

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**BÁO CÁO ĐỒ ÁN** **I**

ỨNG DỤNG PHÂN TÍCH VÀ TÍNH GIÁ TRỊ CỦA BIỂU THỨC SỬ DỤNG THUẬT TOÁN RPN

**GVHD**: ThS. Nguyễn Đăng Quang

**Sinh viên thực hiện**:

Phạm Huỳnh Thanh Lâm 17110168

Chu Minh Hoàng 17110139

***TP. Hồ Chí Minh, ngày tháng 11 năm 2019***

NHẬN XÉT CỦA GIẢNG VIÊN

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

**MỤC LỤC**

[PHẦN MỞ ĐẦU 1](#_Toc24485538)

[1. Lý do chọn đề tài: 1](#_Toc24485539)

[2. Mục đích nghiên cứu đề tài: 1](#_Toc24485540)

[3. Yêu cầu : 1](#_Toc24485541)

[4. Phương pháp nghiên cứu: 1](#_Toc24485542)

[PHẦN NỘI DUNG 2](#_Toc24485543)

[I. Đặc tả 2](#_Toc24485544)

[II. Phân công 3](#_Toc24485545)

[III. Thiết kế 4](#_Toc24485546)

[1) Thuật toán 4](#_Toc24485547)

[2) Thiết kế lớp 6](#_Toc24485548)

[3) Thiết kế giao diện 14](#_Toc24485549)

[IV. Cài đặt và kiểm thử 15](#_Toc24485550)

[PHẦN KẾT LUẬN 22](#_Toc24485551)

[1. Kết quả đạt được 22](#_Toc24485552)

[2. Kết luận 22](#_Toc24485553)

[TÀI LIỆU THAM KHẢO 23](#_Toc24485554)

**DANH MỤC CÁC BẢNG**

[Bảng 1: Bảng phân công việc 4](#_Toc23800019)

[Bảng 2: Bảng các class có trong chương trình 7](#_Toc23800020)

[Bảng 3: Bảng thống kê các phương thức dùng trong class tinhtoan 8](#_Toc23800021)

[Bảng 4: Bảng thống kê các phương thức trong class Operator 13](#_Toc23800022)

**DANH MỤC CÁC HÌNH**

[Hình 1: Ví dụ chuyển infix sang postfix đơn giản (sử dụng biến) 4](#_Toc24485504)

[Hình 2: Ví dụ chuyển đổi một biểu thức bằng thuật toán RPN 5](#_Toc24485505)

[Hình 3: Phương thức chuẩn âm 9](#_Toc24485506)

[Hình 4: Phương thức chuẩn hóa 10](#_Toc24485507)

[Hình 5: Phương thức chuyển đổi infix thành postfix (một phần code) 11](#_Toc24485508)

[Hình 6: Sử dụng phương thức so sánh hai operator 11](#_Toc24485509)

[Hình 7: Các bước làm của thuật toán chuyển đổi infix thành postfix 12](#_Toc24485510)

[Hình 8: Phương thức kiểm tra toán tử 12](#_Toc24485511)

[Hình 9: Tính toán chuỗi postfix 13](#_Toc24485512)

[Hình 10: Đưa ra độ ưu tiên của toán tử 14](#_Toc24485513)

[Hình 11: Giao diện màn hình 14](#_Toc24485514)

[Hình 12: Giao diện nhập biểu thức cần tính 15](#_Toc24485515)

[Hình 13: Kết quả của biểu thức sau khi thực hiện 15](#_Toc24485516)

[Hình 14: Kết quả sẽ không tính được nếu vừa nhập biến vừa nhập số 15](#_Toc24485517)

[Hình 15: Kết quả sẽ ra NULL nếu nhập chuỗi có hàm lượng giác nhưng thiếu dấu ngoặc 16](#_Toc24485518)

[Hình 16: Kết quả sẽ bằng Null nếu nhập liệu hàm lượng giác có chữ viết hoa đầu hoặc viết hoa toàn bộ 16](#_Toc24485519)

[Hình 17: Chưa làm được hàm kiểm tra nhập liệu (nếu có các ký tự là vẫn chuyển đổi thành chuỗi postfix) 16](#_Toc24485520)

[Hình 18: Nhập liệu một biểu thức đơn giản và tính toán 17](#_Toc24485521)

[Hình 19: Chuyển hóa chuỗi gồm hai biến a và b 17](#_Toc24485522)

[Hình 20: Thực hiện tính toán vừa có biến vừa có số 17](#_Toc24485523)

[Hình 21: Nhập liệu một biểu thức phức tập gồm nhiều toán tử, kết quả cho ra một số vô tỷ 18](#_Toc24485524)

[Hình 22: Xử lý và tính số âm 18](#_Toc24485525)

[Hình 23: Xử lý số hữu tỉ và phân số 19](#_Toc24485526)

[Hình 24: Tính toán hàm lượng giác 19](#_Toc24485527)

[Hình 25: Tính toán hàm lượng giác sử dụng thêm +,-,\*, / bên trong hàm lượng giác đó 19](#_Toc24485528)

[Hình 26: Cộng trừ nhiều biến 20](#_Toc24485529)

[Hình 27: Tính toán biểu thức ln (10) +1 20](#_Toc24485530)

[Hình 28: Tính căn của số 20](#_Toc24485531)

[Hình 29: Tính toán hàm mũ 21](#_Toc24485532)

[Hình 30: Thông báo khi một số chia cho 0 21](#_Toc24485533)

# PHẦN MỞ ĐẦU

## Lý do chọn đề tài:

Nhằm củng cố kiến thức đã học ở các môn học trước, lập trình hướng đối tượng, cấu trúc dữ liệu và giải thuật nhóm thực hiện phân tích và tính giá trị biểu thức bằng thuật toán RPN

## Mục đích nghiên cứu đề tài:

Mục đích của đề tài là tạo ra một chương trình tính toán đơn giản thông qua ngôn ngữ Java và các tính chất của một đối tượng, đồng thời củng cố kiếm thức đã học để áp dụng vào thuật toán và tìm hiểu về ngôn ngữ mới (java).

## Yêu cầu :

Áp dụng được phương pháp chuyển từ trung tố sang hậu tố bằng thuật toán RPN. Thực hiện được các phép tính đơn giản của một biểu thức bình thường ở dạng infix, ứng dụng các kiến thức đã học trong môn lập trình hướng đối tượng và các phương pháp chứa dữ liệu vào và ra trong cấu trúc dữ liệu.

## Phương pháp nghiên cứu:

Nghiên cứu tham khảo tra cứu thông tin và tài liệu

Hoàn thiện đề tài với sự hướng dẫn của giảng viên

# PHẦN NỘI DUNG

## Đặc tả

* Tên đề tài: Chương trình phân tích biểu thức và tính giá trị biểu thức đó dựa trên thuật toán RPN
* Dữ liệu đầu vào là một biểu thức hay còn gọi là Infix (trung tố). Sau đó được chuyển sang chuỗi Postfix (hậu tố). Các số và phép tính nhập vào được gọi lần lượt là toán hạng và toán tử.
* Chức năng: Dùng để tính một biểu thức bất kì có giá trị phép tính cộng, trừ, nhân, chia (+, -, \*, /), hàm lượng giác, căn, mũ, các đại lượng ln, log, log10. Với các số thực là: các số tự nhiên, số âm, số hữu tỷ (1/2, 0.12, 1.233 ở đây số thập phân được viết có dấu ‘.’ thay cho dấu ‘,’), các biến (là các chữ cái thường). Đồng thời tính toán các phép tính trong ngoặc tròn.
* Phân tích hướng giải quyết thuật toán : Chia làm hai giai đoạn

a. Chuyển trung tố thành hậu tố: Có 1 stack (chỉ chứa toán tử) và 1 danh sách liên kết đơn

Vấn đề được đặt ra:

* Tại sao lại sử dụng danh sách liên kết đơn?

Do chưa biết người dùng sẽ nhập bao nhiêu ký tự và danh sách liên kết đơn giải quyết được vấn đề đó (Danh sách liên kết là một cấu trúc dữ liệu có thể giữ một số lượng phần tử tùy ý và dễ dàng thay đổi kích thước, cũng như dễ dàng bỏ đi các phần tử bên trong nó)

* Tại sao không sử dụng mảng?

Trả lời: Ta vẫn có thể dùng mảng (phải biết trước kích thước hoặc quy định) bỏ qua điều ở trên thì việc sử dụng mảng hay danh sách liên kết đều khả thi đối với thuật toán RPN.

Ngoài ra lợi ích của việc sử dụng danh sách liên kết còn được thể hiện ở việc: khi sử dụng danh sách liên kết sẽ tối ưu được bộ nhớ.Vùng nhớ sẽ được cấp phát thêm khi cần chèn thêm phần tử mới, vùng nhớ được trống khi xóa phần tử.Trong bài toán chỉ cần một nơi lưu trữ cũng như là duyệt mảng để làm nên việc sử dụng là tối ưu hơn sử dụng mảng.

b. Tính trung tố: Có 1 stack (chỉ chứa toán hạng)

* Tính toán biểu thức tính cộng, trừ, nhân, chia, căn, hàm đại số.
* Xuất kết quả.
* Thiết kế giao diện: Chương trình được chạy trên Console đơn giản

## Phân công

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Công việc | Phạm Huỳnh Thanh Lâm | Chu Minh Hoàng |
| 1 | Tìm hiểu thuật toán và ngôn ngữ java, xác định các đối tượng cần thực hiện | X | X |
| 2 | Thiết kế các lớp cần thực hiện | X | X |
| 3 | Viết hàm xuất nhập biểu thức và đưa chuỗi biểu thức đó vào danh sách liên  kết (biểu thức ở đây là infix) |  | X |
| 4 | Viết hàm chuyển infix thành postfix | X | X |
| 5 | Viết lớp với đối tượng là toán tử dùng lưu toán tử đó và độ ưa tiên | X | X |
| 6 | Viết hàm xử lý các dấu cộng trừ liên tiếp nhau | X |  |
| 7 | Xử lý hàm đại số trong bài toán | X | X |
| 8 | Kiểm tra, sửa lỗi, tinh chỉnh, bổ sung chương trình (nếu có) | X | X |
| 9 | Viết báo cáo đồ án với Word | X | X |
| 10 | Mức độ hoàn thành | 100% | 100% |

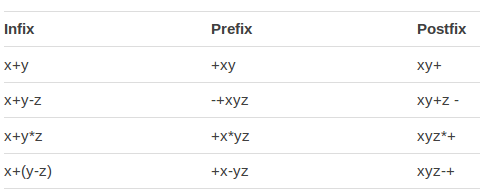
*Bảng 1: Bảng phân công việc*

## Thiết kế

### Thuật toán

Thuật toán được sử dụng cho bài toán được gọi là “ký pháp nghịch đảo Ba Lan” hoặc được viết tắt là RPN(Reverse Polish notation), được phát minh vào khoảng giữa thập kỷ 1950 bởi một triết học gia và nhà khoa học máy tính Charles Hamblin người Úc.

Dưới đây là ví dụ đơn giản cho thuật toán:

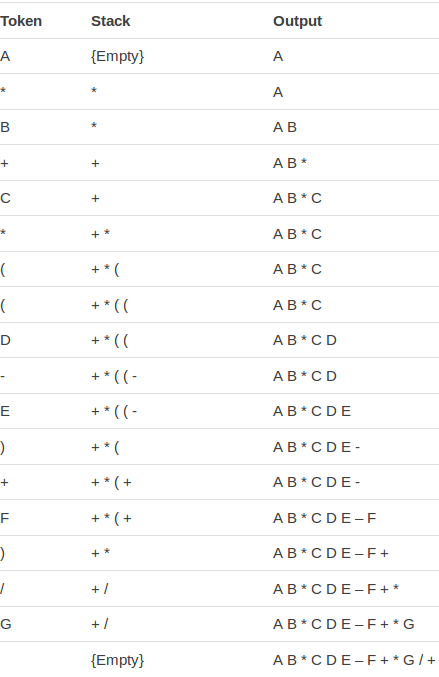


Hình 1: Ví dụ chuyển infix sang postfix đơn giản (sử dụng biến)

Thuật toán sẽ thực hiện chuyển đổi một biểu thức infix thành postfix bằng cách đọc từng token trong biểu thức infix từ trái qua phải, với mỗi token ta thực hiện các bước sau:– Nếu là toán hạng: cho ra output.– Nếu là dấu mở ngoặc “(“: cho vào stack– Nếu là dấu đóng ngoặc “)”: lấy các toán tử trong stack ra và cho vào output cho đến khi gặp dấu mở ngoặc “(“ (Dấu mở ngoặc cũng phải được đưa ra khỏi stack)– Nếu là toán tử: +Chừng nào ở đỉnh stack là toán tử đồng thời toán tử đó có độ ưu tiên lớn hơn hoặc bằng toán tử hiện tại thì lấy toán tử đó ra khỏi stack và cho ra output. +Đưa toán tử hiện tại vào stack

Sau khi duyệt hết biểu thức infix, nếu trong stack còn phần tử thì lấy các token trong đó ra và cho lần lượt vào output.

Để làm rõ thuật toán trên nhóm thực hiện tiến hành ví dụ đơn giản: Chuyển biểu thức A\*B+C\*((D-E) + F)/G từ dạng Infix sang dạng Postfix:



*Hình 2: Ví dụ chuyển đổi một biểu thức bằng thuật toán RPN*

### Thiết kế lớp

#### Danh mục các lớp

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Tên lớp | Mục đích | SV phụ trách |
| 1 | Tinhtoan | Chứa các phương thức để thực hiện bài toán | Chu Minh Hoàng  Phạm Huỳnh Thanh Lâm |
| 2 | Operator | Chứa đồng toán tử,độ ưa tiên, từ đó so sánh chúng với nhau | Phạm Huỳnh Thanh Lâm  Chu Minh Hoàng |
| 3 | Stack : st,sh | Ngăn xếp dùng để chứa toán tử và toán hạng | Giải thích: Stack được hỗ trợ trong java |
| 4 | Programmain | Thiết kế giao diện người nhìn đơn giản trên console và xuất các kết quả trên giao diện | Phạm Huỳnh Thanh Lâm  Chu Minh Hoàng |

*Bảng 2: Bảng các class có trong chương trình*

Giải thêm về các class đã chọn và thực hiện được nêu ở bảng trên:

1. TinhToan

* Mục đích: Chứa các phương thức để tính toán và phân tích bài toán:
* Chuẩn âm.
* Chuẩn hóa.
* Chuyển từ infix sang postpix.
* Tính kết quả khi đã chuyển thành postfix.

2. Operater

* Đối tượng chứa toán tử và độ ưu tiên của toán tử.
* So sánh các operator.

3. Stack

* Sử dụng để lưu trữ trong khi thực hiện quá trình chuyển đổi infix sang postfix. Stack ở đây lưu trữ mỗi phần tử vô là một mảng gồm 2 phần tử (toán tử, độ ưa tiên). Phần tử (toán tử, độ ưa tiên) được xét bên trong Operator, đưa mỗi operator được xét vào mảng gồm hai phần tử.
* Ngoài ra Stack còn được dùng để lưu toán hạng, để thực hiện các phép tính đưa ra kết quả cuối cùng.

#### Phương thức trong mỗi lớp

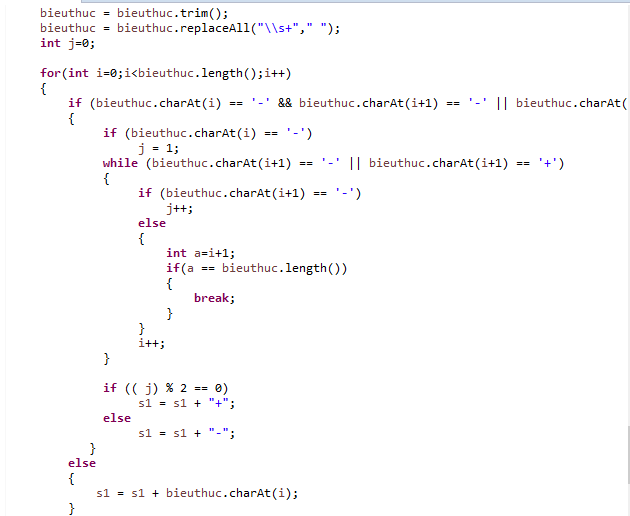
1. TinhToan.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | Phương thức | Mục đích | Tên file số thức tự dòng khai báo | SV phụ trách |
| 1 | **public** **boolean** isOperator(String c)  Input:Kí tự c  Ouput:True hoặc False  Giải thích: Sắp xếp chuỗi tăng dần với phương thức array.sort và array.binarySearch xuất vị trí của toán tử trong mảng | Kiểm tra xem kí tự c có phải là các toán tử "+", "-", "\*", "/", ")", "(" ,"sin","cos","tan","cotg","^","log","ln","sqrt" | Dòng 151 | Chu Minh Hoàng |
| 2 | **public** String Chuanam(String bieuthuc)  Input:String  Ouput:String  Giải thích: xử lý vấn đề về số âm  Ví dụ:-1+2; 1--2=>1+2; --1+2=>+1+2 | Chuẩn hoá số âm hoặc việc nhập nhiều dấu cộng trừ liên tiếp nhau | Dòng 158 | Phạm Huỳnh Thanh Lâm |
| 3 | **public** LinkedList<String> ChuanHoa(String bieuthuc)  Input:Chuỗi biểu thức <String>  Ouput:Danh sách liên kết chứa các phần tử của chuỗi trên | Chuẩn hoá đưa chuỗi lưu thành một danh sách liên kết đơn tương với từng phần tử toán hạng và toán tử | Dòng 118 | Chu Minh Hoàng |
| 4 | **public** Linked List<String> chuyendoi(LinkedList<String> elementMath)  Input: LinkedList <infix>  Ouput: LinkedList <postfix> | Chuyển từ infix sang postfix | Dòng 55 | Chu Minh Hoàng, Phạm Huỳnh Thanh Lâm |
| 5 | **public** String Xuatkq(LinkedList<String> doi)  Input: LinkedList  Ouput:String | Tính kết quả dựa trên chuỗi postfix và xuất kết quả đó ra màn hình | Dòng 7 | Chu Minh Hoàng  Phạm Huỳnh Thanh Lâm |

*Bảng 3: Bảng thống kê các phương thức dùng trong class tinhtoan*

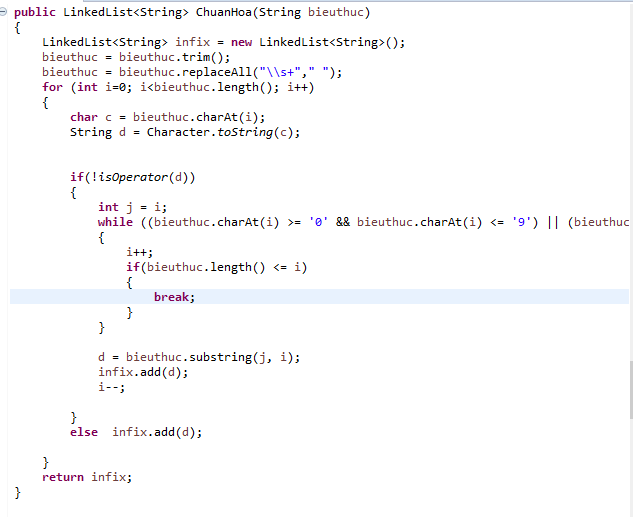
Ở bảng phương thức tinhtoan gồm có các phương thức phục vụ cho việc chuyển đổi cũng như việc tính toán:

* Khi bắt đầu bài toán bắt buộc phải nhập một chuỗi để bài toán xử lý và để máy tính có thể hiểu được nhóm đồ án sử dụng phương thức: **public** String Chuanam (String bieuthuc) để chuẩn hóa chuỗi, nhằm xóa đi các kí tự trắng không cần thiết khi nhập biểu thức và một khó khăn là khi ta nhập biểu thức có dấu trừ hoặc dấu âm của một số thì phương thức chuẩn âm làm điều này, mục đích để khiến cho bài toán được xử lí dễ dàng hơn



*Hình 3: Phương thức chuẩn âm*

Có được chuỗi sau khi đã chuẩn âm, Nhóm bắt đầu chuyển đổi đưa các token vào linkedlist bằng phương thức **public** LinkedList<String> ChuanHoa (String bieuthuc)



*Hình 4: Phương thức chuẩn hóa*

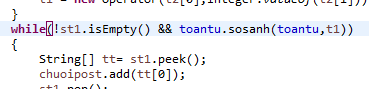
Phương thức chuẩn hóa mục đích đưa từng chuỗi vào linkedlist, một ví dụ cụ thể sin (45) +1 khi ta đưa vào linkedlist kết quả sẽ như sau [sin, (, 45,), +, 1], làm vậy giúp bài toán xác định được phép tính để tính biểu thức ở phía sau đồng thời giải quyết được việc khi nhập vừa số vừa biến (12a)

* Kết thúc phương thức chuẩn âm và chuẩn hóa ta được một chuỗi infix được lưu trong linklist, tiếp đó đến bước thực hiện chuyển infix thành postfix.



*Hình 5: Phương thức chuyển đổi infix thành postfix (một phần code)*

Ở phần này, hai đối tượng được sử dụng là Stack để lưu trữ toán tử và đối tượng Operator được sử dụng lưu trữ sẽ trả ra toán tử và độ ưu tiên, đồng thời so sánh độ ưu tiên đúng với đối tượng ta nêu ở class Operator.



*Hình 6: Sử dụng phương thức so sánh hai operator*

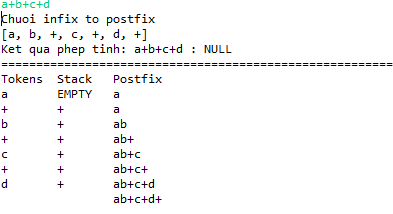
Xử lý độ ưu tiên của toán tử và việc thực hiện push, pop vào Stack trong bài toán đồng thời kết quả chuyển hóa được lưu vào một danh sách liên kết đơn.







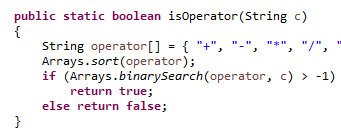
Ở phần chuyển đổi, nhóm xuất từng bước thực hiện của việc chuyển đổi từ infix sang postfix



*Hình 7: Các bước làm của thuật toán chuyển đổi infix thành postfix*

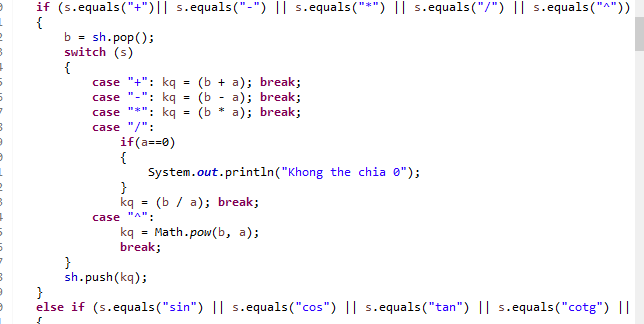
* Tokens là linkedlist đã được chuẩn hóa từ bước trên.
* Stack lưu các toán hạng mô tả từng bước làm của việc chuyển đổi
* Postfix là kết quả thu được trong từng bước chạy đến kết quả cuối cùng khi thu được chuyển Postfix chuyển đổi

Phương thức isOperator (String c) là một phương thức đóng vai trò vô cùng quan trọng trong bài toán, phương thức dùng để xác định trong phần tử trong danh sách liên kết là toán tử hay toán hạng được sử dụng xuyên suốt trong quá trình thực hiện bài toán



*Hình 8: Phương thức kiểm tra toán tử*

* Khi đã chuyển thành postfix, bước tiếp theo là làm sao từ postfix ra được kết quả( tính toán từ chuỗi postfix)



*Hình 9: Tính toán chuỗi postfix*

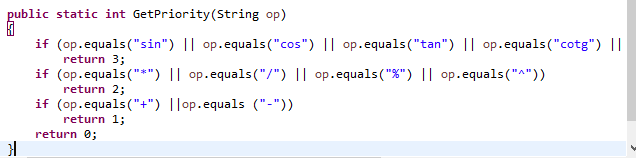
1. Operate.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | Phương thức | Mục đích | Tên file số thức tự dòng khai báo | SV phụ trách |
| 1 | **public char** getkitu()  **public int** getdouutien()  **public** Operator(**char** kitu, **int** douutien)  **public** Operator() | Khai báo operate gồm 2 thành phần toán tử và độ ưa tiên | Dòng 9 tới 29 | Phạm Huỳnh Thanh Lâm |
| 2 | **public void** Chuoi(**char** elementMath)  Input: toán tử | Đưa toán tử vào đối tượng operator | Dòng 31 | Phạm Huỳnh Thanh Lâm |
| 3 | **public** Character[] Xetdouutien(**char** elementMath)  Input: toán tử  Ouput: một mảng 2 phần tử toán tử và độ ưu tiên | Đưa toán tử và độ ưu tiên vào mảng để đưa vào stack | Dòng 38 | Chu Minh Hoàng |
| 4 | **public boolean** sosanh(Operator a, Operator b)  Input: Operate a và b  Ouput: True hay False | So sánh 2 toán tử với nhau coi toán tử a có “<=” toán tử b hay không | Dòng 47 | Phạm Huỳnh Thanh Lâm |
| 5 | **public static int** GetPriority(**char** op)  Input: toán tử char  Ouput: độ ưu tiên int | Gán độ ưu tiên cho toán tử | Dòng 54 | Phạm Huỳnh Thanh Lâm |

*Bảng 4: Bảng thống kê các phương thức trong class Operator*

Dưới đây là một vài giải thích cho các phương thức bên trong lớp Operator:

* Ở đây ta có phương thứcsosanh (Operator a, Operator b) nhằm so sánh các operator là một đối tượng gồm toán tử và độ ưa tiên nhằm xác định đối tượng nào lớn nhất dựa vào độ ưa tiên.
* Ngoài ra ta còn có phương thức:

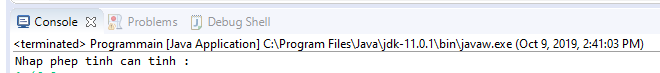


*Hình 10: Đưa ra độ ưu tiên của toán tử*

Ta dùng tính chất so sánh hai chuỗi của String trong java là phương thức “equals”, là phương thức được java hỗ trợ. Ở đây phương thức equals sẽ so sánh một chuỗi với một Object.Kết quả là true nếu và chỉ nếu tham số là không null và là một đối tượng String mà biểu diễn cùng dãy ký tự như đối tượng đó. Ngoài ở đây nếu ta dùng kí hiệu so sánh ‘= =’ sẽ làm cho phần tử op sẽ không nhận được bắt kì chuỗi cần so sánh.

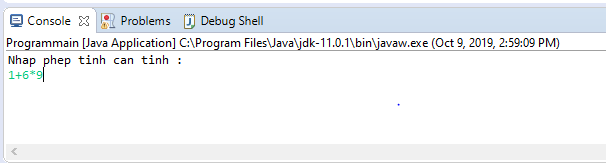
### Thiết kế giao diện

* Dòng in thông báo nhập biểu thức



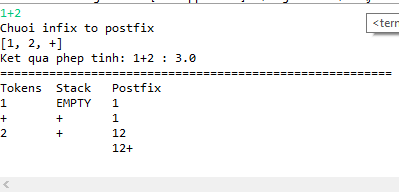
Hình 11: Giao diện màn hình

* Dòng nhập biểu thức



Hình 12: Giao diện nhập biểu thức cần tính

* Dòng in kết quả

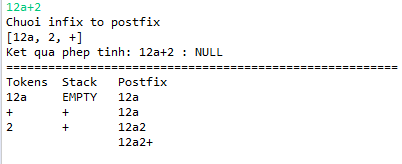


Hình 13: Kết quả của biểu thức sau khi thực hiện

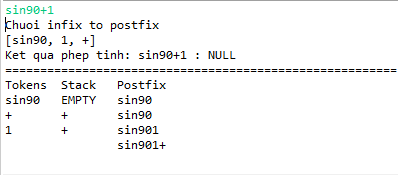
## Cài đặt và kiểm thử

Xét hai trường hợp khi nhập đúng và khi nhập sai:

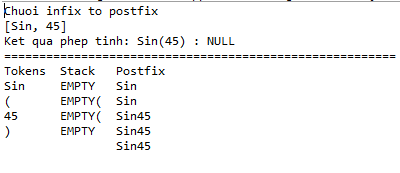
* Các trường hợp sai khi nhập liệu:



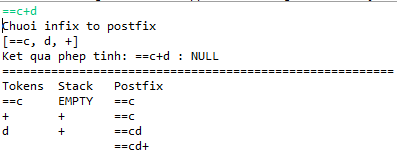
Hình 14: Kết quả sẽ không tính được nếu vừa nhập biến vừa nhập số



Hình 15: Kết quả sẽ ra NULL nếu nhập chuỗi có hàm lượng giác nhưng thiếu dấu ngoặc

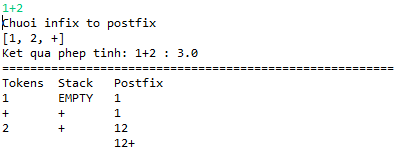


Hình 16: Kết quả sẽ bằng Null nếu nhập liệu hàm lượng giác có chữ viết hoa đầu hoặc viết hoa toàn bộ

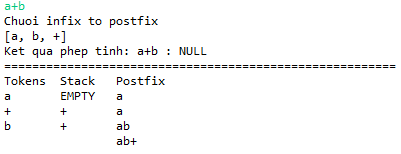


Hình 17: Chưa làm được hàm kiểm tra nhập liệu (nếu có các ký tự là vẫn chuyển đổi thành chuỗi postfix)

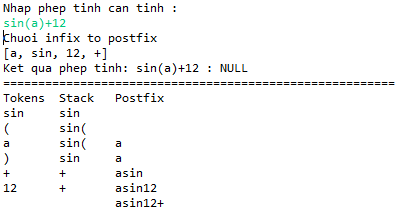
* Kiểm thử các trường hợp nhập đúng



Hình 18: Nhập liệu một biểu thức đơn giản và tính toán

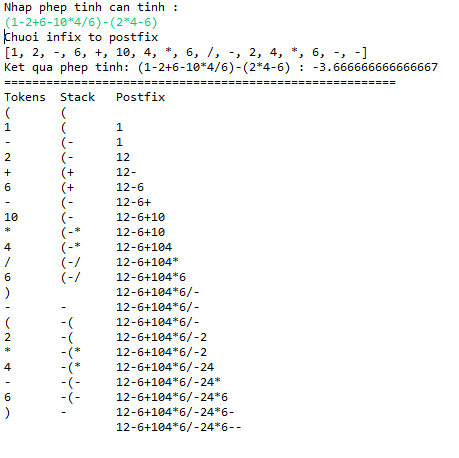


Hình 19: Chuyển hóa chuỗi gồm hai biến a và b

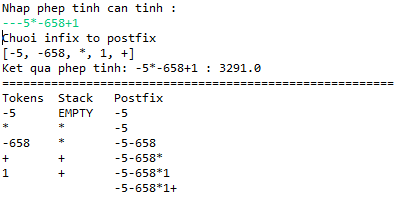


Hình 20: Thực hiện tính toán vừa có biến vừa có số

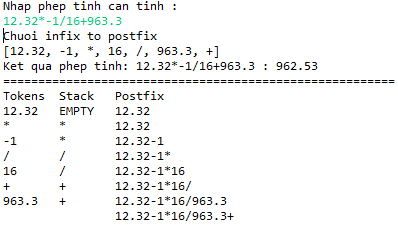
(Biểu thức nhập không thể tính toán được, từ biểu thức trên chỉ có thể chuyển hóa thành chuỗi postfix)



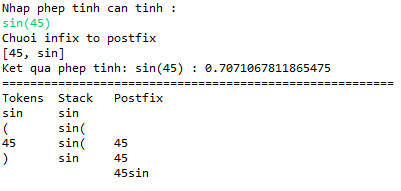
Hình 21: Nhập liệu một biểu thức phức tập gồm nhiều toán tử, kết quả cho ra một số vô tỷ



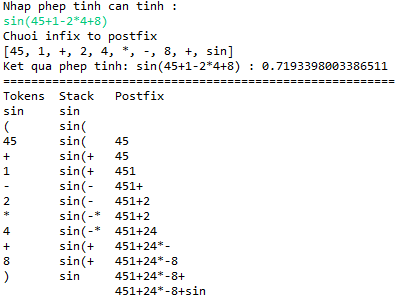
Hình 22: Xử lý và tính số âm



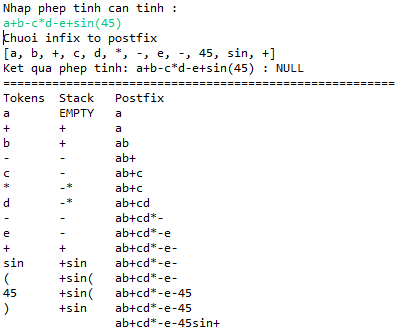
Hình 23: Xử lý số hữu tỉ và phân số



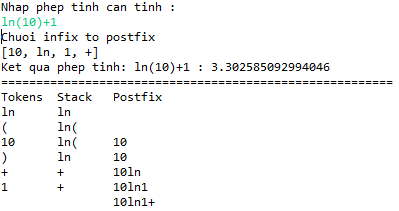
Hình 24: Tính toán hàm lượng giác



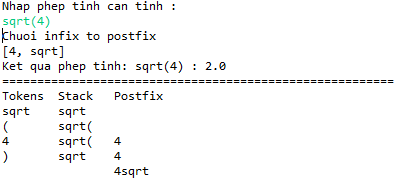
Hình 25: Tính toán hàm lượng giác sử dụng thêm +,-,\*, / bên trong hàm lượng giác đó



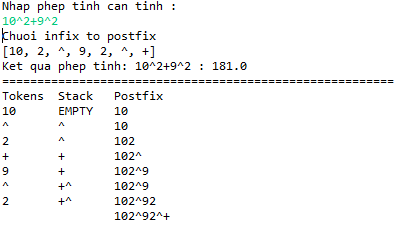
Hình 26: Cộng trừ nhiều biến



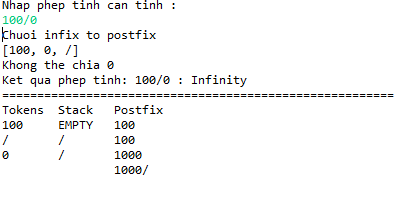
Hình 27: Tính toán biểu thức ln (10) +1



Hình 28: Tính căn của số



Hình 29: Tính toán hàm mũ



*Hình 30: Thông báo khi một số chia cho 0*

# 

# PHẦN KẾT LUẬN

## Kết quả đạt được

Sau quá trình thực hiện đồ án: Ứng dụng phân tích và tính giá trị biểu thức bằng thuật toán RPN, nhóm đã hoàn thành được các mục tiêu:

* Tính toán biểu đơn giản với các phép tính toán học.
* Hiểu rõ được phương pháp chuyển trung tố sang hậu tố (infix => postfix).

Tuy nhiên còn một vài chức năng về phần xuất nhập còn hạn chế, không tính được nếu thiếu dấu ngoặc tròn đối với các hàm đại số, vẫn chưa tối ưu hóa hoàn toàn code

## Kết luận

1. **Ưu điểm**

Sau khi hoàn thành đồ án, nhóm thực hiện đồ án tự nhận thấy đã hoàn thành tốt phần nào những yêu cầu cầu của đề tài đã đặt ra, và cả những mục tiêu mà nhóm đã tự đặt ra:

* Có thể tính toán được các biểu thức khác nhau.
* Tính toán được các biểu thức với nhiều lớp căn một cách chính xác.
* Phân tích chuỗi biểu thức thành chuỗi postfix bất kì cả biến và số.

1. **Nhược điểm**

Tuy hoàn thành nhưng còn một số sai sót:

* Giao diện kém bắt mắt, đơn giản, tính tương tác với người dùng không cao do người dùng chỉ có thể thao tác trên bàn phím.
* Không tính được hàm lượng giác nếu không có ngoặc và không tính được hàm lượng giác nếu viết hoa các chữ cái.
* Nếu nhập thiếu một dấu ngoặc hay sai, chương trình sẽ không tính được và ra Null.
* Không tính được nếu nhập sai chữ kí hiệu toán học.

# TÀI LIỆU THAM KHẢO

-Tham khảo thuật toán:

[1] <https://yinyangit.wordpress.com/2011/01/26/algorithm-chuy%E1%BB%83n-bi%E1%BB%83u-th%E1%BB%A9c-trung-t%E1%BB%91-sang-ti%E1%BB%81n-t%E1%BB%91-va-h%E1%BA%ADu-t%E1%BB%91-b%E1%BA%B1ng-stack/>

-Tham khảo phần xét toán tử:

[2] <https://cachhoc.net/2013/07/07/thuat-toan-java-chuyen-bieu-thuc-trung-to-sang-hau-to-java-converts-infix-to-postfix/>