Q&A Session for Programming Languages Lecture 22

Session Number: 1200964069
Date: 2020-11-24
Starting time: 14:31

ANON - 14:30
Q: Thank you for providing everyone with an extra late day! People are traveling and it is hectic with move out so it is much appreciated!
Priority: N/A
	Konstantin Kuzmin - 14:33
	A: You are welcome!

ANON - 14:32
Q: How big can the teams be for the homework assignment?
Priority: N/A
	Ana L. Milanova - 14:35
	A: It is set the same way as HW5. Min is 1 max is 3.

ANON - 14:42
Q: It's undergrad with undergrad and grad with grad teams right? (for team assignments)
Priority: N/A
	Ana L. Milanova - 14:51
	A: No, there is no restriction. You can have a grad undergrad team.

ANON - 14:43
Q: Strictly speaking, an empty parameter list in a C function declaration declares a function with an indeterminate number of parameters, (void) is required to declare a function with no formal parameters
Priority: N/A
	Ana L. Milanova - 14:52
	A: Yes, that is correct. In C foo(void) means a function with no parameters.

ANON - 14:44
Q: Forgot the question, sorry. I assume that for homework purposes, we should follow the lecture's example?
Priority: N/A
	Ana L. Milanova - 14:54
	A: You may assume that foo() means a function with no parameters too. (Although, strictly, foo() in C means a function of

any number of parameters.) Make sure that you document assumptions you make in your answer.

ANON - 14:47
Q: So I'm confused. Did employee.monthlyrate fail?
Priority: N/A
Ana L. Milanova - 15:03
A: No, it does not fail. It printed the stale value we set in the previous initialization. You can reference the alternative part of the Union.

ANON - 14:50
Q: I assume the teams are for hw7?
Priority: N/A
Steven Haussmann - 14:50
A: Correct.

ANON - 14:53
Q: So I'm confused. Did employee.monthlyrate fail?
Priority: N/A

ANON - 14:56
Q: So I'm confused. Did employee.monthlyrate fail?
Priority: N/A
Steven Haussmann - 14:59
A: to repeat what Milanova said: it will display whatever happens to be in the place that the bogus parameter name points to. The runtime does not notice the incorrect variant record usage.

ANON - 14:56
Q: I'm kinda confused, what exactly did lecture 21 part 4 go over?
Priority: N/A
Steven Haussmann - 14:56
A: Some extra information about how primitive and composite types work.
Ana L. Milanova - 15:08
A: There are questions on array addressing and array layout in the homework and upcoming quiz.

ANON - 15:01
Q: Is a subroutine essentially a method?
Priority: N/A
Ana L. Milanova - 15:04
A: Yes, essentially. Routine/subroutine is a more general

term, it usually refers to a computational unit that triggers its own stack frame.

Ana L. Milanova - 15:05

A: "method" typically refers to a member function in a Java class.

ANON - 15:04
Q: if we have, f(y) but then f(x) is defined as x = x +1. does this mean y is the parameter and x is the argument?
Priority: N/A

Steven Haussmann - 15:05

A: It's the other way around. When we invoke f with y, y is an argument. When f(x) = { x = x + 1} is defined, x is a parameter in f

---

ANON - 15:05
Q: Is a subroutine the same as a member function?
Priority: N/A

Steven Haussmann - 15:06

A: A subroutine is just a function. "member functions" are a concept in object-oriented programming, where functions are stored in specific objects (and can use the fields of their object)

ANON - 15:07
Q: how can we figure out what parameter passing mode the language uses?
Priority: N/A

Steven Haussmann - 15:10

A: By reading about it! You could do some experimentation to figure it out, I suppose (e.g. seeing if f(x) = x += 1 mutates its argument)

Ana L. Milanova - 15:11

A: Yes, what Steven says. It is usually one of the first thing that is announced about a programming language. And you can write some programs (as we shall see shortly in lecture) that can help you figure out the mechanism.

ANON - 15:07
Q: I'm confused why Call-By-Reference falls under the Value Model
Priority: N/A

Ana L. Milanova - 15:14

A: It makes sense under the Value model. It requires the distinction of l-lalue and r-value of a variable which we don't have with the reference model. It requires the language to pass an l-value as an argument.

ANON - 15:16
Q: why do we use aliasing?
Priority: N/A
	Ana L. Milanova - 15:18
	A: In this context (and in other contexts in programming) memory aliasing improves efficiency. In this case, aliasing allows us to work with a large global structure without having to copy this structure on the frame of the callee.
	Steven Haussmann - 15:18
	A: Aliasing is a side-effect of any system where two different variables can refer to the same data. It is often useful for shared access to information, but also can result in bugs (e.g. we don't expect x and y to be linked)

ANON - 15:27
Q: are r and my_huge_struct aliases? but we cannot modify my_huge_struct due to the const?
Priority: N/A
	Steven Haussmann - 15:28
	A: Yes, const results in a read-only alias

ANON - 15:29
Q: for the first example, if we set r =0, will it still point to the huge_struct object?
Priority: N/A
	Steven Haussmann - 15:30
	A: The pointer will be modified. Since the pointer was passed by value, this will not affect the value stored in my_high_struct

---

ANON - 15:49
Q: Is the expression passed as a closure only in Algol? Or in the example would c[2] be modified instead of c[3]?
Priority: N/A
	Ana L. Milanova - 15:52
	A: As far as I know. Call by name is not implemented by any other major language in this family (the family of imperative procedural languages).
	Ana L. Milanova - 15:53
	A: In the example, it is c[3] that gets modified. This is because the code modified m (by value) on the previous line, and c[m] references c[3].
	Ana L. Milanova - 15:54
	A: Sorry, I meant "modified m (by reference)" not "modified m (by value)".

ANON - 15:50

Q: how does call by name and call by value related to normal order reduction?
Priority: N/A
	Steven Haussmann - 15:53
	A: Call by name vs. call by value affect when, exactly, an argument is evaluated.
	Steven Haussmann - 15:54
	A: recall that, in normal order reduction, "arguments" -- i.e. the right side of an application where the left side is an abstraction -- get evaluated *after* that application is reduced. This means we lazily evaluate the "argument"
	Steven Haussmann - 15:54
	A: whilst, in applicative order, we fully reduce the left and right sides before performing the reduction; thus, we eagerly evaluate the "argument"

ANON - 15:51
Q: So is Call By Sharing just Call By Reference in a Language that only passes around References?
Priority: N/A
	Ana L. Milanova - 15:55
	A: Yes, you can think of it this way.

ANON - 15:56
Q: So Call-By-Sharing does not distinguish between l and r values becuase everything is an l value = an address which is why it just passes around addresses only but still is very similar to the Call By Reference?
Priority: N/A
	Ana L. Milanova - 16:07
	A: Yes, that is correct. A variable is an address. What we cannot do in languages like Java that use the reference model is make two stack variables aliases to the same location. We can do this in C/C++.

ANON - 15:59
Q: I was 4 minutes late to Lecture due to Internet Issues. What did I miss? Was it just the Homework 7 Team Information?
Priority: N/A
	Ana L. Milanova - 16:08
	A: Yes. We were also a bit late as we couldn't join the event right away.

ANON - 16:02
Q: So Haskell uses Call-By-Name? Scheme uses Call-By-Value or Applicative and Haskell uses Lazy Evaluation or Call-By-Name?

Priority: N/A
Ana L. Milanova - 16:10
A: Yes, that is correct. "Call by name" is another term commonly used for "lazy evaluation". The term that is usually used with Haskell is "lazy evaluation".

ANON - 16:05
Q: Is there office hour today?
Priority: N/A
Konstantin Kuzmin - 16:05
A: Yes.

ANON - 16:06
Q: For the homework, you don't have to worry about the head of freshVars showing up twice because it's automatically precluded from E1[z/y] correct?
Priority: N/A
Ana L. Milanova - 16:14
A: Yes, this is correct. (I might be misunderstanding the question though.)

ANON - 16:06
Q: do we have ta offic hour tomorrow?
Priority: N/A
Steven Haussmann - 16:07
A: No, office hours aren't being held over the break.

ANON - 16:06
Q: Is there an easy way to strip the value from a Maybe Monad without using a case expr in Haskell?
Priority: N/A
Steven Haussmann - 16:07
A: No -- that's the entire point! You must explicitly handle both cases. Also, for this assignment, you will always want to be making a decision based on whether or not you got Just Expr or Nothing
Steven Haussmann - 16:07
A: See this thread for more thoughts: https://submitty.cs.rpi.edu/courses/f20/csci4430/forum/threads/497

ANON - 16:07
Q: is the best way to become familar with parameter passing to understand the examples + practice with coding?
Priority: N/A
Ana L. Milanova - 16:15
A: You can go over the minimal examples in lecture and in the

textbook. We have more practice problems in the upcoming quiz and HW as well.

ANON - 16:09
Q: Why can you not make two stack variables aliases in a Reference Model Language like Java? Is it because they are not on the heap and the reference model everything is on the heap?
Priority: N/A
Ana L. Milanova - 16:22
A: Parameter passing is by-value, which means that the argument value (a heap address or a value of simple type) is copied into the parameter location. But the parameter and argument are still names of two distinct locations on the stack.
Ana L. Milanova - 16:22
A: We cannot have the aliasing we had in the pseudocode examples or in the C++ examples when we passed by reference.

ANON - 16:10
Q: I'm guessing you return a Maybe monad using "return"?
Priority: N/A
Ana L. Milanova - 16:12
A: Yes. And the "return" of the Maybe monad is Just.
Steven Haussmann - 16:14
A: ^ which is how you'll interact with it if you don't care about the monadic features of Maybe

---

ANON - 16:21
Q: Happy Thanksgiving!!
Priority: N/A
Ana L. Milanova - 16:23
A: Thank you! Happy Thanksgiving to you all and have an enjoyable break!

ANON - 16:25
Q: Oh so the Reference Model does not deal with Aliases but passes by value and may pass the same reference address to a a parameter =>thereby creating an "alias"
Priority: N/A
Ana L. Milanova - 16:27
A: There are aliases, but they are not the <local var,local var> aliases. But you can have x.f and y.f in Java, where x and y are the same address and x.f and y.f are names for the same location on the heap.
Ana L. Milanova - 16:29
A: In Java (and I believe in other languages with this model) we only have aliases for heap locations. In C/C++ we can have aliases

for stack locations, e.g., x = 1; a = &x; and x and *a are aliases.

ANON - 16:25
Q: Thank you Professor Milanova, Professor Kuzmin, TA Steven + Everyone. Hope you have a great break as well!
Priority: N/A

Ana L. Milanova - 16:30
A: Thank you! I hope you have an enjoyable and restful break too!