Технология программирования на ЭВМ
Цикл while

Баев А.Ж.

Казахстанский филиал МГУ

02 октября 2018

Пример. Степень двойки.

Дано целое положительное число n от 1 до 15. Найти минимальную степень двойки, которая больше n.

Ввод	5	3	8
Вывод	8	4	16

Решение if

Будем перебирать все степени двойки 2, 4 и 8. Если степень всё еще меньше, то её можно еще увеличить.

```
int p = 2, n;
scanf("%d", &n);
if (p <= n) {
    p *= 2;
}
if (p <= n) {
    p *= 2;
}
if (p <= n) {
    p *= 2;
}
printf("%d", n);
```

Решение while

Будем перебирать все степени двойки 2, 4 и 8. Если степень всё еще меньше, то её можно еще увеличить.

```
int p = 2, n;
scanf("%d", &n);
while (p <= n) {
    p *= 2;
}
printf("%d", n);
```

Скобки — не обязательны.

Общий вид

Тело цикла body выполняется если условие condition верно.

```
while (condition) {
    body;
}
```

Пример. Количество нулей.

Дано целое положительно число n от 1 до 10^6. Необходимо посчитать, на сколько нулей оканчивается число.

Ввод	123000	1000000	2001
Вывод	3	6	0

Решение

```
int ans = 0, n;
scanf("%d", &n);

while (n % 10 == 0) {
    n /= 10;
    ans++;
}
```

Скобки — обязательны.

Что будет в неправильном решении?

```
int ans = 0, n;
scanf("%d", &n);

while (n % 10 == 0)
    n /= 10;
    ans++;
```

Зацикливание

Цикл не останавливается.

```
int ans = 0, n = 100;

while (n % 10 == 0)
    ans++;
    n /= 10;
```

В таких случае рекомендуется снять процесс комбинацией: `Ctrl + C`.

Эхо.

Программа выводит введенные числа на экран до тех пор, пока не встретится нуль.

Ввод	10 20 0
Вывод	10 20

Решение без вывода нуля.

```
int a;
scanf("%d", &a);
while(a != 0) {
    printf("%d", a);
    scanf("%d", &a);
}
```

Решение с выводом нуля.

```
int a;
do {
    scanf("%d", &a);
    printf("%d", a);
} while(a != 0);
```

Цикл с постусловием do while.

Тело цикла `body` выполняется после чего, проверяется условие `condition`. Если оно верно, то повторяем действия.

```
do {
    body;
} while (condition);
```

Переменная–счетчик.

Вывести текст «HELLO!» 5 раз.

```
int i = 0;
while (i < 5) {
    puts("HELLO!");
    ++i;
}
```

Стоит отметить, что условие $(i < 5)$ будет проверено 6 раз и после завершения цикла, значение переменной i будет равно 5.

Переменная–счетчик.

Дано целое положительное n, вывести все числа меньше n.

```
int i = 0, n;
scanf("%d", &n);
while (i < n) {
    printf("%d␣", i);
    i++;
}
```

Максимальный квадрат

Дано целое n от от 1 до 10000. Найти максимальный квадрат меньший n (то есть $x^2 < n$).

Ввод	30	100
Вывод	25	81

Решение

```c
#include <stdio.h>

int main() {
    int n, m = 1, ans;
    scanf("%d", &n);
    while (m * m < n) {
        m++;
    }
    m--;
    ans = m * m;
    printf("%d\n", ans);
    return 0;
}
```

Число цифр

Дано целое число от 1 до 10^{18}. Посчитать количество десятичных цифр.

Ввод	2017	12345678987654321
Вывод	4	17

Решение

```
#include <stdio.h>

int main() {
    int ans = 0;
    long long n;
    scanf("%lld", &n);

    while (n != 0) {
        n /= 10;
        ans++;
    }
    printf("%d\n", ans);
    return 0;
}
```

Обратный порядок цифр

Дано целое число от 1 до 10^{18}. Вывести цифры числа в обратном порядке.

Ввод	2017	1000000
Вывод	7102	0000001

Решение

```c
#include <stdio.h>

int main() {
    int digit;
    long long n;
    scanf("%lld", &n);

    while (n != 0) {
        digit = n % 10;
        printf("%d", digit);
        n /= 10;
    }
    printf("\n");
    return 0;
}
```

Делители числа

Дано целое n от от 1 до 30000. Вывести все делители числа n через пробел.

Ввод	25	12
Вывод	1 5 25	1 2 3 4 6 12

Решение

```
#include <stdio.h>

int main() {
    int d = 1, n;
    scanf("%d", &n);
    while (d <= n) {
        if (n % d == 0) {
            printf("%d␣", d);
        }
        d++;
    }
    puts("");
    return 0;
}
```

Простое ли число

Дано целое n от от 1 до 30000. Проверить, является ли данное число — простым (вывести `prime` или `not prime` соответственно).

Ввод	19	91
Вывод	`prime`	`not prime`

Решение

```
#include <stdio.h>

int main() {
    int d = 2, tau = 0, n;
    scanf("%d", &n);
    while (d < n) {
        if (n % d == 0) {
            tau++;
        }
    }
    if (tau == 0) {
        printf("prime\n");
    } else {
        printf("not␣prime\n");
    }
    return 0;
}
```

Решение 2

```
#include <stdio.h>

int main() {
    int d = 2, tau = 0, n;
    scanf("%d", &n);
    while (d * d <= n) {
        if (n % d == 0) {
            tau++;
        }
    }
    if (tau == 0) {
        printf("prime\n");
    } else {
        printf("not␣prime\n");
    }
    return 0;
}
```

Последовательная обработка чисел

Дана последовательность целых чисел от -1000 до 1000, причем ввод заканчивается нулем. Найти сумму чисел (гарантируется, что ответ по модулю не превосходит 10^9).

Ввод	1 2 -3 4 5 0	2 0
Вывод	9	2

Решение

```c
#include <stdio.h>

int main() {
    int a, sum = 0;
    scanf("%d", &a);
    while (a != 0) {
        sum += a;
        scanf("%d", &a);
    }
    printf("%d\n", sum);
    return 0;
}
```

Минимум

Дана последовательность целых чисел от -1000 до 1000, причем ввод заканчивается нулем. Найти минимум чисел.

Ввод	2 3 1 4 5 0
Вывод	1

Решение

```c
#include <stdio.h>

int main() {
    int a, min;
    scanf("%d", &a);
    min = a;
    while (a != 0) {
        scanf("%d", &a);
        if (min > a) {
            min = a;
        }
    }
    printf("%d\n", min);
    return 0;
}
```

Позиция минимума

Дано целое n от 1 до 1000. Далее n различных целых чисел от -1000 до 1000. Найти минимум из чисел и его позицию.

Ввод	5 2 3 1 4 5
Вывод	1 3

Решение

```
#include <stdio.h>

int main() {
    int n, a, min, imin = 1, i = 1;
    scanf("%d␣%d", &n, &min);
    while (i < n) {
        i++;
        scanf("%d", &a);
        if (a < min) {
            min = a;
            imin = i;
        }
    }
    printf("%d␣%d\n", min, imin);
    return 0;
}
```

Алгоритм Евклида

Даны два целых числа от 1 до 10^9. Найти наибольший общий делитель этих чисел.

Ввод	40 12	20 17
Вывод	4	1

Решение

$$(40, 12) = (12, 4) = (4, 0)$$

$$(a, b) = (b, a \bmod b)$$

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

int main() {
    int a, b, d;
    scanf("%d␣%d", &a, &b);
    while (b != 0) {
        d = a % b;
        a = b;
        b = d;
    }
    printf("%d\n", a);
    return 0;
}
```

Отладочная печать (простой метод)

Помогут дополнительные puts и printf.

```
int ans = 0, n = 100;
puts("1 step");
while (n % 10 == 0)
    ans++;
    n /= 10;
puts("2 step");
```

Обратите внимание, что printf() не гарантирует вывод на экран при зацикливании, если в вывод нет '\n', например, так

```
while (n % 10 == 0) {
    printf("%d\n", n);
    ans++;
    n /= 10;
}
```

Отладка в gdb (сложный метод)

Компилируем

```
gcc prog.c -o prog -Wall -Werror -lm -g
```

Запускаем отладчик

```
gdb prog
```

Посмотрим код

```
list 1
```

Ставим точку останова (до которой программа будет выполняться в обычном режиме). Лучше ставить сразу после ввода.

```
break 6
```

Запускаем

```
run
```

Добавляем переменную наблюдения (можно несколько переменных)

```
display n
display ans
```

Делаем построчное выполнение (первый раз надо набрать команду целиком, потом просто Enter).

```
next
```

Выход

```
quit
```