UiT Norges arktiske universitet

Hjemmeeksamen i:	INF-1100 Innføring i programmering og datamaskiners virkemåte
Innleveringsfrist (dato og tidspunkt):	Onsdag 27. november kl 14:00
Kursansvarlig:	Førsteamanusensis Einar Holsbø
Antall sider:	6 sider inkludert fremside
Support:	På innleveringsdagen kan du ringe 776 20 880 for support. Dersom du opplever lang ventetid og innleveringfristen går ut, kan du sende en kopi av besvarelsen din til **eksamen@uit.no** mens du venter på svar. **Ikke forlat køen.**
Vekting av spørsmål, eller annen informasjon:	
Viktig informasjon om sitering og plagiering:	1. *Dette er en individuell eksamen som skal besvares uten samarbeid med andre.* 2. *Alle kilder er tillatte (egne notater, pdfer fra forelesningene, lærebok, nettsider, KI-verktøy, etc).* 3. Alle eksamener som leveres i WISEflow blir automatisk sjekket for plagiat. *Det er ikke tillatt å gjengi innhold fra KI-verktøy, medstudenter, nettressurser, litteratur eller andre kilder uten referanser. Husk at det er heller ikke tillatt å kopiere fra egne tidligere innleverte besvarelser uten å referere. Det vises noen eksempler på hvordan man kan referere til kildebruk i selve eksamensteksten.*

INF-1100 - Hjemmeeksamen (Høst, 2024)

I denne oppgaven skal du skrive en enkel interpreter til et lite stack-basert språk https://en.wikipedia.org/wiki/Stack-oriented_programming. Et stack-basert språk fungerer slik at alle språkets operasjoner manipulerer en stack. For eksempel vil en addisjon poppe de to øverste verdiene i stacken, `a` og `b`, og pushe summen deres `a+b`:

```
[  a  ]       add
[  b  ]    --------->   [ a+b ]
[  c  ]                [  c  ]
før add                etter add
```

Språket som du skal lage interpreter til er et hjemmelaget språk kalt `8inf`. Alle operasjoner støttet i `8inf` er definert i fila `README.md` og alle eksisterende `8inf` script i hele verden ligger i mappa `scripts/`. Det ingen poeng I å prøve å google språket for å forstå det: det finnes bare i denne oppgaven. Du trenger ikke forstå hvordan man skriver et program i `8inf`. Det du trenger å forstå er hva hver operasjon isolert sett gjør med stacken og evt. programmets øvrige tilstand.

Merk at det gis ikke utsettelse på eksamen. Start så tidlig som mulig.

Oppgavebeskrivelse

Din oppgave er å implementere en *interpreter* for et lite stack-orientert språk. For å hjelpe deg på vei får du utdelt kode som leser inn script på `.8f` format. Programmet fjerner kommentarer og legger hver separate instruksjon i et array av strenger. Det du skal gjøre er å implementere en runtime stack som kan brukes i kjøringen av programmet og du skal skrive kode som implementerer de forskjellige instruksjonene beskrevet i `README.md`. Instruksjonen `.cjump` krever i tillegg at du holder styr på programflyt, da denne tillater hopp fremover og bakover i programmet.

Prekode

Nyeste versjon av prekode ligger på https://github.com/3inar/inf1100_exam_24 – hvis det oppdages alvorlige bugs i prekode vil oppdateringer publiseres der.

Prekoden har følgende struktur:

- `8inf.c` Er starten på en interpreter: `./8inf scripts/hello_world.8f` vil lese inn scriptet i fila `hello_world.8f` vha funksjonen `load_program` (se nedenfor). Du må fylle inn kode som kjører det innleste scriptet.
- `Makefile` er en makefile som bygger programmet `8inf` beskrevet i `8inf.c`. Du blir antakelig nødt til å legge til noen filer her.
- `inc/` holder headerfiler. Den inneholder kun `lexer_preprocessor.h` som deklarerer funksjonen `load_program` pluss noen hjelpefunksjoner. Du må antakelig legge til flere filer her.
- `scripts/` inneholder en håndfull `.8f` script med varierende kompleksitet.

- `src/` holder implementeringer av funksjoner deklarert i `inc/`. Inneholder kun `lexer_preprocessor.c` som implementerer `load_program` og en håndfull med hjelpefunksjoner til `load_program`. Du må antakelig legge til flere filer.

Funksjonen `load_program` gjør all nødvendig preprossesering av gyldige `.8f` script: den fjerner kommentarer og bryter opp koden i kun instruksjoner. Et script som ser ut som

```
(this is a multi-line
 comment)
~hello, world!~ .print (this is an inline comment) .newline
```

vil leses inn i et dynamisk allokert array av strenger som dette:

```
program[0] -> "~hello, world!~"
program[1] -> ".print"
program[2] -> ".newline"
program[3] -> NULL
```

Kjøring av `hello_world.8f` vil se ut som dette:

```
user@host pre % ./8inf scripts/hello_world.8f
Hello, world!
user@host pre %
```

Rapport

Du skal levere en rapport om løsningen din. Rapporten har en begrensning på 3500 ord. Du kan skrive på norsk, nynorsk, eller engelsk.

Rapporten skal ha følgende struktur:

- Introduksjon
  - Fortell kort hva som er i denne rapporten og hvorfor
- Teknisk bakgrunn
  - Kort bakgrunnsinformasjon om valgt datastruktur og liknende.
- Design og implementering
  - Hvordan henger systemet ditt sammen helt overordnet?
  - Hva er viktige detaljer ved implementeringen av systemet?
- Eksperimenter
  - Beskriv dine eksperimenter for å evaluere aspekter ved løsningen.
- Resultater
  - Gjorde du noen målinger som del av eksperimentene? Vis dem her.
- Diskusjon
  - Hva er fordeler og ulemper med hvordan du valgte å løse oppgaven?
  - Hvordan tolker du resultatene fra dine eksperimenter?
- Konklusjon
  - Gi et kort sammendrag av hva du gjorde og hva du har lært.

Merk at skjermdump av kode + to setninger er ikke en tilfredsstillende rapport. Rapporten skal inneholde dine beskrivelser av løsningen din. Koden er allerede dokumentert i koden.

Referanser, plagiat, etc.

Det skal vises tydelig hvor man henter kunnskap. Dette gjøres ved sitering av kilder slik som her [1]. Hvis man kopierer noe direkte fra en kilde (direkte sitat), så må dette også tydelig merkes slik som her:

> For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. [2]

Merk at tallene [1] og [2] refererer til linjer i seksjonen "Referanser" nederst i dette dokumentet: der står det nøyaktig hvor jeg har informasjonen min fra.

Hvis du bruker kode fra andre steder enn prekoden må dette også merkes tydelig; dette gjør man som kommentarer til koden.

```
/* denne funksjonen har jeg hentet fra www.eksempel.no/exp-funksjon */
float exp(float x) {
 ...
}
```

Poenget er at det skal være helt tydelig hva som er ditt eget arbeid og hva du ikke har gjort selv. Det er lov å bruke kode fra din egen løsning av obligatorisk eller frivillig oppgave til dette kurset i år. Dette skal også merkes, f.eks.:

```
/* denne funksjonen skrev jeg selv til løsningen av arbeidskravet i INF1100,
   høst 24. */
int factorial(int n) {
 ...
}
```

AI-verktøyer som copilot og ChatGPT må også siteres som bruk av eksterne kilder. En tommelfinger-regel er at å bruke kode som disse verktøyene har skrevet er omtrent som å bruke kode funnet i en annen kilde: du må være tydelig på at det er det du gjør. Bruk av AI til skriving av kode merkes altså på lignende måte, den store forskjellen er at man skal legge ved chat-loggen:

```
// denne funksjonen ble skrevet med chat.uit.no; loggen er vedlagt som fil
// chatlog_1.txt
void swap(int *a, int *b) {
 ...
}

// Denne funksjonen har jeg debugget med hjelp fra chat.uit.no; loggen er
// vedlagt som fil chatlog_2.txt
void gc_free(...) {
```

```
  ...
}
```

Chatlog legges ved som en tekstfil, ikke et bilde.

Hvis man bruker en AI til noe som ikke direkte har med koden å gjøre, f.eks. språkvask av rapporten og denslags, nevnes det i rapporten, og chat log legges ved.

Hvis man ikke siterer kilder i rapport eller kode blir dette tolket som plagiat, som er en form for fusk på eksamen. Fusk på eksamen har store akademiske konsekvenser nærmere forklart her: https://uit.no/eksamen

Husk at dette er en eksamen. Du må selv opparbeide deg en forståelse av oppgaven og jobbe selvstendig for å finne løsningen på oppgaven. Hvis du har generelle spørsmål om mulige løsninger så kan du diskutere med hjelpelærerne. De vil også kunne assistere deg hvis det er veldig spesifikke implementeringsdetaljer du lurer på. Spør heller en gang for mye enn en gang for lite. Men husk å aldri kopiere kode eller rapport.

Innlevering

Eksamen skal leveres i to deler: Rapporten leveres som hoveddokument i `.pdf`. Koden leveres som et vedlegg i `.zip` med følgende struktur:

- `inf1100.zip`
  - `code/`
    - `8inf.c`
    - `Makefile`
    - `inc/`
    - `scripts/`
    - `src/`
  - `report.pdf`

Merk at vi vil ha rapporten i `.zip` fila også. Har du laget flere filer som en del av løsningen må naturligvis disse også være med. Legg ikke ved kompilerte programmer, kun kode.

Det er viktig at man ikke skriver hverken navn eller kandidatnummer i kode eller rapport, da dette blir tatt hånd om i Wiseflow.

Vurderingsskjema

Hvis sensor kan svare "ja" på 9 av 12 spørsmål nedenfor vil du få bestått på denne eksamenen:

- Er runtime stack implementert?
- Fungerer `hello_world.8f` som det skal?
- Fungerer `integer_ops.8f`?

- Fungerer `comparisons_and_logicals.8f`?
- Fungerer `loop.8f`?
- Er det gjort gode kommentarer og valgt beskrivende variabelnavn?
- Er der gjort et eller flere eksperimenter med tilhørende målinger?
- Følger rapporten strukturen som er gitt i eksamenstekst, skrevet på et akseptabelt informativt nivå?
- Er eksperimenter beskrevet i rapporten, inklusive målinger?
- Beskriver rapporten hvordan instruksjonene implementeres i praksis?
- Beskriver rapporten mekanismen bak `.cjump`? Eller hvis `.cjump` ikke er implementert rett slik at `loop.8f` ikke fungerer: forklarer rapporten hvordan den i prinsippet kan implementeres?
- Drøfter rapporten fordeler og ulemper med løsningen?

Referanser

- [1] Kernighan, B. W., & Ritchie, D. M. (2002). The C programming language.
- [2] Dijkstra, E. W. (1968). Letters to the editor: go to statement considered harmful. Communications of the ACM, 11(3), 147-148.