**Control Constructs:**

The control constructs in this (simple) version of the language will be **if-else** and **for** loops. An example would be:

if (some\_condition) {

//true part  
} else {

//false part

}

OR

if (some\_condition) {

//true part  
}

Note that the brackets are optional for both the **if** and the **else** statements. A **for**-loop will look something like:

for (init; test; step) {

//body  
}

with the brackets again being optional.

**Types:**

-There will be two types, **int** and **char**. An **int** type is 32-bits (or the size of a word on the x86 architecture) while a **char** type is 1-byte or 8-bits.

**Atomic Units:**

Identifiers

**Primitive Types/Atomic Units:**

-There will be two types, **int** and **char**. An **int** type is 32-bits (or the size of a word on the x86 architecture) while a **char** type is 1-byte or 8-bits.

-An **int** type is any of the following numbers:

-Base ten**:**

-Octal (Any

The first set of features I want to focus on are if-else statements, for loops, variable declarations, variable assignments, and arithmetic and logical expressions. The only types that my language will have are simple integer (int) and char (char) types. I will also provide a print statement that takes in one variable as its argument and prints it to the console [mostly used for debugging]. Example (valid) programs that I am thinking of would look like the following:

int x;

int y;

x = 5;

y = 4;

if (x < 0) {

print x;

} else {

print y;

}

for (x = 0; x < 10; ++x) {

y = y + 5;  
}

print y;

char c = ‘h’;

print c;

if statements can be nested, loops can be nested, if statements can have loops inside them, loops can have if statements inside them, etc. Scoping will be limited to the inner-most scope. For example, if we have

int x;

int y;

for (y = 0; y < 10; ++y) {

int x = 10;

print x;  
}

then the “x” used in the “print x” would be the innermost “x”, i.e. the one in the loop itself.