**Samsung Lab, Ha Noi, 9/7/2019**

**My Program contents**

**Samsung 2019**

**Problem:** Define a = 0, b = 1, c = 2, ... Solve the equation: (a+b)\*c+a\*(b+c).

**Technique:**

* Two Phases Technique
* Clean Room Technique

**Contents: Two phases**

* [Compiler](#CompilerLocation)

* [Execution](#ExecutionLocation)

[***Compiler***](#Compiler)

**Kí pháp Ba Lan**

* **Ký pháp Ba Lan**
* ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): *Polish notation*), còn gọi là ký pháp tiền tố (tiếng Anh: *prefix notation*), là một cách viết một biểu thức đại số rất thuận lợi cho việc thực hiện các phép toán. Đặc điểm cơ bản của cách viết này là không cần dùng đến các dấu ngoặc và luôn thực hiện từ trái sang phải.
* **Tác giả**
* Ký pháp Ba Lan do nhà logic toán Jan Łukasiewicz đề xuất khoảng năm 1920. Jan Łukasiewicz là một nhà toán học người Ba Lan. Ông sinh ra ở Lwów, Galicia (nay là Lviv, Ukraina). Lĩnh vực nghiên cứu chính của ông là logic toán.
* **Các phương pháp biểu diễn phép toán hai ngôi**
* Ta có thể viết hai hạng tử trước và ký hiệu toán tử sau. Chẳng hạn:

*a* + *b* viết là *a* *b* +, *a* \* *b* viết là *a* *b* \*

* Cũng có thể viết toán tử trước, hai toán hạng sau. Chẳng hạn:

*a* + *b* viết là + *a* *b*, *a* \* *b* viết là \* *a* *b*

* **Chuyển đổi từ biểu thức bình thường sang ký pháp Ba Lan**
* Thuật toán chuyển từ ký pháp trung tố sang ký pháp tiền tố hoặc hậu tố rất gần với cách xử lý các phép tính trong máy tính bấm tay (hay máy tính bỏ túi). Một biểu thức chỉ gồm các phép toán hai ngôi bất kỳ luôn có thể được tính bằng máy tính bấm tay mà không cần dùng dấu ngoặc. Các phép toán ở trước nếu có độ ưu tiên (ưu tiên bởi toán tử hoặc bởi dấu ngoặc) thấp hơn một phép toán ở sau được đẩy vào một ngăn xếp (stack), chỉ khi nào các phép toán ưu tiên hơn ở sau được tính xong, các phép toán ở trước mới được xử lý.

[***Execution***](#Execution)

**Equation:** (a+b)\*c+a\*(b+c) **Compile to Polish Form**

* a b + c \* a b c + \* +

**Run step by step:**

a = 0, b = 1, c = 2. Result = 2.

**Code:**

void execution () {

char ch;

int i;

for (i=0; i<ex.length();i++) {

ch = ex.charAt(i);

if (ch>= 'a' && ch <= 'z') {

ctrl++;

stk[ctrl] = ch - 'a';

} else {

switch(ch) {

case '+' :

stk[ctrl-1] += stk[ctrl];

ctrl--;

break;

case '-' :

stk[ctrl-1] -= stk[ctrl];

ctrl--;

break;

case '\*' :

stk[ctrl-1] \*= stk[ctrl];

ctrl--;

break;

case '/' :

stk[ctrl-1] /= stk[ctrl];

ctrl--;

break;

}

}

} // for

System.out.println("\n RESULT: " + stk[0]);

} // Execution

}

**Note:**

**Develop 1-operator. // toan tu 1 ngo***i*

**Divide by zero. // dangerous error.**

*Tuan Anh Pham.*