I117（15）文字列（5）数式評価

知念

北陸先端科学技術大学院大学 情報科学研究科
School of Information Science,
Japan Advanced Institute of Science and Technology

数式評価

ちょっとした計算をするためにプログラム中で入力された式を評価する

```
1+2         =>  3
1+2*3       =>  7
1+2*3+4     =>  11
```

詳細はオートマトンや言語処理系の講義へ
ここでは処理を実感するために紹介

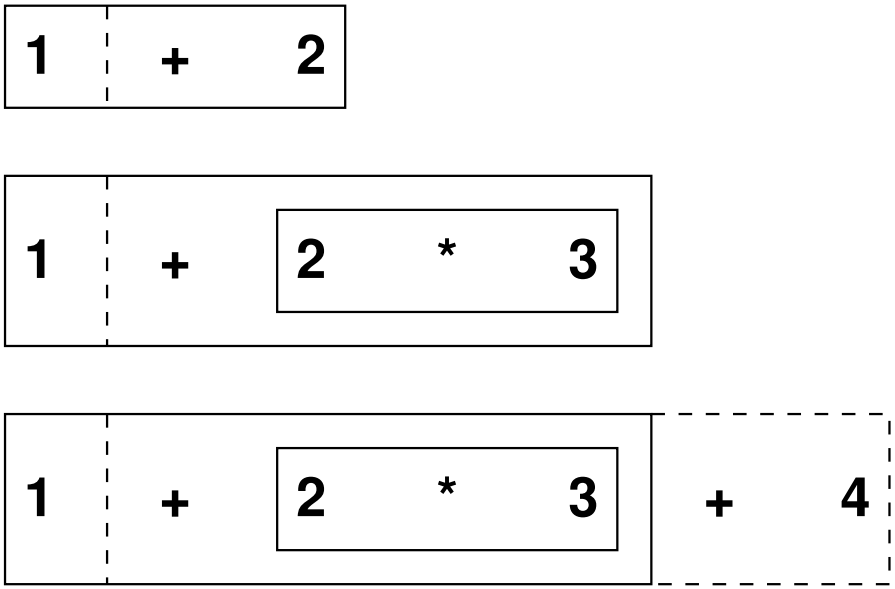
数式評価 *(cont.)*

本講義のホームページから expreval01.c を入手
http://www2.jaist.ac.jp/is/private/lecture/i117/2008/expreval01.c

- 非常に簡単な数式評価プログラム

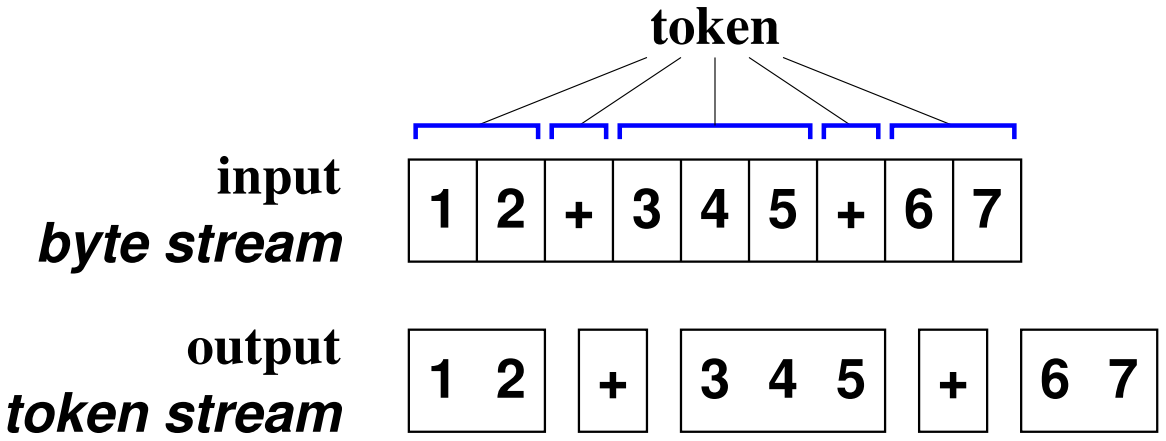
全体の流れ

- 式を分割して考える
- 多くは二項演算（部分式、演算子、部分式）



全体の流れ *(cont.)*

トークン取り出し
評価の最小単位（数や記号）をトークンと呼ぶ
バイトストリームをトークンストリームに変換する



トークンを取り出す関数を gettoken と呼ぶ

全体の流れ *(cont.)*

階層的に呼び出す

```
L1() {
    r  = L2();
    op = token;
    while(op==’+’) {
        gettoken();
        r += L2();
    }
    return r;
}
```

```
main() {
    gettoken();
    L1();
}
```

全体の流れ *(cont.)*

```
L2() {
    r  = resolv();
    op = token;
    while(op==’*’) {
        gettoken();
        r *= resolv();
    }
    return r;
}
```

```
resolv() {
    return
        atoi(token);
}
```

全体の流れ *(cont.)*

各層で欲しい演算子のみ処理する点に注意

	関数名	演算子	演算
3	resolve		token→number
2	L2	*	乗算
1	L1	+	加算

必要な演算の分だけ層を追加する
層追加の例に、expreval01.c では L3 を定義

全体の流れ *(cont.)*

```
expr ::= L1
L1 ::= L2 *( '+' L2 )
L2 ::= resolv *( '*' resolv)
resolv ::= NUM | SYM
```

expreval01.c では L2 と L3 が少し違う

```
L2 ::= L3 *( '*' L3 )
L3 ::= resolv
```

gettoken

gettoken 関連変数
得たトークンの情報を以下の変数に反映

名前	摘要
char*pos	式全体を格納する文字列中の注目位置
char token[]	トークン
int tokentype	トークン種別（TNUM, TSYM）
int tokenop	演算子（OPLUS, OTIMES, ...）

gettoken *(cont.)*

- 数字
  - 数字が途切れるまで token に蓄える
  - tokentype に TNUM を代入
- それ以外
  - 登録されている演算子に該当するか調べる
  - 該当したら tokentype に TSYM、tokenop に演算子を代入

複数文字の演算子がある点に注意、expreval01.c ではインクリメントのために ++ 演算子を登録してある

gettoken *(cont.)*

演算子のチェック（順序に注意）

```
    MATCH(OINC,     "++",    2);
    MATCH(OPLUS,    "+",     1);
    MATCH(OTIMES,   "*",     1);
    if(tokentype==TIG) {
        fprintf(stderr,
            "***ignore token\n");  }
```

プラス二つを見付けると、tokenop に OINC を代入
プラス一つを見付けると、tokenop に OPLUS を代入

多項演算

同じレベルの間、繰り返し演算する

```
r = L2()
while(op=='+') {
    gettoken();
    r += L2(); }
```

この処理がL2 *('+' L2) に相当

```
L2
L2 '+' L2
L2 '+' L2 '+' L2
L2 '+' L2 '+' L2 ... '+' L2
```

動作確認

標準入力から式を与える

```
% echo "1+2" | ./a.out
; 1+2
3
```

```
1+2+3   => 6
1+2*3   => 7
1+2*3+4 => 11
1*2+3*4 => 14
```

優先度も確実に実現できている

演習

expreval01.c 拡張

1) 減算、除算、剰余に対応せよ★

```
18-5  =>  13
34/7  =>  4          34%7  =>  6
```

2) 括弧に対応して優先度変更を実現せよ★★

```
(1+2)*3  =>  9
```

3) 負の値に対応せよ★★★

```
-5+3  =>  2
```

演習 *(cont.)*

大きく変更したプログラム

4) 文字列を扱える数式評価プログラムを作れ★★
   - 文字列用の演算子を設けよ
   - 数字を扱えなくても良い
5) 数字と文字列を扱える数式評価プログラムを作れ★★
6) 実数を扱える数式評価プログラムを作れ★★
   - 実数の型は double とする
7) 変数を扱える数式評価プログラムを作れ★★★

演習 *(cont.)*

- 変数代入の演算子を設けよ