IS105

Gruppe 3
NoName

ICA01

Gruppemedlemmer:
Ali Al Musawi
Tor Borgen
Ann Margrethe Ly Pedersen
Brage Fosso
Adrian Lorentzen
Arne Bastian Wiik
Morten Schibbye

Alle kode henvisninger ligger i README.md

https://github.com/GB-Noname/is105-ica01

1.2.1 Binære tall

Konverter følgende desimaltall til 2-tallssystemet (binært tallsystem):

1 = 1

2 = 10

5 = 101

8 = 1000

16 = 0001 0000

256 = 0001 0000 0000

Vi brukte dele på 2 metoden.

Svaret får vi ved å sette opp resultatetene vertikalt og lese nedenifra og opp.

ex: 5= 1 2: 2= 0 1= 1

Konverter følgende binære tall til desimaltall (mest signifikante bit-en er til venstre):

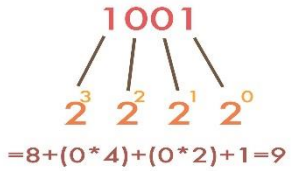
100= 4

1001= 9

1100110011= 819

Vi brukte andrepotensmetoden

ex:



1.2.2 Informasjonsmengde

Flere personer prøver å gjette et tresifret (3-bit) binært tall.

Hvor mye informasjon (i bits) har hver spiller fått?

N er lik	8		
Navn	M= tall de kan ikke kan utelukke	Log2(N/M)	Antall bits med informasjon
Lise	4	1	1
Per	3	1,41503749927884	1,41
Oscar	5	0,678071905112638	1
Louise	1	3	3

Lise = 1-bit

Per = 1,41-bit

Oskar = 1-bit

Louise = 3-bit

Metode brukt--> Log2 (N/M) N = utvalg, M = tall det kan være basert på den informasjonen man har. Altså vi krysser av hvor mange som er i utvalget (N) deretter deler vi det på tilfeller (M) og bruker Log2 på resultatet av N/M

1.2.3 Arbeid med Git

I komandolinjen skriver vi som følgende:

cd c:\github

cd gruppe

git clone "link"

cd is105-uke04

ls

(files are there)

1.2.4 Samarbeid i Git og Introduksjon i Golang

1) Hvilken fordeler og ulemper har en git-flow-modell med en hovedrepository?

Fordeler:

- Man får muligheten til å teste ut ulike muligheter uten å merge inn til Master branchen, dette gir rom for mer eksperimentering.
- Teamet kan jobbe mer produktivt.
- Skaper god oversikt grunnet et systematisk system.
- Parallell utvikling

Ulemper:

- En ulempe kan være at det blir for mye forkasting av branches, hvor du ikke kommer noen vei. Eks. En branch som utvikles i takt med master, men blir aldri merget inn i selve Master branch.
- Kan oppstå mergeconflicter

2) Finn ut hva heter objektfiler for de mest brukte plattformer (Unix/Linux, MS Windows, Mac OS X)! Hvorfor, etter deres mening, har disse plattformene så forskjellige objektfil-formater?

Ut ifra spørsmålet lese vi dette som at objektfil for windows blir .exe, siden når vi kjører "go build main.go" blir det eneste som opprettes en .exe fil. Det vil ut ifra denne logikken være .exec som er objektfiler for mac. Og for linux/unix er det flere "executables" deriblandt .deb og .tar.gz. Men en objektfil i samme forstand som windows og mac finnes egentlig ikke for unix baserte operativsystem.

Er det derimot objektfiler i generell forstand vil svaret være:

Unix/Linux: COFF/ELF(COFF har I stor grad blitt erstattet av ELF): https://en.wikipedia.org/wiki/COFF / https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

Mac OS X: Exex/ mach-o: https://en.wikipedia.org/wiki/Mach-O

MS Windows: exe / Ntddk.h. --> PE32 og PE64: https://msdn.microsoft.com/en-us/library/windows/desktop/aa364395(v=vs.85).aspx

De er forskjellige fordi operativsystemene er bygd opp forskjellig med ulik kildekode. Systemene er bygget på ulike måter, og det er da naturlig at objektfil-formatene er tilpasset systemene og støtter de strukturene som eksisterer i dem.

3)Hvilke forskjeller ser dere i forhold til programmeringsspråket Java?

Under denne øvelsen så ser vi at:

- Import statements er forskjellige fordi man deklarer kun pakken ikke biblioteket
- Små bokstaver i Golang vil si at funksjonen er (private) store = public. I motsetning til public og private statements i Java.
- Man skriver fmt.Println i Golang, mens i Java skriver man System.out.println
- GoLang kan ha en «naked return»

Generellt sett så har Golang raskere compile tid enn de fleste data språk som brukes per dags dato. Det er et mer fleksibelt programmeringsspråk og ikke så streng når det kommer til struktur I syntax.

I Golang er det ikke nødvendig å lukke setninger eller variabler. Det medfører at det blir mer praktisk orientert koding. I tillegg så har Golang en annen måte å deklarere hvilken type variablene er, hvor du skriver det etter parantesene i motsetning til Java, der skrives det før parantesene,

Det er en annen måte å deklarere hvilken type variablene er, i Java skriver man det før parantesene, og i Golang skriver man det etter.

Java er et klassebasert og objektorientert kode språk i motsetning til GoLang som ikke er det.

4)Hvilke viktige poeng illustrerer denne øvelsen når det gjelder bruk av et programmeringsmiljø på en plattform?

- Komprimering
- Quality of service (QoS)
- Estimere filstørrelser eller beregne sannsynligheter
- Statistikk
- Håndtere datamengder

Log kalkulasjonene kan brukes til mange av de overnevnte punktene. Som f.eks. sentrale deler av komprimeringsalgoritme, brukes i QoS-beregning for å kunne forutse og bedre kvalitet på datatrafikk. Det er også sentralt i statistikk, sannsynlighet og big data håndtering.

At mappestruktur kreves for å importere en annen fil, hvis begge filer ligger i samme mappe vil compileren slite med å finne filen som forsøkes importert.

5) Er det hensiktsmessig å legge inn denne filen i git repository? Begrunn svaret!

Ja, da kan alle klone repository og kjøre filen.

6)Hvordan skiller pakken log, som dere har implementer, seg fra andre pakker i go, som for eksempel, fmt?

- Log package lar deg skrive til alle standard enheter. Alt som støtter Input/Output typing interface
- Fmt implementerer formaterte input og output funksjoner som er like print og scan.

Svar på oppgave 5 og 6 ligger på github i folder log

Vi har valgt å kjøre en løsning med en skanner som tar input istedenfor å bruke parametre når vi kjører/builder programmet. Dette fordi det er i våre øyne en bedre løsning å ha et program som kan kjøres og kalkulere uten å måtte builde med ulike parametre hver gang. Dette resulterer i at vi effektivt har et program som alle kan bruke gjennom kommandolinje og kalkulere base 2 og 10 fra hvilke som helst tall.