The PSI Programming Language
Handbook written by David H. Kristensen

Introduction
The PSI Programming Language (from here on, simply Psi) is a learners programming language, devised to introduce everyone to programming. The language is loosely based on C[1], and therefore, it provides a better platform for continuing to real C than other introductionary languages – ex. Pascal and BASIC.

Psi translates cleanly to ANSI-C, which is the worldwide standard for the most widely used programming language, C. The Psi compiler set consists of a compiler, and depends on *Flex* and *GCC*[2].

Program Structure
Psi programs are composed of a long string of **statements**, which do all the hard work. By now, you probably want to see a very basic example program, so here's the ubiquitous Hello World example, written in Psi:

```
+STANDARD
BEGIN
PrintString("Hello, World\n");
END
```

Not too complicated *(the \n means print new line)*. Every Psi program starts with the declaration of **+STANDARD,** which is the C support module. Programs will not function without the inclusion of this module. There are other modules, but for now, we'll just ignore them. By now, you have probably noticed the fact, that all the commands, except those within **BEGIN** and **END** (called the Program Block) are written entirely in capitals. Not only does it increase readability, it also improves the separation of actual program code and *external declarations*. We could also split up the Hello World program:

```
+STANDARD
BEGIN
PrintString("Hello,");
PrintString(" World\n");
END
```

Both programs would do exactly the same. You should also have noticed that all statements end with a ;. You could also go wild and write it like:

```
+STANDARD
BEGIN
PrintString("Hello,"); PrintString(" World");
END
```

Again, it's completely valid.

[1] Dennis Ritchie, et al.
[2] The Free Software Foundation, http://www.fsf.org

Variables and data types

Psi has two fundamental datatypes: **String** and **Number**. Strings are objects, designed to represent text (up to 4092 characters[3]), whereas Numbers contain 64-bit, double-precision floating-point numbers[4]. Objects are declared within the program block:

```
Number Age;
String Name;
```

Declaration of objects are also ended with a semicolon (;) just like all other statements inside the program block. You can assign text to a String. It is done with:

```
AssignString(Name, "John Smith");
```

The string will be filled with the characters between the quotes – or the characters of another string, as in this example:

```
String JohnSmith;
AssignString(JohnSmith, "John Smith");
AssignString(Name, JohnSmith);
```

Subsequently, the string can be printed out:

```
PrintString(Name);
```

Numbers can also be assigned data to. But instead of using AssignString, they use AssignNumber:

```
AssignNumber(Age, 20);
```

Number objects can also be assigned to each other, just like String objects. You can also use mathematics in number assignment:

```
AssignNumber(Age, Age+10);
```

Program Control

Psi programs' execution can be altered by *Control Structures*. Psi provides the **If-Else**, which is the most basic program-altering construct. In Psi, it looks like this:

```
If( Age is_equal_to 20 )
      PrintString("You are 20 years old!");
End
```

The If statement checks, whether Age is equal to 20. The equality check is catered for by the mnemonic **is_equal_to** *(note the underscores)*. The other mnemonics are: **is_not_equal_to**, **is_less_than** and **is_greater_than**. If you want to add functionality, so the program does something, it is done with the Else statement:

```
If( Age is_equal_to 20 )
      PrintString("You are 20 years old!");
End

Else
      PrintString("You aren't 20 years old!");
End
```

[3] In reference implementations
[4] On the most common architectures, including the Intel IA-32/IA-64, IBM PowerPC, Sun SPARC and DEC Alpha

The **Else** statement is optional. You can also insert multiple **Else-If** statements, to check other possible statements, before resorting to the default solution:

```
If( Age is_equal_to 20 )
      PrintString("You are 20 years old!");
End;

Else-If( Age is_less_than 20 )
      PrintString("You are less than 20 years old!");
End;

Else-If( Age is_greater_than 20 )
      PrintString("You are more than 20 years old!");
End;
```

Strings are compared with:

```
If( CompareString(Name, "David") )
      PrintString("Your name is David");
End;
```

Handling input

Psi also caters for the most basic input – reading from the keyboard. This can be done to Strings as well as Numbers. It is easily done with:

```
String Name;
Number Age;
ReadString( Name );
ReadNumber( Age );
```

Psi does not yet offer the capabilities of handling external files. This is, however, subject to change[5].

Comments

You can also add C-style comments in your code to increase readability. This is done by:

```
/* My comment
may span several lines
*/
```

Conclusion

Now, you have a new, brave world of programming unleashed at your fingertips. Use it well ;) And remember to contribute :D

[5] Will be available in PSI-2, scheduled for 1st half 2005.