ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ПРАКТИЧЕСКОЙ РАБОТЕ №1

По дисциплине «Введение в языки программирования»

Выполнил: ст. гр. ТКИ – 142

Плешанов И.С.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2024

Оглавление

[**Задание 1–1** 3](#_Toc91434532)

[**Условие задания** 3](#_Toc91434533)

[**Блок-схема алгоритма** 4](#_Toc91434534)

[**Программа на языке С** 5](#_Toc91434535)

[**Решение тестовых примеров на С** 6](#_Toc91434536)

[**Решение тестовых примеров в Python** 7](#_Toc91434537)

[**Задание 1–2** 8](#_Toc91434538)

[**Формулировка задания** 8](#_Toc91434539)

[**Блок-схема алгоритма** 9](#_Toc91434540)

[**Программа на языке С** 10](#_Toc91434541)

[**Решение тестовых примеров на С** 11](#_Toc91434542)

[**Решение тестовых примеров в** **Python** 12](#_Toc91434543)

[**Задание 1–3** 14](#_Toc91434545)

[**Формулировка задания** 14](#_Toc91434546)

[**Блок-схема алгоритма** 15](#_Toc91434547)

[**Программа на языке С** 16](#_Toc91434548)

[**Решение тестовых примеров на С** 17](#_Toc91434549)

[**Решение тестовых примеров в** **Python** 18](#_Toc91434550)

# **Задание 1–1**

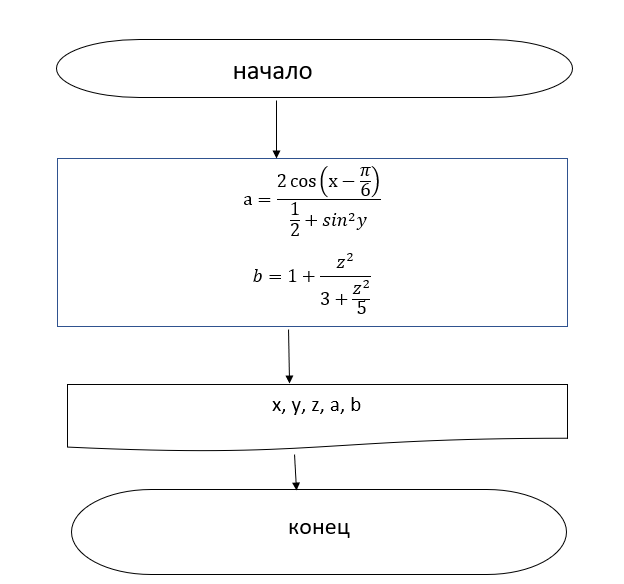
## **Условие задания**

Создать консольное приложение, вычисляющее значения переменных по представленным в таблице формулам. Расчёт примера осуществить по заданным константам. Вывести на экран значения исходных данных, а также результат вычислений. Дополнить свой отчёт блок-схемой алгоритма.

Таблица

|  |  |  |
| --- | --- | --- |
| Номер варианта | Формула | Значения констант |
| 1 |  | x=1.426  y=-1.22  z=3.5 |

## **Блок-схема алгоритма**

****

**Рисунок 1 – Блок-схема 1-1**

## **Программа на языке С**

#include <stdio.h>

#include <math.h>

/\*

\* @brief Функция getA принимает два аргумента x и y и возвращает вычисленное значение.

\* @param x Первый аргумент.

\* @param y Второй аргумент.

\*/

double getA(const double x, const double y);

/\*

\* @brief Функция getB принимает три аргумента x, y и z и возвращает вычисленное значение.

\* @param x Первый аргумент.

\* @param y Второй аргумент.

\* @param z Третий аргумент.

\* @return Вычисленное значение по формуле.

\*/

double getB(const double x, const double y, const double z);

/\*

\* @brief Главная функция программы.

\* Функция main инициализирует значения переменных x, y, z и вычисляет результаты функций getA и getB.

\* Затем выводит значения x, y, z, результат getA и результат getB на стандартный вывод.

\* @return Возвращает 0 при успешном завершении программы.

\*/

int main() {

const double x = 1.426;

const double y = -1.22;

const double z = 3.5;

printf("%lf\n%lf\n%lf\n%lf\n%lf\n", x, y, z, getA(x, y), getB(x, y, z));

return 0;

}

double getA(const double x, const double y) {

return (2 \* cos(x - (M\_PI / 6))) / (0.5 + sin(y) \* sin(y));

}

double getB(const double x, const double y, const double z) {

return (1 + pow(z, 2)) / (3 + (pow(z, 2) / 5));

}

## **Решение тестовых примеров на С**

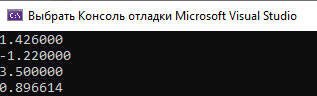


Рисунок 2 – Решение тестового примера для задания 1-1 на С

## **Решение тестовых примеров в Python**

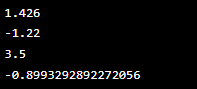


Рисунок 3 - Решение тестового примера задания 1-1 в Python

# **Задание 1–2**

## **Формулировка задания**

Создать консольное приложение для решения задачи, представленной в таблице. Данные для решения вводит пользователь. Вывести результат вычислений на экран. При необходимости дополнить свой отчёт поясняющими формулами, помогающими решить задачу. Дополнить свой отчёт блок-схемой алгоритма.

Таблица 2

|  |  |
| --- | --- |
| Номер варианта | Задача |
| 2 | Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба. |

## **Блок-схема алгоритма**

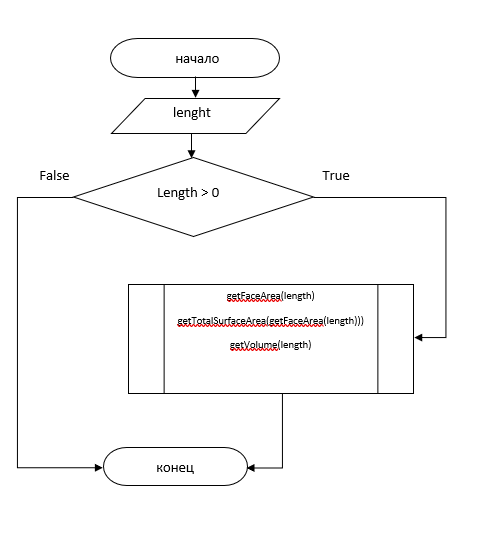


Рисунок 4 – Блок-схема 1-2

## **Программа на языке С**

#define \_USE\_MATH\_DEFINES

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include <locale.h>

/\*

\* @brief Функция ввода числа типа double с консоли с проверкой на корректность ввода.

\* @return Возвращает введенное с консоли число.

\*/

double input(void);

/\*

\* @brief Функция расчета площади грани куба.

\* @param length Длина ребра куба.

\* @return Возвращает площадь грани куба.

\*/

double getFaceArea(const double length);

/\*

\* @brief Функция расчета площади полной поверхности куба.

\* @param face\_area Площадь грани.

\* @return Возвращает площадь полной поверхности куба.

\*/

double getTotalSurfaceArea(const double face\_area);

/\*

\* @brief Функция расчета объема куба.

\* @param length Длина ребра куба.

\* @return Возвращает объем куба.

\*/

double getVolume(const double length);

/\*

\* @brief Точка входа в программу.

\* @return Возвращает в случае успеха.

\*/

int main(void) {

setlocale(LC\_ALL, "Rus");

printf("Введите длину ребра куба:\t");

const double length = input();

printf("Площадь грани куба: %lf\n", getFaceArea(length));

printf("Площадь полной поверхности куба: %lf\n", getTotalSurfaceArea(getFaceArea(length)));

printf("Объем куба: %lf\n", getVolume(length));

return 0;

}

double input(void) {

double number = 0.0;

if (!scanf\_s("%lf", &number)) {

printf("Ошибка ввода\n");

exit(EXIT\_FAILURE);

}

return number;

}

double getFaceArea(const double length) {

return length \* length;

}

double getTotalSurfaceArea(const double face\_area) {

return 6 \* face\_area;

}

double getVolume(const double length) {

return pow(length, 3);

}

## **Решение тестовых примеров на С**

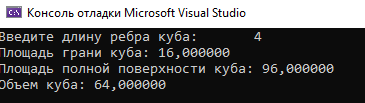


Рисунок 5 – Решение тестового примера для задания 1-2 на С

## **Решение тестовых примеров в Python**

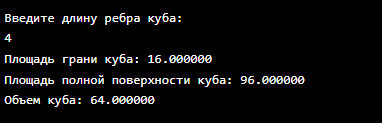


Рисунок 6 – Решение тестового примера для задания 1-2 в Python

# **Задание 1–3**

## **Формулировка задания**

Создать консольное приложение для решения задачи, представленной в таблице. Данные для решения вводит пользователь. Помните, что ввод необходимо проверять на правильность (только числа). Вывести результат вычислений на экран. При необходимости дополнить свой отчёт поясняющими формулами, помогающими решить задачу.

Таблица 3

|  |  |
| --- | --- |
| Номер варианта | Задача |
| 1 | На полу стоит мешок с картошкой массой *m* кг. Вычислите силу тяжести и вес мешка. |

## **Блок-схема алгоритма**

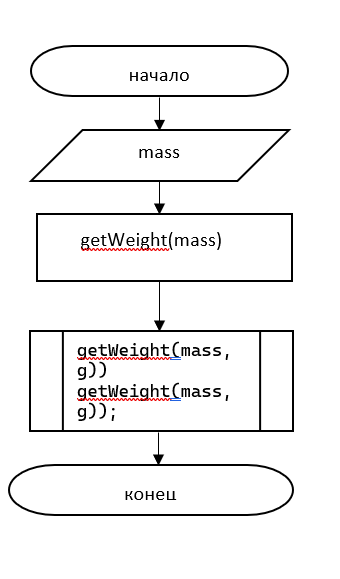


Рисунок 7 – Блок-схема для задания 1-3

## **Программа на языке С**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

/\*

\* @brief Вычисляет вес (силу тяжести) объекта

\* @param mass Масса объекта в килограммах

\* @param g Ускорение свободного падения в м/с²

\* @return Возвращает вес объекта в ньютонах

\*/

double getWeight(const double mass, const double g);

/\*

\* @brief Вводит число с проверкой

\* @return Введенное число типа double

\* Выводит сообщение об ошибке и завершает программу в случае некорректного ввода.

\*/

double input(void);

/\*

\* @brief Главная функция программы

\* Запрашивает у пользователя массу объекта и рассчитывает его вес

\* с использованием фиксированного значения ускорения свободного падения.

\* Результат выводится на экран.

\* @return 0, если программа завершилась успешно.

\*/

int main(void) {

printf("Enter mass:\t");

const double mass = input();

const double g = 9.81;

printf("Weight:\t%lf\n", getWeight(mass, g));

printf("Gravity:\t%lf\n", getWeight(mass, g));

return 0;

}

double input(void) {

double number = 0.0;

if (!scanf\_s("%lf", &number)) {

printf("Input Error\n");

exit(EXIT\_FAILURE);

}

return number;

}

double getWeight(const double mass, const double g) {

return (mass \* g);

}

## **Решение тестовых примеров на С**

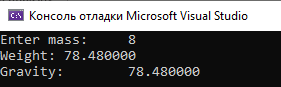


Рисунок 8 – Решение тестового примера для задания 1-3 на С

## **Решение тестовых примеров в Python**

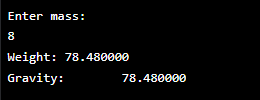


Рисунок 9 – Решение тестового примера для задания 1-3 в Python