```
using Distributions
using QuadGK

println()

Wpl = 1.628e-3

#Teilaufgabe 1
println("Teilaufgabe 1")
muh_x1 = 30
sigma_x1 = 5
muh_x2 = 28.8e4
sigma_x2 = 2.64e4

c03 = -1/Wpl*60
c13 = 1/Wpl*33/5
c14 = -1/Wpl*24/5

beta3 = (c03+c13*muh_x1+muh_x2)/sqrt((c13*sigma_x1)^2 + sigma_x2^2)
beta4 = (c14*muh_x1+muh_x2)/sqrt((c14*sigma_x1)^2+sigma_x2^2)

println("c03: ", c03)
println("c13: ", c13)
println("c14: ", c14)

println("beta3: ", beta3)
println("beta4: ", beta4)


beta = min(beta3, beta4)

Pf = cdf(Normal(),beta)
println("Pf beträgt bei einem beta=",beta)
println(1-Pf)
println("--------------")
println()
#Teilaufgabe 2
println("Teilaufgabe 2")
muh_x1 = 30
sigma_x1 = 5

muh_x2 = 30
sigma_x2 = 5

muh_x3 = 28.8e4
sigma_x3 = 2.64e4

c03 = -1/Wpl*60
c13 = -1/Wpl*-12/5
c23 = -1/Wpl*9
c14 = -1/Wpl*-24/5

beta3 = (c03+c13*muh_x1+c23*muh_x2+muh_x3) / sqrt( (c13*sigma_x1)^2 +
(c23*sigma_x2)^2 + sigma_x3^2)
beta4 = (c14*muh_x1+muh_x3) / sqrt( (c14*sigma_x1)^2+ sigma_x3^2 )

println("c03: ", c03)
println("c13: ", c13)
println("c23: ", c23)
println("c14: ", c14)

println("beta3: ", beta3)
println("beta4: ", beta4)
```

```
beta = min(beta3, beta4)

Pf = cdf(Normal(),beta)
println("Pf beträgt bei einem beta=",beta)
println(1-Pf)

println("--------------")
println()
#Teilaufgabe 3
println("Teilaufgabe 3")


sigma_x1 = 5
muh_x1 = 25
sigma_x2 = 28.8e4
muh_x2 = 2.64e4
x02 = 19.9e4

a = 1/sigma_x1*pi/sqrt(6)
b = muh_x1 - 0.5772/a

sigma_u = sqrt(log(1+(sigma_x2/(muh_x2-x02))^2))

muh_u = log(sigma_x2-x02)-sigma_u^2/2
println()
println("a: ",a,"  b: ", b)
println("sigma_u :",sigma_u,"   muh_u: ",muh_u)
println()

c03 = -1/Wpl*60
c13 = -1/Wpl*12.3
c14 = 1/Wpl*-14.4

println("c03: ", c03)
println("c13: ", c13)
println("c14: ", c14)

# Funktion klein phi
function phi(y)
   return 1/sqrt(2*pi)*exp(-y^2/2)
end

#Funktion groß phi
function Gphi(y)
   return 1/sqrt(2*pi) * quadgk(y -> exp(-y^2/2), -Inf, Inf)[1]
end

# besetzten der Anfangsarrays und Zähler
y = [0.0; 0.0]
alpha = [0.0 0.0 ; 0.0 0.0]
beta = [0.0; 0.0]
p = 0


for i = 1:2000

    h3  = c03 + c13/a * log(log(Gphi(y[1])))+ c13*b + exp(sigma_u*y[2]+muh_u)  +
x02
    h4 = c14/a * log(log(Gphi(y[1]))) + c14*b + exp(sigma_u*y[2]+muh_u)  + x02

    H3 = [c13/a * phi(y[1]) / (Gphi(y[1]) * log(Gphi(y[1]))); sigma_u *
exp(sigma_u*y[2]+muh_u)]
    H4 = [c14/a * phi(y[1]) / (Gphi(y[1]) * log(Gphi(y[1]))); sigma_u *
exp(sigma_u*y[2]+muh_u)]
```

```
    global beta[1] = (h3 - H3' * y) / sqrt(H3' * H3)
    global beta[2] = (h4 - H4' * y) / sqrt(H4' * H4)

    global alpha[1,:] = -H3 / sqrt(H3' * H3)
    global alpha[2,:] = -H4 / sqrt(H4' * H4)

    global y = alpha * beta

    global p += 1
end

println("Für die Iteration erhalten wir nach ", p, " Iterationen, ein beta von:
", beta[1])
println()
Pf = cdf(Normal(),beta[1])
println("Pf beträgt bei einem beta=",beta[1], " : ", 1-Pf)
```