Concepts in Prog Langs - Part 2

/

What other language evolved in parallel with Lisp in the late 1950s?

-

Algol

/

What are the two most prominent Algol family languages?

-

Pascal and C

(although C differs in significant ways?)

/

What are the four main characteristics of the Algol family?

-

1) sequence of statements

2) block structure

3) functions & procedures

4) static typing

/

What was an interesting by-product of Algol 60 design?

-

Backus Normal Form (BNF)

/

What was an important contribution of Pascal?

-

A rich set of data-structuring concepts: records (structs), union type, subranges.

/

Which descendent of Pascal’s main innovation was a grouping sets of related declarations into programming units?

-

Modula (modules)

/

What characteristic most distinguishes C from other popular languages?

-

Its treatment of memory locations, arrays, and pointers. Inherited from B where the only data type is a ‘word’ or ‘cell,’ a fixed-length bit pattern, and memory is presented to the programmers as a linear array of words. Thus pointers and arrays are largely equivalent.

/

Which early language could be considered mostly functional with imperative features or a function-oriented imperative languages, that has concepts from both the Lisp and Algol family of languages?

-

ML

/

How does ML treat identifiers more uniformly that Pascal and C?

-

More uniformly, in C and Pascal, function identifiers are constants, whereas other identifiers are assignable. In ML all declarations are constants (no side effects), but can make them assignable using reference cells, similar to cons cell in lisp.

/

What data-structure types does ML share with Python?

-

Tuple and List

/

What do ML function declarations use?

-

Patterns. The formal parameters are a tuple that must match the unpacked the unpacked actual parameters. This allows pattern matching:

fun f(x, 0) = x

| f(0, y) = y

| f (x, y) = x + y

/

What are L-value and R-value?

-

L-value – location

R-value – value stored in that location

Comes from x:= y + 3

x is on the left side, its value doesn’t matter only its location since the results of the operation are going to be stored there.

y is on the right side, its location doesn’t matter, only its value which is needed for the operation.

/

In ML, reference cells must be initialized to a value of the correct type. What does this prevent?

-

uninitialized variables or dangling pointers

/

What is a type?

-

A collection of computational elements (to include data) that share some common property.

/

What are the three main uses of types?

-

1) naming and organizing concepts

2) ensure correct interpretation of bit sequences in memory

3) provide info about data to the compiler

/

What is a type error?

-

When a computational entity (function or data) is used in a manner inconsistent with the concept it represents.

/

When is a language type safe? Is C type safe?

-

When no program is allowed to violate its type distinctions.

C is not type safe due to pointers.

/

When is a language strongly typed?

-

When every expression has a type and it is type safe.

/

At what two times are types checked?

-

compile-time or run-time

(many languages use a combination of both)

/

What are the tradeoffs between compile-time and run-time type checking?

-

Compile-time checks are faster and more conservative, but may restrict program expressiveness.

/

What is type inference?

-

The process of determining the types of expressions based on the known types of symbols that appear in them. Unlike type-checking, not all information is specified and some must be inferred.

/

What are type variables?

-

Place holders for unknown types. May be resolved to specific types (bool, int, …) by the type-inference algorithm or may not be constrained in such a way by the function definition, so that the function can be applied to any type that matches the form given by the type expression. Java generics? (Type inference and polymorphism are independent concepts, but the latter arises from the way type variables are used in the former.)

/

What are the three steps of the ML type-inference algorithm?

-

1) Assign a type to the expression and each sub expression.

2) Generate a set of constraints on types. (a system of equations from parse tree)

3) Solve the constraints via unification. (solve the equations through substitution)

/

What are the sub-expressions and types of g(x) = 5 + x?

-

&lambda;x.((+5)x) - r

((+5)x) - s

(+5) - t

+ - int -> (int -> int)

5 - int

x - u

/

What is the rule of function application?

-

If type of f is a, type of e is b, and type of fe is c, then a = b -> c

(think that the type of f must be a for the application of e (to f) with type b to result in type c)

/

What is the name for an expression that satisfies all its constraints?

-

Well typed

/

What does it mean if there are type variables in the argument?

-

That the function is polymorphic and may be used for different types of arguments if they match the type expression.

/

What are the three forms of polymorphism?

-

1) Parametric polymorphism (variable argument types to a function)

2) Ad hoc polymorphism (overloading, two+ implementations with same name)

3) Subtype polymorphism

/

What are the two types of parametric polymorphism?

-

1) Implicit (type inference algorithm inserts template)

2) Explicit (special type variable for describe template)

/

What are the two basic forms of type declarations?

-

transparent – new name for another type

opaque – new type

/

What are the two forms of type equality?

-

name & structure

/

What is scope?

-

A region of text in which a declaration is visible.

A programming language feature that allows two identical names to refer to different things (memory locations)

/

What do scope and function calls both require?

-

new memory area to store function parameters and local variables.

/

What do most programming languages provide/call a region of text with begin and end markers, that may contain local declarations?

-

A block

/

What do the storage management mechanisms associated with block structure allow?

-

Recursive functions

/

Can blocks overlap?

-

No, but they can be nested.

/

What is an activation record?

-

Space for local variables declared in the block on the run-time stack. Also called a stack frame.

/

Where are local variable and parameters stored?

Global variables?

-

The current activation record.

A previous activation record. (the nearest enclosing block with the variable)

/

What is stack discipline?

What can cause it to fail?

-

De-allocating first the most recently allocated activation record.

Higher-order functions.

/

What is lifetime?

-

The duration a memory location is allocated for a declaration during the run of a program.

/

Does lifetime coincide with scope?

-

No, a variable my be hidden, or have a hole in its scope, if a nested block declares a variable of the same name.

/

What is the difference between a function and a procedure?

-

A function returns a value, a procedure does not. A function may have side effects, a procedure has only side effects and is a statement, not an expression.

/

What are the two main distinctions between parameter passing mechanisms?

-

1) Time of evaluation

2) Location of storage

/

What are the two most common non-lazy methods (evaluate the actual parameter before executing the function body)?

-

Pass-by-reference – pass the L-value (address) of the parameter

Pass-by-value – pass the R-value (contents of address) of the parameter

(there are other mechanisms)

/

What three things are important between pass-by-value and pass-by-reference?

-

1) Side Effects – pass-by-ref may alter values outside the function

2) Aliasing – pass-by-ref two names may refer to same object (memory)

3) Efficiency – pass-by-value inefficient for large data structures

/

What are the two main rules for finding a global variable?

-

Static Scope: find in closest enclosing block of program text. (most languages)

Dynamic Scope: find in most recent activation record. (think exceptions/stack trace)

/

What is tail recursion?

What is its advantage?

-

When all recursive calls in a function are tail calls, i.e. no computations afterward.

The same activation record can be used and the function can be compiled into an iterative loop.

/

What three things define a language as having first-class functions?

-

1) Functions declared within any scope.

2) Functions passed as arguments.

3) Functions returned as results.

/

In languages with first-class functions and static sope, what general represents a function value?

-

A closure

/

What is a closure?

-

A pair consisting of a pointer to the function code and a pointer to the activation record.

/

How are closures related to objects?

-

They both combine data with code for functions.

/

How can you construct a system using function passing and upcalls (function call up the stack)?

-

Higher levels pass handlers into lower levels which are called when the lower level needs to notify the higher level of something. (Callbacks/Events?)

/

When does stack discipline break down for higher-order functions?

-

During the upward funarg problem, when functions are returned as results, since the activation record can’t be deallocated. During downward funarg, statck discipline still exists but the stack is a tree.

/

What is considered harmful?

-

Go To

/

In modern programming style, what three things help avoid spaghetti code?

-

1) Group code into logical blocks.

2) Avoid explicit jumps, except for function returns.

3) Do not (can not) jump into the middle of a block or function

/

What are four constructs that help to structure jumps?

-

if … then … else … end

while … do … end

for … { … }

case …

/

What proveds a structured jump o leave a construct such as a block or function?

-

An exception.

/

What do you call creating an exception?

What about dealing with an exceptions?

-

raising or throwing

handling or catching

/

What rule determines the correct handler for an exception?

-

Dynamic scoping

/

What is continuations?

-

A programming technique based on higher order functions. Related to upcall and callback functions.