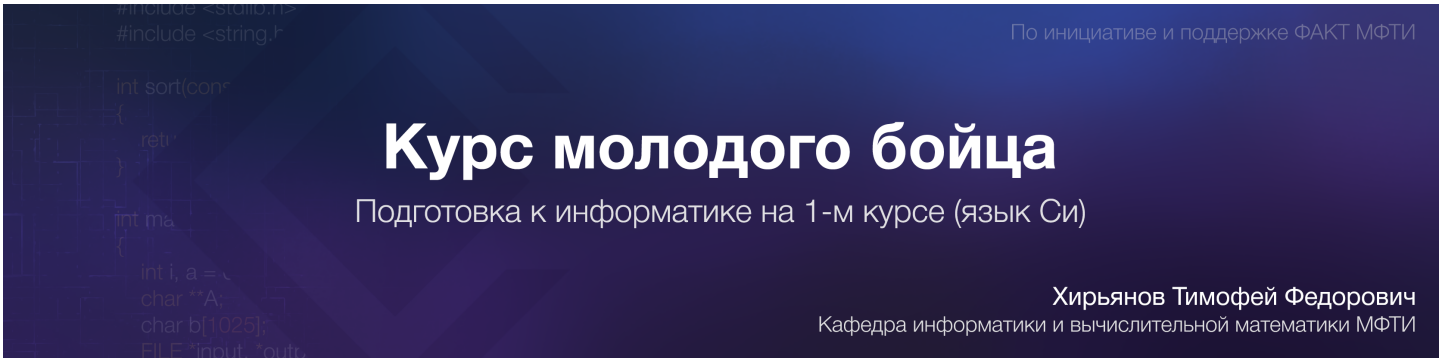


Главная

Основы языка Си

Канал общения с другими участниками курса

Если вы пользуетесь соцсетью "Вконтакте", вступайте в сообщество "Курса молодого бойца" для обсуждений и взаимопомощи.

Установка среды разработки Code::Blocks

Обучение предполагает не только просмотр видео, но и самостоятельный запуск, а также модификацию предлагаемых программ. Для компиляции исходных текстов на Си рекомендуется установить среду Code::Blocks с компилятором MinGW.

К каждому уроку предполагается домашнее задание, однако по техническим причинам его публикация задерживается до 15 августа. Ссылка на него будет находится на главной странице курса справа от содержания урока.

Разбор "Hello, World!" на Си

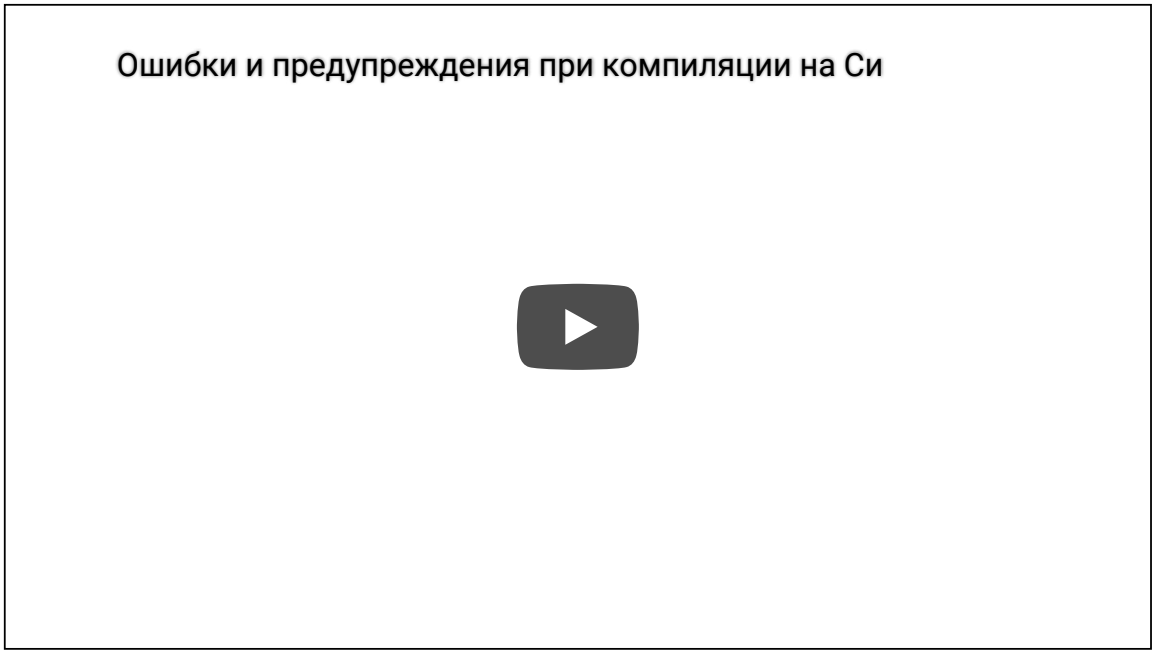
Приветствие целевой аудитории курса. Разбор hello_world.c построчно.

hello_world.c

```
#include <stdio.h>
```

```c
int main(int argc, char* argv[])
{
    printf("Hello, World!\n");
    return 0;
}
```

Ошибки и предупреждения при компиляции

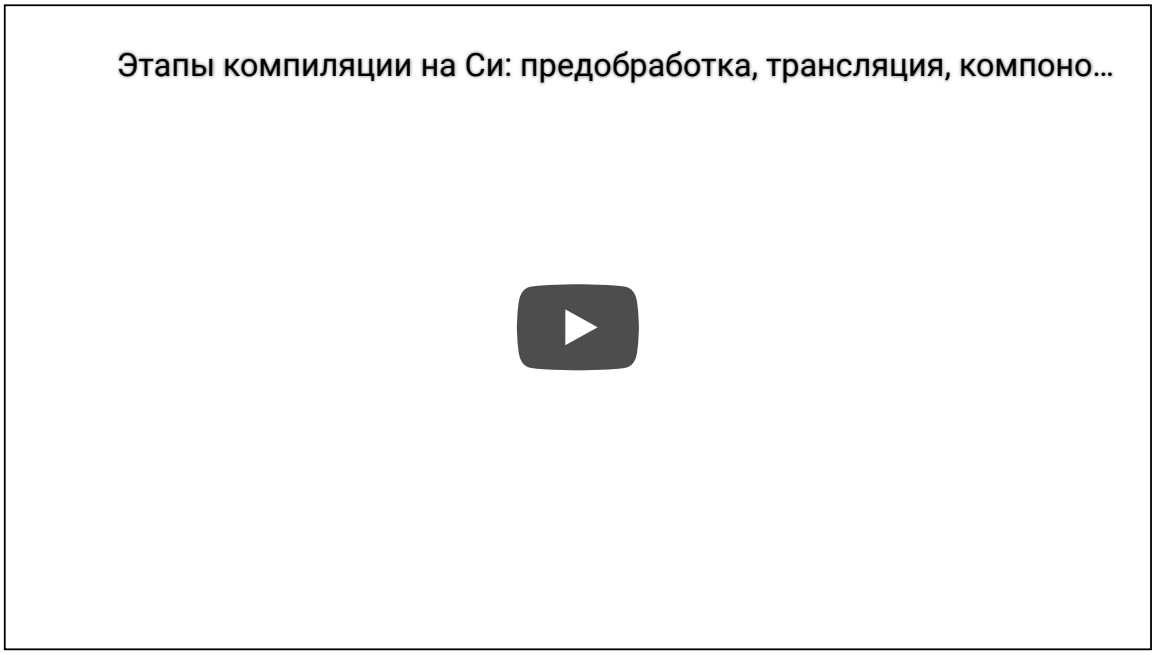


Что такое компиляция. Любите ошибки компиляции! Примеры сообщений об ошибках. Предупреждения при компиляции. Опция -Wall очень полезна!

warnings_errors.c

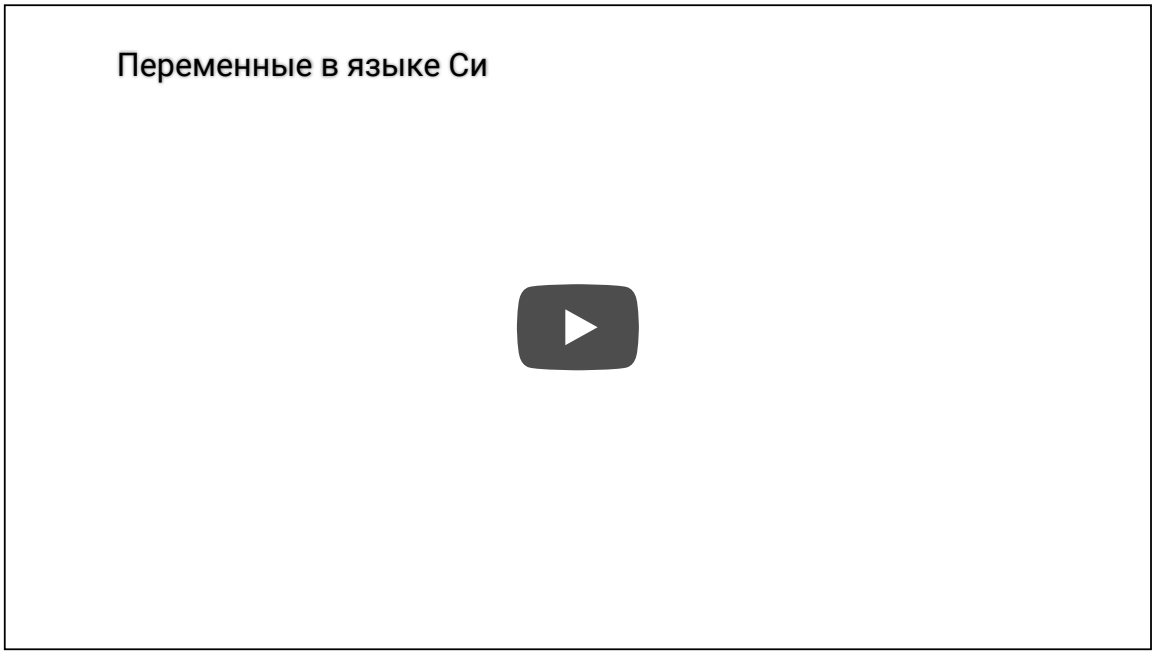
```c
#include <stdio.h>

int main()
{
    printf("I'm a Bot. What's your name?\n");
    char name;
    scanf("%s", name);
    printf("Hello, %s! How old are you?\n");
    int age;
    scanf("%d", age);
    printf("You are looking younger! "
           "I thought you are %d!\n", age-3);
    return 0;
}
```

Этапы компиляции на Си: предобработка, трансляция, компоновка



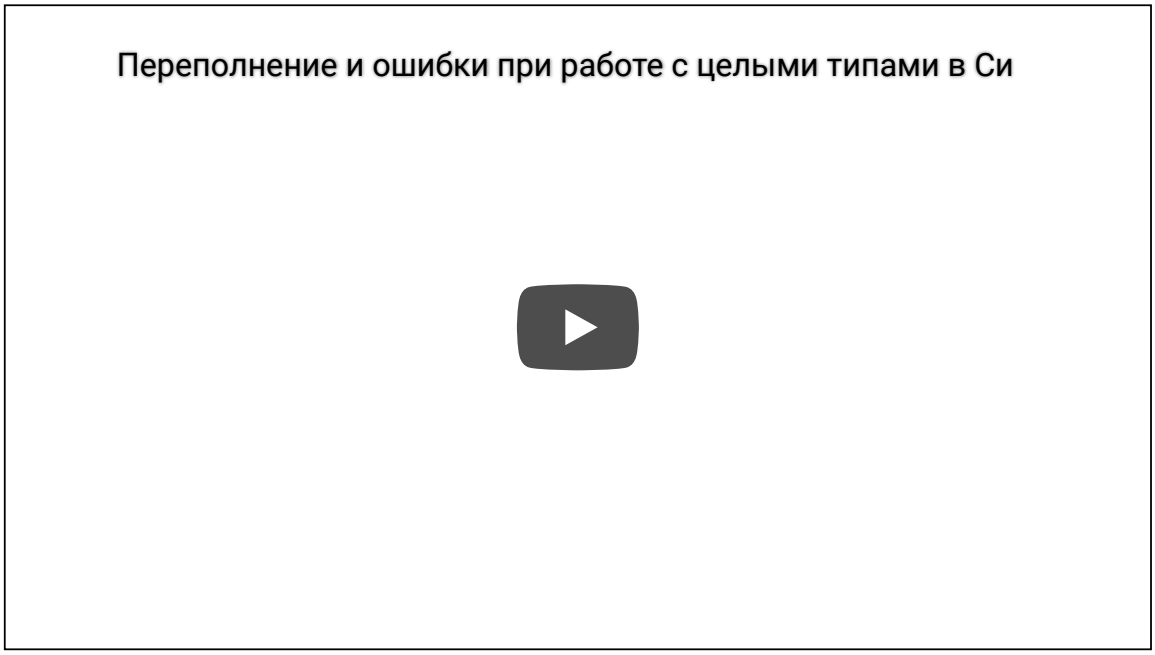
Этапы компиляции: предобработка, трансляция, компоновка. Компилятор GCC. Компиляция в консоли.

Переменные в языке Си



Переменная – это... Что определяет тип. Типизированные значения, не являющиеся переменными. Концепция присваивания в Си. Какими типами пользоваться.

Переполнение и ошибки при работе с целыми типами в Си



Целочисленное переполнение. Декларация переменных целых типов. Демонстрация переполнения. Неявное приведение типов: проблема с unsigned и signed. Явное приведение типов.

declarations.c

```c
#include <stdio.h>
#include <stdbool.h>
#include <inttypes.h>

int main(int argc, char* argv[])
{
    char c;
    int i;
    double d;
    bool b;

    int64_t i64;

    return 0;
}
```

overflows.c

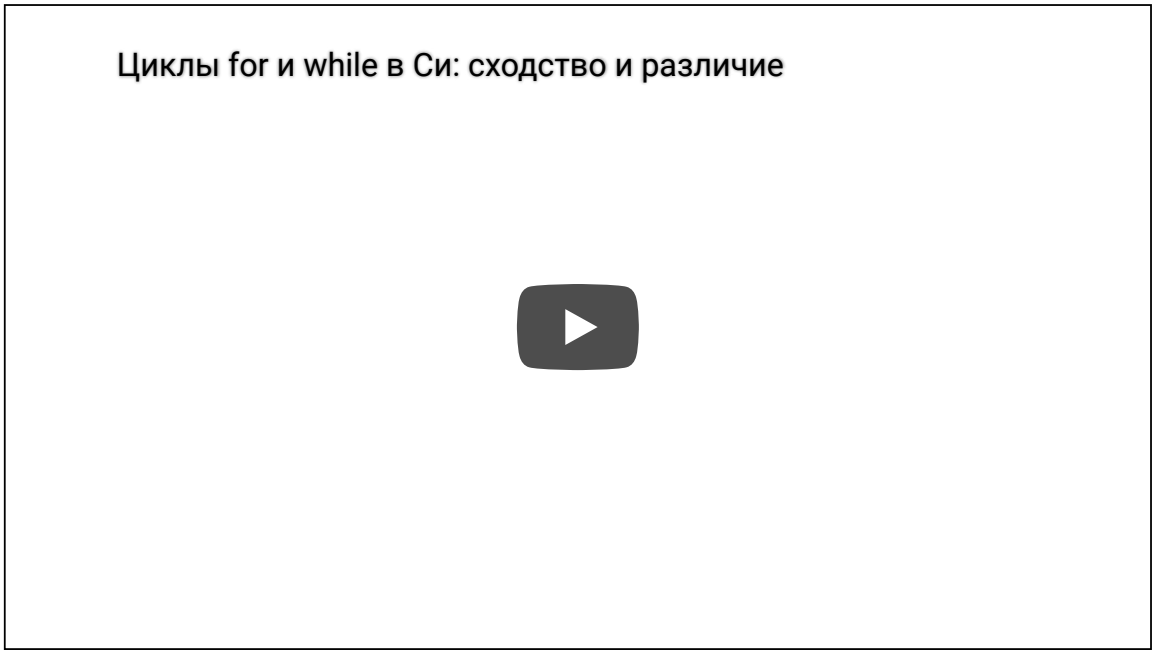
```c
#include <stdio.h>
#include <inttypes.h>

int main(int argc, char* argv[])
{
    int8_t x = 127;
    printf("x = %d\n", x);
    x += 1;
    printf("x = %d\n", x);
    x -= 1;
    printf("x = %d\n", x);

    return 0;
}
```

typecasting.c

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
```

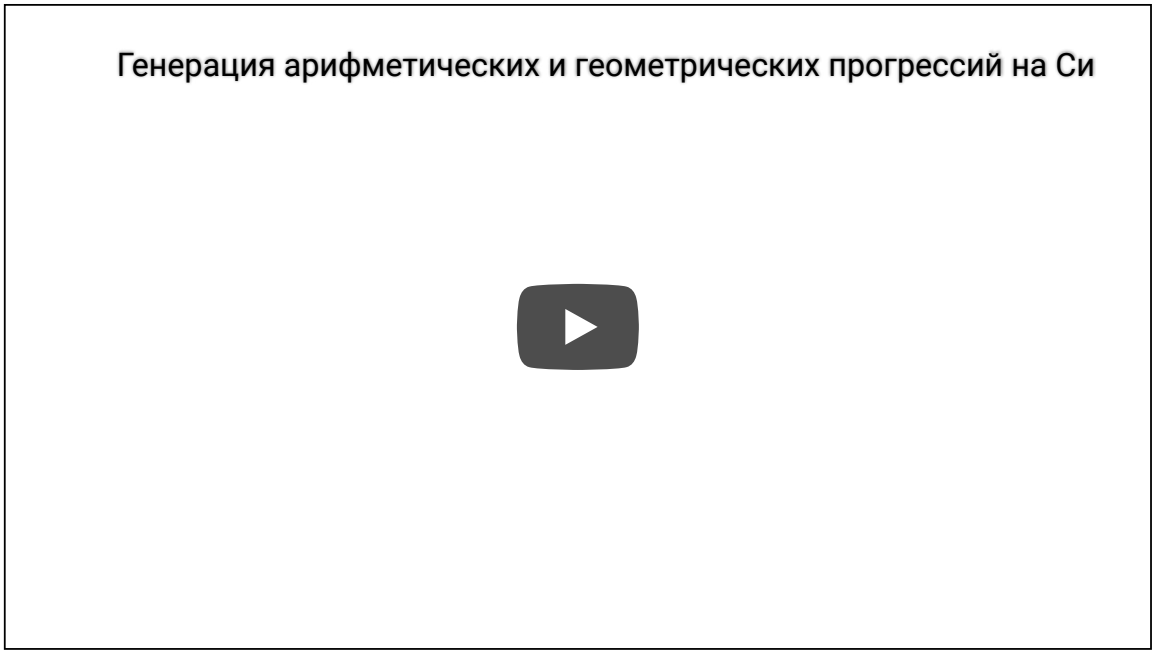
```
    // int is implicitly cast to unsigned int
    int x = -100;
    unsigned int y = 10U;
    long long int z = x + y;
    printf("z = %lld\n", z);

    // char is explicitly cast to int
    char c = 'Я';
    int d = (int)c * 10;
    printf("d = %d\n", d);

    return 0;
}
```

Циклы for и while: сходство и различие



Итерация, тело цикла и заголовок. Синтаксис цикла while. Значение переменной после цикла. Заголовок цикла for и сравнение с циклом while. Опасность зацикливания.

Генерация арифметических и геометрических прогрессий

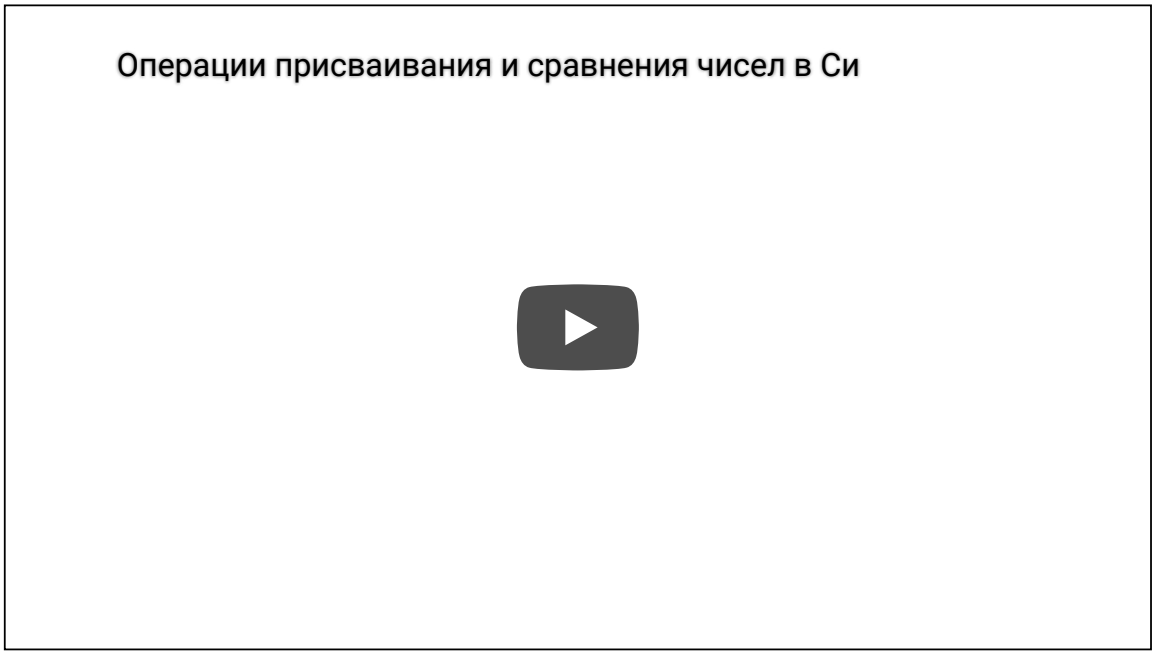


Цикл, генерирующий прогрессиию. Обёртка для диалога с пользователем. Особенности программ, сдаваемых в систему Ejudge. Отладка программы для отрицательного шага.

generation.c

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    int start, stop, step;
    printf("Generator of progression.\n"
           "Enter start, stop, step:");
    scanf("%d%d%d", &start, &stop, &step);

    int sign = (step > 0)? +1: -1;
    int x = start;
    while (sign*x < sign*stop)
    {
        printf("x = %d\n", x);
        x += step;
    }

    printf("After: x = %d\n", x);

    return 0;
}
```

Операции присваивания и сравнения чисел в Си



Операция присваивания – арифметическая операция. Виды операций присваивания. Операция сравнения – арифметическая операция. Результат сравнения – целое число 0 или 1.

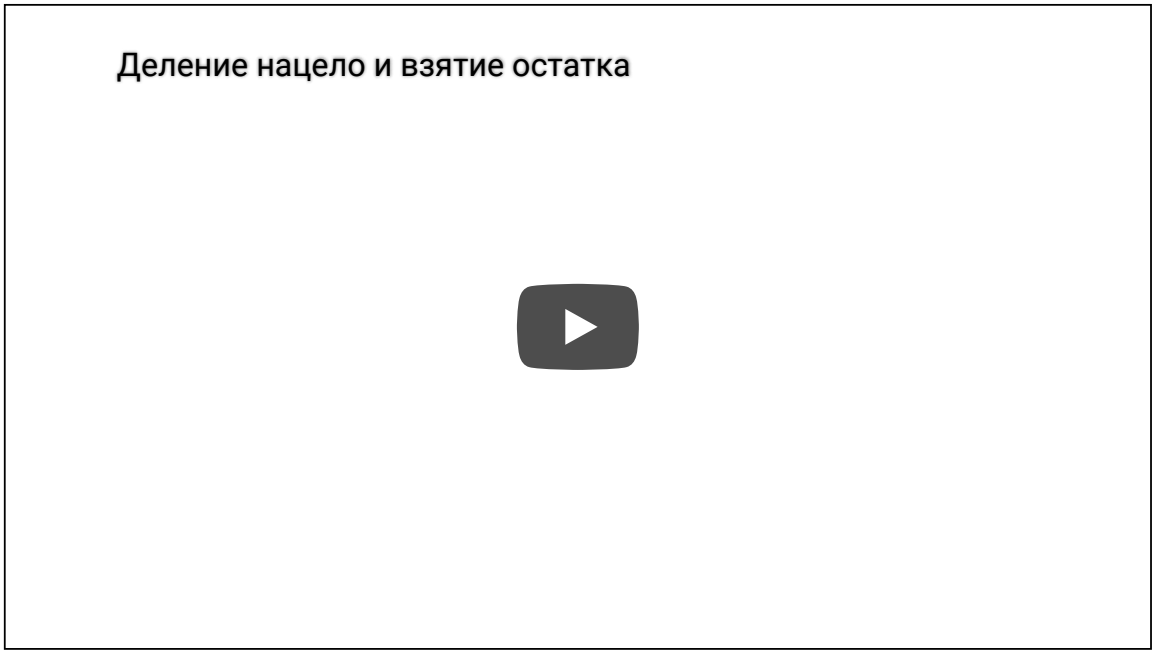
assignment.c

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    int x = 256;

    x += 256; // 512
    x -= 256; // 256
    x *= 2;   // 512
    x /= 8;   // 64
    printf("x = %d\n", x);

    return 0;
}
```

comparisons.c

```c
#include <stdio.h>
#include <stdbool.h>

int main(int argc, char* argv[])
{
    int x;

    x = (5 > -1) + (3 <= 4) +
        (2*2 == 2+2) + (2*3 != 2+3);

    printf("x = %d\n", x);

    return 0;
}
```

Деление нацело и взятие остатка



Деление нацело и взятие остатка. Разложение числа на цифры. Осторожно: отрицательные остатки при делении отрицательного на положительное! Как сделать дробное деление для целых чисел.

division.c

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    int x = -12345;
    while (x != 0)
    {
        printf("%d %d\n", x/10, x%10);
        x /= 10;
    }

    return 0;
}
```

Самостоятельная работа

ОТКРЫТА РЕГИСТРАЦИЯ НА КОНТЕСТ

Дорогой учащийся! Вы – молодец, если вы просмотрели видеоролики.

Но урок ещё не закончился! Вы получите практические навыки при выполнении заданий учебного соревнования в системе автоматической проверки Ejudge или, говоря студенческим жаргоном, *контеста* (англ. contest).

Создание пользователя на сервере Ejudge происходит при регистрации на 1-й на контест. Желаю вам успешно выполнить задачи домашнего задания!