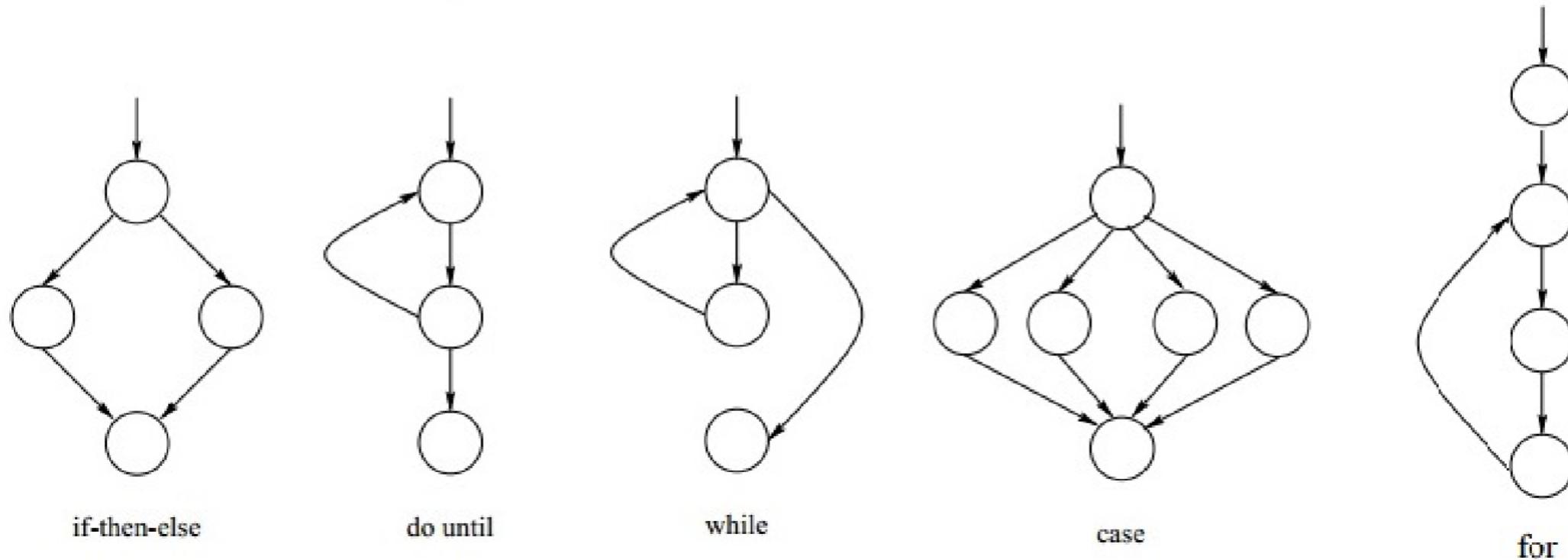


Control Flow

Programming and Music
<http://www.bjarni-gunnarsson.net>

Control Flow



“Control flow is the order function calls, instructions, and statements are executed or evaluated when a program is running. Control flow statements are used to determine what section of code is run in a program at a given time. An example of a control flow statement is an if/else statement.”

Boolean expressions

A **boolean** expression is an expression that results in a **boolean** value, that is, in a value of either **true** or **false**.

Complex boolean expressions can be built out of simple expressions, using the following boolean operators:

& (*and, true if and only if both sides are true*)

|| (*or, true if either side is true (or if both are true)*)

not (*not, changes true to false, and false to true*)

Parentheses can be used for grouping the parts of complex boolean expressions.

Boolean evaluations

Arithmetic tests that can be used to create **boolean values**. These compare two or more objects and the evaluation returns a boolean value used for program logic.

<, less than

<=, less than or equal to

==, equal to

!=, not equal to

>=, greater than or equal to

>, greater than

Truth Tables

Truth Table for AND

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table for OR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Conditionals

Conditional statements are used to test values and perform different actions depending on the result of the test.

The test **condition** must result in a **boolean expression** with only an option of **true** or **false** checked for in the test.

The most commonly used conditional is the **if statement** which tests an input and if it passes the test an action is executed.

If

The **if statement** usually has an **else branch** which specifies actions to take if the test fails.

Related conditionals are **switch** and **case** that offer many branches as well as those used for iteration on collections (**while**, **for**).

Brackets, Braces, and Parentheses

SuperCollider uses **brackets**, **braces**, and **parentheses** in its language syntax.

Brackets [] are used to define arrays of objects (or literals).

Braces { } are used to define function or class bodies.

Parentheses () are used to express events, separate expressions or define function argument lists.

Boolean logic

`(1 == 1) || (1 == 2)`

`1 == 2`

`1 != 2`

`1 <= 2`

`if(0.5.coin, {"true it is"}, {"false sometimes"})`

Iteration

When a task or function has to be executed repeatedly, an **iteration** is applied.

An example of an iteration is a **loop**.

A loop is when a sequence of statements is specified once but may be carried out several times in succession with changing variables.

Iteration is often performed to a **condition** where it iterates until the condition is met.

Iteration coupled with conditions attribute to the **control flow** of a program.

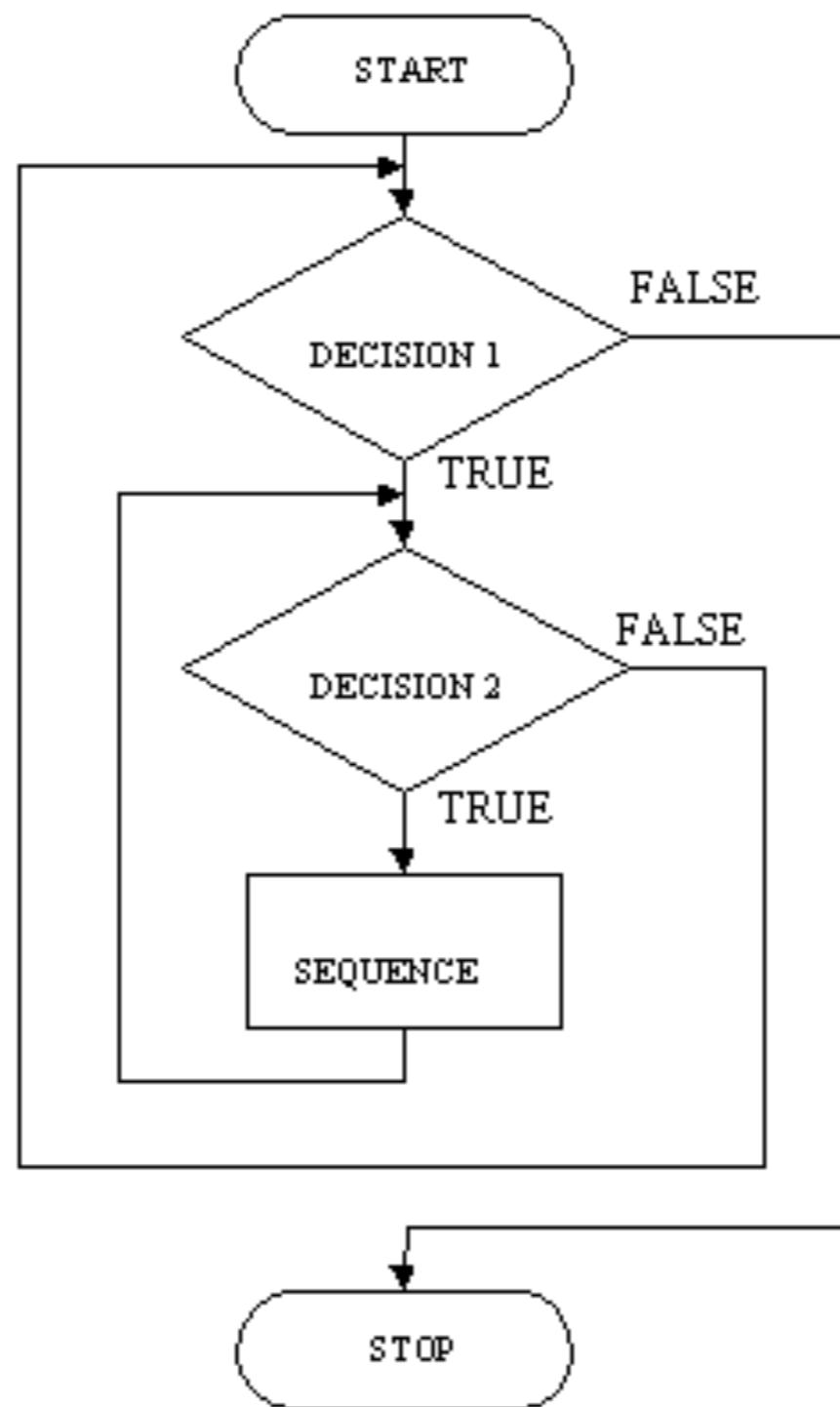
Iteration

In SuperCollider **Iteration** can be executed in various ways such as using the following:

- * **do** (*execute a number of time or iterate a collection*)
- * **for** (*go from a start to a end count and execute a function*)
- * **forBy** (*like for but has a variable step size*)
- * **while** (*execute while a certain test condition fails*)
- * **loop** (*a function method and loops that function*)
- * **repeat** (*repeats an object call a number of times*)

Additionally the collection objects have special iteration methods.

Iteration



Iteration

```
do ( [1,2], {item, i| (item * 10 + i).postln } )
```

```
7.do ( { rrnd(10,100).postln } )
```

```
for (10, 50, { arg i; i.postln } );
```

```
forBy (10, 100, 10, { arg i; i.postln } );
```

```
x = Prand([10, 12]).loop.asStream;  
x.nextN(32);
```

```

        (
            NF(\iop, {|freq=78, mul=1.0, add=0.0|
                var noise = LFNnoise1.ar(0.001).range(freq, freq + (freq * 0.1));
                var osc = SinOsc.ar([noise, noise * 1.04, noise * 1.02, noise * 1.08], 0, 0.2);
                var out = DFM1.ar(osc, freq*4, SinOsc.kr(0.01).range(0.92, 1.05), 1, 0, 0.005, 0.7);
                HPF.ar(out, 40)
            }).play;
        )
    (
        NF(\dsc, {|freq = 1080|
            HPF.ar(
                BBandStop.ar(Saw.ar(LFNnoise1.ar([19, 12]).range(freq, freq*2), 0.2).excess(
                    SinOsc.ar([freq + 6, freq + 4, freq + 2, freq + 8])),
                LFNnoise1.ar([12, 14, 10]).range(100, 900),
                SinOsc.ar(20).range(9, 11)
            ), 80)
        });
        if(cindex.isNil, { cindex = 2000 });
        if(pindex.isNil, { pindex = 1000 });
        pindex = pindex + 1;
        this[pindex] = \filter -> process;
    );
    clearProcessSlots {
        pindex = 1000;
        (this.pindex - 1000).do{|i| this[this.pindex+i] = nil; }
    }
    clearOrInit {|clear=true|
        if(clear == true, { this.clearProcessSlots(), { this.initialize() }});
    }
    transform {|process, index|
        if(index.isNil && pindex.isNil, {
            this.initialize();
        });
        pindex = pindex + 1;
        this[pindex] = \filter -> process;
    };
    control {|process, index|
        var i = index;
        if(i.isNil, {
            this.initialize();
            cindex = cindex + 1;
            i = cindex;
        });
        NF(\depfm, {|freqMin=5, freqMax=20, mul=20, add=80, rate=0.5, modFreq=2100, index=0.3, amp=0.2|
            var trig, seq, freq;
            trig = Dust.kr(rate);
            seq = Diwhite(freqMin, freqMax, inf).midicps;
            freq = Demand.kr(trig, 0, seq);
            HPF.ar(PMOsc.ar(LFCub.kr([freq, freq/2, freq/3, freq/4], 0, mul, add),
                LFNnoise1.ar(0.3).range(modFreq, modFreq*2), index) * amp, 50)
        }).play;
    );
    this[i] = \pset -> process;
}

```

Exercises

Exercises

1. Write a program to find the sum of three numbers, where these numbers can be different each time the program runs.
2. Write a programs that calculates which is the largest of three input numbers.
3. Write a program that generates a random number and based on its value prints the value number of times the word 'Sonology'.
4. Write a program that prints out all odd numbers between 10 and 50

...

Exercises

5. Write a loop that displays the multiplication table of a given number from 1 to 9.

6. Write a nested loop that will print a pattern that follows the logic:

1

22

333

4444

