

Programming Language Issues

What kind(s) of data is manipulated?
How is data referred to?
How is control specified?
What about constants?
Communication with the outside world?
Transparency?
Is it clear to the programmer how each language construct will be implemented?

Marketing?

Training?

...

Data Types

Booleans

Numbers

Integers

Bounded or infinite precision

Reals

Precision

"Fuzzy" numbers

Exceptional values

Infinities

unknowns

Pointers

References

Arrays

Multidimensional

"Ragged" / Triangular

Array bounds

Structures

Powersets

Classes???

Functions

Variables

Scoping

Lexical

Dynamic

Flat global namespace

...

Binding

How do variables get their binding?

Lexically

Dynamically

Control

goto

if statement

loops

while / for

function calls

"Object Oriented"

Self modifying code

Libraries

Are features just subroutines in a library?

How much can the language be extended with an appropriate library?

How is the language specified?

Syntax – "what is a legal program?"

Informal text description

List of rules

"Whatever the compiler accepts"

Formal syntax

Semantics – "what does a legal program do?"

Informal text description

List of rules

"Whatever running the compiled program does"

Formal mathematical description

Denotational semantics

Examples of variable scoping

addlocal.pl

```
#!/usr/local/bin/perl -w
```

```
use strict;
```

```
local $::i = 22;
```

```
sub makeAdder {  
    local $::i = shift;
```

```
    return sub {  
        local $::j = shift;
```

```
        return $::i + $::j;
```

```
    };
```

```
};
```

```
my $add5 = makeAdder(5);  
my $add7 = makeAdder(7);
```

```
print $add5, "\n";
```

```
print $add5->(7), "\n";
```

```
print $add7->(6), "\n";
```

```
swm-kayrun[752]: addlocal.pl
```

```
CODE(0xfedac)
```

```
29
```

```
28
```

addmy.pl

```
#!/usr/local/bin/perl -w

use strict;

my $i = 22;

sub makeAdder {
    my $i = shift;

    return sub {
        my $j = shift;

        return $i + $j;
    };
};

my $add5 = makeAdder(5);
my $add7 = makeAdder(7);

print $add5, "\n";

print $add5->(7), "\n";
print $add7->(6), "\n";
```

```
swm-kayrun[753]: addmy.pl
CODE(0xf096c)
12
13
```

addour.pl

```
#!/usr/local/bin/perl -w

use strict;

our $i = 22;

sub makeAdder {
    our $i = shift;

    return sub {
        our $j = shift;

        return $i + $j;
    };
};

my $add5 = makeAdder(5);
my $add7 = makeAdder(7);

print $add5, "\n";

print $add5->(7), "\n";
print $add7->(6), "\n";
```

```
swm-kayrun[754]: addour.pl
CODE(0xf096c)
14
13
```

add.cpp

```
#include <iostream>

using namespace std;

class Adder {

    int i;

public:

    Adder(int ii) : i(ii) {}

    int operator()(int j) {
        return i + j;
    }
};

int main() {
    Adder add5(5);
    Adder add7(7);

    cout << add5(7) << endl;
    cout << add7(6) << endl;
}
```

```
swm-kayrun[755]: g++ add.cpp
```

```
swm-kayrun[756]: a.out
12
13
```

Add.java

```
class Add {
    interface F {
        int f( int i );
    } // F

    public static F makeAdder( final int i ) {
        return new F(){
            public int f( int j ) {
                return i + j;
            }
        };
    }

    public static void main( String args[] ) {
        F add5 = makeAdder( 5 );
        F add7 = makeAdder( 7 );

        System.out.println( add5.f( 7 ) );
        System.out.println( add7.f( 6 ) );
    }
} // Add
```

```
swm-kayrun[757]: java Add
12
13
```