# Лабораторная работа № 2. Программирование на С++

## Урок № 1

### Здание № 1

***main.cpp:***

#include <iostream>

int readNumber() {

    std::cout << "Введите число: ";

    int number;

    std::cin >> number;

    return number;

}

void writeAnswer(int result) {

    std::cout << "Сумма чисел: " << result << std::endl;

}

int main() {

    int first\_number = readNumber();

    int second\_number = readNumber();

    writeAnswer(first\_number + second\_number);

    return 0;

}

### Здание № 2

***main.cpp:***

int readNumber();

void writeAnswer(int result);

int main() {

    int first\_number = readNumber();

    int second\_number = readNumber();

    writeAnswer(first\_number + second\_number);

    return 0;

}

***io.cpp:***

#include <iostream>

int readNumber() {

    std::cout << "Введите число: ";

    int number;

    std::cin >> number;

    return number;

}

void writeAnswer(int result) {

    std::cout << "Сумма чисел: " << result << std::endl;

}

### Здание № 3

***main.cpp:***

#include "io.h"

int main() {

    int first\_number = readNumber();

    int second\_number = readNumber();

    writeAnswer(first\_number + second\_number);

    return 0;

}

***io.cpp:***

#include <iostream>

int readNumber() {

    std::cout << "Введите число: ";

    int number;

    std::cin >> number;

    return number;

}

void writeAnswer(int result) {

    std::cout << "Сумма чисел: " << result << std::endl;

}

***io.h:***

#ifndef IO\_H

#define IO\_H

int readNumber();

void writeAnswer(int result);

#endif

## Урок № 2

### Здание № 1

**Почему символьные константы лучше литеральных (магических чисел)? Почему использование const лучше использования директив #define?**

Так как использование литеральных констант (магических чисел) не только усложняет программу, но и затрудняет внесение в нее изменений. Символьные константы лучше, потому что они дают понимание того, зачем и почему они используются, а также, если вам понадобится внести изменения — достаточно будет внести правки только в объявлении константы, а не искать их по всему коду. Значения констант, объявленных с помощью #define, не отображаются в отладчике, у таких констант выше вероятность возникновения конфликта имён.

### Здание № 2

**Выберите подходящий тип данных для переменных в каждой из следующих ситуаций. Будьте как можно более конкретными. Если ответом является целочисленный тип данных, то используйте соответствующий тип с фиксированным размером (например, int16\_t). Если переменная должна быть константной, то так и отвечайте.**

* Возраст пользователя.
  + unsigned \_\_int8
* Нравится ли определенный цвет пользователю?
  + bool
* Число Пи.
  + const double
* Количество страниц в учебнике.
  + \_\_int16
* Цена акций в долларах (дробь присутствует).
  + float
* Сколько раз вы моргнули за всю свою жизнь? (Примечание: Ответ исчисляется в миллионах)
  + unsigned \_\_int32
* Пользователь выбирает опцию с помощью ввода определенной буквы.
  + unsigned \_\_int8

### Здание № 3

***main.cpp:***

#include <iostream>

double getDoubleNumber() {

    std::cout << "Введите число с плавающей запятой: ";

    double number;

    std::cin >> number;

    return number;

}

char getMathOperator() {

    std::cout << "Введите один из следующих операторов: +, -, \* или /: ";

    char math\_operator;

    std::cin >> math\_operator;

    return math\_operator;

}

void printResult(double first\_num, char math\_operator, double second\_num) {

    if (math\_operator == '+')

        std::cout << first\_num << " + " << second\_num << " = " << first\_num + second\_num << "\n";

    else if (math\_operator == '-')

        std::cout << first\_num << " - " << second\_num << " = " << first\_num - second\_num << "\n";

    else if (math\_operator == '\*')

        std::cout << first\_num << " \* " << second\_num << " = " << first\_num \* second\_num << "\n";

    else if (math\_operator == '/')

        std::cout << first\_num << " / " << second\_num << " = " << first\_num / second\_num << "\n";

}

int main() {

    double first\_number = getDoubleNumber();

    double second\_number = getDoubleNumber();

    char math\_operator = getMathOperator();

    printResult(first\_number, math\_operator, second\_number);

    return 0;

}

### Здание № 4

***main.cpp:***

#include <iostream>

#include "constants.h"

double getInitialHeight() {

    std::cout << "Введите высоту башни в метрах: ";

    double initial\_height;

    std::cin >> initial\_height;

    return initial\_height;

}

double calculatingHeight(double initial\_height, int seconds) {

    double fall\_distance = (myConstants::GRAVITY \* seconds \* seconds) / 2;

    double current\_height = initial\_height - fall\_distance;

    return current\_height;

}

void printHeight(double height, int seconds) {

    if (height > 0.0)

        std::cout << "На " << seconds << " секунде, мяч находится на высоте " << height << " метр(а,ов).\n";

    else

        std::cout << "На " << seconds << " секунде, мяч на земле.\n";

}

void calculateAndPrintHeight(double initial\_height, int seconds) {

    double height = calculatingHeight(initial\_height, seconds);

    printHeight(height, seconds);

}

int main() {

    const double initial\_height = getInitialHeight();

    calculateAndPrintHeight(initial\_height, 0);

    calculateAndPrintHeight(initial\_height, 1);

    calculateAndPrintHeight(initial\_height, 2);

    calculateAndPrintHeight(initial\_height, 3);

    calculateAndPrintHeight(initial\_height, 4);

    calculateAndPrintHeight(initial\_height, 5);

    return 0;

}

***constants.h:***

#ifndef CONSTANTS\_H

#define CONSTANTS\_H

namespace myConstants {

    const double GRAVITY(9.8);

}

#endif

## Урок № 3

### Здание № 1

**Вычислите результат следующих выражений:**

* (5 > 3 && 4 < 8)
  + true
* (4 > 6 && true)
  + false
* (3 >= 3 || false)
  + true
* (true || false) ? 4 : 5
  + 4

### Здание № 2

**Вычислите результат следующих выражений:**

* 7 / 4
  + 1
* 14 % 5
  + 4

### Здание № 3

**Конвертируйте следующие двоичные числа в десятичную систему счисления:**

* 1101
  + 13
* 101110
  + 46

### Здание № 4

**Конвертируйте следующие десятичные числа в двоичную систему счисления:**

* 15
  + 1111
* 53
  + 110101

### Здание № 5

**Почему вы никогда не должны делать следующее:**

* int y = foo(++x, x);
  + ++ изменяет значение х при передаче аргументов. Неясно, передастся второй аргумент как х или как х + 1.
* int x = 7 / -2; // (до C++11)
  + Неясно, как будет сделано округление.
* int x = -5 % 2; // (до C++11)
  + Неясно, какой будет знак в результате.
* float x = 0.1 + 0.1; if (x == 0.2) return true; else return false;
  + Ошибки округления со значениями типа с плавающей запятой приведут к результату false, хотя кажется, что должно быть true.
* int x = 3 / 0;
  + При делении на ноль будет ошибка.