言語プロセッサ

第9回目

平成26年12月8日

前回演習課題の確認

文法G4について答えよ。

① 終端記号はどれか？
② 非終端記号はどれか？
③ 開始記号はどれか？
④ First集合を求めよ。
⑤ Follow集合を求めよ。
⑥ 構文解析表を求めよ。
⑦ G4はLL(1)か？

G4:

$S \rightarrow i\ C\ t\ S\ S' \mid a$

$S' \rightarrow e\ S \mid \varepsilon$

$C \rightarrow b$

説明

- **if** *condition_is_true* **then** *sentence*
- **if** *condition_is_true* **then** *sentence* **else** *sentence*

`if A then if B then C else D` の解釈は？

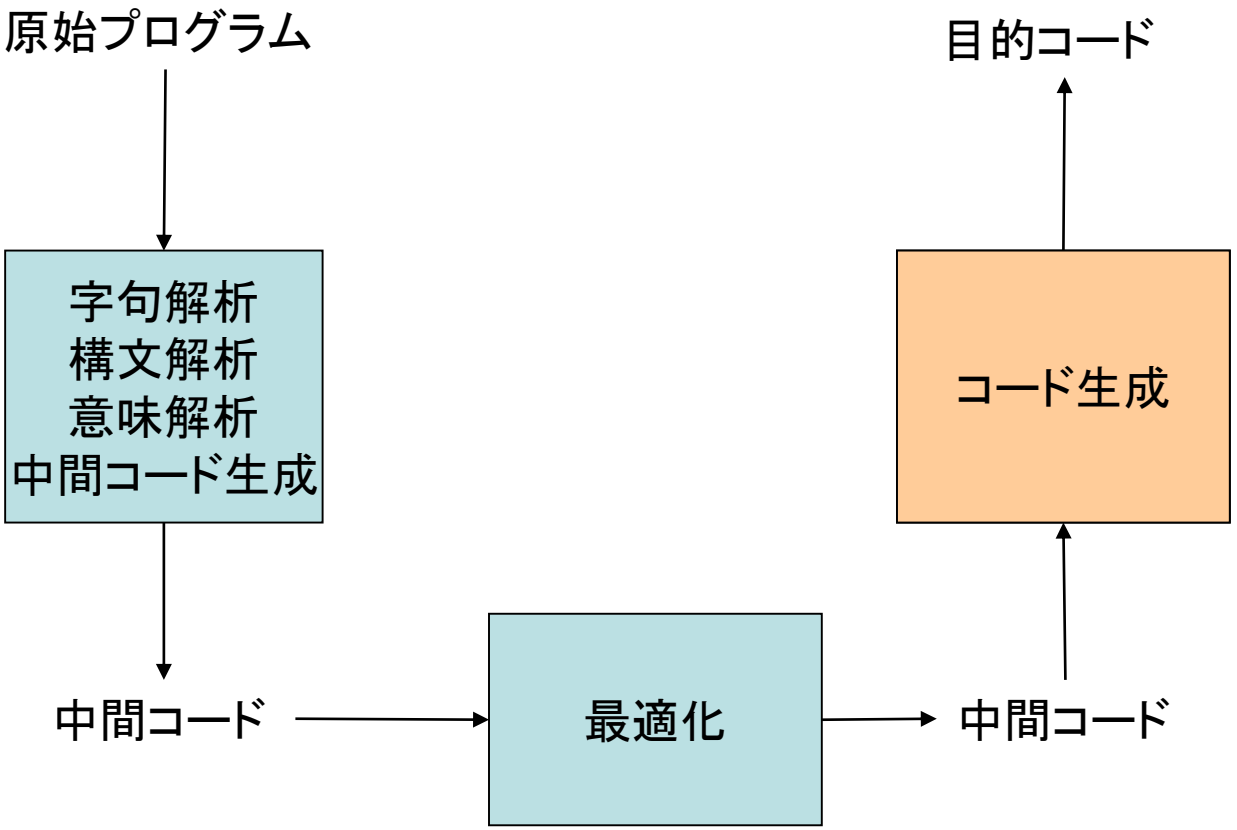
```
if A then
        if B then C else D
```

```
if A then
        if B then C
else D
```

前回までにやったこと

- 言語プロセッサ(コンパイラ，インタープリタ)
- 処理過程の概要
- 字句解析
  - 正規表現，トークン
  - Flex（入門）
- 構文解析
  - 文法（文脈自由文法，LL(1)，LR(1) など）
  - 括りだし，左再帰とその回避法，バックトラック
  - First，Follow，構文解析表，予測的構文解析法

コード生成の概要



もう一度具体例で確認

```
program daikei(input,output);
var abc,xyz,t:integer;
begin
   write('abc xyz = ');
   read(abc,xyz);
   abc := xyz + abc ＊ 123;
   writeln('result = ',abc)
end.
```

abc := xyz + abc * 123;

- （読み込み）

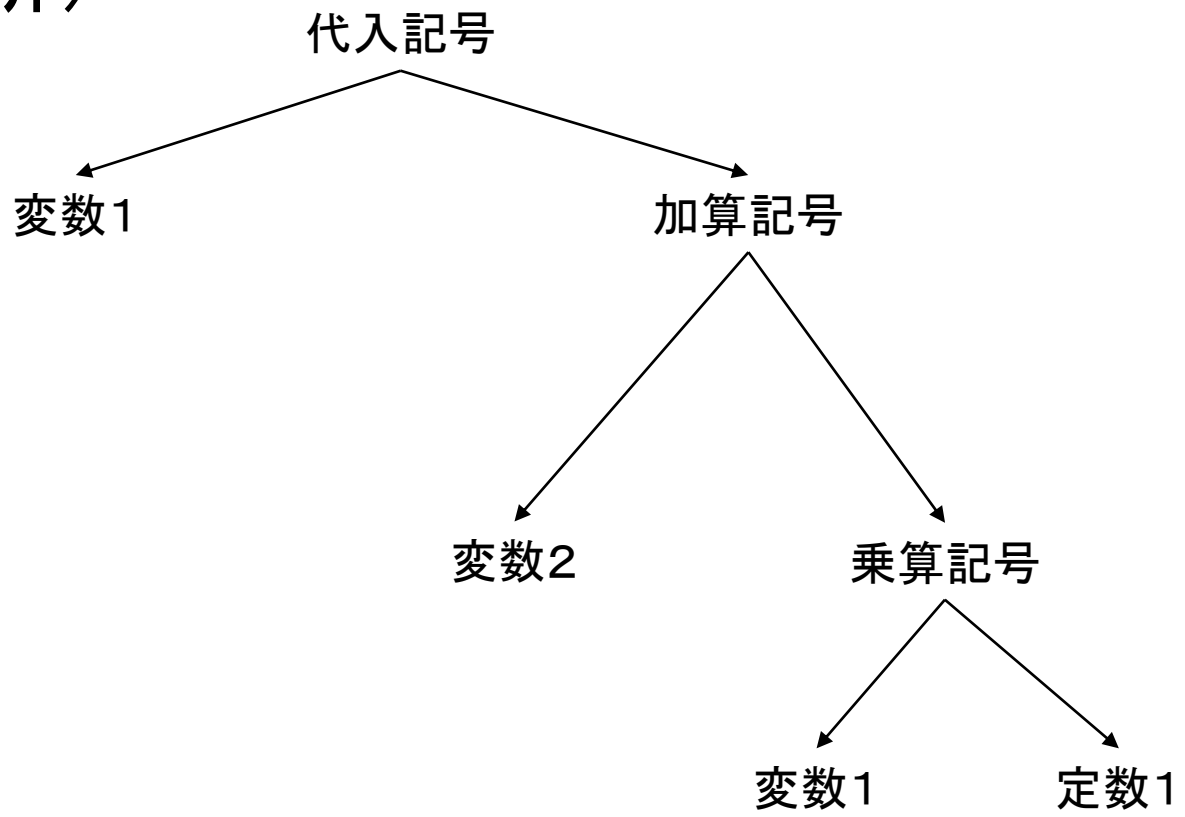
abc := xyz + abc *123;

- トークンへの分解(字句解析)

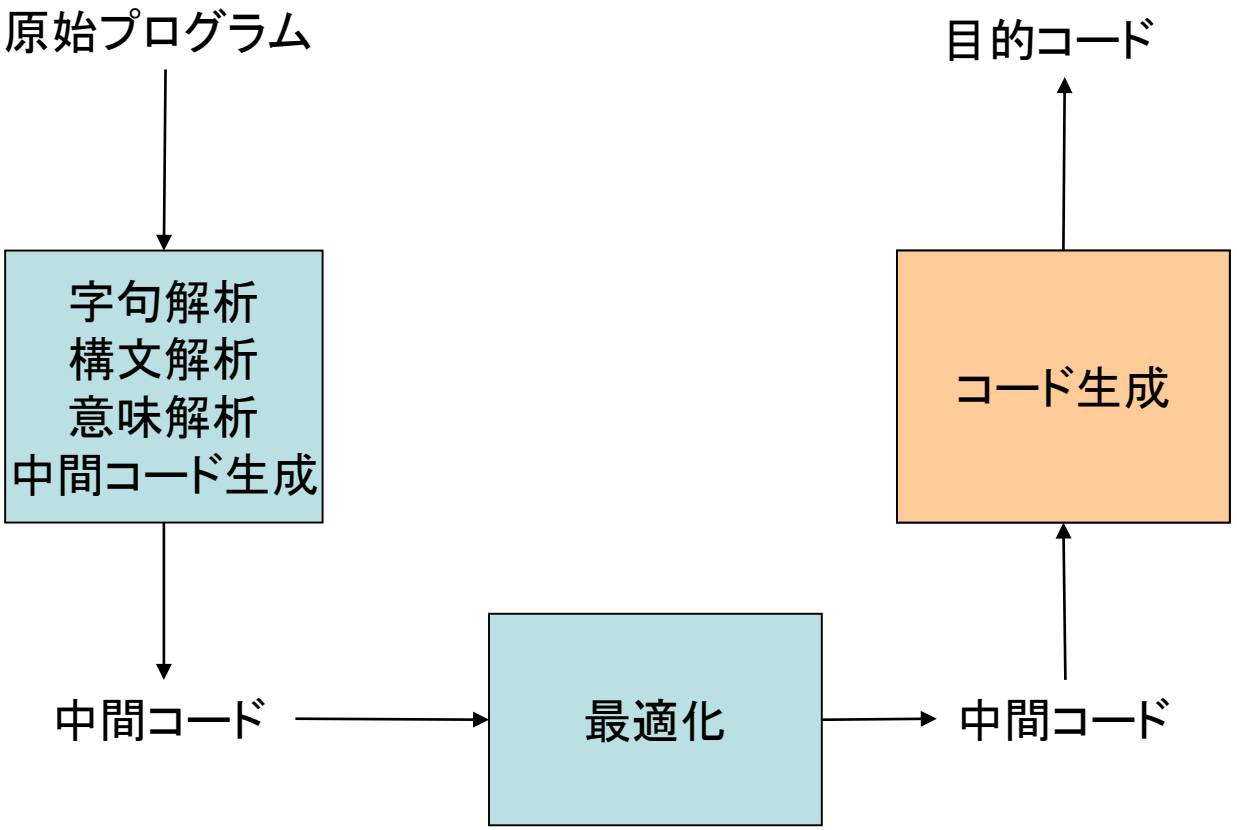
（ 変数1， abc ）
（ 代入記号， := ）
（ 変数2， xyz ）
（ 加算記号， ＋ ）
（ 変数1，abc ）
（ 乗算記号， ＊ ）
（ 定数1， 123 ）
（ 文区切り記号 ， ）

abc := xyz + abc * 123

- （構文解析）



コード生成の概要



意味解析

- 名前の宣言と使用との対応付け

例： int x, y;
float z;

x = z * y;
整数型 = 浮動小数点型 * 整数型

意味解析

例： int x, y;

float z;

x = z * y;

整数型 = 浮動小数点型 * 整数型

整数型 = 浮動小数点型 * 浮動小数点型

整数型 = 浮動小数点型

整数型 = 整数型

- 変数等の宣言された情報を参照する必要がある。

  ＝＞　記号表を用意しよう！

  - 名前 (spell)
  - 型 (int, float, …, struct char *, etc.)
  - 記憶域 (static, auto, …)
  - その他 (const, etc.)

変数の場合(例)

- 型
- 大きさ(バイト数)
- 有効範囲
- 通常変数/仮引数
- 宣言の有無(暗黙宣言が許されている言語)
- 実行時に割り当てられるアドレス　など

関数名・手続き名（例）

- 仮引数の個数および仮引数の型
- 戻り値の型（関数の場合）
- 有効範囲
- コードの先頭番地（entry point）

定数名(例)

- 型
- 定数の内部表現
- アドレス

型名（例）

- 型の種別(int, float, array, structure, etc.)
- 型の種別ごとの情報
（arrayならば、添字の範囲、要素の型など）

記号表に対する操作

- 登録
- 参照
- 更新
- 削除

＝＞表探索問題(Table Search Problem)

記号表の操作は

- 速くなければならない
  ＝＞どうすればいいのか？
  ＝＞　これ以降の議論は、
  「データ構造とアルゴリズム」や
  「計算可能性と計算量」など
  の授業で学んでください。

探索アルゴリズム

- 線形探索法（改良版には番兵法）
- ２分探索法
- ハッシュ法（ハッシング法）
- ２分最適木法
- B木法 etc.

自分で作るときには、まず「線形探索」でOK。
その後、例えばhashing法にしてみよう。

中間言語

- 原始プログラムの構文解析結果は、「解析木」である。
- 実際には、解析木とは異なる内部表現を使うことも多い。
  ＝＞これを「中間言語」と呼ぶ。

中間言語とは

- コンパイラは、原始プログラムを目的プログラムに変換する途中段階で、中間的なプログラムを作る場合がある。これを「中間コード」あるいは、「中間言語プログラム」といい、これを記述する言語を「中間言語」という。

中間言語

1. 構文木
2. 後置記法（Polish notation）
3. 三つ組
4. 四つ組

1. 構文木

- （いままで幾つか例を見てきましたので省略）

2. 後置記法

- 前置記法(prefix notation)
- 中置記法(infix notation)
- 後置記法(postfix notation)

2. 後置記法

- 前置記法(prefix notation)

  ＋ Ｘ Ｙ
- 中置記法(infix notation)

  Ｘ ＋ Ｙ
- 後置記法(postfix notation)

  Ｘ Ｙ ＋

練習問題

次の式の形式を変換せよ.

式 $(x + y) * z$

① 前置記法
② 中置記法
③ 後置記法

後置記法の長短

- 長所:
  - 括弧が不要
  - コード生成がし易い
  - スタックを利用すると、インタープリタが容易に実現可能(教科書pp.15-19参照のこと)
- 短所:
  - 四つ組(後述)と比べ表現に融通性欠如
  - そのため、最適化に不向き

3.三つ組

形式:

番号 （演算子，被演算子1，被演算子2）

例:

7.（＋，X，15）
（意味） ⑦ ← X＋15

二番地命令/コードともいう

例

A = 10 * B — C / D

=>

① (*, 10, B)

② (/, C, D)

③ (—, ①, ②)

④ (=, ③, A)

４．四つ組

形式：

（演算子，被演算子1，被演算子2，結果の変数）

例：

（＋，Ｘ，１５，ｔ）
（意味） ｔ ← Ｘ+15

三番地命令/コードともいう

例

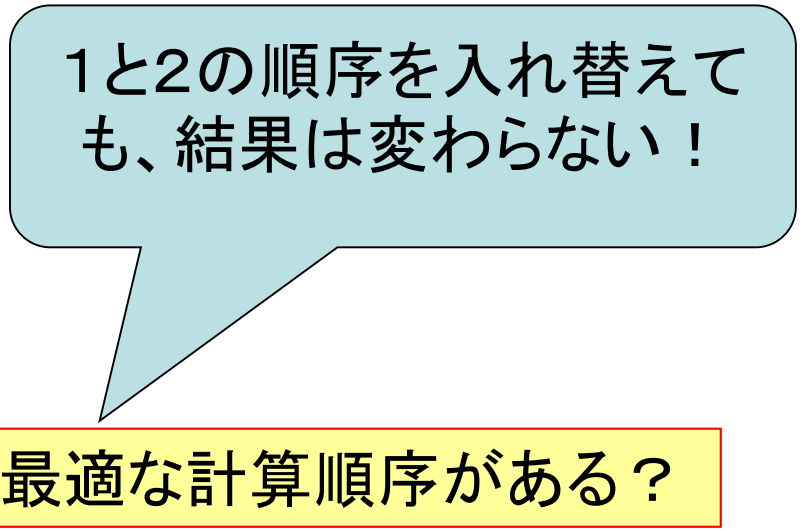
A = 10 * B — C / D

=>

1.（*, 10, B, t1）

2.（/, C, D, t2）

3.（—, t1, t2, A）



例2：X=(A+B-C)/(A+B)

練習問題

（まずは自分でやってみよう）

例2：X=(A+B-C)/(A+B)

（＋，A，B，t1）

（－，t1，C，t2）

（＋，A，B，t3）

（/，t2，t3，X）

例2：X=(A+B-C)/(A+B)

（＋，A，B，t1）

（—，t1，C，t2）

（＋，A，B，t3）

（/，t2，t3，X）

t1とt3は実は同じもの！

最適化しよう！

例2：X=(A+B-C)/(A+B)

（＋，A，B，t1）

（－，t1，C，t2）

（／，t2，t1，X）

（最適化された!）

今日の
練習問題

練習問題

1. 式 $x + y * (z - w)$ を
   a. 構文木
   b. 後置記法
   c. 三つ組　の列
   d. 四つ組　の列

   として表せ。

出席用紙に答案を書いてください。
最後に、提出してもらいます。

コードの最適化

- コンパイル過程において、生成するコードを改良することを「コード最適化」という。

  では、「改良」とはどうすること？

最適化の内容(例)

1. コードを小さくする
2. 実行時の効率をよくする
3. 実行時の使用メモリを小さくする

一般には、2が重要視される。

コード最適化の手法

1. 共通部分の削除
2. 複写伝播
3. 不要コードの削除
4. ループ不変量の抽出とコード移動
5. 演算子の強さの軽減　などなど

1．共通部分の削除

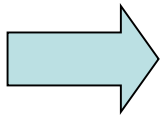
- A=B/(C+D)-(C+D);

  (+, C, D, t1)
  (/, B, t1, t2)
  (+, C, D, t3)
  (-, t2, t3, A)

1．共通部分の削除

- A=B/(C+D)-(C+D);

(+, C, D, t1)
(/, B, t1, t2)
(+, C, D, t3)
(-, t2, t3, A)



(+, C, D, t1)
(/, B, t1, t2)

(-, t2, t1, A)

2. 複写伝播

```
X = Y;
Z = X + 1;
W = X;
```

```
X = Y;
Z = Y + 1;
W = Y;
```

Yだけ。Xなし。

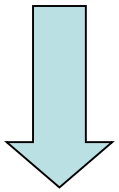
3. 不要コードの削除

```
X = Y;
Z = Y + 1;
W = Y;
```

```
Z = Y + 1;
W = Y;
```

４．ループ不変量の抽出とコード移動

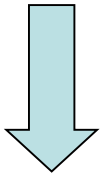
```
for ( i=0; i<100; i++ )
   x[ i ] = 10 * a[ j ]  +  y[ i ];
```



```
w = 10*a[ j ];
for( i = 0; i < 100; i++ )
      x[ i ] = w + y[ i ]
```

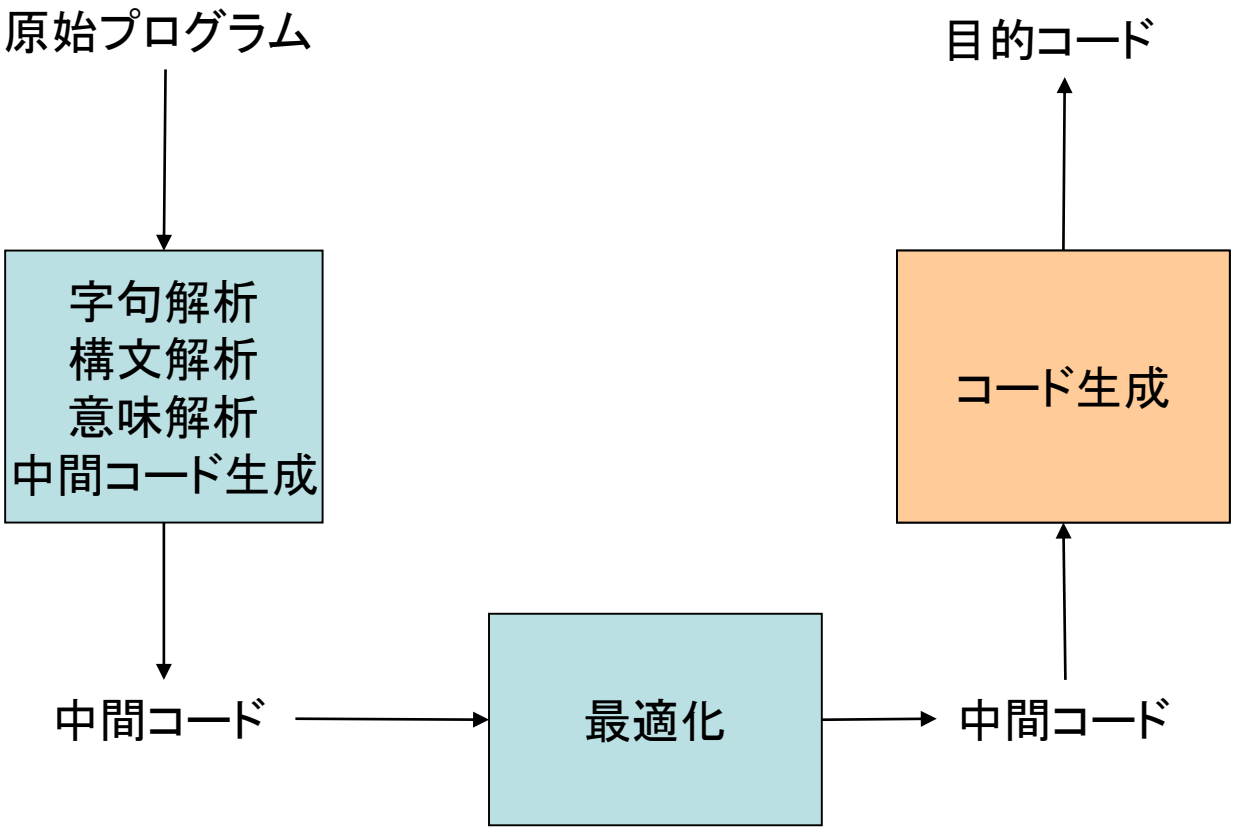
5. 演算子の強さの軽減

Y = A*2



Y = A + A

コード生成の概要(再)



お知らせ

- 来週からPCを用いた演習をやります。
- PCを持参してください。
- gcc, Flex, Bison, Antlrworks　などを使っていきます。事前に準備しください。
- 授業自体は、教壇のPC中心で行います。
- 周りの仲間とわいわいとやりましょう。

最後に

- 寒くなってきました，またインフルエンザの流行時期になりました．手洗い，うがいをしましょう．
- インフルエンザに罹ったら，病院へ行くとともに，他人にうつさないためにも，自宅でしっかりと療養してください．
- その際には，大学・先生に連絡すること．
- 病院で診断書をもらうことを忘れずに！

以上