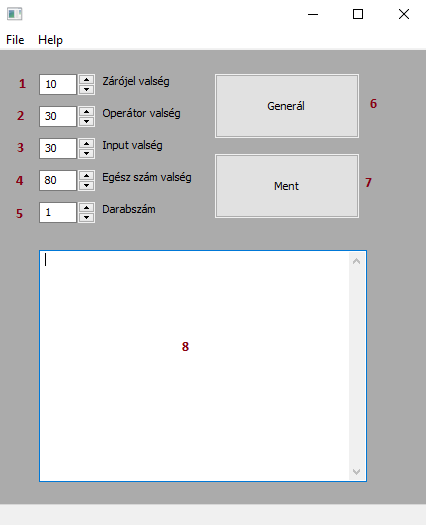
**Használati útmutató**



1-3. A különböző műveleti jelek valószínűségei százalékban megadva. Ezek 0 és 99 között mozoghatnak, mivel a programban az std::rand() működik. A következő művelet akkor lesz az adott típusú, ha egy rand()%100 kisebb, mint az adott érték. Az elágazások a fent látható sorrendben követik egymást, ezért érdemes pl. a zárójel valószínűségét alacsonyan hagyni.

4. Ha a rand()%100 az egész szám valószínűsége felett van, akkor a generált konstans értéke egy (rand()%100/rand()%100) típusú racionális tört lesz. Ez egy elég nagy tartományt lefed, de viszonylag ronda törtek is kijöhetnek. Ha ezt a valséget 99-re állítjátok, akkor csak 1-100 egész számok fognak kijönni.

5. Hány sornyi kódot generáljon a program. Érdemes egyszerre több (legalább 3) sort generálni, mert nem minden sor kifejezetten érdekes.

6. Generálás. Ezen gondolom nincs mit magyarázni.

7. A fájlba mentés egy save típusú dialógussal működik, aminek be van állítva alapértelmezetten a „.txt” kiterjesztés, ezért azt nem kell a végére írni. (Ne lepődjetek meg rajta, ha nem láttok más fájlokat a dialógusban, ez a kiterjesztés beállításának mellékhatása, és nincs kedvem kijavítani :P )

8. Output. Ide kerülnek a generált kódok. ’Nuff said.

(Megjegyzés: Bár ilyet csak egyszer láttam, nekem volt, hogy a program konstansnak egy „inf”-et generált. Ezt azóta sem tudtam reprodukálni, de ha valaki rájön, hogy ez miért van, akkor kérem, hogy szóljon!)

**Hasznos tanácsok**

Igazából csak két olyan dolog van, ami hasznos lehet a házihoz/beadandóhoz, de az egyik nem túl egyszerű, ezért inkább ezt venném előre.

**Tokenek írása:**

A token fogalma nem egy bonyolult dolog, hiszen mindenki használta már őket. Az int, bool, string, 13, void, <<, True, ezek mind tokenek. A tokenek a programkód elemi részei, amiket nem lehet tovább bontani, bár az átlag programozó ezt a szót inkább a „token kifejezésre” érti (Egy olyan dologra, amivel könnyedén meg lehet határozni valaminek a típusát.), ezért igazából a classokat és a structokat is ide lehet sorolni.

A tokenek deklarálása így néz ki:

enum nodetype {constant,input,operation,parentheses};

(Ezt a globális térbe kell írni)

Ebből lesz egy nodetype típusunk, ami azokat az értékeket veheti föl, hogy constant,input,operation és parentheses. (Az enum-al csak ilyen jellegű diszkrét értékeket tudunk megadni. Ha egy halmazt szeretnénk megadni, pl. valós számok, azt máshogy kell, de erre nem is lesz szükségetek. )

Így létrehoztunk egy teljesen új elemi típust. Pl.

nodetype Mynodetype = constant;

Ez egy teljesen valid kód, amiben létrehoztunk egy új nodetype típusú változót.

Ez azért lesz jó, mert így nem kell azzal szórakozni, hogy minden műveletet számokkal reprezentálunk. (Bár ezt is meg tudjuk tenni enum-al. Pl:

enum nodetype {constant = 0,input = 1,operation = 2,parentheses = 3};

Ez minden értékhez egy egész számot is rendel, amiket szabadon fel lehet cserélni. Pl:  
int a = constant;

Itt az a értéke 2 lesz. Ha ezt nehéz megérteni, akkor gondoljatok a bool-ra. Az nagyjából így nézne ki, ha nekünk kéne deklarálni:

enum bool {false = 0,true = 1};

)

Ha ezt megcsináljátok, akkor nem kell azzal bajlódni, hogy akkor most a   
„mytype ==2” az műveletet vagy konstanst jelentett, hanem egyszerűen odaírhatjátok, hogy mytype == constant.

Ez önmagában nem jelent sok segítséget a beadandóhoz, de sok hibát el lehet vele kerülni, és sok szempontból jelentősen kényelmesebb. (Arról nem is beszélve, hogy professzionálisabb.)

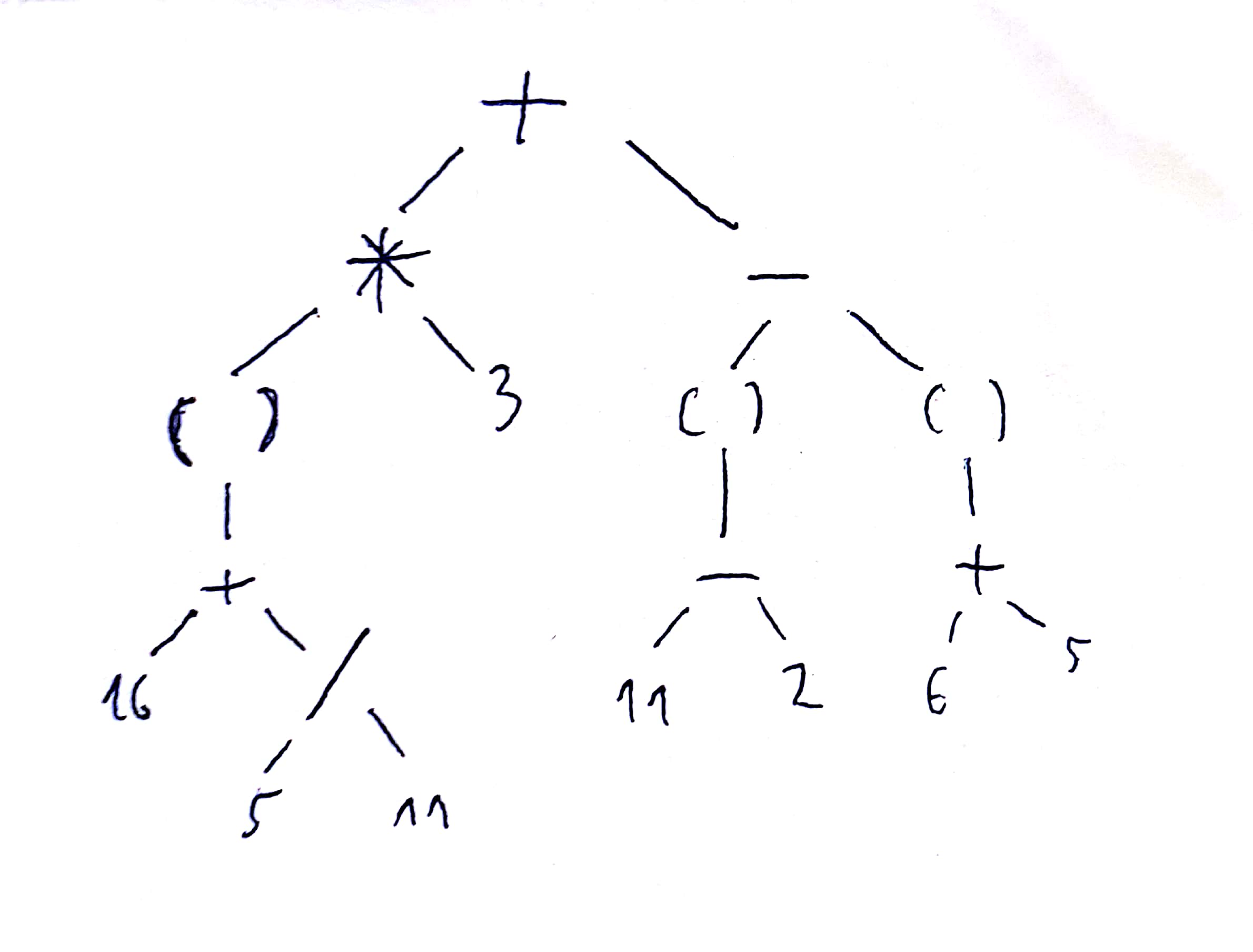
A következő téma az sokkal nagyobb segítség lesz, kifejezetten a parser-höz, de éppen ezért semmi konkrétat nem mondhatok.

**Műveleti fák:**

Ezzel kapcsolatban nem fogok tudni túl sok konkrét segítséget adni, de minden, amire szükségetek lesz a megértéséhez, már fent van a jegyzetben.  
Vegyük a következő kifejezést:

(16+5/11)\*3+(11-2)-(6+5)

Ennek a kifejezésnek a műveleti fája:



Ha nem hiszitek el, építsétek fel belőle a kifejezést józan paraszti ésszel.  
Itt a kifejezés három különböző jelölésrendszerben:

Prefix/lengyel: + \* (+ 16 / 5 11) 3 - (- 11 2) (+ 6 5)  
Infix: (16+5/11)\*3+(11-2)-(6+5)  
Postfix: (16 5 11 / +) 3 \* (11 2 -) (6 5 +) - +

Vonjuk le a logikus következtetéseket (és figyeljük meg, hogy valahol biztosan elrontottam valamit, és hülyét csináltam magamból ☺ )

(Hint: Ezt könnyen meg lehet csinálni kézzel. Csak toll és papír kell hozzá.)