MintaZH – a gyakorlati feladatok megoldásának vázlata

6. Feladat –Mit ír le az alábbi reguláris kifejezés? Rövid, egy mondatos magyarázat mellett adjon egy példát is elfogadott szövegre! Fejtse ki minden csoportosítás (capture group) jelentését külön-külön! (10p)

^((?:special)?entity) ([0-9a-zA-Z]+)(extends [0-9a-zA-Z]+)?$

MEGOLDÁS (példa): *special entity MyEntity extends MyBaseEntity*

Capture group 1: opcionális „special” kulcsszó + entity kulcsszó

Capture group 2: identifier szabály _ nélkül, az entity neve

Capture group 3: „extends” kulcsszó után ős entitás neve, opcionális

7. Feladat – Tekintsük az alábbi attribútum nyelvtant és programkódot! Adja meg a programkód szintaxisfáját és számolja ki a csúcsokhoz tartozó attribútumok értékét! Karikázza be azokat a csúcsokat a fában, ahol típushiba található! (10 pont)

Attribútum nyelvtan:

```
A → T x = E
E.expType = T.type
T.expType = any

E → E+C
E[1].op = GetOperator(+,E[2].type,C.type)
E[1].type = E[1].op.type
E[2].expType = E[1].op.expType
C.expType = E[1].op.expType

E → C
E.type = C.type
C.expType = E.expType
```

```
C → 1
C.type = int

C → "a"
C.type = string

T → int
T.type = int

T → string
T.type = string
```

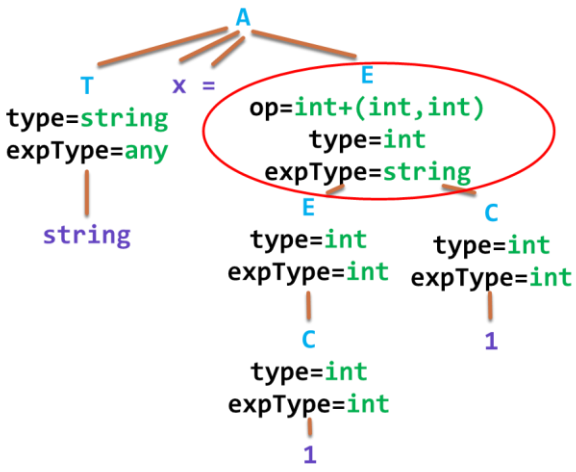
A *GetOperator* függvény az alábbi operátorokat képes feloldani:

`int+(int,int)` – két egész szám összeadása
`string+(string,string)` – két karakterlánc összefűzése

Programkód:

```
string x = 1+1
```

Megoldás:



Magyarázat:

Az attribútumokat abban a sorrendben célszerű kiértékelni, ahogy a lusta kiértékelés lehetővé teszi, vagyis, amikor az attribútum értékét leíró kifejezés minden eleme rendelkezik értékkel. Ennek megfelelően egy jó kiértékelési sorrend lehet a következő:

```
A/T.expType=any      (szabály: A → T x = E)
A/T.type=string      (szabály: T → string)
A/E/C.type=int      (szabály: C → 1)
A/E/E/C.type=int      (szabály: C → 1)
A/E/E.type=int      (szabály: E → C)
```

Mostmár minden adott az operátor feloldásához:

```
A/E.op = int+(int,int)      (szabály: E → E+C)
```

Az operátor eredményének típusa int, így:

```
A/E.type=int      (szabály: E → E+C)
```

Az operátor mindkét operandusának (paraméterének) int-nek kell lennie:

```
A/E/E.expType=int      (szabály: E → E+C)
A/E/C.expType=int      (szabály: E → E+C)
```

A további attribútumok pedig:

```
A/E/E/C.expType=int      (szabály: E → C)
```

Az értékadás jobb oldalának elvárt típusa:

```
A/E.expType=string      (szabály: A → T x = E)
```

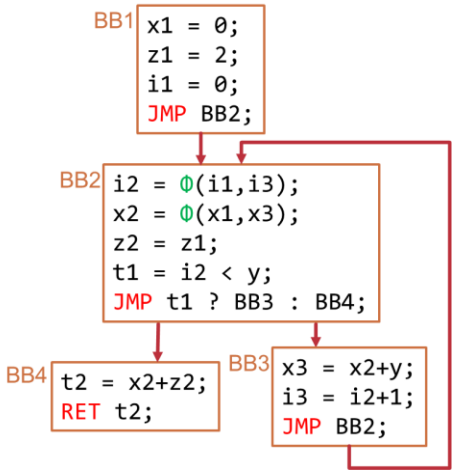
A tanultak alapján az expType az elvárt típus, type pedig a tényleges típus. A tényleges típusnak implicit konvertálhatónak kell lennie az elvárt típusra. Ez az A/E csúcsot kivéve minden más csúcsra teljesül, így típushiba az A/E csúcsnál lép fel.

8. Feladat – Tekintsük az alábbi C# kódot! Alakítsa át a programkódot szöveges SSA formára! Az alapblokkokat az alábbi sorrendben értékelje ki: ciklus előtti kód, ciklusfeltétel, ciklus utáni kód, ciklustörzs. Ügyeljen arra, hogy a változók számozása a kiértékelés sorrendjében sorfolytonos legyen! (10 pont)

Programkód:

```
int foo(int y)
{
  int x = 0;
  int z = 2;
  for (int i = 0; i < y; ++i)
  {
    x += y;
  }
  return x + z;
}
```

Megoldás:



9. Feladat – Szemléltesse a tanult optimalizációs technikákat az alábbi kódrészlet optimalizálásán keresztül! Minden esetben nevezze is meg a felhasznált technikát! Feltételezze, hogy a kódrészlet további részében az b, d, e változók értékére van szükségünk. (10p)

```
…
a = fun();
b = a;
d = b * 2 + c + 5;
e = c + 5;
f = a * 2 + e;
print(e*d);
if (a == 0)
{
   b = e*d;
   print(b + f);
}
else
{
```

```
   print(a+c);
   b = e*d;
}
…
```

Megoldás:

```
a = fun();
b = a;                  // dead code
d = a * 2 + c + 5;      // copy propagation
e = c + 5;
print(e*d);
b = e*d;                // code factoring
if (a == 0)
{
   f = d;               // copy prop + common subexpr. majd code sinking
   print(b+f);
}
else
{
   print(a+c);
}
```