

[Главная](#)

# Циклы и ветвления

## Оператор if и организация ветвления

Оператор if и организация ветвления в Си



Оператор ветвления if в Си. Полная и неполная формы. Когда нужно и не нужно ставить фигурные скобки. Пример применения if для проверки числа на чётность. Вложенные условные инструкции. Каскадное ветвление else-if.

if\_else.c

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    int x = 1;
    scanf("%d", &x);
    while (x != 0)
    {
        if (x%2 == 0)
            printf("Number %d is even.\n", x);
        else
            printf("Number %d is odd.\n", x);
        scanf("%d", &x);
    }

    return 0;
}
```

## nested\_if.c

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    int x, y;
    scanf("%d%d", &x, &y);

    if (y > 0)
        if (x > 0)
            printf("1-st quarter.\n");
        else
            printf("2-nd quarter.\n");
    else
        if (x < 0)
            printf("3-rd quarter.\n");
        else
            printf("4-th quarter.\n");

    return 0;
}
```

## cascade\_elif.c

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    int x, y;
    scanf("%d%d", &x, &y);

    if (y > 0 and x > 0)
        printf("1-st quarter.\n");
    else if (y > 0 and x < 0)
        printf("2-nd quarter.\n");
    else if (y < 0 and x < 0)
        printf("3-rd quarter.\n");
    else if (y < 0 and x > 0)
        printf("4-th quarter.\n");
    else
        printf("Point is on axis.\n");

    return 0;
}
```

## Метки, оператор goto и оператор switch

## Метки, оператор goto и оператор switch в Си



Оператор goto и прыжки на метки. Аргументы против goto. Блок-схема, которую нельзя сделать на циклах. Ветвление switch и почему нужен break.

### never\_goto.c

```
#include <stdio.h>

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

A:  scanf("%d", &x);
    if (x == 0) goto D;
B:  if (x%2 != 0) goto C;
    printf("Number %d is even.\n", x);
    goto A;
C:  printf("Number %d is odd.\n", x);
    goto A;
D:
    return 0;
}
```

### switch\_case.c

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    int x;
    scanf("%d", &x);

    switch (x)
    {
        case 1: printf("One!\n"); break;
        case 2: printf("Two!\n"); break;
        case 3: printf("Three!\n"); break;
        case 0: printf("Zero!\n"); break;
        default: printf("Don't know this number...");
    };

    return 0;
}
```

## Управление циклом: break, continue

## Управление циклом в Си: break, continue



Операторы управления циклом: break и continue. Адекватное использование break. Адекватное использование continue. Тест простоты числа с использованием break и переменной-флага. Тест простоты числа с non-tail return.

### break\_usage.c

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    while (1)
    {
        int x;
        scanf("%d", &x);
        if (x == 0) break;
        printf("Number %d in hexadecimal is %X.\n", x, x);
    }

    return 0;
}
```

### continue\_usage.c

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    for(int i = 1; i < 20; ++i)
    {
        if (i != 13) continue; // Avoid number 13.
        if (i%7 == 0) continue; // Skip multiples of 7.
        printf("Number i = %d. ", i);
        if (i%3 == 0)
            printf("It is a multiple of 3!\n", i);
        else
            printf("It's not a multiple of 3...\n", i);
    }
    return 0;
}
```

### prime\_break.c

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

```
int main(int argc, char* argv[])
{
    int x;
    printf("Enter number to check for simplicity:");
    scanf("%d", &x);

    bool is_prime = true;
    for(int divisor = 2; divisor*divisor <= x; ++divisor)
        if (x%divisor == 0)
        {
            is_prime = false;
            break;
        }

    if (is_prime)
        printf("Number %d is prime!\n", x);
    else
        printf("Number %d is not prime...\n", x);

    return 0;
}
```

### prime\_function.c

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

bool is_prime_number(int x)
{
    for(int divisor = 2;
        divisor*divisor <= x; ++divisor)
        if (x%divisor == 0)
            return false;
    return true;
}

int main(int argc, char* argv[])
{
    int x;
    printf("Enter number to check for simplicity:");
    scanf("%d", &x);

    if (is_prime_number(x))
        printf("Number %d is prime!\n", x);
    else
        printf("Number %d is not prime...\n", x);

    return 0;
}
```

## Разложение числа на множители

## Разложение числа на множители на Си



Постановка задачи. Пример с использованием вложенного цикла. Пример без использования вложенного цикла.

### factorization.c

```
#include <stdio.h>

void print_number_factors(int x)
{
    printf("Number factors: ");
    int divisor = 2;
    while (x != 1)
    {
        while (x%divisor == 0)
        {
            printf("%d ", divisor);
            x /= divisor;
        }
        divisor += 1;
    }
    printf("\n");
}

int main(int argc, char* argv[])
{
    int x;
    printf("Enter number to factorize:");
    scanf("%d", &x);
    print_number_factors(x);

    return 0;
}
```

### Фильтрация потока чисел

## Фильтрация потока чисел на Си



Типы работы с последовательностями чисел. Фильтрация последовательности.

### filtration.c

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    int x;
    printf("Number to split on digits:");
    scanf("%d", &x);
    while (x)
    {
        int digit = x%10;
        if (digit < 5)
            printf("%d ", digit);
        x /= 10;
    }

    return 0;
}
```

## Индуктивные функции: подсчёт, сумма, произведение

### Индуктивные функции на Си: подсчёт, сумма, произведение



Индуктивные функции. Подсчёт цифр числа. Сумма цифр числа. Произведение цифр числа.

### inductive\_count\_production\_sum.c

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    int x;
    scanf("%d", &x);

    int n = 0, s = 0, p = 1;
    while (x)
    {
        int digit = x%10;
        n += 1;
        s += digit;
        p *= digit;
        x /= 10;
    }
    printf("number = %d\n"
           "sum = %d\n"
           "production = %d\n", n, s, p);

    return 0;
}
```

### Индуктивные функции: any of, all of

Индуктивные функции на Си: any of, all of



Проверка соответствия критерию. Алгоритмы any of и all of. Логические вычисления или работа с флажками?

### inductive\_any\_all.c

```
#include <stdio.h>
#include <stdbool.h>
#include <iso646.h>

int main(int argc, char* argv[])
{
    int number;
    scanf("%d", &number);

    bool any = false;
    bool all = true;
    while (number)
```



```

{
    int digit = number%10;
    any = any or (digit < 5);
    all = all and (digit < 5);
    number /= 10;
}
printf("any = %d\n"
       "all = %d\n", any, all);

return 0;
}

```

## Индуктивные функции: поиск максимума

### Индуктивные функции на Си: поиск максимума



Поиск максимума как индуктивная функция. Проблема инициализации временного максимума. Поиск местоположения максимума. Подсчёт элементов, равных максимальному.

### inductive\_maximum.c

В видео при поиске местоположения максимального элемента допущена ошибка – отсутствует инициализация `i = 0`; . В примере ниже она исправлена:

```

#include <stdio.h>

int main(int argc, char* argv[])
{
    int x, i = 0;
    int m = -1000000, m_i = -1;
    scanf("%d", &x);
    while (x != 0) // Zero - terminator of succession.
    {
        if (x%2 == 0)
            if (x > m)
            {
                m = x;
                m_i = i;
            }
        i += 1;
        scanf("%d", &x);
    }
    printf("maximum = %d\n"
           "maximum position= %d\n", m, m_i);

    return 0;
}

```

Можно заметить, что при отсутствии чётных чисел в потоке мы будем наблюдать при выводе результата интерпретацию "мусора" в неинициализированной переменной `m_i`. Можно её, конечно, инициализировать, например, `m_i = -1;`, но это не помогает исправить логику программы, поскольку мы будем выводить то, что математически не определено (максимум среди пустой подпоследовательности). Для отслеживания этого нужно перед выводом результата по значению `m` или `m_i` проверить, а были ли вообще в последовательности чётные числа.

## Самостоятельная работа

Уважаемые студенты!

Ко 2-му уроку есть домашняя работа в форме конкурса: [ссылка на ДЗ №2](#). Ссылка на неё также находится на главной странице сайта.

Если у вас нет логина и пароля, [зарегистрируйтесь на 1-й конкурс](#), и доступ ко 2-му вы получите автоматически.

Сайт построен с использованием [Pelican](#). За основу оформления взята тема от [Smashing Magazine](#). Исходные тексты программ, приведённые на этом сайте, распространяются под лицензией [GPLv3](#), все остальные материалы сайта распространяются под лицензией [CC-BY](#).