nc

note compiler

Phillip Ames (pa2354@columbia.edu)

Describe the language that you plan to implement.

The language that I plan to implement leverages both the simple mathematical nature of music and its associated primitives, along with standard programming constructs such as functions, conditional evaluation and loops to empower users of the language to easily represent and perform computations on individual or multiple musical notes.

Explain what problem your language can solve and how it should be used.

Western music theory consists of a well-defined set of rules that dictate how a given pitch (a note) can be manipulated into various scales and chords. This system is built around the measurement of distance between pitches using the increment known as a half-step. A simple action such as transposing a song from one musical key to another involves shifting all notes by a given number of half-steps. The new representation maintains the same distance between notes while starting on a new scale degree. By providing the constructs mentioned in the language description, *nc* can aid programmers or musicians in solving a wide array of problems they might encounter in the musical domain.

Describe an interesting, representative program in your language.

nc can be used to solve a problem like the tedious task of identifying a chord's functionality in a musical progression. The identification of the distance between notes to distinguish chord quality (scale degree, major, minor, ...) is the first step in classifying a chord's function in a harmonic progression. Relating these chord qualities to a particular scale or tonal key aids in identifying the function within a progression. This requires the ability to perform musical computations such as generating scales from a chord representation, and then comparing the chord member notes against the notes in the scale.

Give some examples of its syntax and an explanation of what it does.

Basic Types:

- int
- bool
- note
- list (array-like container of notes supporting foreach-like iteration)

```
/* function declaration with return value / argument */
list major_scale(note n)
{
  /* c-style variable declarations */
  list l;
```

```
  int i;
  /*
   * A major scale follows the pattern:
   * WWWHWWH
   * where W represents whole step (2 semitones)
   * and H represents half step (1 semitone)
   * from the root note of the scale.
   * The += operator concatenates a note onto a list.
   * the addition operator accepts musical nomenclature such
   * as 'whole' or 'half'
   */
  l += n;
  for (i = 0; i < 6; i++) {
    if (i == 3 || i == 6)
      l += half;
    else
      l += whole;
    l += n;
  }
  return l;
}

/*
 * Returns a list which denotes all the white keys in
 * ascending order on a piano (uses the fact that
 * playing a major scale from C1 touches just the white
 * keys).
 */
list glissando()
{
  int i;
  list l;
  note n = C1;
  /* The lowest two keys on a piano are A0 and B0 */
  l += A0;
  l += B0;

  /* iteration, octave shifting with >> operator */
  for (i = 0; i < 7; i++) {
    l += major_scale(n);
    n = n >> 1;
  }
  l += C8;
  return l;
}

/* entry point for every program */
void main()
{
  /* local variable definitions */
  note n1, n2;
  list l1, l2, l3;

  /*
   * assigns to 'n' the value of an A note in the 4th octave,
   * which corresponds to the first A above middle C (a.k.a. A4).
   */
```

```
  n1 = A;
  n2 = A;

  l1 = major_scale(n1);
  l2 = glissando();

  /*
   * comparison/indexing operators, basic output functionality which
   * accepts string literals only.
   */
  if (l2.length != 52 || l2[0] != A0)
    print "Your piano seems to have an incorrect number of white keys\n";

  /*
   * This shifts n down 1 octave and assigns the resulting value (A3)
   * to n, replacing the value.
   */
  n1 = n1 << 1;

  /*
   * Notes can be compared for equality based on pitch or
   * whether they are in perfect unison.
   */
  if (n1 == n2) {
    print "n1 and n2 are the same note\n";
  }

  /* This will evaluate to false */
  if (n1 === n2) {
    print "n1 and n2 are in perfect unison\n";
  }
}
```