Lab 1 - KOI

Anton Söderlund

Part A

Debuggern säger SIGSEGV, segmentation fault. Vi försöker använda minnet inkorrekt någonstans. Genom att printa varje pekares address ser vi att den senaste som kommer in pekar på 0, dvs nullpekaren. Att läsa eller skriva till nullpekaren ger ett seg fault direkt.

Med en backtrace via gdb kan vi se vilken funktion det är som skickar nullpekaren, rad 247 I abcissa.cpp
int kyss = anslutningspropp(0, kyrkoherde)
Vi löser det genom att skapa och skicka in en deklarerad pekare
*int *dubbelnugge = new int;*
**dubbelnugge = 2;*

Part B
Vilka finns?
Lexikalanalys I lexer.c. Den läser från inputen och tar bort whitespace, ifall det finns. Den tittar sen efter om char-värdet den läst in är ett nummer, bokstav och konverterar dem till tokens.
Syntaxanalys I parser.c. Där beskrivs hur inputen ska hanteras.

Vilka saknas?
Semantisk analys, mellankodsgenerering, maskinoberoende optimering, kodgenering, maskinberoende optimering.

Beräkna inputen
Postfixnotation, eller omvänd polsk notation som det också heter används ofta i datorer och miniräknare genom att pusha värdena och operatorerna till en stack. Det kan göras i emitter.c.

Part C
Att definera den rekursiva funktionen på exakt samma sätt enligt,
*#define FACT(n) ((n==0) ? 1: n*FACT(n-1))*
Denna definitionen kommer inte fungera eftersom preprocessorn ersätter alla FACT(n) skrivet I programmet med funktionen som vi definerat, det betyder att efter preprocessorn är färdig så kommer den här koden:

```
int main() {
FACT(6);
}
```

ersättas till,

```
int main() {

      if(6 == 0)
            return 1;
      else{
            return n*FACT(6-1);
      }
}
```

och här är preprocessorn färdig, och gcc kommer då försöka kompilera denna kod men eftersom FACT(n) inte är definerad någonstans I koden, så kommer vi få "undefined reference to FACT(n)". Vi ser även om vi endast kör preprocessorn med flaggan -E att den endast byter ut FACT(n) två gånger, dvs de två gånger den faktiskt finns I macrot.

Hur kan man modifiera preprocessorn sådan att macrot fungerar?
C-preprocessorn stödjer per definition inte rekursiva anrop, så att det skulle kräva en modifiering av preprocessorn själv. Det finns workarounds genom "variadric macros", dvs man kan definera macrot med att utföra något uttryck på alla argument. T.ex.

```
#define DEBUG(...)  printString (__VA_ARGS__)
```
så kan man sen kalla funktionen med,
DEBUG("hej"), DEBUG("hej","hejhej"),
DEBUG("hej","hejhej","hejhejhej") etc.

Jag hittade ett exempel på ett rekursivt macro här.