INF-1100 Oppgave 2

Universitetet i Tromsø, Fakultet for Naturvitenskap og Teknologi, bachelor Fysikk.

Martin Soria Røvang.

23. September 2016.

1. Introduksjon

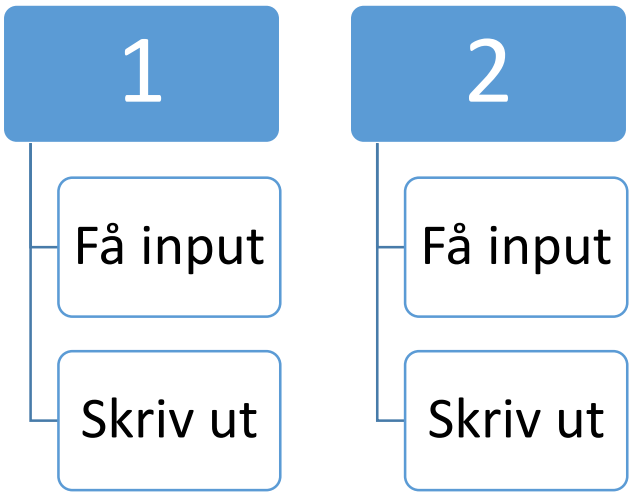
I oppgaven skulle jeg løse for en rekke funksjoner. Jeg skulle lage en vilkårlig trekant, sjekke om et tall hadde en primfaktor, sjekke en rekke tall fra A-B om de var odde- eller partall og bruke funksjonen om primfaktor på alle. Jeg skulle også lage en logaritme «kalkulator» med base 2 og en funksjon som reverserer en string.

2. Teknisk bakgrunn

Kunnskap om C der man forstår hvordan pc-en leser av funksjonene, hvordan logikken fungerer og forståelse av bit og hvordan dens rekke påvirker verdier. Her blir det for det meste brukt loops og funksjoner.

3. Oppbygging av programmet

Jeg lagde en meny for å bedre brukergrensesnittet slik at man ikke bare ble kjørt gjennom alle funksjonene og så kastet ut av programmet. Slik kunne man lettere bruke flere verdier for hver funksjon uten å streve så mye.



Eksempel på hvordan menyen er bygd opp.

Menyen kommer tilbake etter endt bruk av funksjoner, men for at det ikke skal være en evig loop så måtte det bli lagt inn en mulighet for å avslutte programmet ved å legge inn en mulig til å avslutte ved å taste in 0 i menyen.

Oppgave 1 er en pyramidefunksjon der det måtte bli brukt en «nested for loop» [1] for å få til oppgaven. Altså flere for loops inni loopen. Der legger funksjonen en stjerne som vokser med 1 for hver gang loopen går og detter ned 1 linje hver gang for at ikke alle stjernene skulle komme på samme linje.

Oppgave 2 gikk ut på å finne ut om et tall hadde primfaktor. Hvis primfaktortallet ble 0 ved å bli delt på 2 så mange ganger som mulig (bruk av modulus %) så kunne det ikke være primfaktor.

Dette var også slik hvis tallet og primfaktoren ble delt ved modulus og ble ULIKT 0 kunne ikke tallet ikke være primfaktor. Så den eneste muligheten for at det skulle være en primfaktor til tallet var at tallet modulus primfaktoren ble lik 0.

I oppgave 3 brukes funksjonen fra oppgave 2. Det var hovedgrunnen til at det ikke ble brukt brukerinput i funksjonen i oppgave 2. Slik kunne den funksjonen fritt bli kalt med ferdigstilte verdier.

Her må det lages en funksjon som skal finne ut om tallene var enten partall eller oddetall. Og så hente primfaktorfunksjonen og legge in sin printf med de nye verdiene som ble lagt inn i brukerinputen fra denne funksjonen.

I oppgave 4 lages det en logaritmekalkulator med base 2 og dette passer bra siden bits er i 2^n og dermed brukes bitshifting [2] for å få til dette. Først lages en «for loop» som bitshiftet verdien i input mot høyre med en. Legger så til 1 på en annen variabel som i dette tilfelle var «j» (som tilslutt skal bli svaret) helt til ved bitshifting at input variablen ble en også printa programmet ut variabelen som ble plusset opp igjennom loopen altså j.

Oppgave 5 [3] Forsøkte først med brukerinput, men fikk det ikke helt til å fungere med setninger. Hver gang det ble skrevet inn setninger, lagde det problemer med menyen. Å ha et ferdigstilt ord som blir reversert ble til slutt den beste løsningen på problemet.

4. Diskusjon

Det var problematisk å få funksjonen i oppgave 2 til å fungere på en ordentlig måte i oppgave 3 fordi en ny tekst fra en annen funksjon (odd/even) skulle komme med teksten til primfaktorfunksjonen. La først inn scanf i funksjonen i oppgave 2, men oppdaget at det ikke var mulig i oppgave 3 fordi det ble en dobbel brukerinput når det ble kallet på primfaktorfunksjonen.

Her var det også veldig viktig å ikke ha newline i print fra mynumberfunksjonen fordi det skulle printes ut resterende av setningen fra en annen funksjon.

Oppgave 4 og 5 var de som gav mye hodebry. Bitshifting der det flytter bit x antall plasser mot høyre eller venstre (i dette tilfelle mot høyre) f.eks 10000 >> 1 blir 1000 som da er fra 16 (2^4) og blir da 8 (2^3) så ved å bitshifte helt til 1 og for hver runde i loopen så legges det på 1 (Starter også med 1). Da finner man nærmeste n'te heltall og dermed kan man lage logaritmefunksjon med base 2.

Ved å ha brukerinput i oppgave 5 ble først et ord reversert og så kom menyen tilbake der det andre ordet reversert ble skrevet ut. Dette gav mye problemer og det krevde mye tid å prøve å rette det. Til slutt ble det bestemt å fjerne brukerinput og legge inn et ferdigstilt ord. Ved string reverse ble det brukt en «array» som er en slags «hylle» med informasjon som kan gjøres så stor som man vil. Ved bruk av «array» i denne sammenhengen plasseres ord i hver hylle så man kan skifte posisjonen på ordene i hylla og printe ut i rekkefølge. Så hvis «HEI» er på hylleplass 0 1 2 (arrays starter på 0) kan man reversere ordet ved at plassen til I blir til 0, plassen til E blir til 1 og plassen til H blir til 2 og dermed printe ut stringen.

Dette ble gjort med en while loop ved å plusse opp bokstavene i hylla fra start til slutt i ordet og trekke fra posisjonen fra slutt til start. Det måtte også bli brukt en «string terminator» fordi hylla skal ha 100 plasser, noe man ikke hadde trengt fordi det ikke ble brukt brukerinput. Da kunne man fritt velge helt opp til hundre bokstaver. String terminatoren stopper stringen slik at programmet ikke skal printe ut alle 100 som er i «arrayen». Når setningen man la ut var ferdig printet, ligger det alt mulig av minneplasseringer som ikke har blitt definert, slik at det går fra f.eks 1.2.3…..5 til 1218438194164 osv så '\0' stanser stringen på siste bokstav.

5. Konklusjon

Denne oppgaven var en utfordring, det krevde at man leste seg opp fra bøker og nettsider. Med tanke på at det ikke ble brukt brukerinput i oppgave 5 fikk jeg ikke gjort alt jeg ville gjøre. I sin helhet var det godt å ha forskjellige funksjoner å jobbe med og bruke funksjoner i funksjonene.

6. Referanser

1) Introduction to computing systems, second edition, Yale N. Patt, Sanjay J. Patel Side 357.

2) Introduction to computing systems, second edition, Yale N. Patt, Sanjay J. Patel Side 319.

3) http://www.c4learn.com/c-programs/c-program-reverse-string-without-using.html 14.09.2016