Arrays, Conditionals and Loops

Arrays

The type of an array containing elements of type `t` is `t[]`. Example: the type of an int array is `int[]`. To create an array one must always know the size of the array to be created beforehand. To create an int array having `n` elements, we use the statement:

```
int n = 100;
int[] A = alloc_array(int, n);
```

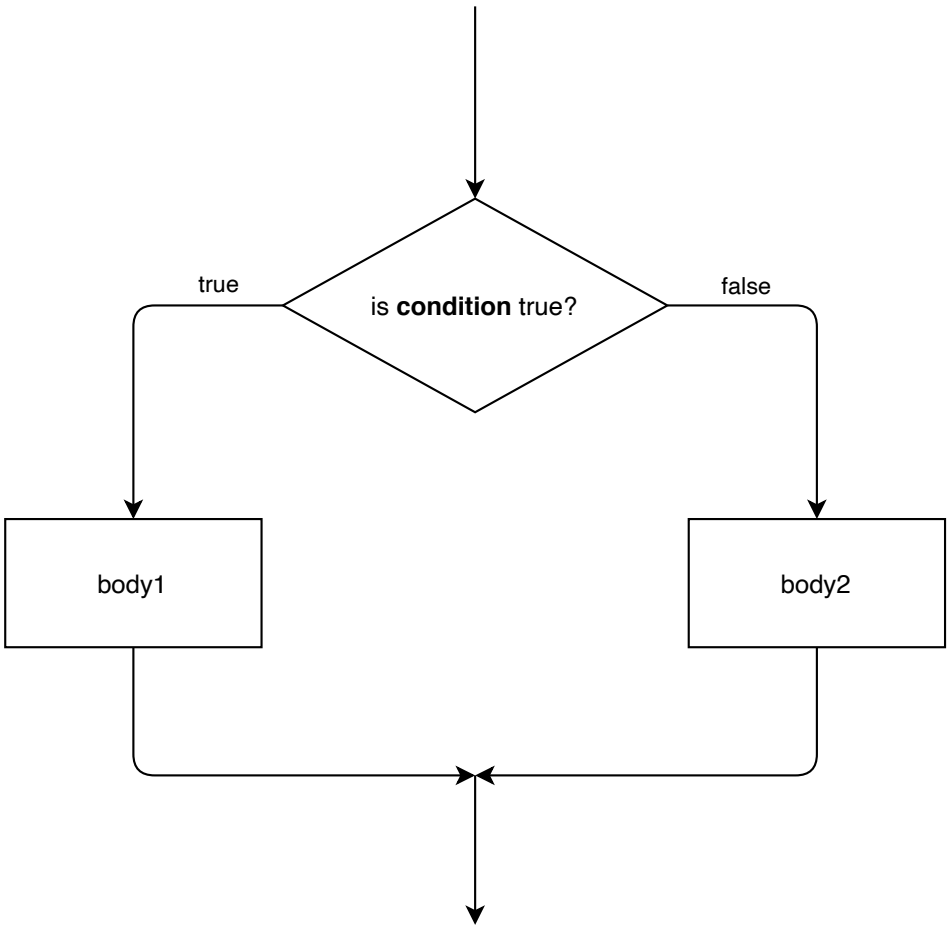
The only valid indices are the values from 0 to n-1.

Conditionals

A conditional block in C_0 looks like the following:

```
if (condition) {
    body1
}
else {
    body2
}
```

The `condition` is a boolean expression (without a semicolon) in the code above.



Example Code

The following code checks if an int variable x is even:

```
bool even(int x) {
    if (x%2 == 0) {
        return true;
    }
    else {
        return false;
    }
}
```

If the else block is not needed then it can be skipped. The format of such statements is:

```
if (condition) {
    body
}
```

Loops

for loop

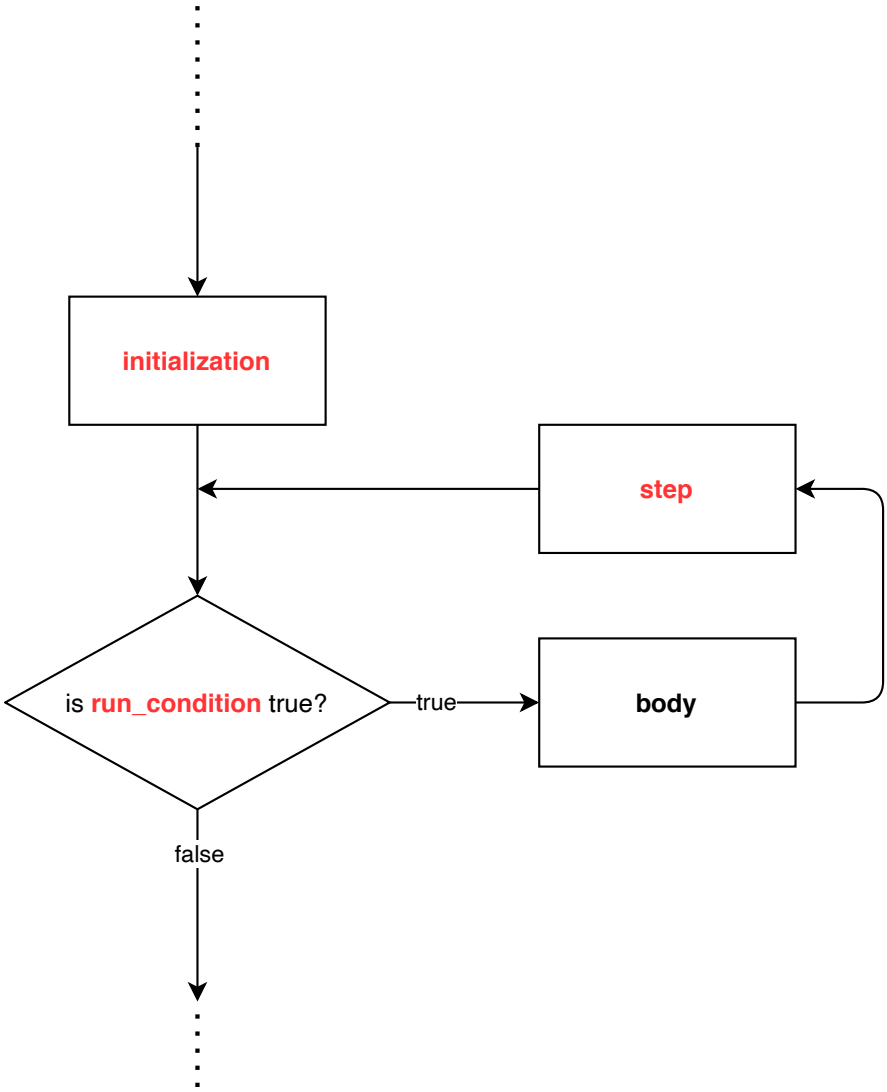
A for loop has the following form:

```
for (initialization;  run_condition; step) {
    body
}
```

In the code above `initialization` and `step` are statements, and the `run_condition` is a boolean expression.

The order of execution is as follows:

1. Execute `initialization`. (NB. It is executed precisely once)
2. Evaluate the condition `run_condition`. If it is false exit the loop, this means that the first statement following the `body` is executed.
3. Execute the statements in the `body`.
4. Execute the statements in `step`.
5. Go to line # 2 above.



Example Code

The following code counts the number of even elements in an input array.

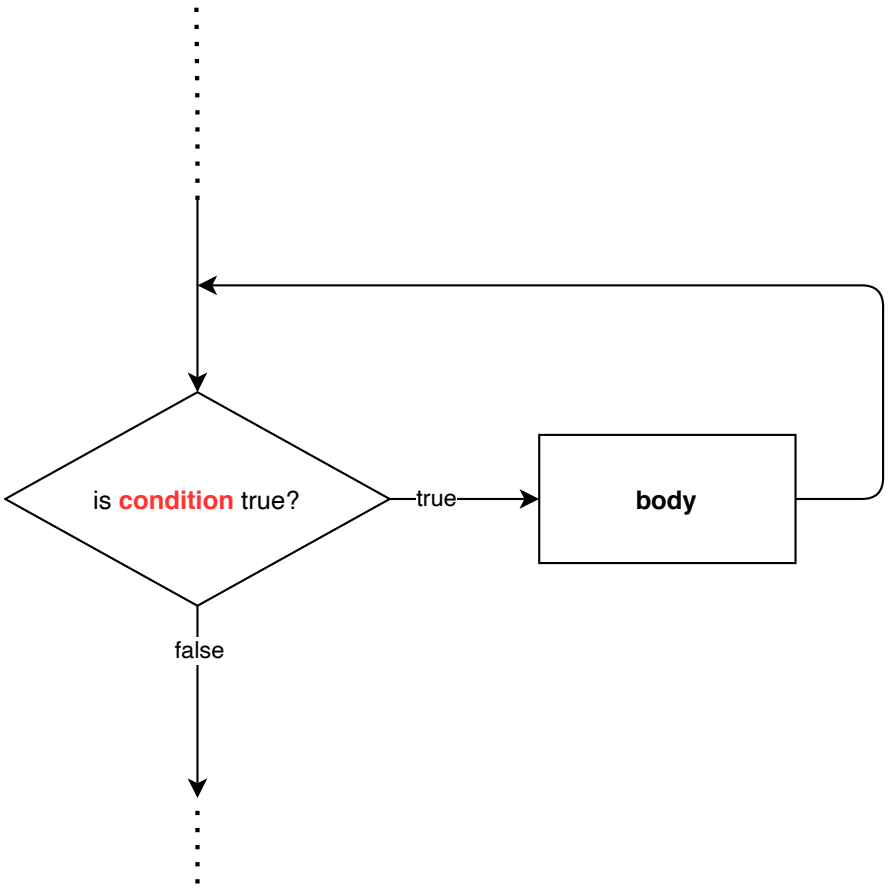
```
int countEven(int[] A, int n) {
    int index;
    int count = 0;
    for (index = 0; index < n; index = index + 1) {
        if ( even( A[index] ) ) {
            count = count + 1;
        }
    }
    return count;
}
```

while loop

A while-loop is a more flexible looping mechanish as compared to for-loops. It has the following syntax:

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

In the code above, `condition` is a boolean expression, and the `body` consists of statements. The statements in `body` are executed as long as `condition` is true, otherwise the next statement following `body` is executed.



Example Code

The following code computes the length of a collatz sequence for an input int. For 4, the length is 3 as the sequence consists of 4, 2, 1.

```
int collatz(int x) {
    //@requires x>0;
    int count = 1;
    while (x != 1) {
        // as long as x is not 1 compute the next number in the sequence
        if ( even(x) ) {
            x = x / 2;
        }
        else {
            x = 3*x + 1;
        }
        count = count + 1;
    }
    return count;
}
```