Лабораторная работа 0.

Знакомство с C++

Вводная лабораторная работа для знакомства с языком программирования C++. Во время выполнения данной работы вы должны научиться запускать программу, сохранять, открывать уже разработанные программы, написать простые программы.

Каждая лабораторная работа состоит из двух частей:

Часть 1. Набора задач, который необходимо выполнить в системе с автоматической проверкой решения. Для зачета необходимо выполнить не менее 80% заданий. Задача считается выполненной если у нее статус **ОК** в посылке.

Часть 2. Индивидуального или уникального задания, непосредственно связанного с лабораторной работой.

Настоятельно рекомендую прочитать: Как задавать вопросы, чтобы получить максимум пользы от них на гите.

Все лабораторные работы сдаются очно, мною может быть проверена любая задача из части 1. Задания части 2 проверяются полностью.

Для успешной сдачи работы, вам необходимо выполнить 80% первой части работы и вторую полностью.

Часть 0. Исправь программу

Цель: научиться находить и исправлять типичные синтаксические ошибки.

Что нужно сделать:

- Скопируйте код ниже в main.cpp.
- Добейтесь, чтобы программа компилировалась и корректно работала.
- Программа должна считывать два числа и выводить сумму, разность, произведение, частное (с учётом деления на ноль) и корень из первого числа.
- Форматируйте числа с 3 знаками после запятой.
- Не забывайте про математические ограничения которые могут быть в этой программе, корректно их обрабатывайте.

```cpp
# include <Iostream>
#include "cmath"
// #include <iomanip>   // возможно, пригодится?

usingn namespace std

int main() {
    int a, b;
    cout << "Введите два числа: "
    cin >> a; cin << b;

    double sum = a + b;
    double diff = a - b
    double mul = a * b;
    double div = a / b;

    setprecision(3);
    cout << fixed;
    cout << "Сумма: " << sum << end1;
    cout << "Разность: " << diff << endl;
    cout << "Произведение: " << mul << std::endl
    cout << "Частное: " << div << endl;

    double root = sqrt(a);
    cout << "Корень из a: " << root << endl;

    return 0
}
```

Периметр и площадь — почему «ломается»?

Цель: на практике понять, что такое переполнение целых типов в C++, как его распознать и устранить.

Что программа должна уметь: считать стороны прямоугольника a и b, посчитать периметр P = 2*(a+b) и площадь S = a*b.

Скопируйте код, соберите, запустите.

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    int a, b;
    cout << "Введите a и b: ";
    cin >> a >> b;

    int sum = a + b;        // Шаг 1: сумма
    int P   = 2 * sum;      // Шаг 2: периметр
    int S   = a * b;        // Шаг 3: площадь

    cout.setf(std::ios::fixed);
    cout << setprecision(3);

    cout << "sum = " << sum << "\n";
    cout << "P   = " << P   << "\n";
    cout << "S   = " << S   << "\n";
    return 0;
}
```

Тесты, которые нужно выполнить и понимать результат:

1. a = 3, b = 4.
2. a = 1000000000, b = 1000000000 (1e9 и 1e9).
3. a = 2000000000, b = 2000000000 (2e9 и 2e9).

Для каждого теста:

- Что вывела программа?
- Напишите, что *должно* было получиться по математике.
- Выделите, на каком шаге «начинаются» странности: на сумме, на умножении на 2, на площади?

Выясняем причину

Добавьте в начало main() такие строки и снова выполните тест 2:

```
#include <limits>   // (добавьте выше)
...
cout << "INT_MAX = " << std::numeric_limits<int>::max() << "\n";
cout << "INT_MIN = " << std::numeric_limits<int>::min() << "\n";
```

Сравните математическое значение 2*(a+b) с INT_MAX. Почему результат «сломался»?

Почему иногда «ломается» уже a*b?

Попытка исправления №1 (и ещё один подвох)

Замените типы P и S на long long, НО оставьте a и b типа int:

```
long long P2 = 2 * (a + b);
long long S2 = a * b;
cout << "P2  = " << P2 << "\n";
cout << "S2  = " << S2 << "\n";
```

Повторите вычисления. Объясните, почему P2 и S2 **всё ещё могут быть неверными**, хотя переменные результата — широкого типа.

Сделайте так, чтобы **всё выражение** вычислялось в широком типе:

```
long long P_ok = 2LL * (static_cast<long long>(a) + b);
long long S_ok = static_cast<long long>(a) * b;
cout << "P_ok = " << P_ok << "\n";
cout << "S_ok = " << S_ok << "\n";
```

Повторите тесты 1–3 и запишите результаты.

Коротко объясните, почему теперь корректно.

Защита от переполнения (опционально)

Добавьте проверки до вычисления (если хотите, только для суммы/периметра):

```cpp
#include <climits> // INT_MAX
...
bool safe_sum = (a <= INT_MAX - b);
if (!safe_sum) {
    cout << "Сумма не помещается в int\n";
} else {
    int sum = a + b;
    // и т.д.
}
```

Часть 1. Выполните следующие задачи по ссылке:

https://informatics.msk.ru/mod/statements/view.php?id=78406#1

Часть 2. Выполните задания

Задание 1. Консольный калькулятор (базовые операции)

Цель: закрепить ввод/вывод в C++ и работу с простыми арифметическими операциями.

Требования

- Программа читает **три** элемента: первое число, оператор, второе число. Формат ввода: <число> <оператор> <число> (Оператор как один символ (char), Оператор >> **пропускает пробелы и перевод строки)**.
  Примеры операторов: +, -, *, /.
- Числа — вещественные (используйте double).
- Вывод: одно число — результат операции в **фиксированном формате** с **3 знаками** после запятой.
- Если оператор неизвестен — вывести: Ошибка: неизвестная операция.
- Если деление на ноль — вывести: Ошибка: деление на ноль.

Учтите, что интерфейс программы должен быть **максимально дружелюбен к пользователю**. При проверке данного задания, преподаватель выступает в роли человека, который с компьютером совсем не знаком и программа должна помогать ему разобраться с тем как работает ваш калькулятор

Задание 2. Функции: выражения и простые утилиты

Цель: привыкнуть раскладывать задачу на небольшие функции без ввода/вывода внутри них.

Реализуйте только вычисления в функциях, а ввод/вывод выполните в main().

1) Математические функции (тип double)

```
double f1(double x);         // x^2 + 2x + 1
double f2(double x);         // sin(x)/x, считать что f2(0) = 1 (по пределу)
double hyp(double a, double b); // √(a^2 + b^2)
double deg2rad(double deg); // перевод градусов в радианы
double rad2deg(double rad); // перевод радиан в градусы
```

2) Базовые утилиты

```
double sum(double a, double b);
double diff(double a, double b);
double prod(double a, double b);

// Возвращает частное; если b == 0, не меняет result и возвращает false
bool try_div(double a, double b, double& result);

int    min3(int a, int b, int c);
bool   is_even(int n);
```

Требования к программе

- Подключите заголовки: <iostream>, <iomanip>, <cmath>.
- В функциях нет cout/cin. Они только считают и возвращают значение.
- В main():
  - Считайте тестовые значения и вызовите все функции.
  - Выводите результаты в фиксированном формате с 3 знаками после запятой (где уместно).
  - Для деления используйте try_div: при b == 0 выведите Ошибка: деление на ноль.

Задание 3. Найти значение алгебраического выражения, соответствующего варианту задания. Вывести результаты на печать. Все результаты выводить в развернутом виде (например: «Сумма чисел A и B равна 3.7854»). Значения вводимых величин должны иметь не менее четырех значащих цифр и задаются студентом самостоятельно. Через a и b обозначены подлежащие вводу числа с плавающей точкой.

1) $\sin^2(a+b^3)\sqrt{\frac{e^{a^2-9,4}}{(a+b)^3}}$

2) $\ln\left(\frac{\sin a^2+\cos b}{\sqrt{1+\frac{e}{a^3+3,4b}}}\right)$

3) $\frac{arctg\left(\frac{\sin(a+\pi)}{\cos(b+2,87)}\right)}{\sqrt{a+\cos^2 b}}$

4) $arctg\left(\frac{\sin a^2+b^3}{\sqrt{1+\frac{b}{a+\cos(\pi\cdot b^2)}}}\right)$

5) $\sqrt{\frac{\sin(\pi+b^2)}{a^2(\sin a+b^2\cos 7,2)}}$

6) $\frac{17,8a+\cos^2(13,2b^2+2,4b+3,7)}{\sqrt{e+\frac{13,7-\sin(a^2+b)}{-17,478a+13,2b}}}$

8) $\ln\left(\frac{\cos(a^2+8,72)+\sin^2(\pi\cdot b)}{\sqrt{1,2+\frac{a}{a^3+3,47b}}}\right)$

9) $arctg\left(\frac{\cos(2\pi\cdot b^2)+a^3}{\sqrt{1,87+\frac{a^2}{b+\cos(3,42b^2)}}}\right)$

10) $\sqrt{\frac{\cos(\pi\cdot a)-1,2b^2}{b^2(\cos a^2+b^2\cos(13,4b))}}$

11) $\sqrt{\frac{e^{-\frac{a^2+e}{a^3+b}}(1,7a^2+b^3)}{\sin(b+a^2)+\cos(1,92b/a)}}$

12) $\cos a^2\frac{(a+b)^3-\cos^2(a^2+b)}{\sqrt{\sin^2 b+\cos^2(\pi-a)^3}}$

13) $\cos^2(a^3-\pi)\sqrt{\frac{e^{-b^2+4,9}}{(a-b)^3}}$

15) $ctg\left(\frac{\sqrt{e+\frac{a+4,8}{b-a^2}}}{\sqrt{a+\cos^2(a+b)}}\right)\cdot e^{\frac{\sin a}{\cos b}}$

16) $\sqrt{\frac{\cos(a-b^2+2ab)}{b^2(\cos(b-a)^2+\pi^2\sin 7,4)}}\times$
$\times\ln\frac{\cos(a-b)^3}{\sin(a+b)^3}$

17) $\frac{7,8b+\sin^2(8,5a^2-2ab+4,7b)}{\sqrt{2,6+\frac{1,3\pi-\cos(b^2-a)}{18,87b+7,85a}}}$

18) $\cos^3(b^2-a^2)\sqrt{\frac{e^{(-a^2+2ab-b^2)}}{(\pi-b)+(a+\pi)^3}}$

19) $\ln\left(\frac{\cos(a^2+b)+\sin^2(7,6b)}{\sqrt{2,2e+\frac{b}{a-\cos^2(b^2+a^2)}}}\right)$

20) $arctg\left(\frac{\sin(2,48a^2)-b^3}{\sqrt{1,8+\frac{\pi}{a+\cos(3,45b^2)}}}\right)$

Задание 4. Самопроверка и мини-юнит-тесты

Цель: научиться писать небольшие функции и проверять их корректность с помощью простого тест-набора без сторонних библиотек.

1. Реализуйте функции (только вычисления, без cin/cout внутри):

```
double mean(double a, double b);            // среднее арифметическое
bool   try_div(double a, double b, double& q); // q = a/b; вернуть false, если b == 0
double hyp(double a, double b);             // √(a^2 + b^2)
```

2. Сделайте в main() простой «тест-раннер», который:

- вызывает функции на ряде входов,
- сравнивает результат с ожидаемым,

- печатает «PASS/FAIL» и итог: ALL TESTS PASSED или TESTS FAILED: N.

3. Добавьте минимум 6 проверок, среди них обязательно:
   - «нормальные» случаи (например, hyp(3,4) = 5),
   - пограничный случай для деления (b == 0),
   - проверка на близость вещественных чисел с допуском (эпсилон).

```
#include <iostream>
#include <cmath>
#include <string>
#include <iomanip>
using namespace std;

// ---------- 1) Функции, которые вы тестируете ----------
double mean(double a, double b) {
    // TODO: вернуть среднее арифметическое
    return 0.0;
}

bool try_div(double a, double b, double& q) {
    // TODO: если b == 0 -> вернуть false и НЕ менять q
    // иначе записать a/b в q и вернуть true
    return false;
}

double hyp(double a, double b) {
    // TODO: вернуть sqrt(a*a + b*b)
    return 0.0;
}

// ---------- 2) Мини-тестовый фреймворк ----------
bool approx_equal(double x, double y, double eps = 1e-6) {
    return std::fabs(x - y) <= eps;
}
int fails = 0;

void expect_close(const string& name, double got, double expected, double eps = 1e-6) {
    if (approx_equal(got, expected, eps)) {
        cout << "[PASS] " << name << "\n";
    } else {
        cout << "[FAIL] " << name << "  got=" << fixed << setprecision(6) << got
             << "  expected=" << expected << "\n";
        fails++;
    }
}

void expect_true(const string& name, bool cond) {
    if (cond) cout << "[PASS] " << name << "\n";
    else { cout << "[FAIL] " << name << "\n"; fails++; }
}
```

```cpp
int main() {
    cout.setf(std::ios::fixed);
    cout << setprecision(3);

    // БАЗОВЫЕ ПРОВЕРКИ (добавьте свои!)
    expect_close("mean: (2,4) -> 3", mean(2,4), 3.0);
    expect_close("hyp: (3,4) -> 5", hyp(3,4), 5.0);

    double q = -123.0;
    bool ok = try_div(10, 4, q);
    expect_true ("try_div ok (10/4)", ok);
    expect_close("try_div q (10/4)",  q, 2.5);

    // Деление на ноль: ok == false, q не меняется
    q = -123.0;
    ok = try_div(5, 0, q);
    expect_true ("try_div false on /0", !ok);
    expect_close("try_div q unchanged on /0", q, -123.0);

    // Примеры дополнительных ваших тестов:
    // expect_close("mean: (-1,1) -> 0", mean(-1,1), 0.0);
    // expect_close("hyp: (0,0) -> 0",  hyp(0,0), 0.0);

    if (fails == 0) cout << "ALL TESTS PASSED\n";
    else            cout << "TESTS FAILED: " << fails << "\n";
    return 0;
}
```