Циклы

Основы языка C, лекция 4

Задача

- Отряд туристов отправился в поход.
  В первый день они прошли **L** км.
- Каждый следующий день проходили на **k** км больше
- Сколько км они прошли за **3** дня?
- Пусть дано: L = 4
  k = 2
- Объявим переменные: Начальное значение
  i — закончилось полных дней 0
  step — прошли за 1 день L
  path — прошли с начала похода 0

графическое решение

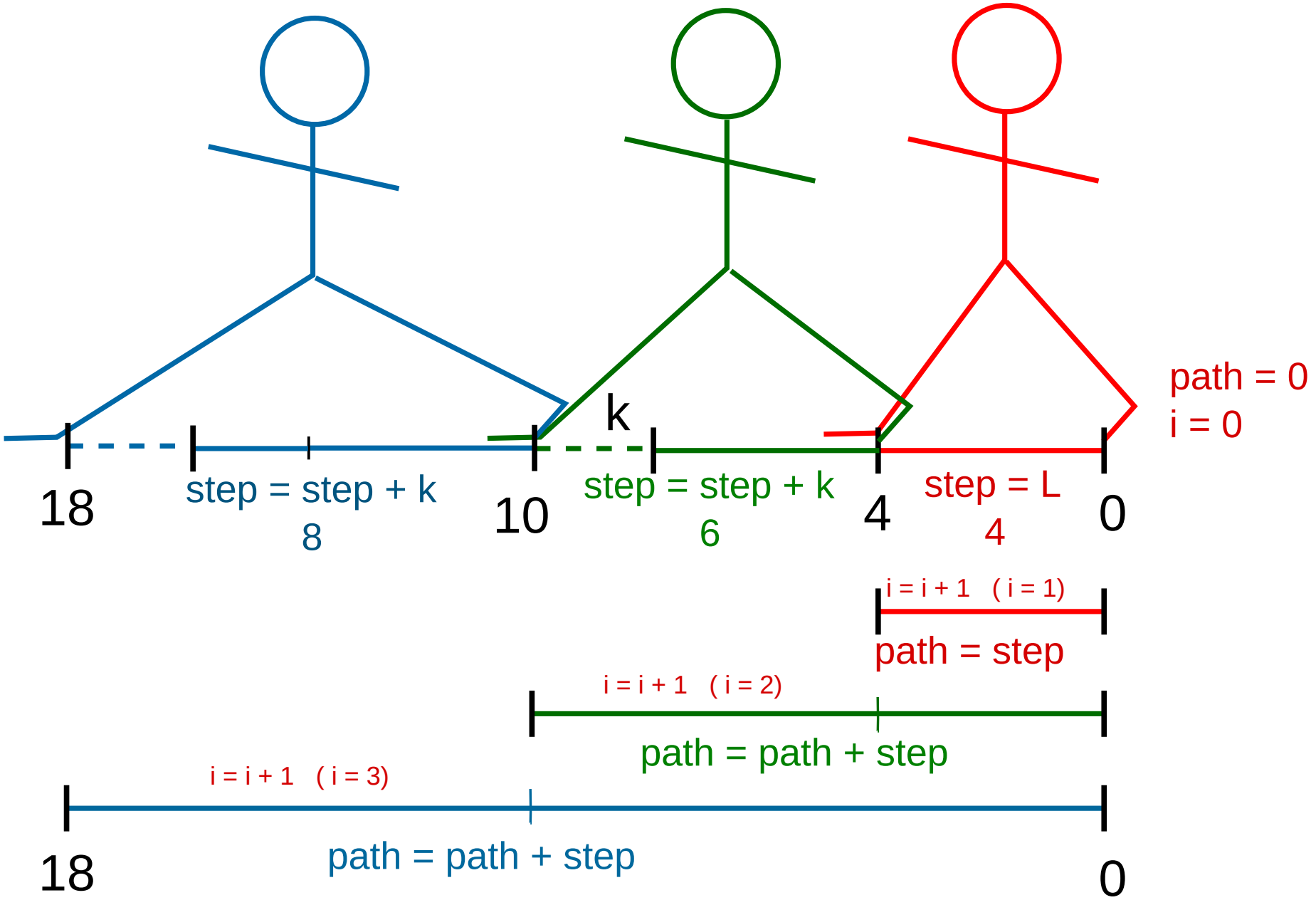
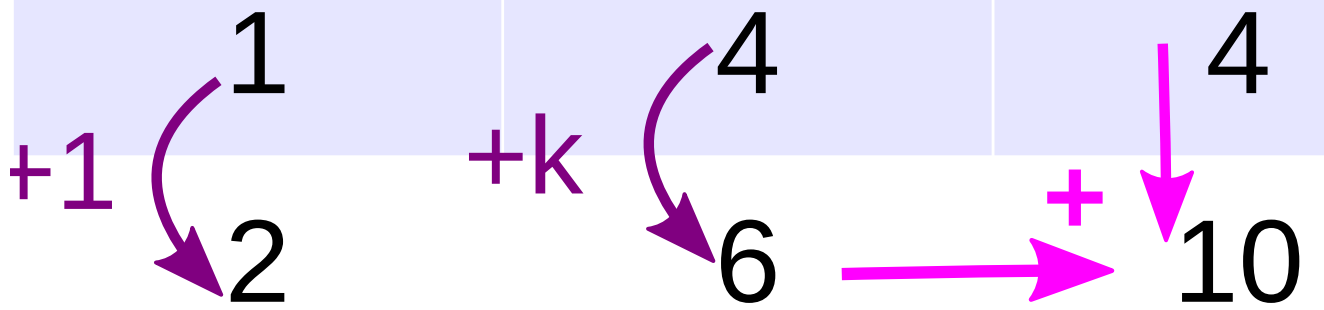
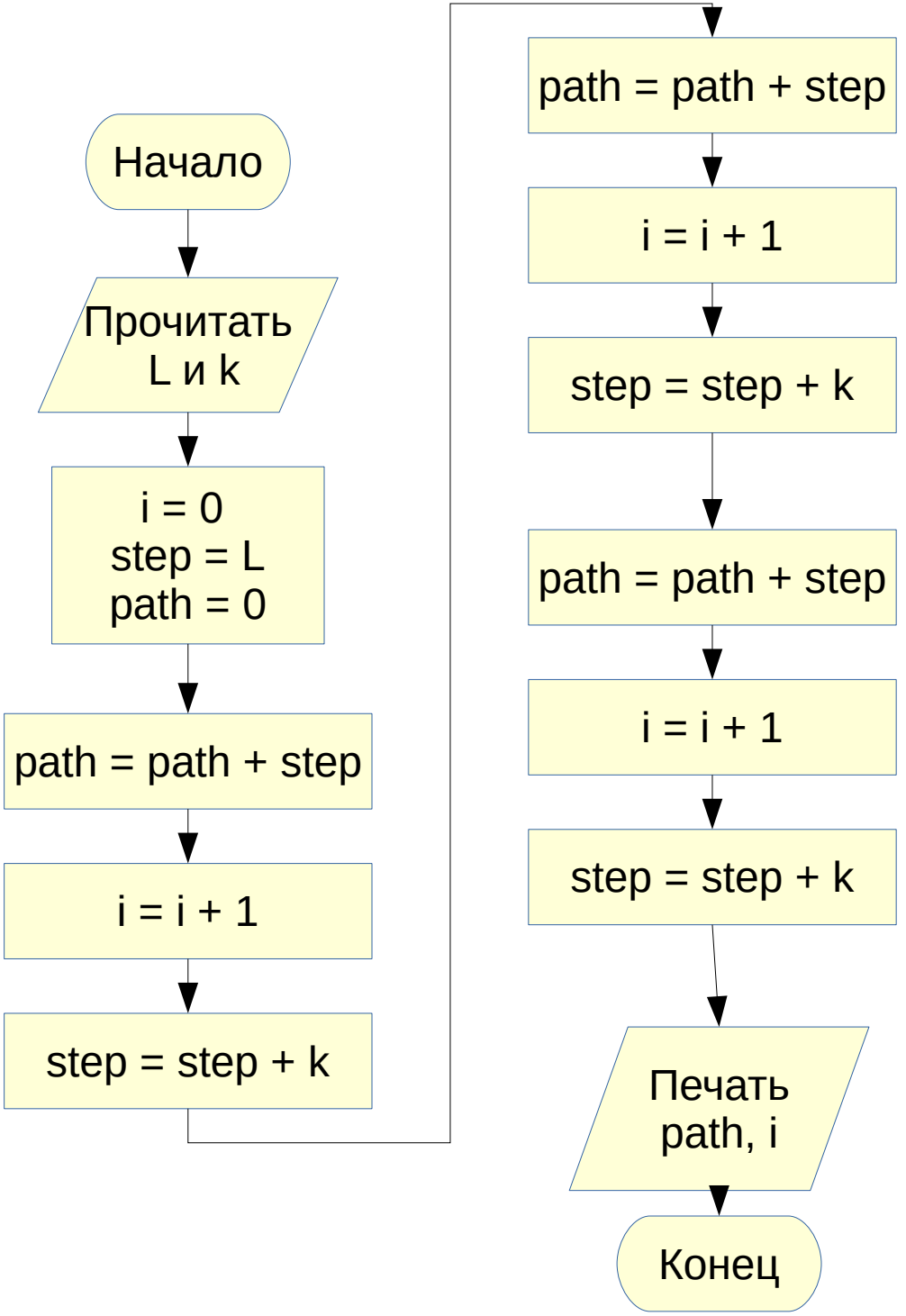


Таблица изменений

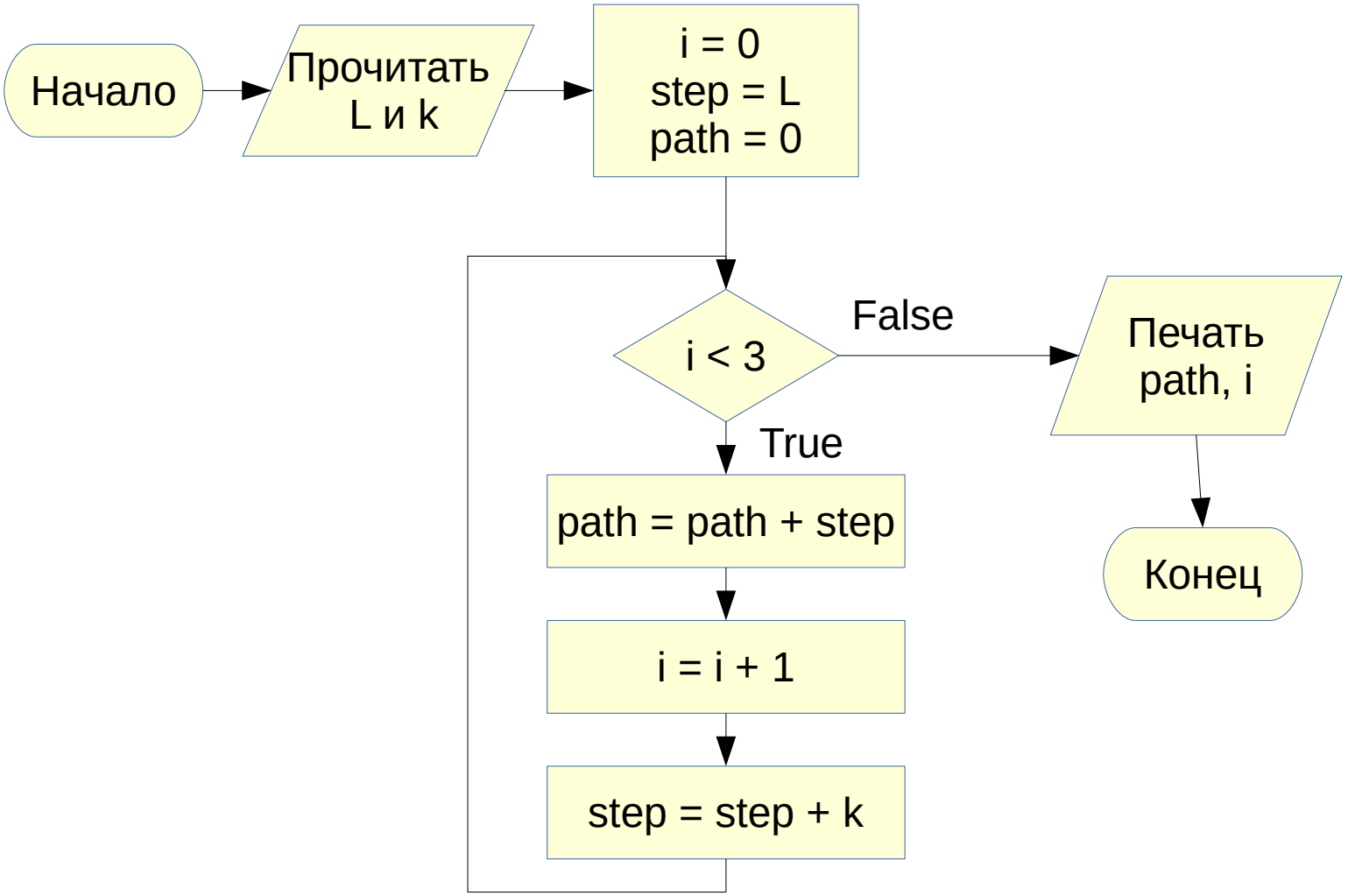
i = i + 1	step = step + k	path = path + step
0	L = 4	0
i	**step**	**path**
1	4	4
2	6	10
3	8	**18**



Блок-схема


Начало
Прочитать
L и k
i = 0
step = L
path = 0
path = path + step
i = i + 1
step = step + k
path = path + step
i = i + 1
step = step + k
path = path + step
i = i + 1
step = step + k
Печать
path, i
Конец

Блок-схема с повторением



Решим в лоб (много кода)

- i = 0; step = L; path = 0;

```
// день первый
path += step;          i ++;
printf("День %d, прошли сегодня %d, всего %d\n",
            i, step, path);
// готовимся с следующему дню
step += k;
// день второй
path += step;  i ++;
printf("День %d, прошли сегодня %d, всего %d\n",
            i, step, path);
// готовимся с следующему дню
step += k;      // еще код про третий день...
```

цикл while

- за 100 дней
- "за сколько дней прошли **X** км"

```
// до цикла 1 раз
i = 0; step = L; path = 0;
while ( i < 4 ) {
    // внутри цикла много раз повторяем
    path += step;
    i ++;
    printf("День %d, прошли сегодня %d, всего %d\n",
            i, step, path);
    // готовимся с следующему дню
    step += k;
}
// после цикла 1 раз
printf ("Всего прошли %d\n", path);
```

Оператор while

- **while (условие) {**

  тело_цикла

  }
- **условие –** условие **продолжения** работы цикла
- тело цикла – один оператор (быть может блочный)
- уровень вложенности – отступом
- **while (1)**

  printf ("Hello\n");
- **while (1) ;** // Что делает?

  printf ("Hello\n");

Оператор while и do..while

- Сначала думаем – потом делаем
- **while (условие) {**
  тело_цикла
  }

- Сначала делаем – потом думаем
- **do {**
  тело цикла
  } while (условие) ;

используем только если тело цикла должно выполниться хотя бы 1 раз

Алгоритм Евклида. НОД и НОК

- НОД (123, 21)
  123 % 21 = 18
  21 % 18 = 3
  18 % 3 = 0
- НОД (21, 125)
  21 % 125 = 21
  125 % 21 = 20
  21 % 20 = 1
  20 % 1 = 0
  Взаимно простые числа
- НОД(a,b)·НОК(a,b) = a·b

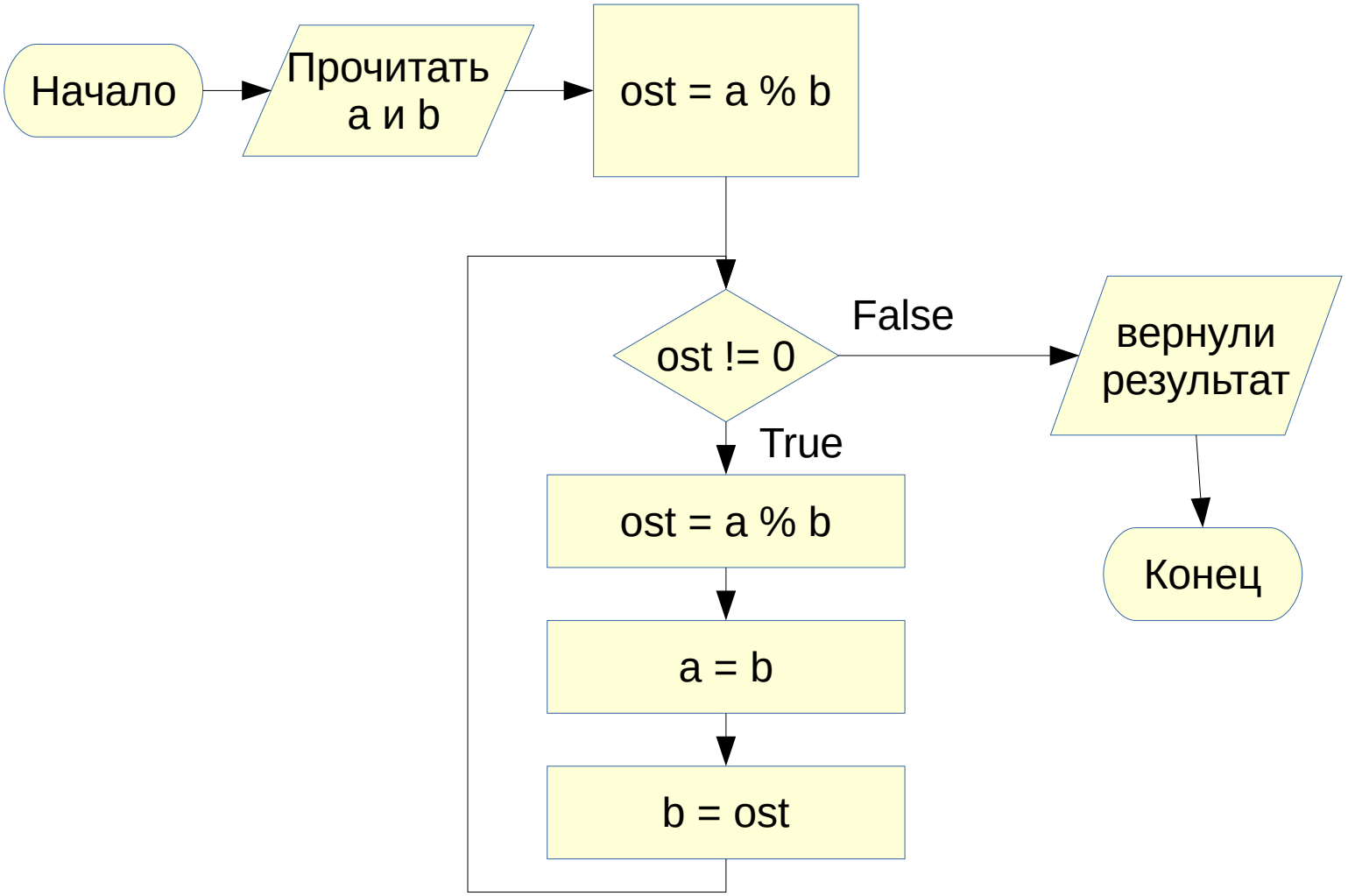
НОД - наибольший общий делитель
GCD greatest common divisor

НОК - наименьшее общее кратное
LCM least common multiple

Таблица для алгоритма

a	b	ost
123	21	% 18
21	**18**	**3**
18	3	0
3	0	0

Блок-схема while



Блок-схема do .. while

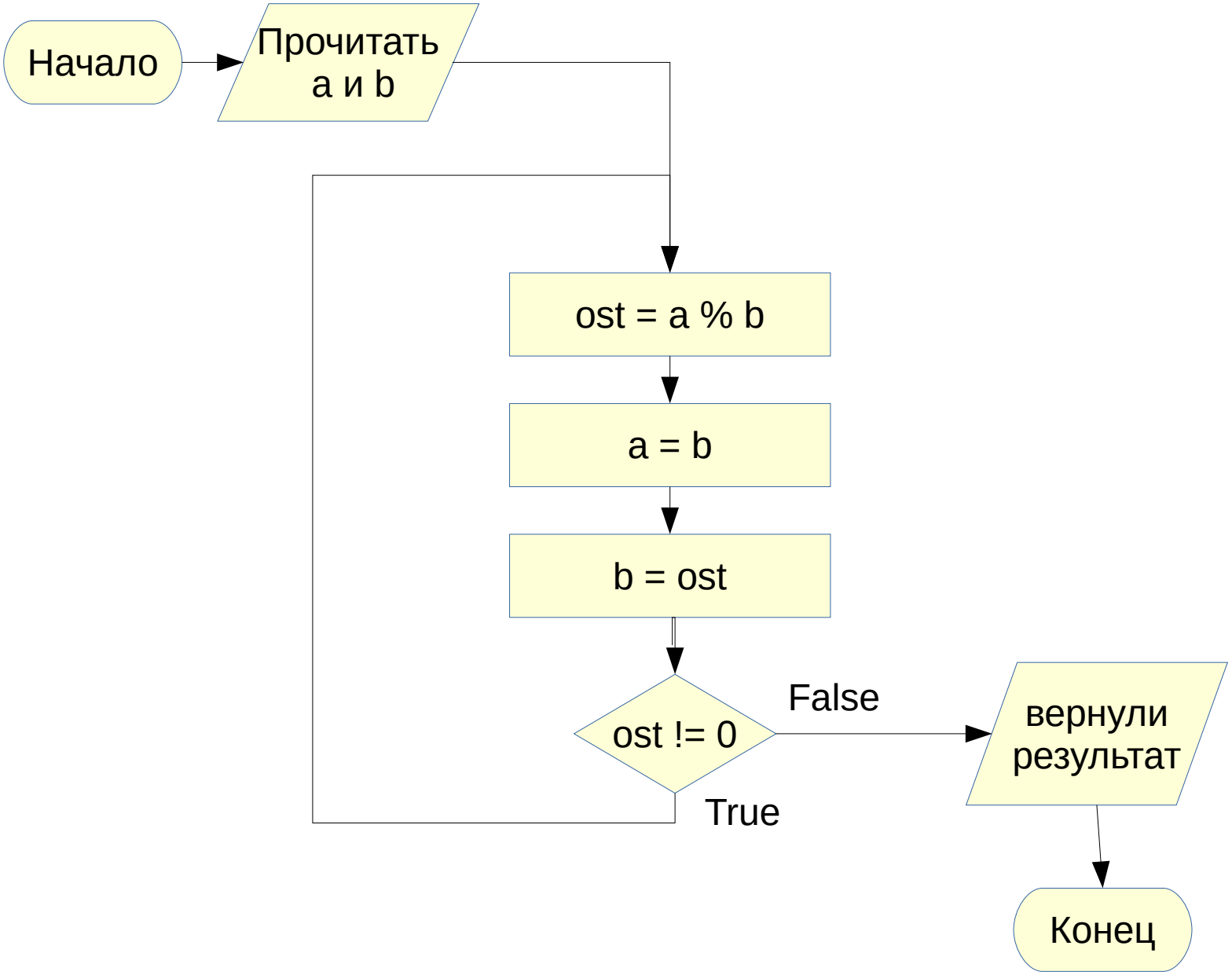
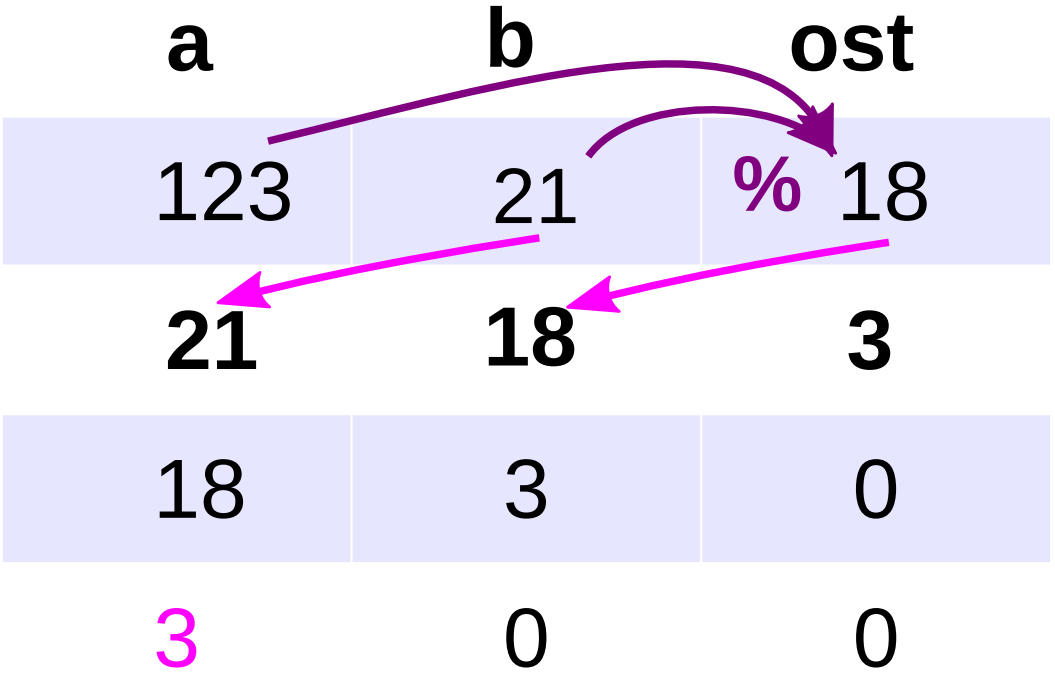


Таблица для алгоритма

- int nod (int a, int b) {

```
    do {
        ost = a % b;
        a = b;
        b = ost
    } while (ost != 0 ) ;

    return ???;
}
```



Собрать яблоки в ряду

- В ряду растет **n** яблонь.
- На каждой i-той яблоне $\mathbf{a_i}$ яблок
- Сколько яблок на всех яблонях?
- Пример входных данных:
  3
  5 2 10
- int a, b, c, n, sum;
  scanf("%d%d%d%d", &n, &a, &b, &c);
  sum = a + b + c;
  printf("Всего %d яблок\n", sum);

Поменьше переменных

- int x, sum = 0; // зачем нужно = 0?
- scanf("%d", &x);
  sum += x;
  printf("x is %d and sum is %d\n", x, sum);
- scanf("%d", &x);
  sum += x;
  printf("x is %d and sum is %d\n", x, sum);
- scanf("%d", &x);
  sum += x;
  printf("x is %d and sum is %d\n", x, sum);
- printf("%d\n", sum);

Собрать яблоки в ряду

- В ряду растет **n** яблонь.
- На каждой i-той яблоне $\mathbf{a_i}$ яблок
- Сколько яблок на всех яблонях?
- Пример входных данных:
  3
  5 2 10
- n — всего чисел
  i — чисел обработано
  x — очередное число
  sum — сумма

i = i + 1

scanf ("%d", &x)

sum = sum + x

i	x	sum
0		0
1	5	5
2	2	7
3	10	**17**

+1

+

Сделаем цикл со счетчиком

- int n, i, x, sum;
- scanf("%d", &n);
  i = 0; // инициализация счетчика
  sum = 0;
- **while (i < n) {**
  scanf("%d", &x);
  sum += x;
  printf("x is %d and sum is %d\n", x, sum);
  i ++;
  }
- printf("%d\n", sum);

Если забудем?

Если забудем?

Не забудем

for (выр1; выр2; выр3)
оператор

- int n, i, x, sum;
- scanf("%d", &n);
  sum = 0; // корзина пуста
- **for (i = 0; i < n; i++) {**
  scanf("%d", &x); // яблок на 1 яблоне
  sum += x; // добавили яблоки в корзину
  printf("x is %d and sum is %d\n", x, sum);
  }
- printf("%d\n", sum);

for "два забора"

for

- **выр1** — инициализация управляющей переменной
- **выр2** — условие продолжение цикла
- **выр3** — изменение управляющей переменной
- **for (выр1; выр2; выр2) {**
  тело цикла
  }
- for (i = 1**,** a = 1**;** i <= 5**;** a *= i**,** i++)
  ;
  printf("5! = %d\n", a);
- for (; ;) // бесконечный цикл

for "два забора" и грядки

Вернемся к яблоням

- int n; // сколько всего яблонь
  int x; // сколько яблок на очередной яблоне
  int sum;// сколько яблок уже сосчитали
  int i; // сколько яблонь уже сосчитали

- // 1 ряд яблонь, сколько яблонь в ряду?
  scanf("%d", &n);

- // перед началом подсчета ряда
  sum = 0;   // еще ни одного яблока не посчитано

- // считаем сумму яблок в ряду
  for (i = 0; i < n ; i++) {
  scanf("%d", &x);   // яблок на очередной яблоне
  sum = sum + x;   // эти яблоки кладем в корзину
  printf("i=%d x=%d sum=%d\n", i, x, sum);
  }

- // в конце печатаем результат:
  printf("Всего %d яблок\n", sum);

Страшная ворона

- На некоторых яблонях сидят вороны.
  Они съели все яблоки на своей яблоне и хотят еще яблок.
- Вороны налетают на корзину и каждая ворона уносит 1 яблоко.
- Если на дереве 5 ворон, будем обозначать их **-5**.
- Задача: собрать все яблоки с ряда с учетом ворон.
- Обратите внимание, что после ворон в корзине не может остаться -2 яблока. Яблоки настоящие.

Трусливый сборщик

- Пусть сборщик яблок, когда видит ворон, не подходит к дереву, а убегает из ряда.
- Он не собирает яблоки в ряду дальше. Убежал, сломал **break** забор.
- ```
  for (i = 0, sum = 0; i < n; i++) {
      scanf("%d", &x);
      if (x < 0)
          break;
      sum += x;
  }
  // сюда передаст управление break
  ```
-
  ```

Храбрый сборщик

- Пусть сборщик яблок, когда видит ворон, не подходит к дереву, а обходит его.
- Он собирает яблоки в ряду дальше. Продолжает **continue** собирать.
-
```
for (i = 0, sum = 0; i < n; i++) {
    scanf("%d", &x);
    if (x < 0)
        continue;
    sum += x;
    // сюда передаст управление break
}
```

Яблоневый сад

- В саду **k** рядов яблонь.
  В каждом ряду **n** яблонь, на каждой яблоне $\mathbf{a}_i$ яблок
- Сколько яблок собрали в этом саду
- **3**
  2 15 24
  3 7 82 15
  1 54
- Надо сложить числа (15+ 24) + (7 82 15) + (54)
- Код для подсчета яблок в 1 ряду у нас уже есть.

Функция сбора 1 ряда

- int **pikup_1row** ( ) {
  int i, n, x, sum;
  scanf("%d", &n);
  for (i = 0, sum = 0; i < n; i++) {
  scanf("%d", &x);
  sum += x;
  }
  return sum;
  }

Собираем весь сад

- int rows; // сколько рядов яблонь
  int j; // сколько рядов уже собрали
  int total; // сколько яблок собрали во всех рядах
- scanf("%d", &rows);
- for (j=0, total = 0; j<rows; j++) {
  total = total + **pickup_1row**();
  printf("Закончили %d-ый ряд.
  Собрали %d яблок\n", j, total);
  }
- // в конце печатаем результат:
  printf("В саду всего %d яблок\n", total);

Вложенный цикл

- int rows, j, total;
  scanf("%d", &rows);
- for (j=0, total = 0; j<rows; j++) {
  int n, i, x, sum;
  scanf("%d", &n);
  for (i = 0, sum = 0; i < n; i++) {
  scanf("%d", &x);
  sum += x;
  }
  total = total + **sum**;
  printf("Ряд %d. Собрано всего %d яблок\n", j, total);
  }
  printf("В саду всего %d яблок\n", total);

Очень страшные вороны

- Если встретилась яблоня с воронами,
  - убегаем из всего сада,
  - уносим корзину с уже собранными яблоками.
- **goto *METKA***; // передать управление на строку,
  // где была поставлена *METKA* (label)
- ***METKA* :** // поставить метку
- К одной метке может вести много разных goto эта метка.

break по внешнему циклу

```
for (j=0, total = 0; j<rows; j++ ) {
    scanf("%d", &n);
    for (i = 0, sum = 0; i < n; i++) {
        scanf("%d", &x);
        if (x < 0)
            goto AWAY;
        sum += x;
    }
    total = total + sum;
    printf("Ряд %d. Собрано всего %d яблок\n", j, total);
}
AWAY:   printf("В саду всего %d яблок\n", total);
```

continue по внешнему циклу

```
for (j=0, total = 0; j<rows; j++ ) {
    scanf("%d", &n);
    for (i = 0, sum = 0; i < n; i++) {
        scanf("%d", &x);
        if (x < 0)
            goto NEXT_ROW;
        sum += x;
    }
    total = total + sum;
    printf("Ряд %d. Собрано всего %d яблок\n", j, total);
NEXT_ROW:
}
printf("В саду всего %d яблок\n", total);
```

Возведение в степень a^n

- Алгоритм "в лоб" $a^n = a \cdot a^{n-1}$
- $a^{12} \rightarrow a^{11} \rightarrow a^{10} \rightarrow a^9 \rightarrow a^8 \rightarrow a^7 \rightarrow a^6 \rightarrow a^5 \rightarrow a^4 \rightarrow a^3 \rightarrow a^2 \rightarrow a^1$
- Сложность алгоритма n-1
- Умный алгоритм $a^{2n} = a^n \cdot a^n$
- $a^8 \rightarrow a^4 \rightarrow a^2 \rightarrow a^1$
- Сложность: число знаков в двоичной записи
- Доработаем для нечетных степеней $a^{2n+1} = a \cdot a^{2n}$
- $a^{12} \rightarrow a^6 \rightarrow a^3 \rightarrow a^2 \rightarrow a^1$
- Число знаков + число единиц в двоичной записи

  $12_{10} = 1100_2$