acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . CoffeeFor :

- Циклы на примере суммирования чисел
  - Постановка задачи
  - Тупое решение без циклов.
  - Решение без циклов чуть поумнее
  - Нужно научиться повторять код несколько раз
  - while
  - while: суммирование НЕизвестного количества чисел
  - while: суммирование известного количества чисел
  - Типичные ошибки
  - for : ничто не забыто
- Циклы на примере общей длины пути
  - Код без циклов
  - Через цикл while
  - Через цикл for
  - Через цикл do while
  - Вопросы для самопроверки:
- Циклы на примере подсчета яблок
  - 1 ряд
  - Много рядов (через функцию)
  - Много рядов (вложенный цикл)
  - break
  - continue
  - break во вложенном цикле
  - continue во вложенном цикле

Циклы на примере суммирования чисел

Постановка задачи

Вычислить сумму нескольких чисел.

Тупое решение без циклов.

Допустим, что нужно просуммировать 3 введеных целых числа.

Простой подход: посмотрим на программу сложения двух чисел и увеличим количество слагаемых.

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

Если нужно просуммировать 5 или 15 чисел, добавим количество переменных, допишем нужные scanf и не забудем поправить вычисление суммы.

Простота этого решения оборачивается его неэффективность, если нам нужно сложить 100 или 1000 чисел. Не хватает фантазии на имена переменных, для каждого дополнительного числа нужно править программу в трех разных местах (завести новую переменную, считываем число в эту переменную, добавляем значение переменной к сумме). Легко ошибиться, трудно найти где ошиблись.

Решение без циклов чуть поумнее

Подумаем, как упростить задачу, не используя новые знания. Обойдемся только двумя переменными:

- int x; // сюда мы будем считывать очередное число
- int sum; // здесь накапливаем сумму уже считанных чисел

```
int main() {
   int x, sum;
   sum = 0;   // зачем нужна эта строка?

   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);
    return 0;
}
```

Уже лучше. Чтобы добавить еще число к этому ряду можно бездумно использовать технологию copy-paste в одном месте. Труднее ошибиться и легче найти ошибку (у нас появилась отладочная печать на каждом шаге!).

- Зачем мы добавили sum=0 ? Чему будет равна сумма, если эту строку убрать?

Минусы решения: программа складывает фиксированное количество чисел, а хочется, чтобы она работала для произвольного количества.

Нужно научиться повторять код несколько раз

Подумаем, что нам хочется от языка программирования, чтобы не приходилось писать с помощью copy-paste такие неудобные программы?

Умения повторить указанный кусок кода несколько раз, указывая условие повторения кода. Например:

```
ПОКА (читается число в x) {
    увеличить sum на x
    печатать x и  sum
}
печатать sum
```

По возвращаемому значению функции scanf можно узнать, прочлось очередное число или нет. Если scanf("%d", &x) вернул 1, то очередное число прочиталось и записано в x. **TODO: ссылка на страницу про scanf**

while

Цикл **while** дает возможность многократного выполнения *оператора*, пока истино *логическое выражение*.

```
while (логическое выражение)
    оператор;
```

Если в цикле нужно выполнить несколько операторов, то нужно все их объединить в один *блочный* оператор (не забываем про фигурные скобки)

```
while (логическое выражение) {
    оператор1;
    ...
    операторN;
}
```

Как работает цикл?

1. Вычисляется *логическое выражение* (условие **продолжения** цикла)
2. Если оно ложно, то управление передается в точку программы сразу за оператором.
3. Если оно истино, то выполняется *оператор*
4. Управление передается на п.1

while: суммирование НЕизвестного количества чисел

Запишем алгоритм на языке C с помощью оператора while

```
int main() {
    int x, sum;
    sum = 0;

    while (1 == scanf("%d", &x)) {
        sum += x;
        printf("x is %d and sum is %d\n", x, sum);
    }

    printf("%d\n", sum);
    return 0;
}
```

Убедитесь, что программа работает. Если вы вводите числа вручную, в конце либо введите EOF (Ctrl+Z в Windows и Ctrl+D в Linux), либо напечатайте буквы, чтобы условие проложения цикла стало ложным.

while: суммирование известного количества чисел

Изменим условие задачи. Пусть на вход дается сначала ожидаемое количество чисел, потом собственно числа. Например, если нужно просуммировать 7, -2 и 10, вводится:

```
3
7 -2 10
```

Умный человек сразу догадается, что первое число можно прочитать и проигнорировать, сведя задачу к предыдущей. Но вдруг у нас введено 3 7 -2 10 34 -71 и суммировать, как нас просят по условию задачи, нужно только 7, -2, 10.

Заведем дополнительные переменные:

- n - сколько чисел нужно суммировать (в этом примере 3)
- i - сколько чисел мы УЖЕ просуммировали. Изначально i ноль и увеличивается на 1 после каждого шага цикла (считываем и прибавляем).

Условие продолжения цикла изменилось - количество УЖЕ просуммированных чисел меньше, чем нужно просуммировать.

```
int main() {
    int x, sum;
    int i, n;

    sum = 0;

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

    printf("%d\n", sum);
    return 0;
}
```

Типичные ошибки

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

- Что делает эта программа?
  - Считывает число n и печатает n чисел от 0 до n-1 включительно.
- Что будет, если в цикле удалить (закоментировать) оператор i++ ?
  - Программа бесконечно будет печатать нули.
    - Ctrl+C позволит прервать выполнение зациклившейся программы.
- Что будет, если i++ в цикле есть, а i=0 перед началом цикла мы забыли?
  - Зависит от ~~звезд~~ компилятора. По стандарту во вновь созданной переменной i может лежать *любое* число (в том числе 0). Если в i оказалось число -47, а вы вводили программе число 3, то вы можете поразиться тому, что напечатает программа.
- Как не забывать i=0 и i++ ?
  - Нужно использовать цикл **for**.

for : ничто не забыто

```
for (выражение1 ; логическое выражение; выражение3)
    оператор;
```

Как работает цикл?

1. **один раз** выполняется *оператор1*
2. Вычисляется *логическое выражение* (условие **продолжения** цикла)
3. Если оно ложно, то управление передается в точку программы сразу за оператором.
4. Если оно истино, то выполняется *оператор*
5. Выполняется выражение3
6. Управление передается на п.2 (условие продолжение цикла)

Просуммируем n чисел, используя оператор **for**

```
int main() {
    int x, sum;
    int i, n;
```

```
    sum = 0;

    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &x);
        sum += x;
        printf("i is %d, x is %d and sum is %d\n",
            i, x, sum);
    }

    printf("%d\n", sum);
    return 0;
}
```

Циклы на примере общей длины пути

Задача Туристы отправились в пеший поход. Поход длился 4 дня.
В первый день туристы прошли L км.
Каждый следующий день они проходили больше на k км.
Сколько всего км прошли туристы?

Код без циклов

Очень долго писать. Трудно написать без ошибок для 100 дней похода.

```
#include <stdio.h>

/* Туристы отправились в пеший поход. Поход длился 4 дня.
 * В первый день туристы прошли L>0 км.
 * Каждый следующий день они проходили больше на k>0 км.
 * Сколько всего км прошли туристы? */

int main()
{
    int L;  // сколько туристы прошли в первый день
    int k;  // на сколько больше они проходили на следующий день

    int x;  // проходят за день
    int s;  // прошли всего
    int i;  // закончилось дней похода

    // Дано:
    scanf("%d", &L);
    scanf("%d", &k);

    // перед началом похода:
    i = 0;      // поход еще не начался
    s = 0;      // еще ничего не прошли
    x = L;      // пройдут в первый день

    // первый день похода
    s = s + x;  // прошли всего
                // готовимся к следующему дню
    x = x + k;  // завтра пройдем больше
    i = i + 1;  // закончился 1 день
    printf("i=%d s=%d x=%d\n", i, s, x);

    // второй день похода
    s = s + x;  // прошли всего
                // готовимся к следующему дню
    x = x + k;  // завтра пройдем больше
    i = i + 1;  // закончился 2 день
    printf("i=%d s=%d x=%d\n", i, s, x);


    // в конце печатаем результат:
    printf("Всего прошли %d км\n", s);

    return 0;
}
```

Через цикл while

```
#include <stdio.h>

int main()
{
    int L;  // сколько туристы прошли в первый день
    int k;  // на сколько больше они проходили на следующий день

    int x;  // проходят за день
    int s;  // прошли всего
```

```
    int i;  // закончилось дней похода

    // Дано:
    scanf("%d", &L);
    scanf("%d", &k);

    // перед началом похода:
    i = 0;      // поход еще не начался
    s = 0;      // еще ничего не прошли
    x = L;      // пройдут в первый день


    while ( i < 4 ) {
        s = s + x;  // прошли всего
                    // готовимся к следующему дню
        x = x + k;  // завтра пройдем больше
        i = i + 1;  // закончился 1 день
        printf("i=%d s=%d x=%d\n", i, s, x);
    }

    // в конце печатаем результат:
    printf("Всего прошли %d км\n", s);

    return 0;
}
```

Через цикл for

```
#include <stdio.h>

int main()
{
    int L;  // сколько туристы прошли в первый день
    int k;  // на сколько больше они проходили на следующий день

    int x;  // проходят за день
    int s;  // прошли всего
    int i;  // закончилось дней похода

    // Дано:
    scanf("%d", &L);
    scanf("%d", &k);

    // перед началом похода:
    s = 0;      // еще ничего не прошли
    x = L;      // пройдут в первый день

    for (i = 0; i < 4 ; i = i+1) {
        s = s + x;  // прошли всего
                    // готовимся к следующему дню
        x = x + k;  // завтра пройдем больше
        printf("i=%d s=%d x=%d\n", i, s, x);    // печатается другое i
    }

    // в конце печатаем результат:
    printf("Всего прошли %d км\n", s);

    return 0;
}
```

Через цикл do while

Будет ли цикл работать правильно для 0 дней похода?

```
#include <stdio.h>

int main()
{
    int L;  // сколько туристы прошли в первый день
    int k;  // на сколько больше они проходили на следующий день

    int x;  // проходят за день
    int s;  // прошли всего
    int i;  // закончилось дней похода

    // Дано:
    scanf("%d", &L);
    scanf("%d", &k);

    // перед началом похода:
    i = 0;      // поход еще не начался
    s = 0;      // еще ничего не прошли
```

```
    x = L;      // пройдут в первый день

    do {
        s = s + x;  // прошли всего
                    // готовимся к следующему дню
        x = x + k;  // завтра пройдем больше
        i = i + 1;  // закончился 1 день
        printf("i=%d s=%d x=%d\n", i, s, x);
    } while ( i < 4 );

    // в конце печатаем результат:
    printf("Всего прошли %d км\n", s);

    return 0;
}
```

Вопросы для самопроверки:

- Как нужно изменить код, чтобы посчитать:
  - В первый день отряд прошел L км, каждый следующий день - на k км меньше. Обратно отряд не шел (стоял на месте). Сколько км прошел отряд за t дней?
  - До берега моря S км. В первый день отряд прошел L км, каждый следующий день - на k км больше. За сколько целых дней отряд дойдет до моря?

Циклы на примере подсчета яблок

1 ряд

```
#include <stdio.h>

/* В ряд растет N яблонь. На каждой яблоне ni яблок.
 * Сколько всего яблок на всех яблоках?
 * Формат входных данных: сначала число N, потом ряд чисел через пробел ni.
 * Пример: 3 10 2 5
 * */

int main()
{
    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 = i+1) {
        scanf("%d", &x);    // яблок на очередной одной яблоне
        sum = sum + x;
        printf("i=%d x=%d sum=%d\n", i, x, sum);
    }

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

    return 0;
}
```

Много рядов (через функцию)

Напишем функцию pickup_1row, которая считает сколько яблок в одном ряду.

Вызовем ее для каждого ряда.

```
#include <stdio.h>

// считает сколько яблок в 1 ряду
int pickup_1row()
{
    int n;  // сколько всего яблонь в ряду
    int x;  // сколько яблок на очередной яблоне

    int sum;// сколько яблок уже сосчитали (в ряду)
    int i;  // сколько яблонь уже сосчитали (в ряду)

    // 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 = i+1) {
        scanf("%d", &x);    // яблок на очередной одной яблоне
        sum = sum + x;
        printf("i=%d x=%d sum=%d\n", i, x, sum);
    }

    return sum;
}
int main()
{
    int rows; // сколько рядов яблонь
    int j;    // сколько рядов уже посчитали
    int total;// сколько яблонь уже посчитали во всех рядах

    scanf("%d", &rows);

    for (j=0; j<rows; j++ ) {
      total = total + pickup_1row();
      printf("Закончили %-ый ряд. Собрали % яблок\n", j, total);

   }

    // в конце печатаем результат:
    printf("В ряду всего %d яблок\n", sum);

    return 0;
}
```

Дописать по решению в классе

Много рядов (вложенный цикл)

Изменим постановку задачи: Яблони растут рядами. Дано количество рядов яблонь n. Далее в начале строки пишется сколько яблонь в ряду и количество яблок на каждой яблоне через пробел. Сколько яблок в саду?

Пример входных данных:

```
3
2 15 24
3 7 82 15
1 54
```

Нужно просуммировать числа (15+ 24) + (7 82 15) + (54). Заметим, что код, считающий сколько яблок в *одном* ряду у нас уже есть. Нужно выполнить его столько раз, сколько будет заявлено рядов.

```
#include <stdio.h>

int main()
{
    int n;  // сколько всего яблонь
    int x;  // сколько яблок на очередной яблоне

    int sum;// сколько яблок уже сосчитали (в ряду)
    int i;  // сколько яблонь уже сосчитали (в ряду)

    int rows; // сколько рядов яблонь
    int j;    // сколько рядов уже посчитали
    int total;// сколько яблонь уже посчитали во всех рядах


    // сколько рядов?
    scanf("%d", &rows);
    for ( j = 0; j < rows ; j ++ ) {


        // 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 = i+1) {
            scanf("%d", &x);    // яблок на очередной одной яблоне
            sum = sum + x;
            printf("i=%d x=%d sum=%d\n", i, x, sum);
        }

        // в конце ряда печатаем результат:
```

```
        printf("В ряду всего %d яблок\n", sum);

        total += sum;
        printf("Закончили ряд %d, итого %d яблок\n", j, total);
        printf("---\n");
    }

    // в конце печатаем результат:
    printf("Итого %d яблок\n", total);

    return 0;
}
```

break

Оператор **break** передает управление за пределы текущего цикла (или оператора switch/case).

```
#include <stdio.h>

/* В ряд растет N яблонь. На каждой яблоне ni яблок.
 * В саду могут быть вороны, которые отнимают яблоки (с яблони собирается отрицательное число яблок).
 * Напечатать  один раз YES если ворона в саду. Иначе ничего не печатать.
 * Формат входных данных: сначала число N, потом ряд чисел через пробел ni.
 * Пример1: 3 10 -2 5
 * Пример2: 5 10 -2 5 1 -7
 * */

int main()
{
    int n;  // сколько всего яблонь
    int x;  // сколько яблок на очередной яблоне

    int i;  // сколько яблонь уже сосчитали

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

    // перед началом подсчета ряда

    // считаем сумму яблок в ряду
    for (i = 0; i < n ; i = i+1) {
        scanf("%d", &x);    // яблок на очередной одной яблоне
        if (x < 0) {        // ВОРОНА !!!!
            printf("YES\n");
            break;
        }
        printf("i=%d x=%d\n", i, x);
    }

    printf("Закончили собирать яблоки с %d деревьев\n", i);

    return 0;
}
```

continue

Оператор **continue** передает управление за последний оператор цикла.

Пропускаем яблони с воронами. Ничего им не отдаем!

```
#include <stdio.h>

/* В ряд растет N яблонь. На каждой яблоне ni яблок.
 * В саду могут быть вороны, которые хотят отнять яблоки (с яблони собирается отрицательное число яблок).
 * Яблоки воронам не отдавать, пропускать эти деревья.
 * Напечатать сколько всего собрали яблок.
 * Формат входных данных: сначала число N, потом ряд чисел через пробел ni.
 * Пример: 3 10 -2 5
 * Решение: 10 + 5 = 15
 * */

int main()
{
    int n;  // сколько всего яблонь
    int x;  // сколько яблок на очередной яблоне

    int i;  // сколько яблонь уже сосчитали
    int sum;// сколько яблок собрали

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

    // перед началом подсчета ряда
```

```
    sum = 0;

    // считаем сумму яблок в ряду
    for (i = 0; i < n ; i = i+1) {
        scanf("%d", &x);    // яблок на очередной одной яблоне
        if (x < 0) {        // ВОРОНА !!!!
            continue;       // пропускаем дерево
        }
        sum += x;
        printf("i=%d x=%d sum=%d\n", i, x, sum);
    }

    printf("В ряду собрали %d яблок\n", sum);

    return 0;
}
```

break во вложенном цикле

Если сад из нескольких рядов, то как прекратить собирать яблоки во всем саду, а не только в текущем ряду?

Оператор **goto _метка_** передает управление на метку с указанным именем. Каждая метка должна иметь свое имя. Можно делать несколько переходов на одну и ту же метку.

В примере имя метки OUT_GARDEN

```
#include <stdio.h>

/* В ряд растет N яблонь. На каждой яблоне ni яблок.
 * Сколько всего яблок на всех яблоках?
 * Как только встретим в саду ворону (яблок на яблоне < 0, она отнимает яблоки), надо убежать из сада.
 * Формат входных данных: сначала число N, потом ряд чисел через пробел ni.
 * Пример: 3 10 2 5
 * */

int main()
{
    int n;  // сколько всего яблонь
    int x;  // сколько яблок на очередной яблоне

    int sum;// сколько яблок уже сосчитали (в ряду)
    int i;  // сколько яблонь уже сосчитали (в ряду)

    int rows; // сколько рядов яблонь
    int j;    // сколько рядов уже посчитали
    int total;// сколько яблонь уже посчитали во всех рядах


    // сколько рядов?
    scanf("%d", &rows);
    for ( j = 0; j < rows ; j ++ ) {

        // 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 = i+1) {
            scanf("%d", &x);    // яблок на очередной одной яблоне
            if (x < 0) {        // ВОРОНА
                goto OUT_GARDEN;
            }
            sum = sum + x;
            printf("i=%d x=%d sum=%d\n", i, x, sum);
        }

        // в конце ряда печатаем результат:
        printf("В ряду всего %d яблок\n", sum);

        total += sum;
        printf("Закончили ряд %d, итого %d яблок\n", j, total);
        printf("---\n");
    }

OUT_GARDEN:
    // сюда перейдем с помощью goto или дойдем, выполняя код как обычно (обойдем все яблони сада)

    // в конце печатаем результат:
    printf("Итого %d яблок\n", total);

    return 0;
}
```

continue во вложенном цикле

```
#include <stdio.h>

/* В ряд растет N яблонь. На каждой яблоне ni яблок.
 * Сколько всего яблок на всех яблоках?
 * Как только встретим в саду ворону (яблок на яблоне < 0, она отнимает яблоки), надо убежать из сада.
 * Формат входных данных: сначала число N, потом ряд чисел через пробел ni.
 * Пример: 3 10 2 5
 * */

int main()
{
    int n;  // сколько всего яблонь
    int x;  // сколько яблок на очередной яблоне

    int sum;// сколько яблок уже сосчитали (в ряду)
    int i;  // сколько яблонь уже сосчитали (в ряду)

    int rows; // сколько рядов яблонь
    int j;    // сколько рядов уже посчитали
    int total;// сколько яблонь уже посчитали во всех рядах


    // сколько рядов?
    scanf("%d", &rows);
    for ( j = 0; j < rows ; j ++ ) {

        // 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 = i+1) {
            scanf("%d", &x);    // яблок на очередной одной яблоне
            if (x < 0) {        // ВОРОНА
                goto OUT_ROW;
            }
            sum = sum + x;
            printf("i=%d x=%d sum=%d\n", i, x, sum);
        }

        // в конце ряда печатаем результат:
        printf("В ряду всего %d яблок\n", sum);

        total += sum;
        printf("Закончили ряд %d, итого %d яблок\n", j, total);
        printf("---\n");

OUT_ROW:
    // здесь конец ряда
    }

OUT_GARDEN:
    // здесь конец сада

    // в конце печатаем результат:
    printf("Итого %d яблок\n", total);

    return 0;
}
```