**Algorithm for Prefix to Infix Conversion**

1. Initialize/Input a string containing prefix expression.
2. Create a stacks of input string. // Code is provided in sample code.
3. Traverse from the last element to first of the string.
4. If the current element is not an operator, push it as a string in the stack.
5. If the current element is an operator, pop the two top elements from the stack and concatenate them as a single string with a current operator in between.

Push the string back into the stack.

1. Return the top of the stack.

**Programming precautions**

1. To traverse from the last element to first of the string, you can use a stack (denoted as Stack\_Prefix) to store the elements of Prefix expression.
2. For the step 3-2) of **Algorithm**, if the operator is “+” or “-” and there is more than one element left in Stack\_Prefix, include the concatenated string in “(” and “)”. For example, the two elements are operand1 and operand2, the operator is “+”, the resulted string is “(operand1+operand2)”, not just “operand1+operand2”.

**Some useful code segments**

* To transfer a string obtained from stack to float type, you can use code like:

string str;

float f;

str = Stack\_Prefix.top();

sscanf(str.c\_str(), "%f", &f);

**DO NOT** use the c++ 11 functions like “stof”.

* To concatenate operands and operator, you can use code like:

expression = "(" + operand1 + Stack\_Prefix.top() + operand2 + ")";

算法实现过程示例

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 输入 | Infix | \* - 5 6 7 |  |  |  |  |  |  |  |
| 变量类型 |  | stack<string> | stack<string> | string | string | string | double\* | int | result  备注 |
| 变量说明 | 扫描到的元素 Stack\_Prefix.top() | Stack\_Prefix (栈底->栈顶) 分号表示栈元素分隔 | Stack\_Infix  (栈底->栈顶) 分号表示栈元素分隔 | operand1 | operand2 | expression | result (index 0->n) 数组，计算和存储运算结果 | i=0 标记所使用的 result 中elements和要存储结果的位置索引 |  |
| step 1 |  | \*;-;5;6;7 | 空 |  |  |  |  |  |  |
| step 2 | 7 | \*;-;5;6 | 7 |  |  |  | [7] | 1 |  |
| step 3 | 6 | \*;-;5 | 7;6 |  |  |  | [7 6] | 2 |  |
| step 4 | 5 | \*;- | 7;6;5 |  |  |  | [7 6 5] | 3 |  |
| step 5 | - | \* | 7;(5-6) | 5 | 6 | (5-6) | [7 -1 5] | 2 | 计算 5-6，存到result[1] |
| step 6 | \* | 空 | (5-6)\*7 | (5-6) | 7 | (5-6)\*7 | [-7 -1 5] | 1 | 计算 -1\*7, 存到result[0] |

最终结果 Infix 表达式为Stack\_Infix.top()

表达式计算结果为 result[0]