

Koi - Lab 3

Anton Söderlund, DT501G

Part A

Makefilen bygger `lex.yy.c` med ett flex kommando och `example.tab.c`, `example.tab.h` från `example.y` med bison kommandot. Den kompilar sedan till en exekveringsfil med gcc. gcc kommandet är: `gcc $(CFLAGS) -o example example.tab.o lex.yy.o` och den nämner inte `.c` eller `.h` filerna, men make kan härleda till dem genom namnen på filerna, dvs om om `.o` och `.c` filer har samma namn behöver man inte skriva ut hela receptet som `gcc -c main.c -o main.o`

Det är divisionoperatorn som saknas, och den är tillagt i exemplet.

Part B&C

I `parser.y` finns grammatiken som jag utgick ifrån föreläsning 5 anteckningarna. Jag strukturerade det liknande som i lab 2 att uttryck består står av:

```
expr 'operator' term
| term
```

En term består av:

```
term 'operator' factor
| factor
```

Faktor består av:

```
faktor 'operator' exponent
| exponent
```

Exponent består av:

```
(' expr ')
| id
| NUM
```

Association och prioritet ordnades enligt hur C rankar dem¹.

I `lexer.c` ändrade jag variabeln `token_value` till `yyval`, vilket är namnet på variabeln som bison skapar för `token_value` i `parser.tab.h`. Det behövs för att läsa in korrekt till \$\$-variablerna i grammatiken, så att alla beräkningar kan göras direkt i `parser.y`.

De enda shift/reduce-konflikter jag hade var ifrån `?:`-operatorn, vilket jag kunde lösa genom att antingen specificera både `'?'` och `':'` som operatorer, eller att deklarerar `%prec '?'` i den produktionen, vilket betyder att hela produktionen ska ha prioriteteten som `'?'` operatorn.

Part D

Jag ändrade alla `.c`, `.h`-filer till `.cpp`, `.hpp` och kompilarar med g++ istället för gcc, och ändrade lite inkluderingar, och programmet fungerade likadant utan error förutom i `init.cpp` med ett litet error vilket löstes med `(char*) cast`.

¹ https://en.wikipedia.org/wiki/Order_of_operations#Programming_languages