基本情報第2回　　2013/4/23
情報、通信、制御に関する理論

例題1:逆ポーランド記法
後置表記法(逆ポーランド表記法)では，例えば，式　Y＝(A－B)×C　を　YAB－C×＝　と表現する。
　次の式を後置表記法で表現したものはどれか。
　　Y＝(A＋B)×(C－(D÷E))

- YAB＋C－DE÷×＝　　ア
- YAB＋CDE÷－×＝　　イ
- YAB＋EDC÷－×＝　　ウ
- YBA＋CD－E÷×＝　　エ

例題２：状態遷移表
次の表は，文字列を検査するための状態遷移表である。検査では，初期状態をａとし，文字列の検査中に状態がｅになれば不合格とする。
　解答群で示される文字列のうち，不合格となるものはどれか。ここで，文字列は左端から検査し，解答群中の△は空白を表す。

		文字				
		空白	数字	符号	小数点	その他
現在の状態	a	a	b	c	d	e
	b	a	b	e	d	e
	c	e	b	e	d	e
	d	a	e	e	e	e

- ＋0010　　ア
- －1　　イ
- 12.2　　ウ
- 9.△　　エ

例題３：BNF

次の規則から生成することができる式はどれか。

〔規則〕

＜式＞ ::＝＜変数＞｜(＜式＞＋＜式＞)｜＜式＞＊＜式＞

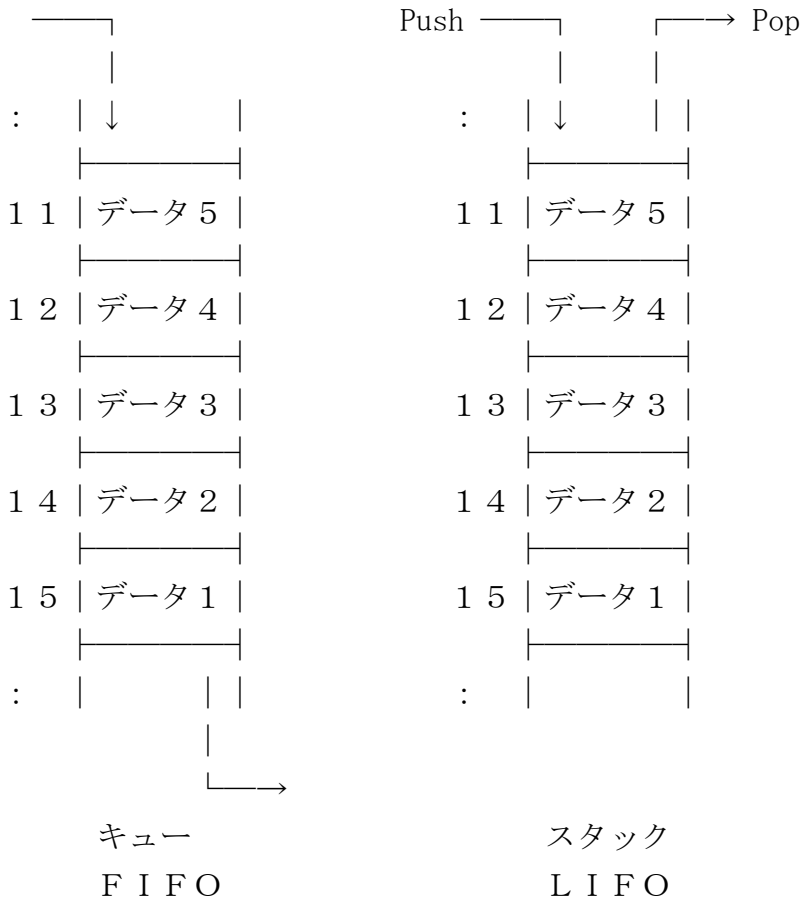
＜変数＞ ::＝A｜B｜C｜D

- A＋(B＋C)＊D　ア
- (A＋B)＋(C＋D)　イ
- (A＋B)＊(C＋D)　ウ
- (A＊B)＋(C＊D)　エ

解説１：**逆ポーランド表記法**

スタック、キュー、逆ポーランド記法

キューとスタック



配列の特殊な構造に、**キュー**と**スタック**があります。

キューは、FIFO（First In First Out：先入れ先出し）方式で、待ち行列ともいいます。プリンタへのスプールによる出力処理やデータを入力された順番通りに処理するときなどに用いられます。

それに対して、**スタック**は、LIFO（Last In First Out：後入れ先出し）方式です。新しいデータを入れる（Push）と、それまでのデータが下に押し下げられ、取り出すとき（Pop）ときは、１番上のデータが取り出され、２番目以降のデータが上に押し上げられます。

スタックの利用

コンピュータでは、スタックが広い分野で用いられています。

- CPUでは、命令を実行するためのレジスタの一つとして、スタックポインタというレジスタを用いています。
- コンパイラが数式を命令に展開する技法の一つに「逆ポーランド記法」がありますが、これにはスタックが重要な役割を持っています（これに関しては後述します）。
- 再帰呼び出しを実現するには、スタックが用いられます。
- 複雑な探索問題や最適化問題では、分枝限定法がとられますが、その中間結果を記録する手段（バックトラッキング）として、スタックが用いられます。

逆ポーランド記法との関係

逆ポーランド記法

数学の一般的な記法での数式 A＋B×C を計算するには、発生順序により「(A＋B)×C」とするのは誤りです。日本語では「AにBにCを×したものを＋する」ことになりますが、その順に「ABC×＋」と記述する記法を**逆ポーランド記法**といいます。

もう少し複雑な例にして、数式を逆ポーランド記法にする手順を説明します。

数式 A＋B×(C＋D)＋E は、演算の優先順位を( )で明示すれば、

((A＋(B×(C＋D)))＋E)

になります。

最も深い( )から、数式を逆ポーランド記法に置き換えていきます。

((A＋（B×（C＋D)))＋E)

└─①─┘

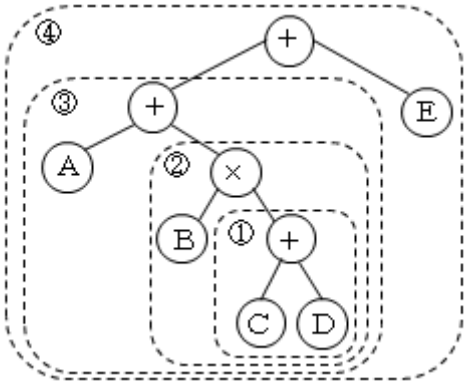
CD＋

└────②────┘

BCD＋×

└──────③──────┘

ABCD＋×＋



└──────────④──────────┘

ABCD＋×＋E＋

となり、逆ポーランド記法の

ABCD＋×＋E＋

になります。

なお、これを木構造（演算木）を用いて表現することもできます。

練習問題

問題１：　(A＋B)×(C－D)　→　AB＋CD－×

問題２：　A／(B－C)＋D　→　ABC－／D＋

スタックによる計算方法

数学での記法を逆ポーランド記法に変換すれば、スタックを利用して簡単に計算することができます。

そのルールは、次の通りです。

逆ポーランド記法で記述した数式を左から順番に処理する。スタックは空である。

数値ならば、スタックに Push する

演算子ならば、

スタックの1番上の数値を Pop して、その数値をXとする。

新しくスタックの1番上にきた数値をYとして、YにXを演算する。

その結果をYに代入する。

数式がなくなったとき、スタックには唯一の数値が残っており、それが数式の結果である。

このように、逆ポーランド記法では、＋×や( )などの優先順位を考慮する必要がなく、式の解釈が簡単になり、スタック操作だけで行えるので、初期の電卓では逆ポーランド記法の順に入力するものもありました。また、コンパイラで数式を解釈するときにも利用されています。

例題

「ABC／－」を例にして説明します。

Step1:Aを Push する。

Step2:Bを Push する。

Step3:Cを Push する。

Step4:／になった。Cが Pop され、スタックの最上段がBになる。

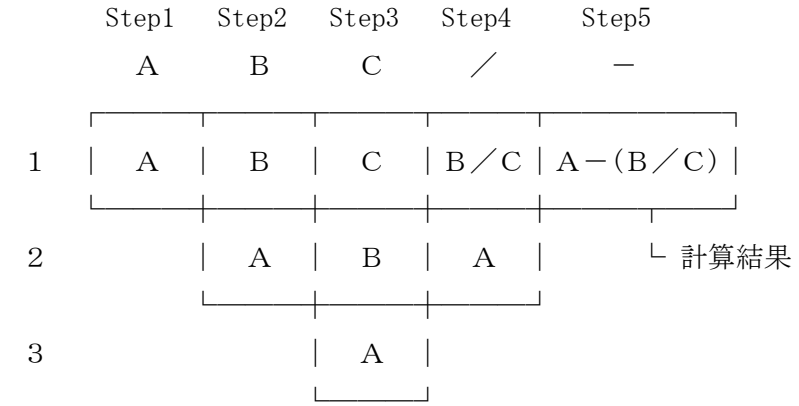
最上段の値を B／C の計算結果に置き換える。

Step5:－になった。B／C が Pop され、スタックの最上段がAになる。

最上段の値を A－(B／C) の計算結果に置き換える。

数式がなくなったので、計算結果は A－(B／C) となる。

スタックの状況は、次のようになります。



＜例題１の解説＞

通常の式を、逆ポーランド表記法で表現するための基本は、**A＋B を AB＋**で表すことです。これと一回使った演算子は 2 度使わないことに注意して、普通に計算式を解くのと同じ要領で行っていくことで逆ポーランド表記法の式になります。

Y＝(A＋B)×(C－(D÷E))を、一つずつ順番に逆ポーランド表記法に変換していきましょう。

1. まず括弧内の A＋B と D÷E を変換します。
   Y＝AB＋×(C－DE÷)
2. 次にもう一つの括弧内の(C－DE÷)を変換します。
   Y＝AB＋×CDE÷－

この時「DE÷」を一つの項として考えると、「C」－「DE÷」⇒CDE÷－となることを理解しやすいかと思います。

3. 次に右辺でまだ演算をしていない、"×"の左側と右側で演算します。先程と同様に「AB＋」×「CDE÷－」⇒AB＋CDE÷－×と考えます。
   Y＝AB＋CDE÷－×
4. 最後に 左辺と右辺を"＝"で演算して逆ポーランド表記法への変換が完了します。
   YAB＋CDE÷－×＝

したがって答えは、「イ」となります。

通常の式から、逆ポーランド表記法への変換はそれほど難しくありませんが、その逆(逆ポーランド⇒普通の式)は、迷ってしまう人もいるのではないでしょうか。今後も出題される可能性がありますので押さえておきたいですね。

＜例題２の解説＞：状態遷移

- ＋0010

c(符号)→b(数字)→b(数字)→b(数字)→b(数字)と遷移するので問題ありません。

- －1

c(符号)→b(数字)と遷移するので問題ありません。

- 12.2

b(数字)→b(数字)→d(小数点)と遷移し、現在の状態が d であり次の文字が数字であるので e に遷移します。したがって不合格となります。

- 9.△

b(数字)→d(小数点)→a(空白)と遷移するので問題ありません。

＜例題3の解説＞BNF

問題文のような構文定義法は、**BNF**(Backus-Naur Form, バッカス・ナウア記法)と呼ばれ、XML をはじめ多くのプログラム言語の構文定義に用いられています。

BNF で使われている各記号は、「::＝」が左辺と右辺の区切り,「｜」が or(または),「＜＞」は非終端記号を表しています。

各式を規則に従って変換していき、最後に<式>の形になれば規則どおりであることになります。

- A＋(B＋C)＊D

A＋(B＋C)＊D→<式>＋<式>

- (A＋B)＋(C＋D)

(A＋B)＋(C＋D)→<式>＋<式>

- (A＋B)＊(C＋D)

正しい。

(A＋B)＊(C＋D)→<式>＊<式>→<式>

- (A＊B)＋(C＊D)

(A＊B)＋(C＊D)→(<式>)＊(<式>)

第２回　小テスト(2013/4/24)　　学番　　氏名

問題１：

A＝1，B＝3，C＝5，D＝4，E＝2 のとき，逆ポーランド表記法で表現された式 AB＋CDE／－* の演算結果はどれか。

- -12　ア、　2　イ、　12　ウ、　14　エ

正解　ウ

解説

逆ポーランド記法で表現された計算式には計算順番があります。以下のようにして解いてみましょう。

まず演算子(+-*/)の左側にある 2 つの項に注目します。この計算式の場合では、+の左にある AB と、／の左側にある DE です。

まず最初に、この演算子と 2 つの項を計算します。

AB+　＝1＋3＝4，DE／＝4÷2＝2

計算式をまとめると 4C2－* になります。

再び演算子(+-*/)の左側にある 2 つの項に注目し計算します。今回は"－"の左にある C と 2 です。

C2-　＝5－2＝3

計算式をまとめると 43* になります。

再び演算子(+-*/)の左側にある 2 つの項に注目し計算します。最後に残った 43*です。

43*　＝12

よって答えは 12 になります。

問題２：

次の状態遷移表をもつシステムの状態が S1 であるときに，信号を t1，t2，t3，t4，t1，t2，t3，t4 の順に入力すると，最後の状態はどれになるか。ここで，空欄は状態が変化しないことを表す。

状態 / 信号	S1	S2	S3	S4
t1		S3		
t2	S3		S2	
t3			S4	S1
t4		S1		S2

- S1 ア、 S2 イ、 S3 ウ、 S4 エ

正解 ア

解説

1. [状態 S1，入力 t1] 空欄なので変化しません。
2. [状態 S1，入力 t2] S3 に遷移します。
3. [状態 S3，入力 t3] S4 に遷移します。
4. [状態 S4，入力 t4] S2 に遷移します。
5. [状態 S2，入力 t1] S3 に遷移します。
6. [状態 S3，入力 t2] S2 に遷移します。
7. [状態 S2，入力 t3] 空欄なので変化しません。
8. [状態 S2，入力 t4] S1 に遷移します。

状態 / 信号	S1	S2	S3	S4
t1	1	S3 5		
t2	S3 2		S2 6	
t3		7	S4 3	S1
t4		S1 8		S2 4

したがって最後の状態は「S1」になります。

問題３：

次の BNF で定義されるビット列 S であるものはどれか。

<S> ::= 01|0<S>1

- 000111 ア、 010010 イ、 010101 ウ、 011111 エ

正解 ア

解説

BNF(Backus-Naur Form, バッカス・ナウア記法)は、コンピュータ言語の構文などを記述するために使用される表記法で、プログラム言語 ALGOL(アルゴル)の文法を表現するためにジョン・バッカスなどによって考案されました。

「::＝」は等号「|」は論理和(OR)を表しているので、問題文の BNF は「<S>は、01 または 0<S>1 である」と解釈します。また<S>は両辺に表れているので再帰的に定義されていることになります。
<S>に右辺のいずれかを代入していくと、

<S>→0<S>1→00<S>11→000111

というように定義できることができるのは「000111」とわかります。