bb0(%0 : $Expr):
debug_value %0 : $Expr, let, name "self", argno 1 // id: %1
switch_enum %0 : $Expr, case #Expr.Int!enumelt.1: bb1, case #Expr.Var!enumelt.
bb1(%3 : $(n: Int)): // Preds: bb0
br bb3 // id: %4
bb2(%5 : $<τ_0_0> Box (var τ_0_0) <(x: String)>): // Preds: bb0
br bb3 // id: %6
bb5(%22 : $<τ_0_0> Box (var τ_0_0) <(f: Expr, g: Expr)>): // Preds: bb0
%23 = project_box %22 : $<τ_0_0> Box (var τ_0_0) <(f: Expr, g: Expr)>, 0 /
%24 = tuple_element_addr %23 : $*(f: Expr, g: Expr), 0 // user: %25
%25 = load %24 : $*Expr // users: %28, %30
%26 = tuple_element_addr %23 : $*(f: Expr, g: Expr), 1 // user: %27
%27 = load %26 : $*Expr // users: %29, %30
debug_value %25 : $Expr, let, name "f" // id: %28
debug_value %27 : $Expr, let, name "g" // id: %29
br bb8(%25 : $Expr, %27 : $Expr) // id: %30
bb6(%31 : $<τ_0_0> Box (var τ_0_0) <(f: Expr, g: Expr)>): // Preds: bb0
%32 = project_box %31 : $<τ_0_0> Box (var τ_0_0) <(f: Expr, g: Expr)>, 0 /
%33 = tuple_element_addr %32 : $*(f: Expr, g: Expr), 0 // user: %34
%34 = load %33 : $*Expr // users: %37, %39
%35 = tuple_element_addr %32 : $*(f: Expr, g: Expr), 1 // user: %36
%36 = load %35 : $*Expr // users: %38, %39
debug_value %34 : $Expr, let, name "f" // id: %37
debug_value %36 : $Expr, let, name "g" // id: %38
br bb8(%34 : $Expr, %36 : $Expr) // id: %39
bb7(%40 : $<τ_0_0> Box (var τ_0_0) <(f: Expr, g: Expr)>): // Preds: bb0
%41 = project_box %40 : $<τ_0_0> Box (var τ_0_0) <(f: Expr, g: Expr)>, 0 /
%42 = tuple_element_addr %41 : $*(f: Expr, g: Expr), 0 // user: %43
%43 = load %42 : $*Expr // users: %46, %48
%44 = tuple_element_addr %41 : $*(f: Expr, g: Expr), 1 // user: %45
%45 = load %44 : $*Expr // users: %47, %48
debug_value %43 : $Expr, let, name "f" // id: %46
debug_value %45 : $Expr, let, name "g" // id: %47
br bb8(%43 : $Expr, %45 : $Expr) // id: %48
bb3: // Preds: bb2 bb1
%7 = integer_literal $Builtin.Int64, 1 // user: %8
%8 = struct $Int (%7 : $Builtin.Int64) // user: %9
br bb9(%8 : $Int) // id: %9
bb4(%10 : $<τ_0_0> Box (var τ_0_0) <(f: Expr)>): // Preds: bb0
%11 = project_box %10 : $<τ_0_0> Box (var τ_0_0) <(f: Expr)>, 0 // user: %
%12 = tuple_element_addr %11 : $*(f: Expr), 0 // user: %13
%13 = load %12 : $*Expr // users: %19, %17, %14, %18
debug_value %13 : $Expr, let, name "f" // id: %14
// function_ref Expr.count.getter
%15 = function_ref @_T05deriv4Expr33_13DFEAE3F6B32E19B1D7387847AB0BACLLO5count
retain_value %0 : $Expr // id: %16
retain_value %13 : $Expr // id: %17
%18 = apply %15(%13) : $@convention(method) (@guaranteed Expr) -> Int // user:
release_value %13 : $Expr // id: %19
release_value %0 : $Expr // id: %20
br bb9(%18 : $Int) // id: %21
bb8(%49 : $Expr, %50 : $Expr): // Preds: bb7 bb6 bb5
// function_ref Expr.count.getter
%51 = function_ref @_T05deriv4Expr33_13DFEAE3F6B32E19B1D7387847AB0BACLLO5count
retain_value %49 : $Expr // id: %52
retain_value %50 : $Expr // id: %53
%54 = apply %51(%49) : $@convention(method) (@guaranteed Expr) -> Int // user:
%55 = apply %51(%50) : $@convention(method) (@guaranteed Expr) -> Int // user:
%56 = struct_extract %54 : $Int, #Int._value // user: %59
%57 = struct_extract %55 : $Int, #Int._value // user: %59
%58 = integer_literal $Builtin.Int1, -1 // user: %59
%59 = builtin "sadd_with_overflow_Int64"(%56 : $Builtin.Int64, %57 : $Builtin.
%60 = tuple_extract %59 : $(Builtin.Int64, Builtin.Int1), 0 // user: %65
%61 = tuple_extract %59 : $(Builtin.Int64, Builtin.Int1), 1 // user: %62
cond_fail %61 : $Builtin.Int1 // id: %62
release_value %50 : $Expr // id: %63
release_value %49 : $Expr // id: %64
%65 = struct $Int (%60 : $Builtin.Int64) // user: %66
br bb9(%65 : $Int) // id: %66
bb9(%67 : $Int): // Preds: bb8 bb3 bb4
return %67 : $Int // id: %68