



Lesson 1

Компиляция программы. Схема компиляции. Ввод/Вывод данных. C++ типы данных. `sizeof()`. Переменные. Операции над переменными. Логические операции. Shift операции. Операторы инкремента (`++`), декремента (`--`). Условная конструкция (`if-else`). Примеры.



Что такое программирование? Кто такой программист?

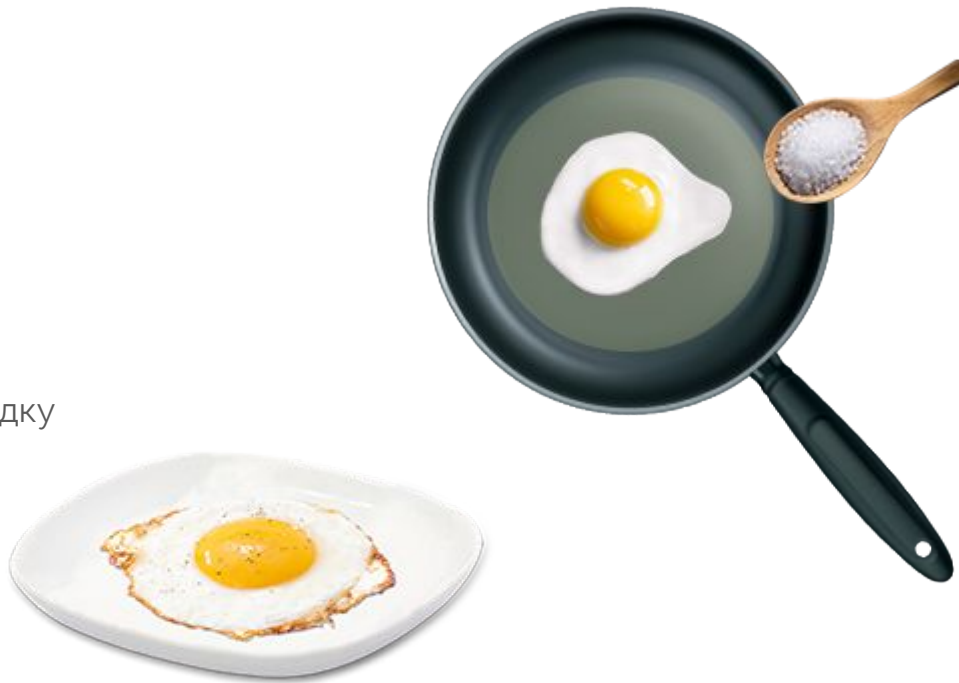



Простая Яичница (ингредиенты):

- масло (растительное или сливочное);
- 1 яйцо;
- соль.

Шаги приготовления:

- налить или положить масло на сковородку
- разбить яйцо
- посолить
- жарить пока не приготовится





Программирование - это как писать рецепт для компьютера. Вместо ингредиентов и шагов приготовления, мы используем код и команды.

Программист - это тот кто готовит этот рецепт, тот кто пишет программы (тот кто кодит). Если словами кухни - это создатель (повар).



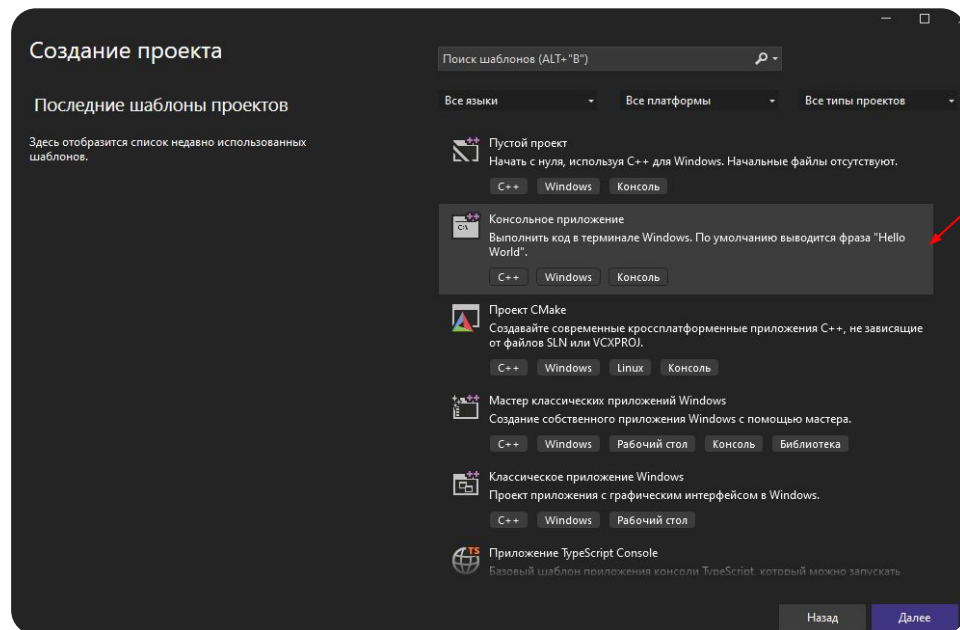
Что такое среда разработки (IDE)?

Среда разработки (IDE) - "*Integrated Development Environment*" это рабочее пространство для программиста. Она предоставляет удобный интерфейс, в котором можно писать код, проверять его на ошибки, выполнять и отлаживать программы, а также управлять проектами.

Примеры: Visual Studio, VScode, Clion...

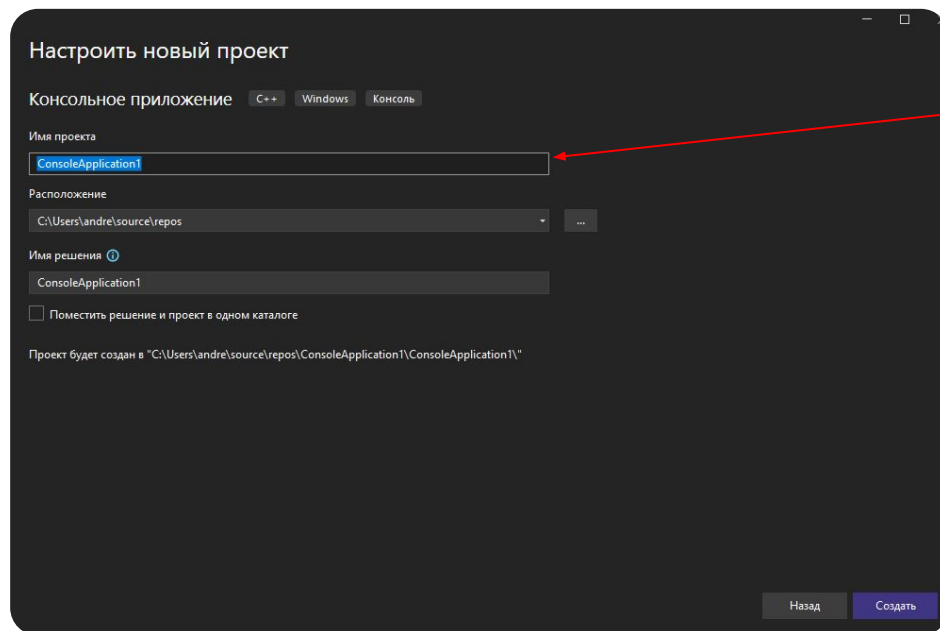
<https://visualstudio.microsoft.com/ru/vs/features/cplusplus/>

Создание проекта в IDE. Visual Studio



*выбираем консольное
приложение C++*

Создание проекта в IDE. Visual Studio



Настроить новый проект

Консольное приложение C++ Windows Консоль

Имя проекта
ConsoleApplication1

Расположение
C:\Users\andre\source\repos

Имя решения ⓘ
ConsoleApplication1

☐ Поместить решение и проект в одном каталоге

Проект будет создан в "C:\Users\andre\source\repos\ConsoleApplication1\ConsoleApplication1\."

Назад Создать

указываем имя проекта C++



Версии C++.

98++:

Основные синтаксические нормы (которые перетекли прямо из языка C (СИ))

Основные функции:

- Введение шаблонов;
- Виртуальные функции;
- Пространства имен (namespaces);
- Вывод типов по умолчанию (type inference)

11++:

- **Автоматическое выведение типов** с использованием auto
- **Лямбда-функции**
- **Смарт-указатели** (std::shared_ptr, std::weak_ptr)
- **Статические утверждения** (static_assert)
- **Перемещение семантики и конструкторы перемещения**
- **Стандартная библиотека** (например, std::array, std::tuple, std::unordered_map)
- **Константные выражения** (constexpr)



Версии C++.

14++:

изменение/добавления функций:

- **Обобщенные лямбды** (generic lambdas)
- **Расширенные constexpr**
- **Автоматическое выведение типов** для возвращаемых значений функций
- **Двоичный литерал** (например, 0b1101)

17++:

- **Структурное приведение** (structured bindings)
- **Введение std::optional, std::variant, std::any**
- **Расширенные constexpr**
- **if constexpr** (упрощает написание шаблонного кода)
- **Внутренние пространства имен** (nested namespaces)
- **Файловая система** (std::filesystem)



Версии C++.

20++:

- **Концепты** (concepts)
- **Диапазоны** (ranges)
- **Корутины** (coroutines)
- **Модули** (modules)
- **Трилинейные операторы** (<=>, также известные как «spaceship operator»)
- **Константные выражения** улучшены (расширенные constexpr)
- **Классическая библиотека** (например, std::span)

23++:

- **Стековые корутины** (stackless coroutines)
- **Автоматическое выведение типов для this** (deduced this)
- **Улучшения в работе с модулями**
- **Улучшенные constexpr функции**
- **Параллельные алгоритмы** с поддержкой параллельных исполнений
- **Обработка исключений** в constexpr функциях

Преимущества версий C++.

C++11

Рекомендован для: Проектов, которым нужна стабильная и хорошо поддерживаемая версия. Обучения современным концепциям C++.

Причины использования:

- **Семантика перемещения:** Уменьшает количество копий объектов, что повышает производительность.
- **Смарт-указатели:** Управляют временем жизни объектов автоматически, предотвращая утечки памяти.
- **Лямбда-выражения:** Упрощают написание анонимных функций.
- **Автоматическое выведение типов (auto):** Уменьшает количество шаблонного кода и делает код более читаемым.

C++14

Рекомендован для: Проектов, которые хотят воспользоваться улучшенными возможностями C++11.

Причины использования:

- **Обобщенные лямбды:** Делают лямбда-функции более гибкими.
- **Расширенные constexpr:** Позволяют выполнять более сложные вычисления во время компиляции.
- **Улучшенные возможности выведения типов:** Делают код более чистым и читаемым.

C++17

Рекомендован для: Проектов, которые хотят использовать современные возможности и улучшения производительности.

Причины использования:

- **Структурные приведения:** Упрощают распаковку кортежей и пар.
- **Введение новых контейнеров (std::optional, std::variant, std::any):** Повышает гибкость и безопасность типов.
- **Файловая система (std::filesystem):** Обеспечивает стандартные средства для работы с файлами и каталогами.

Преимущества версий C++.



C++20

Рекомендован для: Проектов, которые стремятся быть на переднем крае технологий и использовать новейшие возможности языка.

Причины использования:

- **Концепты:** Улучшают проверку шаблонов на этапе компиляции, делая код более безопасным и читаемым.
- **Диапазоны:** Обеспечивают новый способ работы с коллекциями данных.
- **Корутины:** Упрощают написание асинхронного и ленивого кода.
- **Модули:** Улучшают время компиляции и управление зависимостями.

C++23

Рекомендован для: Проектов, которые могут использовать новейшие версии компиляторов и хотят воспользоваться последними улучшениями языка.

Причины использования:

- **Стековые корутины:** Делают асинхронное программирование еще более эффективным.
- **Улучшенные модули:** Продолжают улучшать управление зависимостями и время компиляции.
- **Параллельные алгоритмы:** Обеспечивают встроенную поддержку параллельных вычислений.



Рекомендуемые версии C++.

- **C++98, C++8** лучше не использовать в современном мире т.к версии уже устарели;
- **C++11 или C++14**, если вам нужна стабильная, хорошо поддерживаемая версия с современными возможностями.
- **C++17**, если вам нужны улучшенные возможности производительности и новые стандартные библиотеки.
- **C++20 или C++23**, если вы хотите использовать новейшие возможности языка, а также если ваш компилятор и инструменты поддерживают эти версии.

Простой код на C++.

Каждая строчка в коде на языке программирования C++ оканчивается символом ";". Это очень важно!!!

```
#include <iostream>
```

#include - подключает файлы в код для использования их содержимого.

```
int main()
```

main() - основная функция, вызываемая при запуске программы. **int** - тип возвращаемого значения.

```
{
```

```
    std::cout << "Hello, world!" << std::endl;
```

cout - вывод текста в командную строку. Оператор "<<" - в данном коде он используется для передачи текста, в двойных кавычках, после которого следует **endl** для перевода строки с очисткой буфера.

```
    return 0;
```

```
}
```



Ввод/Вывод данных (потоки).

Cin это объект входного потока пространства имен `std::namespace`. **Cout** является объектом выходного потока пространства имен `std:: namespace`. (простыми словами, **cin** — это ввод чего-то в консоль, **cout** — это вывод чего-то на консоль)

Примеры (без подключения пространства имен `std: namespace`):

```
std::cout << "Hello, world!";
```

```
int x;  
std::cin >> x;
```

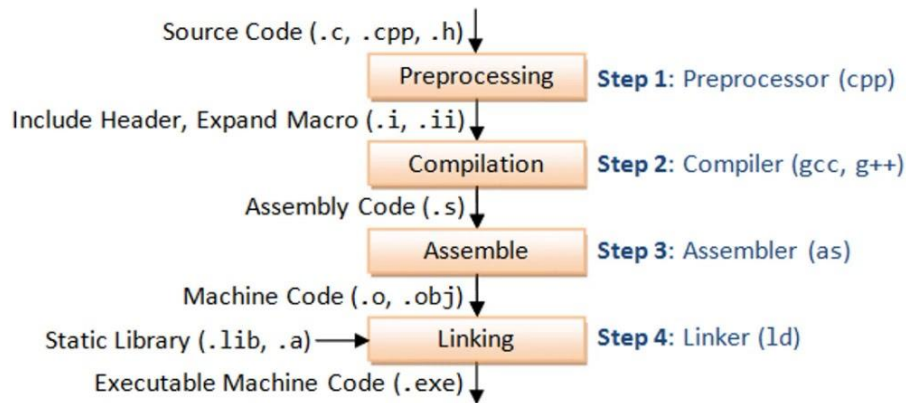


Компиляция программы.

Компиляция программы на C++ - это процесс преобразования исходного кода на языке программирования C++ в исполняемый файл. Процесс компиляции включает несколько этапов:

- *Препроцессинг (Preprocessing)*
- *Компиляция (Compilation)*
- *Сборка (Linking)*
- *Генерация исполняемого файла*

Схема компиляции программы.



1. **Препроцессинг** - обработка кода до компиляции, вставка содержимого и обработка макросов.
2. **Компиляция** - перевод кода на языке C++ в машинный код или язык ассемблера.
3. **Сборка** - объединение отдельных файлов в один исполняемый файл.
4. **Генерация исполняемого файла** - создание файла, который может быть запущен на выполнение.



С++ типы данных.

Целочисленные типы данных: `short int`, `unsigned short int`, `int`, `unsigned int`, `long`, `unsigned long`, `long long`, `unsigned long long`

Типы данных с плавающей точкой: `float`, `double`, `long double`

Символьные типы данных: `char` (`signed char`), `unsigned char`, `wchar_t`

Логический тип данных: `bool`

Числовой тип данных: `enum`

*Типы данных, начинающиеся с префикса `unsigned`, могут содержать только положительные числа.

С++ байтовый размер типов данных.



Тип данных	Размер в байтах
char (<i>signed char</i>), unsigned char, bool	1
short int, unsigned short int, enum, wchar_t	2
int, unsigned int, float	4
double, long long, unsigned long long	8
long double	8-16 (зависит от платформы)

long 4 (обычно), 8 (в зависимости от платформы)

unsigned long 4 (обычно), 8 (в зависимости от платформы)



Чем полезно знать байтовый размер типов данных?

1. **Эффективное использование памяти** - при работе с большими объемами данных важно оптимизировать использование памяти, особенно на ресурс ограниченных системах.
2. **Переносимость кода** - размеры типов данных могут различаться на разных платформах, и знание этих размеров помогает писать переносимый код.
3. **Избежание переполнений и потери данных** - понимание диапазонов значений, которые может хранить тип данных, помогает избежать переполнений и потери данных в процессе выполнения программы.



Переменные.

Переменная, говоря простым языком, - это хранилище данных, значений. В языке C++ переменная объявляется с помощью следующего синтаксиса: `тип_данных` `название_переменной` = `значение_переменной`;

Пример:

```
int x = 1;  
float z;  
double y = 2.42;  
char c;
```

*Присваивать значение переменной не обязательно.

Константная переменная - это переменная с фиксированным значением, которое не может быть изменено каким-либо образом в будущем. (ключевое слово const)

```
const int x = 10; // const variable
```



sizeof() + преобразование типов данных

sizeof() - это унарный оператор, возвращающий длину в байтах переменной или типа, заключенного в скобки.

```
sizeof(int);  
sizeof(double);  
sizeof(char);  
sizeof(float);
```

Чтобы, например, преобразовать **double** в **int**, нужно рядом с переменной типа double в скобках написать (int). Таким образом, вещественное число превратится в целое число (часть после запятой будет отброшена)

```
double val = 5.35241;  
int res = (int)val; // 5
```



Операции над переменными.

```
int x = 10, y = 5, z = 0;  
z = x + y; // adding  
z = x - y; // subtracting  
z = x * y; // multiplication  
z = x / y; // division  
z = x % y; // remainder of division
```

При присвоении значения переменной, которая уже существует, я использую сокращенную форму. Например, так:

```
x = x + 4;  
x += 4; // abbreviated  
x = x - 6;  
x -= 6; // abbreviated  
x = x * 2;  
x *= 2; // abbreviated
```

```
x = x / 2;  
x /= 2; // abbreviated
```



Логические операции.

В языке программирования C/C++ используются следующие логические операции:

- 1) `&&` - логическое **AND**;
- 2) `||` - логическое **OR**;
- 3) `!` - логическое **NO**.

Существуют также побитовые логические операции:

- 1) `&` - побитовое **AND** (**AND**);
- 2) `^` - bitwise addition modulo 2 (**XOR** - exclusive OR);
- 3) `|` - побитовое **OR** (**OR**);
- 4) `~` - побитовое (**NOT**).


```
#include <iostream>
#include <locale>

int main() {
    setlocale(LC_ALL, "Russian");
    int x = 5;
    int y = 10;

    // Логическое AND
    if (x > 0 && y > 0) {
        std::cout << "Оба x и y положительны." << std::endl;
    }

    // Логическое OR
    if (x > 0 || y > 0) {
        std::cout << "Хотя бы одно из x или y положительно." << std::endl;
    }

    // Логическое NOT
    if (!(x > 0)) {
        std::cout << "x не положительно." << std::endl;
    }

    return 0;
}
```

Консоль отладки Microsoft Visual Studio

Оба x и y положительны.
Хотя бы одно из x или y положительно.

```
#include <iostream>
#include <locale>

int main() {
    setlocale(LC_ALL, "Russian");
    int a = 5;    // 0101 в двоичной системе
    int b = 10;   // 1010 в двоичной системе

    // Побитовое AND
    int result_and = a & b; // 0000 в двоичной системе
    std::cout << "Побитовое AND: " << result_and << std::endl;

    // Побитовое XOR
    int result_xor = a ^ b; // 1111 в двоичной системе
    std::cout << "Побитовое XOR: " << result_xor << std::endl;

    // Побитовое OR
    int result_or = a | b; // 1111 в двоичной системе
    std::cout << "Побитовое OR: " << result_or << std::endl;

    // Побитовое NOT
    int result_not = ~a; // 1010 в двоичной системе (побитовое отрицание)
    std::cout << "Побитовое NOT: " << result_not << std::endl;

    return 0;
}
```

Консоль отладки Microsoft Visual Studio

```
Побитовое AND: 0
Побитовое XOR: 15
Побитовое OR: 15
Побитовое NOT: -6
```



Shift операции (или операции смещения).

В языке C/C++ предусмотрены две операции побитового сдвига:

- 1) `<<` - сдвигает значение операнда влево на заданное количество бит.
- 2) `>>` - сдвигает значение операнда вправо на заданное количество бит.

```
// Shift operations  
int a = 15, b = -5, c;  
  
// shift to the left - multiplication  
c = a << 1; // c = a * 2^1 = 30  
c = b << 2; // c = b * 2^2 = -20  
  
// shift to the right - division  
c = a >> 3; // c = a / 2^3 = 1  
c = b >> 1; // c = b / 2^1 = -3
```



Операции инкремента (++), декремента (--).

В языке **C++** существует два оператора, которые увеличивают или уменьшают целочисленное значение на 1:

- 1) оператор **++** - инкремент; (увеличивает значение операнда на 1)
- 2) оператор **--** - декремент. (уменьшает значение операнда на 1)

```
int x = 12, y = 7;

// value changes immediately
++x; // x = 13
--y; // y = 6

// value changes after this line
x++; // x = 14
y--; // y = 5
```



Задачи. Переменные и операции над переменными.

1. Сумма двух целых чисел (num1, num2). Записать сумму чисел в переменную res. Вывести результат на экран.
2. Найти **площадь прямоугольника**. ($S = \text{length} * \text{width}$). (тип данных double). Вывести результат на экран.
3. SWAP задача - обмен значениями между переменными. Объявить две переменные (a, b целого типа данных) и сделать операцию swap. Вывести результат на экран.
4. Запросить у пользователя число (вещественного типа данных используя 4 байта). Преобразовать к целому типу данных. И вывести результат на экран.
5. Запросить у пользователя число (типа данных int). Вывести тип данных переменной на экран.
6. Запросить два числа (целого типа данных). Вычислить остаток от деления первого числа на второе число. Вывести результат на экран.



Задачи. Переменные и операции над переменными.

1. **Калькулятор BMI** - Создайте программу для вычисления индекса массы тела (BMI). Запросите у пользователя **вес** и **рост**.

$$BMI = \frac{weight(kg)}{height^2(m^2)}$$

2. **Перевод температуры** - Запросите у пользователя температуру в градусах Цельсия и выведите ее в градусах Фаренгейта.

$$T_f = \frac{9}{5}T_c + 32$$



ASCII - CHAR.

ASCII (American Standard Code for Information Interchange) — это стандартный код для представления текстовых символов в компьютерах и других устройствах. Каждый символ ASCII представлен числом от 0 до 127. Этот стандарт используется для кодирования букв, цифр, знаков препинания и управляющих символов.

В C++ символы представляются типом `char`, и каждый символ соответствует числу в таблице ASCII. Например, символ 'a' имеет числовое представление 97, а символ 'A' — 65.

```
{ a=97, b=98,...}  
{ A=65, b=66, c=67, ....}
```



Условная конструкция (if-else).

Условный оператор **if** позволяет выбрать путь выполнения программы. Выбор осуществляется по некоторому условию. Если условие выполняется, то программа выполняется одним путем. В противном случае (**else**) программа выполняется по другому пути. Таким образом, в программе создается ветвление.

Конструкция:

```
if ( condition )  
{  
    // body if  
}  
else  
{  
    // body else  
}
```




Пример. Условная конструкция.

```
#include <iostream>

int main() {
    // Вводим число
    int number;
    std::cout << "Введите число: ";
    std::cin >> number;

    // Проверка условия
    if (number > 0) {
        std::cout << "Число положительное." << std::endl;
    }
    else if (number < 0) {
        std::cout << "Число отрицательное." << std::endl;
    }
    else {
        std::cout << "Число равно нулю." << std::endl;
    }

    return 0;
}
```

Задачи. Условная конструкция.

1. **Четность числа** - Попросите пользователя ввести число и определите, является ли оно четным или нечетным.

Четное число - это число, которое делится на 2 без остатка.

2. **Возрастная категория** - Спросите у пользователя его возраст и определите, к какой возрастной категории он принадлежит.

Периодизация возрастов человека		
Возрастной период	Продолжительность возрастного периода	
	Мужской пол	Женский пол
Новорожденный	1—10 дней	1—10 дней
Грудной	10 дней — 1 год	10 дней — 1 год
Раннее детство	1—2 года	1—2 года
Первый период детства	3—7 лет	3—7 лет
Второй период детства	8—12 лет	8—11 лет
Подростковый	13—16 лет	12—15 лет
Юношеский	17—21 год	16—20 лет
Средний (взрослый):		
	22—35 лет	21—35 лет
первый период	36—60 лет	36—55 лет
второй *	61—75 лет	56—75 лет
Пожилый	76—90 лет	76—90 лет
Старческий	76—90 лет	76—90 лет
Долгожители	Старше 90 лет	Старше 90 лет



Задачи. Условная конструкция.

4. **Максимальное из двух чисел** - Запросите два числа у пользователя и определите, какое из них больше.
5. **Проверка на делимость на 3 и 5** - Введите число и определите, делится ли оно и на 3, и на 5, и выведите соответствующее сообщение. В противном случае скажите, что число не делится на 3&5.
6. **Проверка на кратность 6** - Попросите пользователя ввести число и определите, является ли оно кратным 6.

Например:

число 36 - кратно 6 (т.к $36 / 6 = 6$ остаток 0)

число 31 - не кратно 6 (т.к $31 / 6 = 5$ остаток 1)

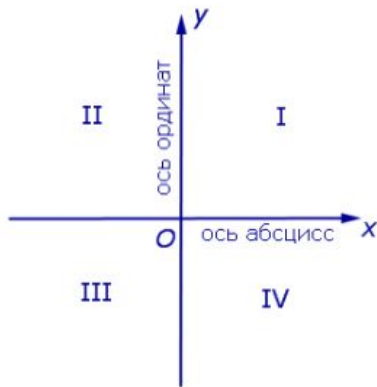
Задачи. Условная конструкция.

7. **Определение квартала года** - Введите номер месяца и определите, к какому кварталу года относится этот месяц.



Задачи. Условная конструкция.

8. **Определение четверти графика** - Запросите у пользователя координаты точки (x, y) и определите, в какой четверти находится эта точка.





Задачи. Условная конструкция.

9. **Калькулятор скидки** - Введите общую стоимость товара и определите сумму скидки в зависимости от суммы покупки:
 - a. 10% скидка, если сумма больше 1000 рублей,
 - b. 5% скидка, если сумма больше 500 рублей,
 - c. 2% скидка в остальных случаях.
10. **Проверка на положительное, отрицательное или ноль** - Запросите у пользователя число и если оно > 0 вывести надпись "положительное число", если число < 0 то вывести "отрицательное число", в противном случае (иначе) это ноль.



Thank you for your attention!