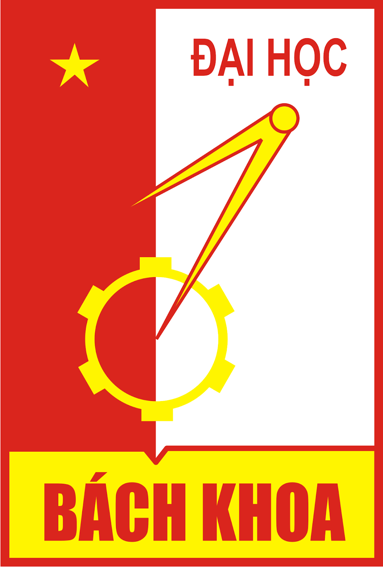


HANOI UNIVERSITY OF SCIENCE & TECHNOLOGY SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY





**BÁO CÁO PROJECT CUỐI KỲ**



**IT3280 – THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

|  |  |  |
| --- | --- | --- |
|  | **Thông tin học phần** |  |
| **Mã HP** | **Tên học phần** | **Mã lớp** |
| IT3280 | Thực hành kiến trúc máy tính |  |

|  |  |  |
| --- | --- | --- |
| **Thông tin sinh viên** | | |
| **Họ và tên**  Nguyễn Kiều Duyên | **Lớp**  IT-E6 04 K67 | **Mã sinh viên**  20225618 |
| Nguyễn Tuấn Đạt | IT-E6 04 K67 |  |

**Giảng viên:** Đỗ Công Thuần



**Hanoi, January 2024**



Table of Contents

[I. BÀI 4: BIỂU THỨC TRUNG TỐ HẬU TỐ 3](#_Toc167234487)

[1. Mô tả yêu cầu đề bài 3](#_Toc167234488)

[2. Thực hiện project 3](#_Toc167234489)

[a. Thuật toán 3](#_Toc167234490)

[b. Giải thích code 4](#_Toc167234491)

[3. Kết quả 14](#_Toc167234492)

[a. Các trường hợp ngoại lệ 14](#_Toc167234493)

[b. Các kết quả khác 15](#_Toc167234494)

# BÀI 4: BIỂU THỨC TRUNG TỐ HẬU TỐ

**Người thực hiện: Nguyễn Kiều Duyên**

# Mô tả yêu cầu đề bài

* Viết chương trình tính giá trị biểu thức bất kỳ bằng phương pháp duyệt biểu thức hậu tố.
* Các yêu cầu cụ thể:  
  1. Nhập vào biểu thức trung tố, ví dụ: 9135 + 2 + 8 \* 6  
  2. In ra biểu thức ở dạng hậu tố, ví dụ: 9135 2 + 8 6 \* +  
  3. Tính ra giá trị của biểu thức vừa nhập.  
  4. Các toán hạng là số nguyên dương, độ lớn nằm trong phạm vi biểu diễn 32 bit (một word).  
  5. Các toán tử bao gồm: phép cộng, trừ, nhân, chia lấy thương.

# Thực hiện project

# Thuật toán

* ***Xử lý ngoại lệ***
* *Kiểm tra dấu ngoặc đơn:* Chương trình theo dõi số lượng dấu ngoặc mở và đóng ('(' và ')') gặp phải trong biểu thức infix. Nếu tại bất kỳ thời điểm nào có dấu ngoặc đóng mà không có dấu ngoặc mở tương ứng, hoặc nếu có dấu ngoặc mở không khớp khi kết thúc, biểu thức được xem là không hợp lệ.
* *Toán tử và toán hạng liên tiếp:* Mã kiểm tra sự xuất hiện liên tiếp của các toán tử hoặc toán hạng trong biểu thức infix. Nếu có hai toán tử liên tiếp, biểu thức được xem là không hợp lệ (ví dụ: 5 \* \* 3 không hợp lệ). Nếu có hai toán hạng liên tiếp, biểu thức cũng không hợp lệ (ví dụ: 2 3 + 4 không hợp lệ).
* *Kết thúc bằng toán tử*: Mã kiểm tra xem biểu thức infix có kết thúc bằng một toán tử hay không. Nếu có, biểu thức được xem là không hợp lệ (ví dụ: 3 + 4 \* không hợp lệ).
* *Chia cho 0:* Chương trình bao gồm kiểm tra để đảm bảo rằng việc chia cho 0 được tránh. Nếu toán tử là '/', và số chia là 0, kết quả được đặt thành 0 mặc định.
* ***Chuyển đổi biểu thức infix sang postfix***
* Với tính tiện lợi và dễ triển khai, ngăn xếp là một trong những phương pháp phổ biến để chuyển đổi biểu thức infix sang postfix.

**Bước 1:** Đưa một biểu thức infix vào một chuỗi ký tự gọi là infix.

**Bước 2:** Tạo một chuỗi mới để lưu trữ biểu thức postfix, gọi là postfix, và khởi tạo một ngăn xếp.

**Bước 3:** Xem xét các ký tự infix từng cái một và làm theo các bước cho đến khi kết thúc biểu thức:

* Nếu ký tự là một số, lưu nó vào postfix.
* Nếu ký tự là "(", đặt nó vào ngăn xếp.
* Nếu ký tự là ")": lấy tất cả các phần tử ra khỏi ngăn xếp và đặt vào postfix cho đến khi gặp "(" (cũng được lấy ra khỏi ngăn xếp).
* Nếu ký tự là một toán tử và nếu ngăn xếp rỗng, đẩy toán tử vào ngăn xếp.
* Nếu toán tử đang xét có độ ưu tiên cao hơn toán tử ở đỉnh ngăn xếp, đẩy toán tử đó vào ngăn xếp.
* Nếu toán tử đang xét có độ ưu tiên thấp hơn hoặc bằng toán tử ở đỉnh ngăn xếp, lấy toán tử từ đỉnh ngăn xếp ra lần lượt đến khi nào toán tử đang xét có độ ưu tiên cao hơn toán tử ở đỉnh ngăn xếp thì dừng lại và đẩy toán tử đang xét vào đỉnh ngăn xếp.

**Bước 4:** Thực hiện bước 3 cho đến khi kết thúc biểu thức và tất cả các toán hạng và toán tử được đặt vào postfix. Cuối chuỗi infix, nếu có các phần tử trong ngăn xếp, lấy các phần tử đó ra khỏi ngăn xếp và đặt vào postfix. Kết quả, chúng ta có biểu thức postfix.

* ***Tính toán biểu thức***

**Bước 1:** Duyệt toàn bộ biểu thức postfix từ trái sang phải.

**Bước 2:** Tạo một ngăn xếp giá trị mới.

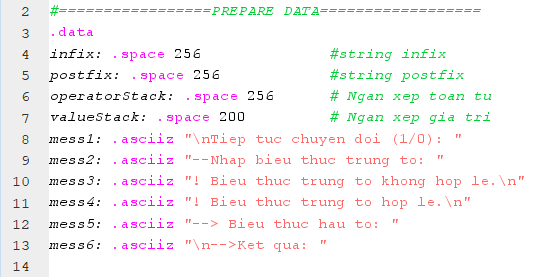
**Bước 3:** Nếu phần tử đang xét là một toán hạng, đẩy nó vào ngăn xếp.

**Bước 4:** Nếu phần tử đang xét là một toán tử, lấy 2 toán hạng từ đỉnh ngăn xếp, sau đó kiểm tra phép toán là gì và thực hiện, kết quả được đẩy lại vào ngăn xếp.

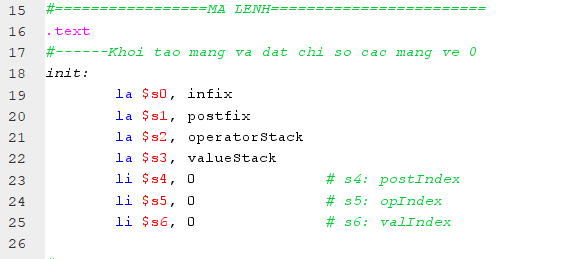
* ***Luồng thực thi mã***
* Chương trình đọc một biểu thức infix từ người dùng.
* Nó kiểm tra tính hợp lệ của biểu thức.
* Nếu biểu thức hợp lệ, nó chuyển đổi sang biểu thức postfix và tính toán kết quả.
* Kết quả và biểu thức postfix sau đó được in ra.
* Chương trình hỏi người dùng nếu họ muốn tiếp tục, và vòng lặp lặp lại nếu người dùng chọn tiếp tục.
* ***Ghi chú***
* Mã giả định rằng biểu thức infix đầu vào được phân cách bằng dấu cách và kết thúc bằng ký tự xuống dòng.
* Chương trình xử lý các toán tử số học cơ bản (+, -, \*, /).
* Việc chuyển đổi và tính toán được thực hiện bằng cách sử dụng ngăn xếp.

# Giải thích code

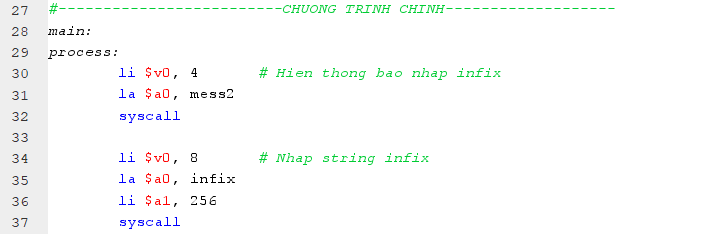
* ***Phần chuẩn bị dữ liệu***



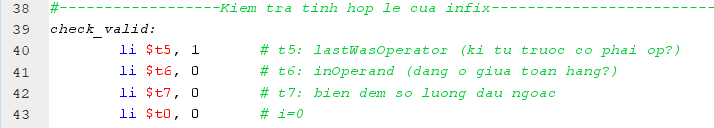
* **infix, postfix**: 2 mảng để lưu biểu thức trung tố và hậu tố.
* **operatorStack, valueStack:** 2 mảng để lưu trữ ngăn xếp các toán tử ( phục vụ cho việc chuyển đổi từ trung tố sang hậu tố ), và ngăn xếp giá trị ( phục vụ cho việc tính giá trị biểu thức từ biểu thức hậu tố )
* **mess1 đến mess6:** Các chuỗi ký tự dùng để hiển thị thông báo.
* **Khởi tạo biến**
* Khởi tạo các thanh ghi để trỏ đến các mảng và đặt các chỉ số tương ứng cho các mảng (Index) ban đầu là 0

****

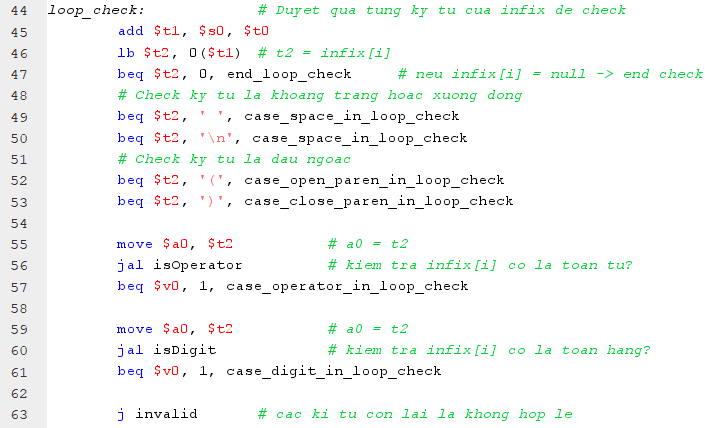
* **Chương trình chính**
* **process:** Quá trình chính để đọc các biểu thức trung tố, kiểm tra tính hợp lệ của biểu thức, chuyển biểu thức trung tố sang hậu tố, tính toán giá trị biểu thức và in ra kết quả

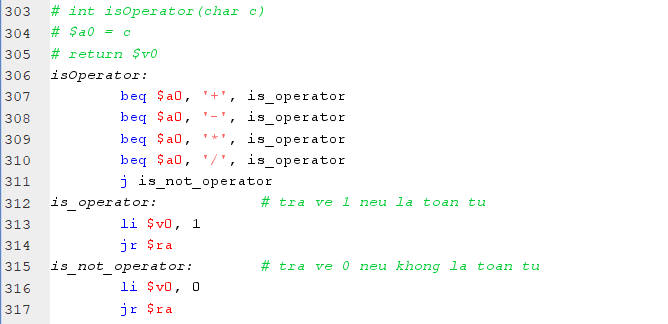
****

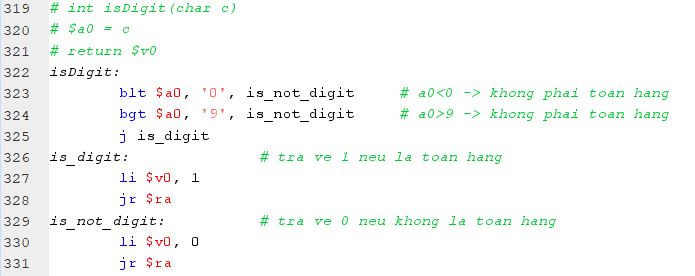
* **Kiểm tra tính hợp lệ của biểu thức**
* **check\_valid:** Kiểm tra tính hợp lệ của biểu thức trung tố bằng cách sử dụng các cờ (flags):
* t5: lastWasOperator: biến cờ kiểm tra ký tự trước đó có phải toán tử hay không
* t6: inOperand: biến cờ kiểm tra xem có đang ở giữa toán hạng hay không
* t7: Biến đếm số lượng ngoặc

****

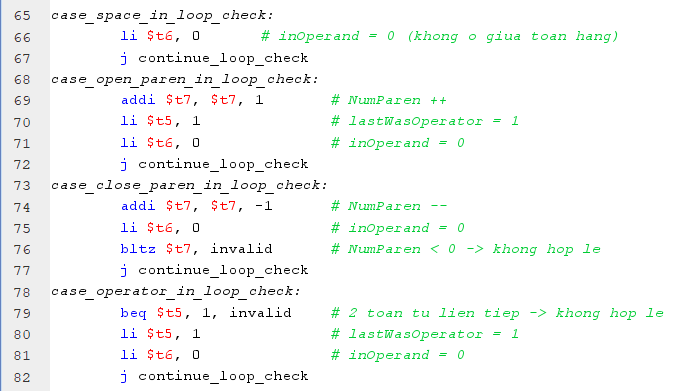
* **loop\_check:** Vòng lặp duyệt qua từng ký tự của biểu thức trung tố để kiểm tra tính hợp lệ

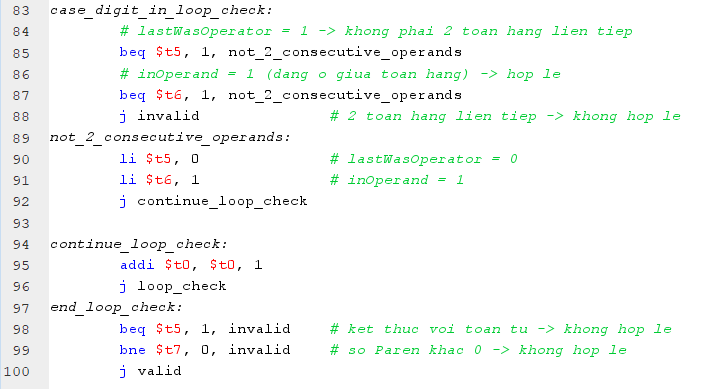
****

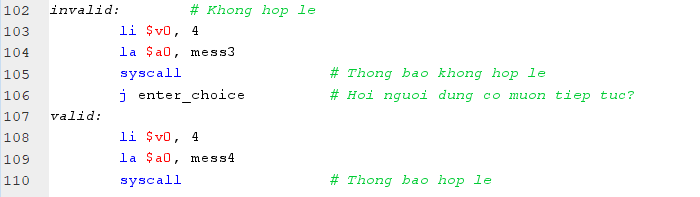
****

****

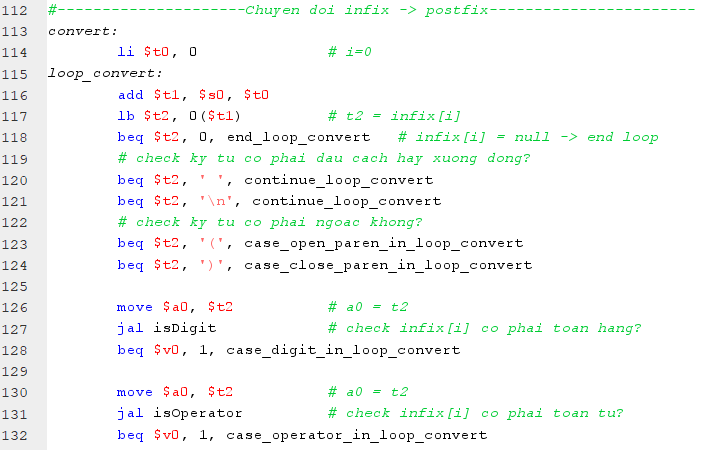
* Kiểm tra tính hợp lệ bao gồm các nhánh:
* **case\_space\_in\_loop\_check:** Kiểm tra nếu kí tự là khoảng trắng hoặc xuống dòng thì đặt t6 (inOperand) bằng 0 và tiếp tục check
* **case\_open\_paren\_in\_loop\_check:** Kiểm tra nếu kí tự là dấu mở ngoặc đơn thì tăng biến số lượng ngoặc 1 đơn vị, đặt cờ kí tự trước đó là toán tử và tiếp tục check.
* **case\_close\_paren\_in\_loop\_check:** Kiểm tra nếu kí tự là dấu đóng ngoặc đơn thì trừ biến số lượng ngoặc 1 đơn vị, kiểm tra nếu số lượng ngoặc nhỏ hơn 0 thì biểu thức không hợp lệ.
* **case\_operator\_in\_loop\_check, case\_digit\_in\_loop\_check, not\_2\_consecutive\_operands:** Nếu kí tự là toán tử hoặc toán hạng thì kiểm tra sự xuất hiện liên tiếp của toán tử hoặc toán hạng. Nếu có thì biểu thức không hợp lệ

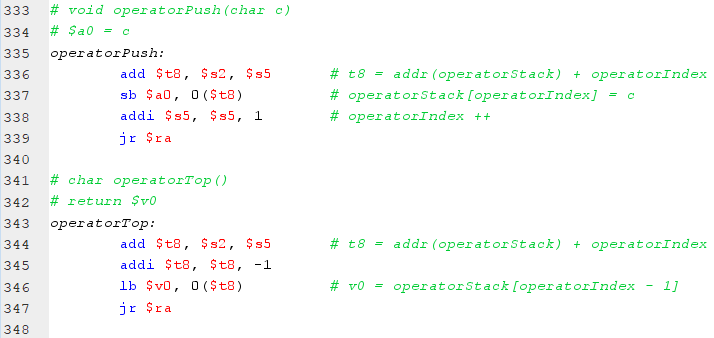


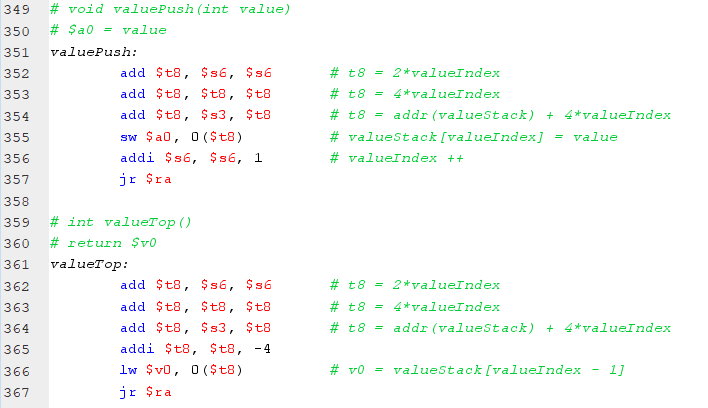


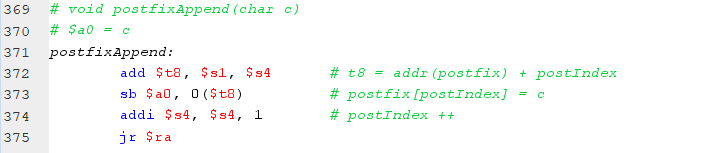


* **Chuyển đổi biểu thức trung tố sang hậu tố và tính toán biểu thức hậu tố**
* **convert:** Chuyển đổi biểu thức trung tố sang hậu tố
* Các hàm sử dụng để chuyển đổi:
  + **operatorPush:** Đẩy một toán tử vào ngăn xếp toán tử.
  + **operatorTop:** Lấy toán tử ở trên đỉnh của ngăn xếp toán tử.
  + **postfixAppend:** Thêm 1 ký tự vào chuỗi hậu tố.
  + **valuePush:** Đẩy một giá trị vào ngăn xếp giá trị.
  + **valueTop:** Lấy giá trị ở trên đỉnh của ngăn xếp giá trị.
  + **calculate:** Thực hiện phép tính dựa trên giá trị và toán tử.

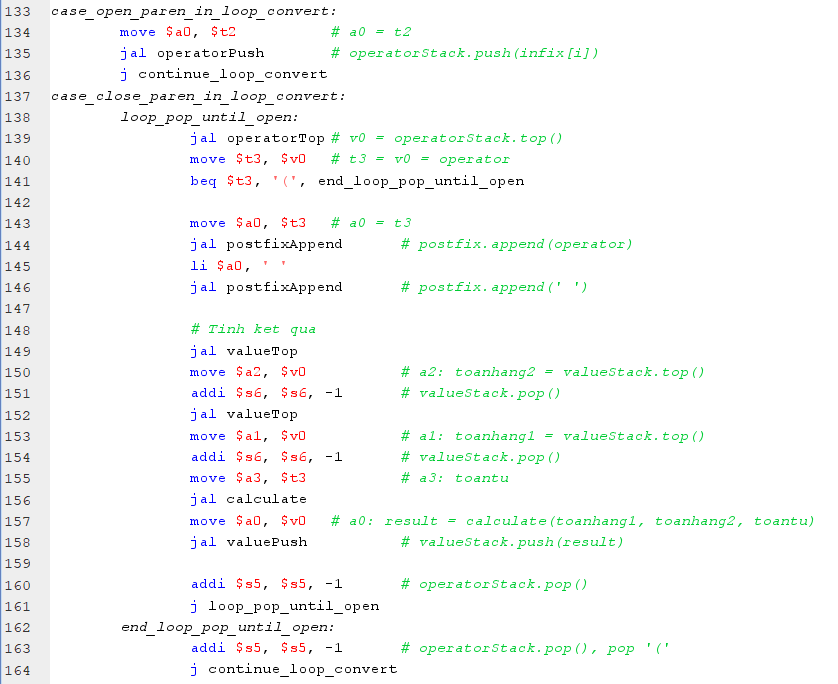




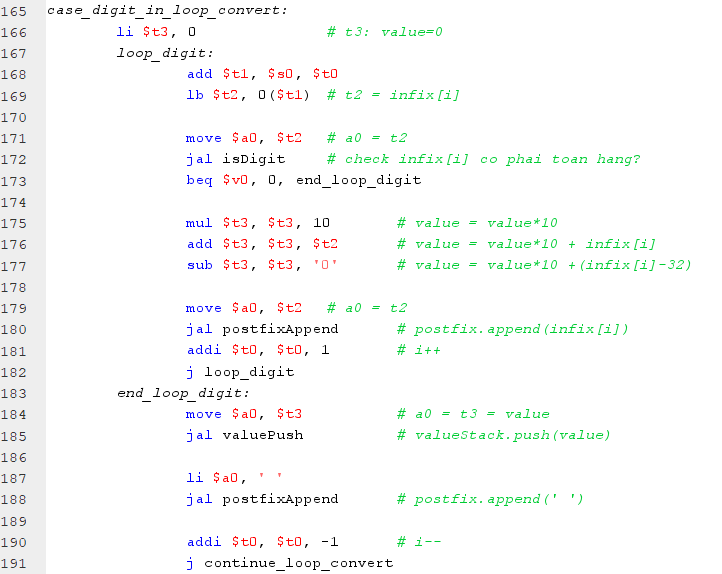




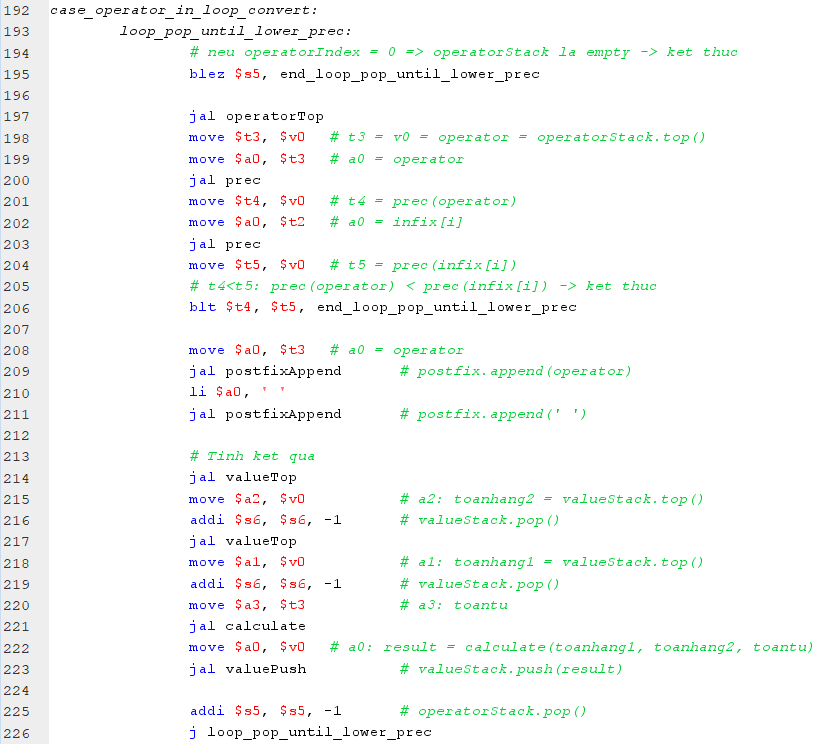
* **case\_open\_paren\_in\_loop\_convert:** Xử lý khi gặp dấu ngoặc mở ‘(‘ trong biểu thức. Đẩy dấu ngoặc vào trong ngăn xếp toán tử rồi tiếp tục xử lý ký tự tiếp theo.
* **case\_close\_paren\_in\_loop\_convert:** Xử lý khi gặp dấu ngoặc đóng ‘)’ trong biểu thức. Lấy các toán tử từ ngăn xếp toán tử ra đồng thời thực hiện các phép toán tương ứng sử dụng ngăn xếp giá trị để tính giá trị biểu thức, cho đến khi gặp dấu ngoặc mở ‘(‘ đầu tiên thì dừng lại . Sau đó tiếp tục xử lý các ký tự tiếp theo trong vòng lặp.

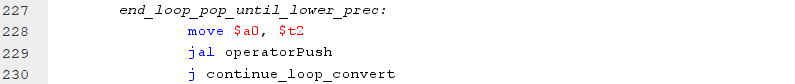


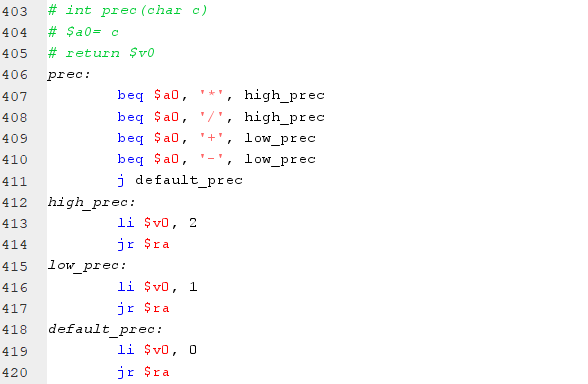
* **case\_digit\_in\_loop\_convert:** Xử lý chuyển đổi các chữ số liên tiếp trong biểu thức trung tố thành 1 số. Tích lũy giá trị của các chữ số thành số 32 bit (word), thêm chữ số vào biểu thức hậu tố và đẩy giá trị tích lũy vào ngăn xếp giá trị.



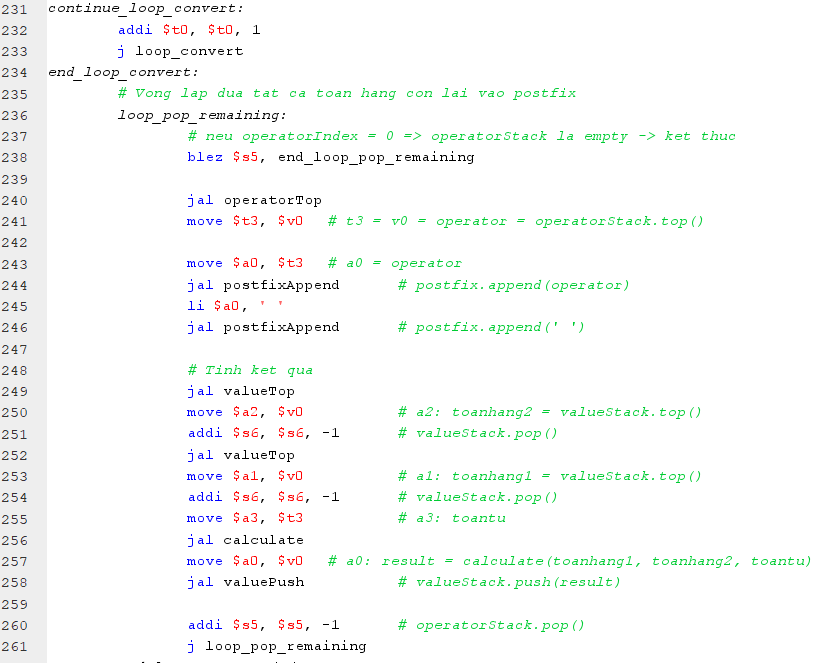
* **case\_operator\_in\_loop\_convert:** Đảm bảo rằng các toán tử có độ ưu tiên cao hơn được lấy ra từ ngăn xếp toán tử, thêm vào biểu thức hậu tố và thực hiện các phép tính tương ứng để tính giá trị của biểu thức từ ngăn xếp giá trị. Vòng lặp tiếp tục cho đến khi gặp một toán tử có độ ưu tiên thấp hơn hoặc ngăn xếp toán tử trở nên rỗng. Cuối cùng, toán tử hiện tại trong biểu thức trung tố được đẩy vào ngăn xếp toán tử.

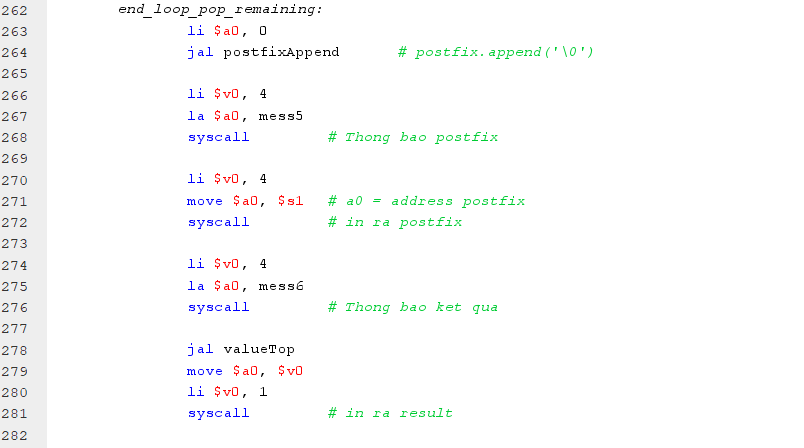




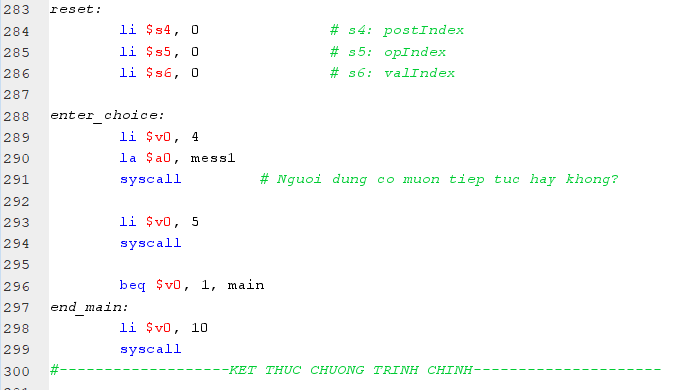


* **end\_loop\_convert:** Bước này hoàn tất chuyển đổi biểu thức trung tố sang hậu tố bằng cách lấy ra các toán tử còn lại trong ngăn xếp toán tử thêm vào biểu thức hậu tố, đồng thời thực hiện các phép tính biểu thức tương ứng cùng với ngăn xếp giá trị.





* **reset và user\_choice**: hỏi người dùng có muốn chuyển đổi tiếp không và lặp lại.



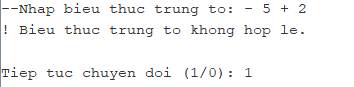
# Kết quả

# Các trường hợp ngoại lệ

* Toán tử và toán hạng không hợp lệ

|  |  |
| --- | --- |
|  |  |

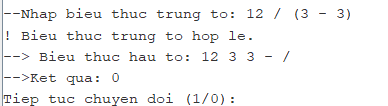
* Bắt đầu bằng toán tử



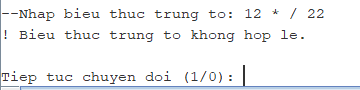
* Lỗi về dấu ngoặc đơn

|  |  |
| --- | --- |
|  |  |

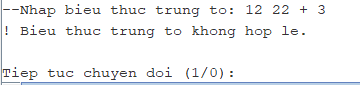
* Lỗi chia cho 0 ( Kết quả luôn được đặt là 0 )



* 2 toán tử liên tiếp

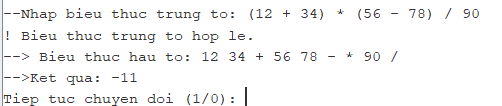


* 2 toán hạng liên tiếp

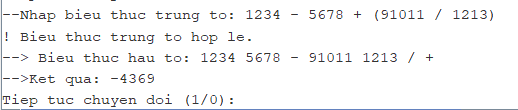


# Các kết quả khác

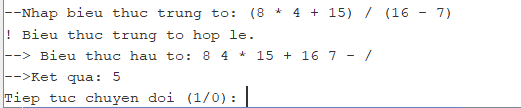
* (12 + 34) ∗ (56 − 78) / 90 = −11.378



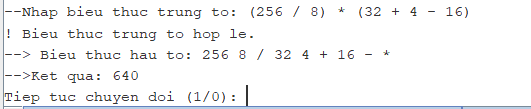
* 1234 – 5678 + (91011 / 1213) = −4369.974



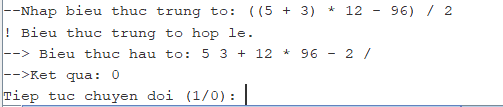
* (8 ∗ 4 + 15) / (16 − 7) = 5.444



* (256 / 8) ∗ (32 + 4 − 16) = 640.000



* ((5 + 3) ∗ 12 − 96) / 2 = 0.000



* *Note:* Vì ở đây dùng phép chia ‘/’ lấy phần thương nên kết quả đều đã được làm tròn thành số nguyên