Bài 16:

Thực hiện: Lê Minh An

MSSV: 20184024

Phân tích cách thực hiện:

* Đọc chuỗi cần phân tích vào mảng có kích thước 100 bytes (Message).
* Sử dụng stack có độ rộng 25 thanh ghi để lưu trữ các kí tự khác nhau xuất hiện trong chuỗi đầu vào (appear\_char).
* Xét lần lượt các kí tự trong chuỗi đầu vào. Nếu kí tự đó chưa xuất hiện trong stack thì tiến hành thêm kí tự đó vào stack.   
  Nếu kí tự là nằm ngoài khoảng ‘a’ đến ‘z’ thì sẽ bỏ qua và xét kí tự tiếp theo.  
  Nếu stack đã đầy thì sẽ thoát quá trình xét các kí tự.
* In ra chuỗi đã nhập vào và số lượng các kí tự khác nhau.

Ý nghĩa các thanh ghi được sử dụng:

* $s0: Lưu địa chỉ kí tự đầu tiên của chuỗi nhập vào (Message).
* $t0: Lưu giá trị biến i – số thứ tự của kí tự nhập vào đang xét (Message[i]).
* $t1: Lưu giá trị biến count – đếm số kí tự khác nhau đã xuất hiện.
* $t2: Lưu giá trị biến j – số thứ tự của kí tự đang xét trong stack (appear\_char[j]).
* $t3: Địa chỉ của Message[i]
* $t4: Giá trị của Message[i]
* $t5: Lưu giá trị thanh ghi $sp để truy cập stack các kí tự đã xuất hiện
* $t6: Sử dụng để lưu kết quả so sánh kí tự nhập vào đang xét với khoảng ‘a’ – ‘z’
* $t7: Lưu giá trị của kí tự đã xuất hiện để so sánh ( appear\_char[j] )

Kết quả thực hiện chương trình:

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Vậy, chương trình hoạt động đúng chức năng

Bài 5:

Thực hiện: Nguyễn Năng Hưng

MSSV: 20184118

Đề bài: Chuyển biểu thức trung tố thành hậu tố và tính giá trị biểu thức

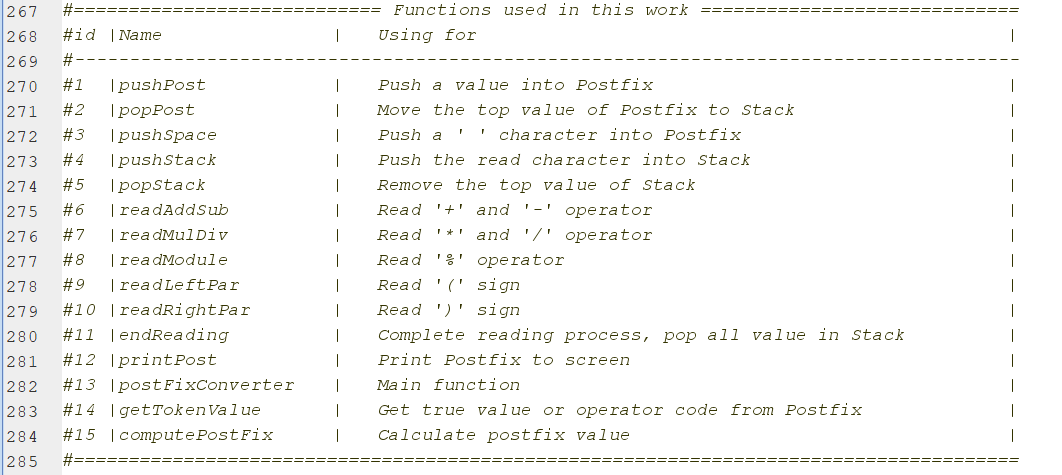
Phân tích cách thực hiện:

* Đọc vào một chuỗi các ký tự là một biểu thức số học, không chứa biến
* Lần lượt duyệt từ phần từ đầu tiên đến phần thử cuối cùng.
  + Nếu gặp kí tự là số, tức số hạng:
    - Đọc ký tự tiếp theo, nếu là tiếp tục là toán hạng thì đẩy (10\*toán hạng trước + toán hạng sau) vào postfix, nếu không phải toán hạng thì đẩy toán hạng đầu tiên vào postfix
    - Chèn dấu cách vào postfix
  + Nếu gặp kí tự là ký hiệu toán tử:
    - Nếu stack rỗng: đẩy vào stack
    - Nếu đỉnh stack là toán tử có độ ưu tiên cao hơn hoặc bằng toán tử hiện tại: pop phần tử đó ra khỏi stack, push vào postfix. Lặp lại cho đến khi gặp phần tử không thỏa mãn. Sau đó đẩy toán tử hiện tại vào stack
    - Nếu đỉnh stack là toán tử có độ ưu tiên thấp hơn: đẩy vào stack
  + Nếu gặp kí tự là dấu cách, bỏ qua và đọc tiếp
* Kết thúc việc đọc từ đầu đến cuối biểu thức nhập vào thu được dạng hậu tố cũng là một xâu ký tự gồm các toán hạng và toán tử, cách nhau bằng dấu cách.
* Thực hiện việc tính giá trị của xâu postfix:
  + Nạp lần lượt các ký tự của xâu postfix vào:
    - Nếu là toán hạng, đẩy vào stack
    - Nếu là toán tử, lấy 2 phần tử của toán hạng ra, thực hiện phép toán rồi đẩy lại vào stack
  + Giá trị của biểu thức cuối cùng là phần tử đầu tiên của stack

Các thanh ghi sử dụng:

* $t9: Expression (xâu đầu vào)
* $s0: Stack
* $s1: Postfix
* $s2: stackSize
* $s3: postSize
* $t0: value push stack
* $t1: look ahead value
* $t2: token from postfix
* $t3: index of token
* $t4: pop stack value
* $t5: iterator

Bảng hướng dẫn các hàm trong chương trình:

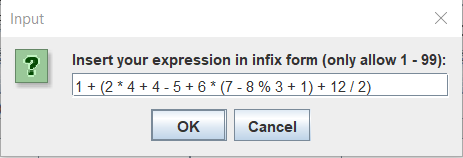


Bảng: Các hàm được dùng trong chương trình

Chạy kiểm thử chương trình với tính đúng đắn được thử nghiệm bằng:

* Biến đổi hậu tố: <https://www.mathblog.dk/tools/infix-postfix-converter/>
* Tính giá trị biểu thức: <https://google.com>

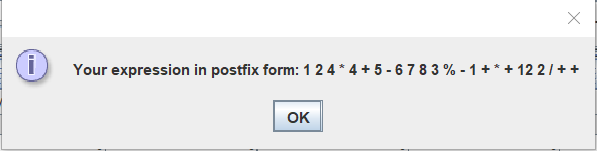
Hình ảnh chạy chương trình:



Ảnh: nhập vào xâu biểu thức

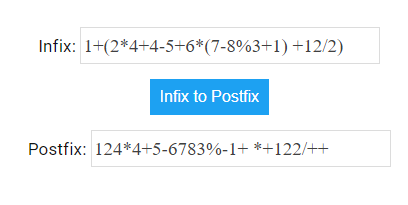
**Chú ý:** biểu thức không nhất thiết phải định dạng có dấu cách

Chương trình được thiết kế để tự động loại bỏ các dấu cách



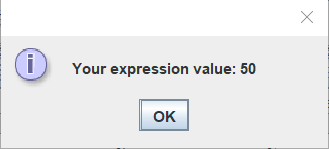
Ảnh: Kết quả thực thi chương trình

In ra màn hình biểu thức dưới dạng post fix

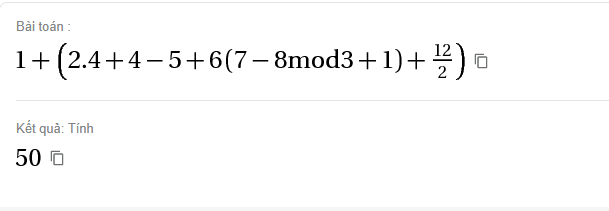


Hình: Kết quả biến đổi hậu tố kiểm thử trên website

**Kết luận:** Chương trình biến đổi hậu tố chính xác



Hình: Kết quả biểu thức nhập vào

Hình: Kết quả biểu thức kiểm thử bằng google

**Kết luận:** Chương trình tính biểu thức chính xác

Như vậy, chương trình hoạt động đúng chức năng.