

Технология программирования на ЭВМ

Цикл for

Баев А.Ж.

Казахстанский филиал МГУ

12 ноября 2019

Числа от 10 до 20

```
1 int i;  
2 i = 10;  
3 while (i <= 20) {  
4     printf("%d□", i);  
5     i++;  
6 }
```

Числа от 10 до 20

```
1 int i;
2
3 for (i = 10; i <= 20; i++) {
4     printf("%d ", i);
5 }
```

```
инициализация      i = 10
условие продолжения: i <= 20
тело цикла:         printf("%d ", i)
итерирование:       i++
```

Общий вид.

цикл while:

```
1  init;  
2  while (condition) {  
3      body;  
4      iteration;  
5  }
```

цикл for:

```
1  for (init; condition; iteration) {  
2      body;  
3  }
```

Нечетные числа от 1 до n

Задача: распечатать все нечетные числа от 1 до n .

Ввод

16

Вывод

1 3 5 7 9 11 13 15

Нечетные числа от 1 до n

```
int i, n;
scanf("%d", &n);
for (i = 1; i <= n; i += 2) {
    printf("%d ", i);
}
```

Буквы английского алфавита

Задача: распечатать все буквы из данного диапазона.

Ввод

kr

Вывод

k l m n o r

Буквы английского алфавита

```
char ch, first, last;
first = getchar();
last = getchar();
for (ch = first; ch <= last; ch++) {
    putchar(ch);
}
```

Квадраты чисел

Задача: распечатать все числа, квадраты которых не превосходят n

Ввод

90

Вывод

1 2 3 4 5 6 7 8 9

Квадраты чисел

```
int i, n;  
scanf("%d", &n);  
for (i = 1; i * i <= n; i++) {  
    printf("%d□", i);  
}
```

Числа от n до 1

Задача: Вывести все числа от n до 1.

Ввод

5

Вывод

5 4 3 2 1

Квадраты чисел

```
int i, n;  
scanf("%d", &n);  
for (i = n; i > 0; i--) {  
    printf("%d□", i);  
}
```

Делители числа n

Задача: Вывести количество делителей числа n .

Ввод

6

Вывод

4

Делители числа n

```
1 int n, d, ans = 0;
2 scanf("%d", &n);
3 for (d = 1; d <= n; d++) {
4     if (n % d == 0) {
5         ans++;
6     }
7 }
8 printf("%d", ans);
```

Типичная ошибка

Что выведет данный код?

```
1 int n, d, ans = 0;
2 scanf("%d", &n);
3 for (d = 1; d <= n && n % d == 0; d++) {
4     ans++;
5 }
6 printf("%d", ans);
```


Оператор break

Досрочно завершает цикл

```
1  int i;  
2  for (i = 0; i < 5; i++) {  
3      printf("start_□d\n", i);  
4      if (i == 2) {  
5          break;  
6      }  
7      printf("finish_□%d\n", i);  
8  }  
9  printf("last_□%d\n", i);
```

Что выведет?

Оператор break

```
start 0  
finish 0  
start 1  
finish 1  
start 2  
last 2
```

Оператор continue

Переходит на следующую итерацию

```
1 int i;
2 for (i = 0; i < 5; i++) {
3     printf("start_\d\n", i);
4     if (i == 2) {
5         continue;
6     }
7     printf("finish_%d\n", i);
8 }
9 printf("last_%d\n", i);
```

Что выведет?

Оператор continue

```
start 0  
finish 0  
start 1  
finish 1  
start 2  
start 3  
finish 3  
start 4  
finish 4  
last 5
```

Вложенные циклы

Задача: Распечатать таблицу умножения размера $n \times m$.

Ввод

3 5

Вывод

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15

Вложенные циклы

```
1 int i, j, n, m;
2 scanf("%d%d", &n, &m);
3 for (i = 1; i <= n; i++) {
4     for (j = 1; j <= m; j++) {
5         printf("%3d", i * j);
6     }
7     putchar('\n');
8 }
```

Ферзь

Задача: Дана позиция ферзя на шахматной доске.

Отметить все клетки которые бьет ферзь.

Ввод

F3

Вывод

8	X	X	.	.
7	.	X	.	.	.	X	.	.
6	.	.	X	.	.	X	.	.
5	.	.	.	X	.	X	.	X
4	X	X	X	.
3	X	X	X	X	X	X	X	X
2	X	X	X	.
1	.	.	.	X	.	X	.	X
	A	B	C	D	E	F	G	H

Ферзь

```

1 char i, j, r, c, rook, bishop;
2 c = getchar();
3 r = getchar();
4 for (i = '8'; i > '0'; i--) {
5     printf("%c_", i);
6     for (j = 'A'; j <= 'H'; j++) {
7         rook = (i == r) || (j == c);
8         bishop = abs(r - i) == abs(c - j);
9         if (rook || bishop) {
10             printf("X_");
11         } else {
12             printf(". _");
13         }
14     }
15     putchar('\n');
16 }

```


Ферзь

```
1 printf("□□");  
2 for (j = 'A'; j <= 'H'; j++) {  
3     printf("%c□", j);  
4 }
```

Ненужный двойной цикл

О «вредности» некоторых вложенных циклов.

Задача: дано вещественное число x и натуральное число n . Вычислить частичную сумму ряда экспоненты:

$$1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!}$$

Долгое решение

Долгое решение

```

1  int i, j, n;
2  double sum = 1, add;
3  scanf("%lf %d", &x, &n);
4  for (i = 1; i <= n; i++) {
5      add = 1;
6      for (j = 1; j <= i; j++) {
7          add *= x;
8      }
9      for (j = 1; j <= i; j++) {
10         add /= j;
11         sum += add;
12     }

```

Быстрое решение

Вместо вычисления степени заново на каждом шаге x^i , достаточно взять результат с предыдущего шага x^{i-1} и умножить один раз на x . Тоже самое с факториалом.

```
1  int i, j, n;
2  double sum = 1.0, add = 1.0;
3  scanf("%lf□%d", &x, &n);
4  for (i = 1; i <= n; i++) {
5      add *= x;
6      add /= j;
7      sum += add;
8  }
```

Разложить на множители

Задача: Дано целое положительное число от 1 до 10^5 .
Разложить его на простые множители (с учетом кратности).

Ввод

12

Вывод

2 2 3

Разложить на множители

Неправильное решение. Почему?

```
1  for (d = 2; d <= n; d++) {
2      if (n % d == 0) {
3          printf("%d□", d);
4          n /= d;
5      }
6  }
```

Разложить на множители

Правильное решение. Почему?

```
1  for (d = 2; d <= n; d++) {
2      while (n % d == 0) {
3          printf("%d□", d);
4          n /= d;
5      }
6  }
```

Разложить на множители

Задача: Посчитать количество делителей числа n от 1 до 10^9 .

Ввод

12

Вывод

6

Посчитать количество делителей

Долгое решение. Почему?

```
1 int n, d, tau = 0;
2 scanf("%d", &tau);
3 for (d = 2; d <= n; d++) {
4     if (n % d == 0) {
5         tau++;
6     }
7 }
8 printf("%d\n", tau);
```

Попробуйте ввести $10^9 + 7$.

Посчитать количество делителей

Идея: все делители числа n разбиваются на пары x и y такие, что

$$x \cdot y = n.$$

Например, делители 72:

$$1 \cdot 72 = 2 \cdot 36 = 3 \cdot 24 = 4 \cdot 18 = 6 \cdot 12 = 8 \cdot 9$$

Осторожно с полными квадратами.

$$1 \cdot 25 = 5 \cdot 5$$

Перенаправление ввода

Чтобы во время тестирования не вводить постоянно одну и ту же входную строку можно её перенаправлять из файла.

1) записываем в файл строку (один раз)

```
1 nano f.txt
```

2) запускаем с перенаправлением из файла (каждый раз после изменения кода)

```
1 ./prog < f.txt
```

3) также можно сохранить в файл результаты работы программы

```
1 ./prog > f.txt
```

Замер времени работы

Чтобы во время тестирования замерить время работы, обязательно сделайте перенаправление из файла (иначе ваш ввод тоже будет считаться).

1) записываем в файл строку (один раз)

```
1 nano f.txt
```

2) запускаем с перенаправлением из файла (каждый раз после изменения кода)

```
1 time ./prog < f.txt
```

3) смотрим вывод

```
1 real      0m1,004s
2 user      0m0,003s
3 sys       0m0,000s
```