Zadání úlohy je poměrně jednoduché a umožňuje i přímý přístup k problému a přímou implementaci. My však zvolíme trochu vědečtější přístup, jak se na správného důstojníka sluší a patří.

Náš program tedy rozdělíme na dvě části. Nejprve si připravíme takzvaný *lexikální analyzátor*, který bude rozpoznávat lexémy a poté vyrobíme *syntaktický analyzátor*, nebo též parser, který bude tyto lexémy dále zpracovávat. Lexikální jednotky jsou oddělené mezerou a na každém řádku, krom prvního, jsou tři takovéto jednotky.

K výrobě lexikálního analyzátoru je potřeba přesně specifikovat, jak takový lexém vypadá. V našem případě máme čísla a operátory. Čísla se skládají z číslic 0 až 9 a mohou obsahovat znaménko mínus. Operátory jsou přesně definované v zadání. Abychom si zjednodušili práci, použijeme generátor lexikálních analyzátorů – Flex – a analyzátor popíšeme v jeho jazyce:

```
D            [0-9-]
%%
{D}+         {
                 yylval = strtoul(yytext, NULL, 10);
                 return NUM;
             }
"<="         return OP_LE;
">="         return OP_GE;
"=="         return OP_EQ;
"!="         return OP_NE;
"<"          return '<';
">"          return '>';
```

Když máme lexikální analýzu připravenou, zbývá nám dát lexémům smysl, neboli syntaxi. Jinými slovy, musíme popsat, jak se z lexémů skládají jednotlivé věty (v našem případě řádky). Opět k tomu můžeme použít nějaký z vhodných nástrojů pro generovaní syntaktických analyzátorů. My sáhneme po nástroji Bison, kterému řekneme, že náš vstup obsahuje na prvním řádku číslo a poté následují řádky se zadáním. Na každém takovém řádku jsou dvě čísla oddělená operátorem. Řádek můžeme po přečtení ihned vyhodnotit a vypsat výsledek. Popis by vypadal zhruba takto:

```
program     : NUM lines
            ;

lines       : lines expression
            | expression
            ;

expression  : NUM OP_EQ NUM { printf("%s\n", $1 == $3 ? "TRUE" : "FALSE");
}
            | NUM OP_NE NUM { printf("%s\n", $1 != $3 ? "TRUE" : "FALSE");
}
            | NUM OP_LE NUM { printf("%s\n", $1 <= $3 ? "TRUE" : "FALSE");
}
            | NUM OP_GE NUM { printf("%s\n", $1 >= $3 ? "TRUE" : "FALSE");
}
            | NUM '<' NUM   { printf("%s\n", $1  < $3 ? "TRUE" : "FALSE");
}
            | NUM '>' NUM   { printf("%s\n", $1  > $3 ? "TRUE" : "FALSE");
}
            ;
```