**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**BÁO CÁO PROJECT CUỐI KỲ**

**Thực hành kiến trúc máy tính**

**Mã học phần: IT3280**

Họ và tên sinh viên: Phạm Trần Nguyên Phú

GV hướng dẫn: Hoàng Văn Hiệp

**5. Biểu thức trung tố hậu tố**

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: 9 + 2 + 8 \* 6

2. In ra biểu thức ở dạng hậu tố, ví dụ: 9 2 + 8 6 \* +

3. Tính ra giá trị của biểu thức vừa nhập

Các hằng số là số nguyên, trong phạm vi từ 0 → 99.

Toán tử bao gồm phép cộng, trừ, nhân, chia lấy thương

* Mã nguồn:

.data

infix: .asciiz "9+2+8\*6"

postfix: .space 100

msg\_result: .asciiz "Gia tri cua bieu thuc: "

.text

main:

# Chuyển đổi biểu thức trung tố sang hậu tố

la $a0, infix # Tải địa chỉ biểu thức trung tố

la $a1, postfix # Tải địa chỉ để lưu biểu thức hậu tố

jal infixToPostfix # Gọi hàm chuyển đổi

# Tính giá trị biểu thức hậu tố

la $a0, postfix # Tải địa chỉ biểu thức hậu tố

jal evaluatePostfix # Gọi hàm tính giá trị

# In kết quả

li $v0, 4 # Syscall để in chuỗi

la $a0, msg\_result # Tải chuỗi "Gia tri cua bieu thuc: "

syscall

li $v0, 1 # Syscall để in số nguyên

move $a0, $v1 # Giá trị kết quả (lưu trong $v1)

syscall

li $v0, 10 # Thoát

syscall

# Hàm chuyển đổi trung tố sang hậu tố

infixToPostfix:

la $t0, 0($a0) # Địa chỉ biểu thức trung tố

la $t1, 0($a1) # Địa chỉ để lưu biểu thức hậu tố

la $t2, 0x10010000 # Stack để lưu toán tử

addi $sp, $sp, -4 # Khởi tạo stack

convert\_loop:

lb $t3, 0($t0) # Đọc ký tự tiếp theo

beq $t3, $zero, end\_convert\_loop # Nếu là ký tự null thì kết thúc

# Nếu là số, sao chép vào biểu thức hậu tố

blt $t3, '0', check\_operator

bgt $t3, '9', check\_operator

sb $t3, 0($t1)

addi $t1, $t1, 1

j next\_char

check\_operator:

# Nếu là toán tử, xử lý độ ưu tiên

beq $t3, '(', push\_operator

beq $t3, ')', close\_parenthesis

# Xử lý các toán tử khác

handle\_operator:

lw $t4, 0($sp) # Lấy toán tử trên cùng của stack

beq $t4, $zero, push\_operator

# Kiểm tra độ ưu tiên

li $t5, 1 # Độ ưu tiên của + và -

li $t6, 2 # Độ ưu tiên của \* và /

beq $t3, '+', check\_priority

beq $t3, '-', check\_priority

move $t5, $t6

beq $t3, '\*', check\_priority

beq $t3, '/', check\_priority

check\_priority:

lw $t7, 0($sp)

beq $t7, '(', push\_operator

beq $t7, $zero, push\_operator

li $t8, 1 # Độ ưu tiên của + và -

li $t9, 2 # Độ ưu tiên của \* và /

beq $t7, '+', compare\_priority

beq $t7, '-', compare\_priority

move $t8, $t9

beq $t7, '\*', compare\_priority

beq $t7, '/', compare\_priority

compare\_priority:

bge $t8, $t5, pop\_operator

push\_operator:

addi $sp, $sp, -4

sw $t3, 0($sp)

j next\_char

pop\_operator:

lw $t4, 0($sp)

addi $sp, $sp, 4

sb $t4, 0($t1)

addi $t1, $t1, 1

j handle\_operator

close\_parenthesis:

lw $t4, 0($sp)

beq $t4, '(', next\_char

addi $sp, $sp, 4

sb $t4, 0($t1)

addi $t1, $t1, 1

j close\_parenthesis

next\_char:

addi $t0, $t0, 1

j convert\_loop

end\_convert\_loop:

lw $t4, 0($sp)

beq $t4, $zero, end\_convert

addi $sp, $sp, 4

sb $t4, 0($t1)

addi $t1, $t1, 1

j end\_convert\_loop

end\_convert:

sb $zero, 0($t1) # Kết thúc chuỗi hậu tố

jr $ra

# Hàm tính giá trị biểu thức hậu tố

evaluatePostfix:

la $t0, 0($a0) # Địa chỉ biểu thức hậu tố

la $t1, 0x10010000 # Khởi tạo Stack Pointer tại địa chỉ an toàn

eval\_loop:

lb $t2, 0($t0) # Đọc ký tự tiếp theo

beq $t2, $zero, end\_eval\_loop # Nếu là ký tự null thì kết thúc

# Nếu là số, đẩy vào stack

blt $t2, '0', check\_eval\_operator

bgt $t2, '9', check\_eval\_operator

sub $t2, $t2, '0' # Chuyển đổi ký tự thành số

sw $t2, 0($t1)

addi $t1, $t1, 4

j next\_eval\_char

check\_eval\_operator:

# Xử lý toán tử

addi $t1, $t1, -4

lw $t3, 0($t1) # Pop op2

addi $t1, $t1, -4

lw $t4, 0($t1) # Pop op1

# Thực hiện phép toán

beq $t2, '+', eval\_add

beq $t2, '-', eval\_sub

beq $t2, '\*', eval\_mul

beq $t2, '/', eval\_div

eval\_add:

add $t5, $t4, $t3

j eval\_push\_result

eval\_sub:

sub $t5, $t4, $t3

j eval\_push\_result

eval\_mul:

mul $t5, $t4, $t3

j eval\_push\_result

eval\_div:

div $t4, $t3

mflo $t5

j eval\_push\_result

eval\_push\_result:

sw $t5, 0($t1)

addi $t1, $t1, 4

next\_eval\_char:

addi $t0, $t0, 1

j eval\_loop

end\_eval\_loop:

addi $t1, $t1, -4

lw $v1, 0($t1) # Lấy kết quả cuối cùng

jr $ra

* Kết quả sau khi thực thi chương trình:

A screenshot of a computer

Description automatically generated

* Giải thích:

 **main**: Hàm chính của chương trình.

* Chuyển đổi biểu thức trung tố sang hậu tố:
  + Tải địa chỉ của biểu thức trung tố và biểu thức hậu tố.
  + Gọi hàm infixToPostfix để chuyển đổi biểu thức.
* Tính giá trị của biểu thức hậu tố:
  + Tải địa chỉ của biểu thức hậu tố.
  + Gọi hàm evaluatePostfix để tính toán kết quả.
* In kết quả:
  + In chuỗi "Giá trị của biểu thức: ".
  + In giá trị của biểu thức tính toán.
* Thoát.

 **infixToPostfix**: Hàm chuyển đổi biểu thức trung tố sang hậu tố.

* Sử dụng một ngăn xếp để lưu trữ các toán tử trong quá trình chuyển đổi.
* Duyệt qua từng ký tự trong biểu thức trung tố:
  + Nếu là số, thêm vào biểu thức hậu tố.
  + Nếu là toán tử, thực hiện xử lý độ ưu tiên và đưa vào biểu thức hậu tố hoặc ngăn xếp.
* Khi gặp dấu đóng ngoặc, đẩy toàn bộ các toán tử từ ngăn xếp vào biểu thức hậu tố cho đến khi gặp dấu mở ngoặc.
* Kết thúc khi duyệt hết biểu thức trung tố.

 **evaluatePostfix**: Hàm tính giá trị của biểu thức hậu tố.

* Sử dụng một ngăn xếp để lưu trữ các số và thực hiện phép tính.
* Duyệt qua từng ký tự trong biểu thức hậu tố:
  + Nếu là số, đẩy vào ngăn xếp.
  + Nếu là toán tử, lấy hai số từ ngăn xếp, thực hiện phép tính và đẩy kết quả vào ngăn xếp.
* Kết thúc khi duyệt hết biểu thức hậu tố, kết quả cuối cùng là số trên đỉnh của ngăn xếp.
* Thuật toán:
  + **Chuyển đổi trung tố sang hậu tố (‘infixToPostfix’):**

Giả sử biểu thức trung tố là "9+2+8\*6":

* **Bước 1**: Đọc '9' (số) -> Thêm vào biểu thức hậu tố: "9"
* **Bước 2**: Đọc '+' (toán tử) -> Đẩy vào stack: stack = ['+']
* **Bước 3**: Đọc '2' (số) -> Thêm vào biểu thức hậu tố: "92"
* **Bước 4**: Đọc '+' (toán tử):
  + Toán tử trên đỉnh stack là '+' với độ ưu tiên bằng với toán tử hiện tại.
  + Đẩy '+' từ stack vào biểu thức hậu tố và sau đó đẩy toán tử mới vào stack: "92+" ; stack = ['+']
* **Bước 5**: Đọc '8' (số) -> Thêm vào biểu thức hậu tố: "92+8"
* **Bước 6**: Đọc '\*' (toán tử):
  + '*' có độ ưu tiên cao hơn '+' nên đẩy '*' vào stack: stack = ['+', '\*']
* **Bước 7**: Đọc '6' (số) -> Thêm vào biểu thức hậu tố: "92+86"
* **Bước 8**: Hết chuỗi, đẩy tất cả toán tử còn lại từ stack vào biểu thức hậu tố: "92+86\*+"

Vậy biểu thức hậu tố là "92+86\*+".

* + **Tính giá trị biểu thức hậu tố (‘evaluatePostfix’):**

Giả sử biểu thức hậu tố là "92+86\*+":

* **Bước 1**: Đọc '9' (số) -> Đẩy vào stack: stack = [9]
* **Bước 2**: Đọc '2' (số) -> Đẩy vào stack: stack = [9, 2]
* **Bước 3**: Đọc '+' (toán tử):
  + Lấy 2 và 9 từ stack, thực hiện phép cộng: 9 + 2 = 11
  + Đẩy kết quả vào stack: stack = [11]
* **Bước 4**: Đọc '8' (số) -> Đẩy vào stack: stack = [11, 8]
* **Bước 5**: Đọc '6' (số) -> Đẩy vào stack: stack = [11, 8, 6]
* **Bước 6**: Đọc '\*' (toán tử):
  + Lấy 6 và 8 từ stack, thực hiện phép nhân: 8 \* 6 = 48
  + Đẩy kết quả vào stack: stack = [11, 48]
* **Bước 7**: Đọc '+' (toán tử):
  + Lấy 48 và 11 từ stack, thực hiện phép cộng: 11 + 48 = 59
  + Đẩy kết quả vào stack: stack = [59]
* **Bước 8**: Kết quả cuối cùng là số trên đỉnh stack: 59

Vậy giá trị của biểu thức hậu tố "92+86\*+" là 59.