_a^b f(x)dx \approx \int_a^b L_n(x)dx = \sum_{i=0}^{n} f(x_i) c_n^i$$

Метод прямоугольников

На каждом шаге интегрирования функция аппроксимируется полиномом нулевой степени – отрезком, параллельным оси абсцисс. Площадь криволинейной трапеции приближенно заменяется площадью многоугольника, составленного из *n*- прямоугольников. Таким образом, вычисление определенного интеграла сводится к нахождению суммы *n*- элементарных прямоугольников.

$$\int_a^b f(x)dx \approx S_n = \sum_{i=1}^{n} f(\xi_i)\Delta x_i$$

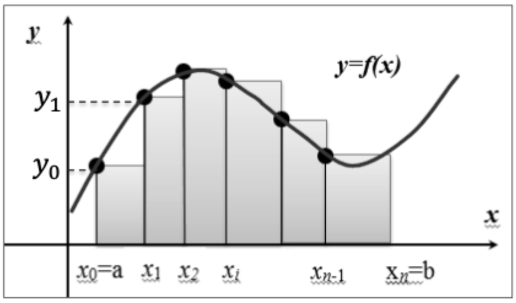
Различают метод левых, правых и средних прямоугольников.

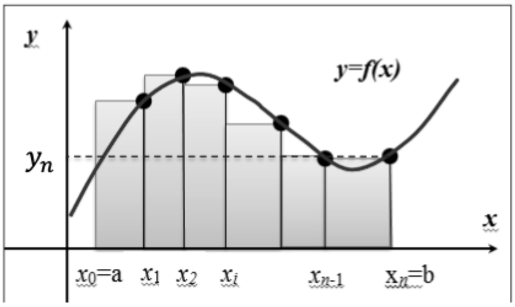
В качестве точек ξ_i могут выбираться левые ($\xi_i = x_{i-1}$) или правые ($\xi_i = x_i$) границы отрезков, получим формулы левых и правых прямоугольников.

Обозначим:

$f(x_i) = y_i$, $f(a) = y_0$, $f(b) = y_n$

$\Delta x_i = x_i - x_{i-1} = h_i$





$\int_a^b f(x)dx \approx h_1y_0 + h_2y_1 + \cdots + h_ny_{n-1} = \sum_{i=1}^{n} h_i\, y_{i-1}$ - левые прямоугольники

$\int_a^b f(x)dx \approx h_1y_1 + h_2y_2 + \cdots + h_ny_n = \sum_{i=1}^{n} h_iy_i$ - правые прямоугольники

При $h_i = h = \frac{b-a}{n} = const$:

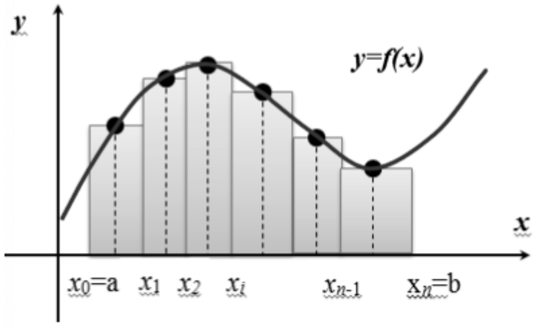
$$\int_a^b \boldsymbol{f(x)dx = h\sum_{i=1}^{n} y_{i-1}}$$

$$\int_a^b \boldsymbol{f(x)dx = h\sum_{i=1}^{n} y_i}$$

Для аналитически заданных функций более точным является использование значений в средних точках элементарных отрезков (полуцелых узлах):

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} h_i\, f(x_{i-1/2})$$

$$x_{i-1/2} = \frac{x_{i-1}+x_i}{2} = x_{i-1} + \frac{h_i}{2}, i = 1, 2, \dots n$$



При $h_i = h = \frac{b-a}{n} = const$:

$$\int_a^b \boldsymbol{f(x)dx = h\sum_{i=1}^{n} f(x_{i-1/2})}$$

Метод трапеций

Подынтегральную функцию на каждом отрезке $[x_i; x_{i+1}]$ заменяют интерполяционным многочленом первой степени:

$$f(x) \approx \varphi_i(x) = a_i x + b$$

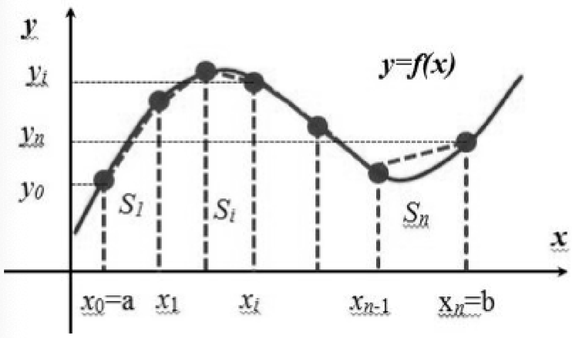
Используют линейную интерполяцию, т.е. график функции $y = f(x)$ представляется в виде ломаной, соединяющий точки (x_i, y_i) . Площадь всей фигуры (криволинейной трапеции):

$$S_{\text{общ}} = S_1 + S_2 + \dots + S_n = \frac{y_0+y_1}{2}h_1 + \frac{y_1+y_2}{2}h_2 + \dots + \frac{y_{n-1}+y_n}{2}h_n$$

$$y_0 = f(a), \quad y_n = f(b), \quad y_i = f(x_i), \quad h_i = x_i - x_{i-1}$$

Складывая все эти равенства, получаем формулу трапеций для численного интегрирования:

$$\int_a^b f(x)dx = \frac{1}{2}\sum_{i=1}^{n} h_i(y_{i-1}+y_i)$$



При $h_i = h = \frac{b-a}{n} = const$ формула трапеций:

$$\int_a^b \boldsymbol{f(x)dx = h\cdot\left(\frac{y_0+y_n}{2} + \sum_{i=1}^{n-1} y_i\right)}$$

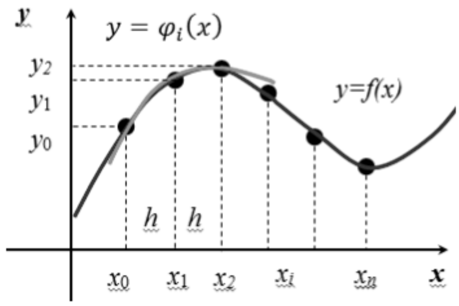
или $$\int_a^b \boldsymbol{f(x)dx = \frac{h}{2}\cdot\left(y_0+y_n+2\sum_{i=1}^{n-1} y_i\right)}$$

Разобьем отрезок интегрирования [*a, b*] на четное число *n* равных частей с шагом *h.* На каждом отрезке $[x_0, x_2], [x_2, x_4], \ldots, [x_{i-1}, x_{i+1}], \ldots, [x_{n-2}, x_n]$ подынтегральную функцию заменим интерполяционным многочленом второй степени:

$$f(x) \approx \varphi_i(x) = a_i x^2 + b_i x + c_i, \quad x_{i-1} \le x \le x_{i+1}$$

Коэффициенты этих квадратных трехчленов могут быть найдены из условий равенства многочлена и подынтегральной функции в узловых точках.

В качестве $\varphi_i(x)$ можно принять интерполяционный многочлен Лагранжа второй степени, проходящий через точки $(x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1})$.



3. Вычислительная часть

Интеграл для вычислений:

10	$\int_2^4 (x^3 - 3x^2 + 7x - 10)dx$

Точное значение интеграла = 26

По формуле Ньютона-Котеса при n = 6:

I = (**41*** (**b** − **a**) / **840**) * f(2)=0 + (2**16***(**b − a**) **/ 840**) * f(2,33)=2,67 + (**27***(**b − a**) **/ 840**) * f(2,66)=6.21 + (**272***(**b − a**) **/ 840**) * f(2,99)=10.84 + (**27***(**b − a**) **/ 840**) * f(3,32)= 16.77 + (2**16***(**b − a**) **/ 840**) * f(3.65)=24.2 + (**41*** (**b** − **a**) / **840**) * f(4)=34 ≈ 25,64

R = |26 – 25,64| = 0,36

По формуле средних прямоугольников при n = 10:

h = (4 - 2) / 10 = 0.2

x[i – ½] = 2.2, 2.4, 2.6, 2.8, 3, 3.2, 3.4, 3.6, 3.8, 4

y[xi – ½] = 1.95312, 15.3906, 26.7969, 0.923828, 3.09961, 5.79102, 13.0957, 17.8965, 23.5879, 30.2637

I = 0.2 * 129,8 = 25,96

R = |26 – 25,96| = 0,04

По формуле трапеций при n = 10:

h = (4 - 2) / 10 = 0.2

x[i] = 2.2, 2.4, 2.6, 2.8, 3, 3.2, 3.4, 3.6, 3.8, 4

y[i] = 1.528, 3.344, 5.496, 8.032, 11, 14.448, 18.424, 22.976, 28.152, 34

I = 0.2 / 2 * (1.528 + 34 + 2 * 111,872) = 25,9272

R = |26 – 25,9272| = 0,0728

По формуле Симпсона при n = 10:

h = (4 - 2) / 10 = 0.2

x[i] = 2.2, 2.4, 2.6, 2.8, 3, 3.2, 3.4, 3.6, 3.8, 4

y[i] = 1.528, 3.344, 5.496, 8.032, 11, 14.448, 18.424, 22.976, 28.152, 34

I = 0,2 / 3 * (1.528 + 4 * (3.344 + 8.032 + 14.448 + 22.976) + 2 * (5.496 + 11 + 18.424 + 28.152) + 34) ≈ 23,8

R = |26 – 23,8| = 2,2

4. Программная часть

Листинг программы:

Функции и их производные в header файле:

```
#ifndef MATPLOTLIB_H_FUNCTIONS_H
#define MATPLOTLIB_H_FUNCTIONS_H

#include <cmath>

double F1(double x) {
    return pow(x,3) - 3 * pow(x, 2) + 7 * x - 10;
}

double F2(double x) {
    return 2 * pow(x,3) - 3 * pow(x, 2) + 5 * x - 9;
}

double F3(double x) {
    return 2 * pow(x,3) - 9 * pow(x, 2) - 7 * x + 11;
}

double F4(double x) {
    return 1 / sqrt(x);
}

double F5(double x) {
    return 1 / 1 - x;
}

#endif
```

Метод Прямоугольников:

```
#ifndef MATPLOTLIB_H_RECTANGLEMETHOD_H
#define MATPLOTLIB_H_RECTANGLEMETHOD_H
```

```
#include "Functions.h"

class RectangleMethod {
public:
    static double LeftRectangleSolving(double a, double b, double n, unsigned short
functionChoice) {
        double h = (b - a) / n;
        double integral = 0;

        switch (functionChoice) {
            case 1:
                for (int i = 0; i < n; ++i) {
                    integral += F1(a + i * h);
                }
                return integral * h;
            case 2:
                for (int i = 0; i < n; ++i) {
                    integral += F2(a + i * h);
                }
                return integral * h;
            case 3:
                for (int i = 0; i < n; ++i) {
                    integral += F3(a + i * h);
                }
                return integral * h;
        }
    }

    static double RightRectangleSolving(double a, double b, double n, unsigned short
functionChoice) {
        double h = (b - a) / n;
        double integral = 0;

        switch (functionChoice) {
            case 1:
                for (int i = 1; i <= n; ++i) {
                    integral += F1(a + i * h);
                }
                return integral * h;
            case 2:
                for (int i = 1; i <= n; ++i) {
                    integral += F2(a + i * h);
                }
                return integral * h;
            case 3:
                for (int i = 1; i <= n; ++i) {
                    integral += F3(a + i * h);
                }
                return integral * h;
        }
    }

    static double CentralRectangleSolving(double a, double b, double n, unsigned short
functionChoice) {
        double h = (b - a) / n;
        double integral = 0;

        switch (functionChoice) {
            case 1:
                for (int i = 0; i < n; ++i) {
                    integral += F1(a + (i + 0.5) * h);
                }
                return integral * h;
            case 2:
                for (int i = 0; i < n; ++i) {
                    integral += F2(a + (i + 0.5) * h);
```

```
                }
                return integral * h;
            case 3:
                for (int i = 0; i < n; ++i) {
                    integral += F3(a + (i + 0.5) * h);
                }
                return integral * h;
        }
    }

    static double RectangleIntegral(double a, double b, double n, double EPS, unsigned
short functionChoice) {
        std::cout << "Введите модификацию метода: \n" << "1) Метод левых \n" << "2)
Метод правых \n"
                  << "3) Метод средних \n" << "4) Вывести решение всеми методами \n";
        unsigned short methodChoice;
        std::cin >> methodChoice;

        switch (methodChoice) {
            case 1: {
                double I0 = LeftRectangleSolving(a, b, n, functionChoice);
                double n1 = n * 2;
                double I1 = LeftRectangleSolving(a, b, n1, functionChoice);

                while (std::abs(I1 - I0) > EPS) {
                    I0 = I1;
                    n = n1;
                    n1 *= 2;
                    I1 = LeftRectangleSolving(a, b, n1, functionChoice);
                }
                std::cout << "Значение определенного интеграла = " << I0 << "\n"
                          << "Количество разбиений = " << n;

                return I0;
            }
            case 2: {
                double I0 = RightRectangleSolving(a, b, n, functionChoice);
                double n1 = n * 2;
                double I1 = RightRectangleSolving(a, b, n1, functionChoice);

                while (std::abs(I1 - I0) > EPS) {
                    I0 = I1;
                    n = n1;
                    n1 *= 2;
                    I1 = RightRectangleSolving(a, b, n1, functionChoice);
                }
                std::cout << "Значение определенного интеграла = " << I0 << "\n"
                          << "Количество разбиений = " << n;

                return I0;
            }
            case 3: {
                double I0 = CentralRectangleSolving(a, b, n, functionChoice);
                double n1 = n * 2;
                double I1 = CentralRectangleSolving(a, b, n1, functionChoice);

                while (std::abs(I1 - I0) > EPS) {
                    I0 = I1;
                    n = n1;
                    n1 *= 2;
                    I1 = CentralRectangleSolving(a, b, n1, functionChoice);
                }
                std::cout << "Значение определенного интеграла = " << I0 << "\n"
                          << "Количество разбиений = " << n;

                return I0;
            }
```

```
            case 4:
                double I0 = LeftRectangleSolving(a, b, n, functionChoice);
                double n1 = n * 2;
                double I1 = LeftRectangleSolving(a, b, n1, functionChoice);
                while (std::abs(I1 - I0) > EPS) {
                    I0 = I1;
                    n = n1;
                    n1 *= 2;
                    I1 = LeftRectangleSolving(a, b, n1, functionChoice);
                }
                std::cout << "Значение определенного интеграла методом левых
прямоугольников = " << I0 << "\n"
                          << "Количество разбиений = " << n << "\n";

                I0 = RightRectangleSolving(a, b, n, functionChoice);
                n1 = n * 2;
                I1 = RightRectangleSolving(a, b, n1, functionChoice);
                while (std::abs(I1 - I0) > EPS) {
                    I0 = I1;
                    n = n1;
                    n1 *= 2;
                    I1 = RightRectangleSolving(a, b, n1, functionChoice);
                }
                std::cout << "Значение определенного интеграла методом правых
прямоугольников = " << I0 << "\n"
                          << "Количество разбиений = " << n << "\n";

                I0 = CentralRectangleSolving(a, b, n, functionChoice);
                n1 = n * 2;
                I1 = CentralRectangleSolving(a, b, n1, functionChoice);
                while (std::abs(I1 - I0) > EPS) {
                    I0 = I1;
                    n = n1;
                    n1 *= 2;
                    I1 = CentralRectangleSolving(a, b, n1, functionChoice);
                }
                std::cout << "Значение определенного интеграла методом средних = " <<
I0 << "\n"
                          << "Количество разбиений = " << n << "\n";
        }
    }
};

#endif //MATPLOTLIB_H_RECTANGLEMETHOD_H
```

Метод трапеций:

```
#ifndef MATPLOTLIB_H_TRAPEZIUMMETHOD_H
#define MATPLOTLIB_H_TRAPEZIUMMETHOD_H

#include "Functions.h"

class TrapeziumMethod {
public:
    static double TrapeziumSolving(double a, double b, double n, unsigned short
functionChoice) {
        double h = (b - a) / n;
        double integral = 0;

        std::vector<double> x(n + 1, 0);
        std::vector<double> y(n + 1, 0);

        switch (functionChoice) {
            case 1:
                y[0] = F1(a);
```

```
                y[n] = F1(b);
                break;
            case 2:
                y[0] = F2(a);
                y[n] = F2(b);
                break;
            case 3:
                y[0] = F3(a);
                y[n] = F3(b);
                break;
        }

        for (int i = 1; i <= n; i++) {
            x[i] = a + i * h;
            switch (functionChoice) {
                case 1:
                    y[i] = F1(x[i]);
                    break;
                case 2:
                    y[i] = F2(x[i]);
                    break;
                case 3:
                    y[i] = F3(x[i]);
                    break;
            }
        }

        for (int i = 1; i < n; i++) {
            integral += y[i];
        }

        integral += (y[0] + y[n]) / 2;
        integral *= h;

        return integral;
    }

    static double CalculateIntegral(double a, double b, double n, double EPS, unsigned
short functionChoice) {
        double I0 = TrapeziumSolving(a, b, n, functionChoice);
        double n1 = n * 2;
        double I1 = TrapeziumSolving(a, b, n1, functionChoice);

        while (std::abs(I1 - I0) > EPS) {
            I0 = I1;
            n = n1;
            n1 *= 2;
            I1 = TrapeziumSolving(a, b, n1, functionChoice);
        }
        std::cout << "Значение определенного интеграла = " << I0 << "\n"
                  << "Количество разбиений = " << n;

        return I0;
    }
};

#endif //MATPLOTLIB_H_TRAPEZIUMMETHOD_H
```

Метод Симпсона:

```
#ifndef MATPLOTLIB_H_SIMPSONMETHOD_H
#define MATPLOTLIB_H_SIMPSONMETHOD_H

#include <cmath>
#include <vector>
```

```cpp
#include <iostream>
#include "Functions.h"

class SimpsonMethod {
public:
    static double SimpsonSolving(double a, double b, double n, unsigned short
functionChoice) {
        double h = (b - a) / n;
        double integral = 0;
        double even = 0, odd = 0;

        std::vector<double> x(n + 1, 0);
        std::vector<double> y(n + 1, 0);

        switch (functionChoice) {
            case 1:
                y[0] = F1(a);
                y[n] = F1(b);
                break;
            case 2:
                y[0] = F2(a);
                y[n] = F2(b);
                break;
            case 3:
                y[0] = F3(a);
                y[n] = F3(b);
                break;
        }

        for (int i = 1; i < n; i++) {
            x[i] = a + i * h;
            switch (functionChoice) {
                case 1:
                    y[i] = F1(x[i]);
                    break;
                case 2:
                    y[i] = F2(x[i]);
                    break;
                case 3:
                    y[i] = F3(x[i]);
                    break;
            }
        }

        for (int i = 1; i < n; i++) {
            if (i % 2 == 0) {
                even += y[i];
            } else {
                odd += y[i];
            }
        }
        even *= 2;
        odd *= 4;
        integral = (h / 3) * (y[0] + even + odd + y[n]);

        return integral;
    }

    static double CalculateIntegral(double a, double b, double n, double EPS, unsigned
short functionChoice) {
        double I0 = SimpsonSolving(a, b, n, functionChoice);
        double n1 = n * 2;
        double I1 = SimpsonSolving(a, b, n1, functionChoice);

        while (std::abs(I1 - I0) > EPS) {
            I0 = I1;
            n = n1;
```

```
            n1 *= 2;
            I1 = SimpsonSolving(a, b, n1, functionChoice);
        }
        std::cout << "Значение определенного интеграла = " << I0 << "\n"
                  << "Количество разбиений = " << n;

        return I0;
    }
};

#endif
```

Реализация программы:

```
#include "SimpsonMethod.h"
#include "TrapeziumMethod.h"
#include "RectangleMethod.h"

int main() {
    std::cout << "Введите номер функции, которую хотите проинтегрировать:" << "\n" <<
              "1) F1(x) = x^3 - 3x^2 + 7x - 10" << "\n" <<
              "2) F2(x) = 2x^3 - 3x^2 + 5x - 9" << "\n" <<
              "3) F3(x) = 2x^3 - 9x^2 - 7x + 11" << "\n";
    unsigned short functionChoice;
    std::cin >> functionChoice;

    std::cout << "Выберите способ решения:" << "\n" << "1) Метод прямоугольников" <<
"\n" << "2) Метод трапеций" << "\n"
              << "3) Метод Симпсона" << "\n";
    unsigned short methodChoice;
    std::cin >> methodChoice;

    std::cout << "Введите пределы интегрированя:" << "\n";
    double a, b;
    std::cin >> a >> b;

    std::cout << "Введите точность интегрированя:" << "\n";
    double EPS;
    std::cin >> EPS;

    if (methodChoice == 1) {
        RectangleMethod::RectangleIntegral(a, b, 4, EPS, functionChoice);
    } else if (methodChoice == 2) {
        TrapeziumMethod::CalculateIntegral(a, b, 4, EPS, functionChoice);
    } else if (methodChoice == 3) {
        SimpsonMethod::CalculateIntegral(a, b, 4, EPS, functionChoice);
    }
}
```

5. Примеры и результаты работы программы

```
Введите номер функции, которую хотите проинтегрирова
1) F1(x) = x^3 - 3x^2 + 7x - 10
2) F2(x) = 2x^3 - 3x^2 + 5x - 9
3) F3(x) = 2x^3 - 9x^2 - 7x + 11
1
Выберите способ решения:
1) Метод прямоугольников
2) Метод трапеций
3) Метод Симпсона
3
Введите пределы интегрированя:
2 4
Введите точность интегрированя:
0.01
Значение определенного интеграла = 26
Количество разбиений = 4
Process finished with exit code 0
```

```
Введите номер функции, которую хотите проинтегрировать:
1) F1(x) = x^3 - 3x^2 + 7x - 10
2) F2(x) = 2x^3 - 3x^2 + 5x - 9
3) F3(x) = 2x^3 - 9x^2 - 7x + 11
2
Выберите способ решения:
1) Метод прямоугольников
2) Метод трапеций
3) Метод Симпсона
1
Введите пределы интегрированя:
1 2
Введите точность интегрированя:
0.01
Введите модификацию метода:
1) Метод левых
2) Метод правых
3) Метод средних
4) Вывести решение всеми методами
4
Значение определенного интеграла методом левых прямоугольников = -1.01952
Количество разбиений = 256
Значение определенного интеграла методом правых прямоугольников = -0.980453
Количество разбиений = 256
Значение определенного интеграла методом средних = -1.00001
Количество разбиений = 256

Process finished with exit code 0
```

6. Вывод:

В ходе выполнения работы мы познакомились с численными методами решения определенных интегралов, научились решать их вручную и с помощью программы.