Федеральное государственное автономное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА _____КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)_____

О т ч е т

по домашнему заданию № 1

Название домашнего задания: Программирование с использованием разветвленных и циклических процессов.

Дисциплина: Алгоритмизация и программирование.

Студент гр. ИУ6-13Б _____ **С.М Соболев**
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ **О.А. Веселовская**
(Подпись, дата) (И.О. Фамилия)

Москва, 2025

Часть 1. Вычисления. Погрешности вычислений.

Задача 1.

Цель работы: изучение и оценка точности представления чисел.

Код программы представлен на рисунке 1:

```
#include <iostream>
#include <math.h>

using namespace std;

int main()
{
    float y;
    cout << "До инициализации y = " <<  y  << endl;
    y = 1;
    cout << "После инициализации y = " << y << endl;
    y = y / 6000;
    y = exp(y);
    y = sqrt(y);
    y = y / 14;
    y = 14 * y;
    y = y * y;
    y = log(y);
    y = 6000 * y;
    cout << "После преобразований y = " << y << endl;
    return 0;
}
```

Рисунок 1 – Текст программы

Примерные результаты при запуске программы:

До инициализации y = 4.59037e-41

После инициализации y = 1

После преобразований y = 0.999844

Абсолютная погрешность: $\Delta = |1 — 0.999844| = 0.000156$

Относительная погрешность: $\delta = \Delta / |A| = 0.000156 / 1 = 0.000156$

В данной задаче главными источниками погрешностей будут следующие:

Погрешности округления: связаны с ограничением количества десятичных знаков при хранении чисел типа float. Такие ошибки появляются на каждом шаге арифметических вычислений и постепенно накапливаются.

Погрешности вычислений: возникают при выполнении математических операций над приближенными значениями. Например, функции exp, sqrt и log из стандартной библиотеки C++ могут содержать небольшие неточности из-за используемых алгоритмов их расчёта.

Вывод: в рамках этой лабораторной работы была разработана и проверена программа на C++, предназначенная для оценки ошибок представления чисел и вычислений с числами типа float. Полученные абсолютные и относительные погрешности оказались достаточно большими, что подчёркивает важность учета подобных ошибок при проведении вычислений с высокой точностью.

Задача 2.

Цель: создать программу для вычисления значений гиперболических функций и анализа погрешностей при расчетах с использованием различных типов данных (float, double, long double).

```cpp
#include <iostream>
#include <math.h>
#include <stdio.h>

using namespace std;

int main()
{
    float x, y, y1, y2;
    puts("Enter x:");
    scanf_s("%f", &x);
    y1 = (exp(x) - exp(-x)) / 2;
    y2 = (exp(x) + exp(-x)) / 2;
    y = pow(y2, 2) - pow(y1, 2);
    printf_s("%20.16Lf %20.16Lf %20.16Lf %20.16Lf %20.16Lf", y1, y2, y, fabs(1 - y), fabs(1 - y) / 1);
    return 0;
}
```

Рисунок 2 – Код программы

Таблица 1 – Результаты при использовании float

x	y1	y2	y	Δ	δ
5	74.2032089233398438	74.2099533081054688	1.0009554624557495	0.0009554624557495	0.00095546245574
10	11013.2324218750000000	11013.2324218750000000	0	1	1
15	1634508.6250000000000000	1634508.6250000000000000	0	1	1
20	242582592.000000000000000	242582592.0000000000000000	0	1	1
25	36002451456.0000000000000	36002451456.00000000000000	0	1	1

Таблица 2 - Результаты при использовании double

x	y1	y2	y	Δ	δ
5	74.2032105777887523	74.2099485247878476	1.0000000000018190	0.0000000000018190	0.0000000000
10	11013.2328747033934633	11013.2329201033244317	1.0000000298023224	0.0000000298023224	0.0000000298
15	1634508.6862359023652971	1634508.6862362083047628	1	0	0
20	242582597.7048951387405396	242582597.7048951387405396	0	1	1
25	36002449668.6929397583007812	36002449668.6929397583007812	0	1	1

Таблица 3 - Результаты при использовании long double

x	y1	y2	y	Δ	δ
5	74.2032105777887590	74.2099485247878444	0.9999999999999991	0. 0.00000000000009	0. 0.0000000
10	11013.2328747033933771	11013.2329201033231394	1	0	0
15	1634508.6862359023683666	1634508.6862362082707705	1.0000002384185791	0.0000002384185791	0.0000002384
20	242582597.7048951379547361	242582597.7048951400211081	1.0039062500000000	0.0039062500000000	0.0039062500
25	36002449668.6929362639784813	36002449668.6929362639784813	0	1	1

Вывод: изменение типов переменных на double и long double значительно влияет на точность вычислений, особенно при больших значениях аргумента. Использование типов double и long double позволяет достичь более высокой точности, что особенно важно в задачах, требующих высокого уровня точности и минимизации ошибок округления.

Задача 3.

Цель: создание программы для проверки тригонометрического тождества с использованием численных методов.

```cpp
#include <iostream>
#include <math.h>
#include <stdio.h>
#include <iomanip>

using namespace std;

int main()
{
    long double x, e = 0.00001;
    cout << "Введите x:\n";
    cin >> x;
    long double res = pow(sin(x), 2) + pow(cos(x), 2);
    cout <<  setprecision(20) << res << endl;
    cout << "Абсолютная погрешность:" << fabs(res - 1);
    cout << "Относительная погрешность:" << fabs(res - 1) / 1;
    return 0;
}
```

Рисунок 3 – Код программы

```
Введите x:
5.14
0.99999999999999988898
Абсолютная погрешность:1.1102230246251565404e-16
Относительная погрешность:1.1102230246251565404e-16
```

Рисунок 4 – Результат работы программы при введенном значении 5.14

```
Введите x:
10.1
1
Абсолютная погрешность:0
Относительная погрешность:0
```

Рисунок 5 - Результат работы программы при введенном значении 10.1

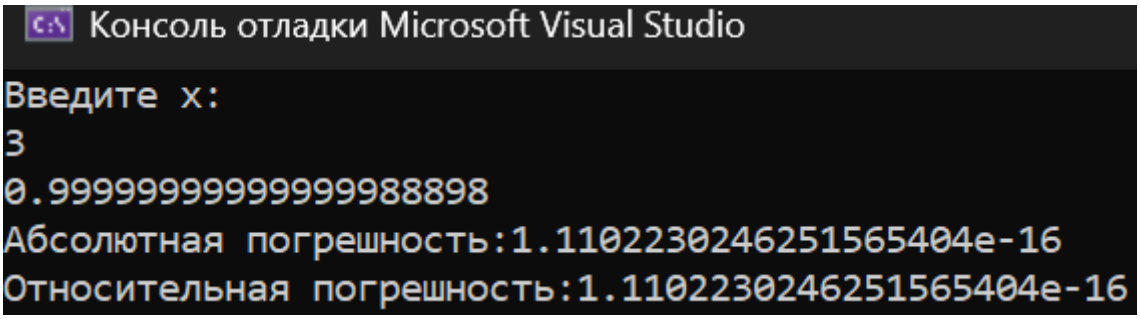

Рисунок 6 - Результат работы программы при введенном значении 3

Вывод: в ходе данного задания была разработана и протестирована программа, которая проверяет тождество sin ^ 2 (x) + cos ^ 2 (x) = 1.

Часть 2. Программирование разветвляющегося вычислительного процесса.

Цель: разработка программы, определяющей выражение max^2(X+Y+2, X+1/2, Y+XW) +1, и ее тестирование.

Задание: Даны действительные числа X, Y и W. Определить max^2(X+Y+2, X+1/2, Y+XW) +1. Протестировать все ветви алгоритма.

Схема алгоритма программы:

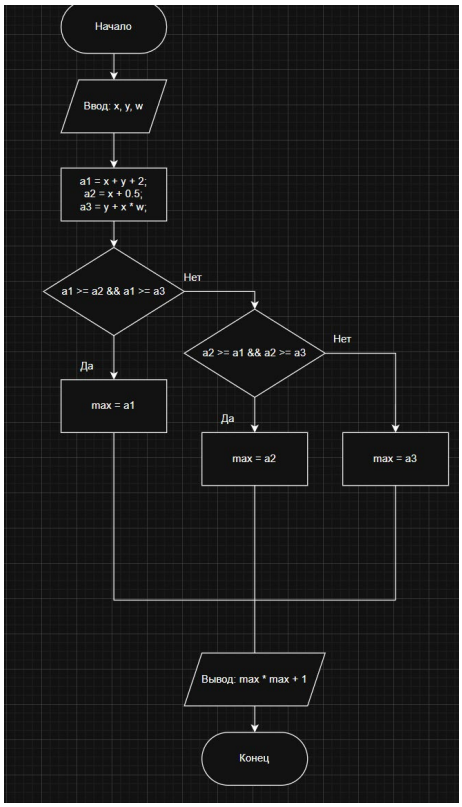

Рисунок 7 – Схема алгоритма программы

```cpp
#include <iostream>
#include <math.h>
using namespace std;

int main()
{
    double x, y, w, max, a1, a2, a3;
    cout << "Введите x, y, w:" << endl;
    cin >> x >> y >> w;
    a1 = x + y + 2;
    a2 = x + 0.5;
    a3 = y + x * w;
    if (a1 >= a2 && a1 >= a3) {
        max = a1;
    }
    else{
        if (a2 >= a1 && a2 >= a3) max = a2;
        else max = a3;
    }

    cout << max * max + 1;
    return 0;
}
```

Рисунок 8 – Код программы

Таблица 4 – Тестирование программы

X	Y	W	Вывод
1	1	1	17
10	-5	-1	111.25
1	5	9	197

Вывод: в результате выполнения задания была создана и протестирована программа, которая определяет значения выражения данного в условии. Программа корректно обрабатывает все возможные случаи.

Часть 3. Программирование циклического процесса.

Цель: создать программу для вычисления суммы ряда, организовав итерационный цикл с точностью E. Значение точности вводится с клавиатуры. Проверить программу при E=10-4, 10-5. Определить, как изменяется число итераций при изменение точности.

Задача: Решить задачу, организовав итерационный цикл с точностью E. Значение точности вводится с клавиатуры.

Вычислить сумму ряда: $S = \sum_{k=1}^{\infty} \frac{1}{k(k+1)}$

Считать точным значение, равное 1.

Проверить программу при E=10-4, 10-5. Определить, как изменяется число итераций при изменение точности.

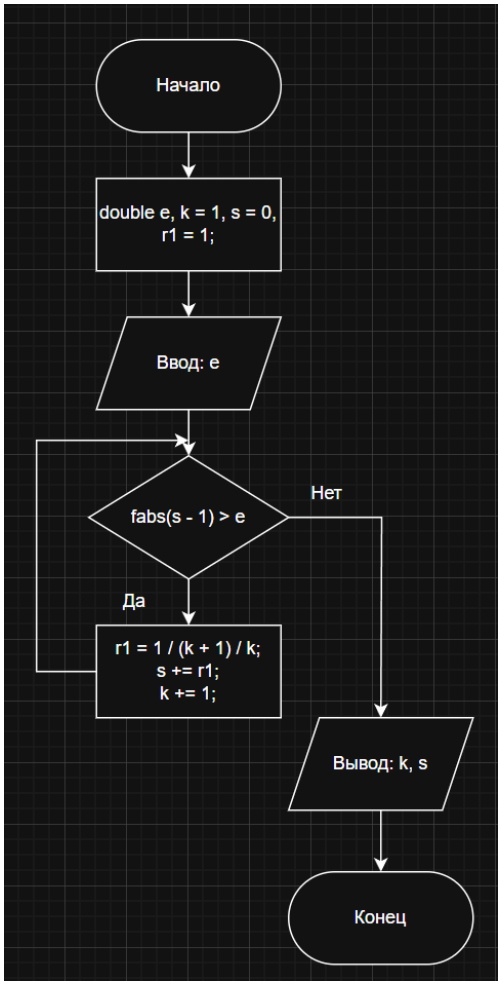

Рисунок 9 – Схема программы

```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;

int main() {
    double e, k = 1, s = 0, r1 = 1, r2 = 0;
    cout << "Введите погрешность:" << endl;
    cin >> e;
    while (fabs(s - 1) > e) {
        r1 = 1 / (k + 1) / k;
        s += r1;
        k += 1;
    }
    cout << k << " " << setprecision(20) << s;
    return 0;
}
```

Рисунок 10 – Код программы

Таблица 5 – Тестирование программы

E	**Вывод**	**Кол-во итераций**
0.01	0.99009900990098975715	100
0.001	0.99900000000000066525	1000
0.0001	0.99990000000000067715	10000
0.00001	0.99999000000001314614	100000

Вывод: программа вычисляет сумму ряда с разной точностью и показывает, что число итераций увеличивается при уменьшении значения є. Результаты программы близки к точному значению, что свидетельствует о корректности работы программы.