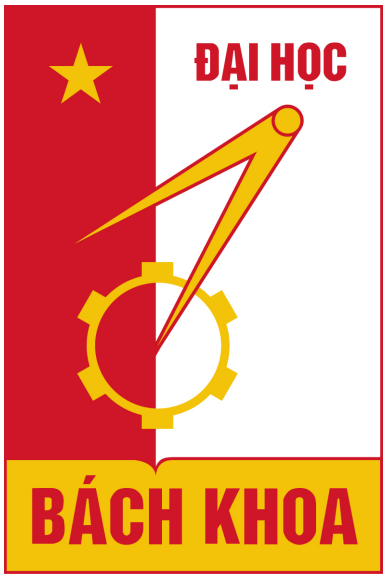
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

o0o



Thực hành kiến trúc máy tính - IT3280

Biểu thức trung tố hậu tố

Giáo viên hướng dẫn: Hoàng Văn Hiệp
Sinh viên thực hiện : Lê Thành Đạt - 20225804
Lớp : CNTT Việt Nhật K67

Hà Nội - 2024

Mục lục

1 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:

- Nhập vào biểu thức trung tố, ví dụ: 9 + 2 + 8*6
- In ra biểu thức ở dạng hậu tố, ví dụ: 9 2 + 8 6 * +;
- 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.

2 Thuật toán

2.1 Dữ liệu và cấu trúc

Dữ liệu:

- **expression**: Vùng nhớ để lưu trữ biểu thức của người dùng nhập vào
- **result_msg**: Chuỗi thông báo kết quả "The result is: "
- **msg**: Chuỗi thông báo nhập "Input: "
- **error_msg**: Chuỗi thông báo lỗi "Input is invalid"

Cấu trúc:

- Sử dụng stack để lưu trữ các toán hạng và kết quả trung gian.
- Các hàm con thực hiện kiểm tra toán tử, thực hiện phép tính và tính toán giá trị của biểu thức hậu tố.

2.2 Các hàm trong chương trình

2.2.1 Hàm isOperator

Hàm này kiểm tra xem ký tự có phải là một toán tử hay không.

- Lưu trữ các thanh ghi vào stack.
- So sánh ký tự với các toán tử $(+, -, *, /,)$.
- Trả về 1 nếu ký tự là toán tử, ngược lại trả về 0.

2.2.2 Hàm compute

Hàm này thực hiện các phép toán dựa trên toán tử và các toán hạng đã cho.

- Lưu trữ các thanh ghi vào stack.
- Thực hiện các phép toán $(+, -, *, /,)$ dựa trên các toán tử.
- Trả về kết quả của phép toán

2.2.3 Hàm tinhGiaTriBieuThucHauTo

Hàm này tính toán giá trị của biểu thức hậu tố

- Lưu trữ giá trị của thanh ghi ra vào stack.
- lặp qua từng ký tự của biểu thức.
- Kiểm tra nếu ký tự là toán tử, thực hiện phép toán bằng hàm **compute** và lưu kết quả vào stack.
- Nếu ký tự là toán hạng, chuyển đổi ký tự sang số nguyên và lưu vào stack.
- Sau khi duyệt hết biểu thức, in kết quả ra màn hình.

2.2.4 Các hàm phụ trợ

- **print_result**: In kết quả ra màn hình.
- **error**: Hiển thị thông báo lỗi và quay lại đầu chương trình

2.3 Mô tả thuật toán

- Khởi tạo: Tạo một ngăn xếp trống để lưu trữ các toán hạng.
- Duyệt qua biểu thức: Đọc từng phần tử (toán hạng hoặc toán tử) từ trái sang phải trong biểu thức hậu tố.
- Xử lý từng phần tử:
  - Nếu phần tử là toán hạng: Đẩy toán hạng đó vào ngăn xếp.
  - Nếu phần tử là toánt tử (như $+, -, *, /,$):
    - Lấy hai toán hạng từ đầu ngăn xếp ra (toán hạng thứ nhất và toán hạng thứ hai).
    - Thực hiện phép toán tương ứng với hai toán hạng vừa lấy ra.
    - Đẩy kết quả của phép toán vào lại ngăn xếp.
- Kết thúc: Khi đã duyệt hết tất cả các phần tử trong biểu thức, kết quả của biểu thức sẽ nằm ở đầu ngăn xếp.

3 Code

```
.data
    expression: .space 100
    result_msg: .asciiz "The result is: "
    msg: .asciiz "Input: "
    error_msg: .asciiz "Input is invalid"
    nline: .asciiz "\n"
.text
move $fp, $sp
main:
    move $sp, $fp
    li $v0, 4
    la $a0, msg
    syscall   #Show the string

    li $v0, 8
    la $a0, expression
    li $a1, 100
    syscall   #Nhap bieu thuc

    lb $s7, 0($a0)
    beq $s7, 'e', exit

    jal tinhGiaTriBieuThucHauto

isOperator:
#luu tru thanh ghi a0 va ra vao stack
    addi $sp, $sp, -8
    sw $a0, 0($sp)
    sw $ra, 4($sp)

#Kiem tra toan tu
    li $t0, '+'
    beq $a0, $t0, isOperatorTrue
    li $t0, '-'
    beq $a0, $t0, isOperatorTrue
    li $t0, '*'
    beq $a0, $t0, isOperatorTrue
    li $t0, '/'
    beq $a0, $t0, isOperatorTrue
    li $t0, '^'
    beq $a0, $t0, isOperatorTrue

#Neu la toan tu thi tra ve 1 nguoc lai ve 0
isOperatorFalse:
    li $v0, 0
    j endCheck
```

```
isOperatorTrue:
    li $v0, 1

endCheck:
    lw $a0, 0($sp)
    lw $ra, 4($sp)
    addi $sp, $sp, 8
    jr $ra

compute:
#Luu tru cac thanh ghi vao stack
    addi $sp, $sp, -16
    sw $a0, 0($sp)
    sw $a1, 4($sp)
    sw $a2, 8($sp)
    sw $ra, 12($sp)

#Thuc hien cac phep toan
    li $t0, '+'
    beq $a2, $t0, compute_add
    li $t0, '-'
    beq $a2, $t0, compute_sub
    li $t0, '*'
    beq $a2, $t0, compute_mul
    li $t0, '/'
    beq $a2, $t0, compute_div
    li $t0, '^'
    beq $a2, $t0, compute_pow

compute_add:
    add $v0, $a0, $a1
    j compute_end

compute_sub:
    sub $v0, $a0, $a1
    j compute_end

compute_mul:
    mul $v0, $a0, $a1
    j compute_end

compute_div:
    beq $a1, 0, error
    div $a0, $a1
    mflo $v0
    j compute_end
```

```
compute_pow:
    addi $t1, $a0, 0
    addi $t2, $a1, 0
    li $v0, 1
    pow_loop:
        beq $t2, $zero, pow_end
        mul $v0, $v0, $t1
        addi $t2, $t2, -1
        j pow_loop
    pow_end:
        j compute_end

compute_end:
    lw $a0, 0($sp)
    lw $a1, 4($sp)
    lw $a2, 8($sp)
    lw $ra, 12($sp)
    addi $sp, $sp, 16
    jr $ra

tinhGiaTriBieuThucHauto:
    addi $sp, $sp, -4
    sw $ra, 0($sp)
    la $s0, expression
    li $t0, 0
    li $t1, 1
#Lap qua tung ki tu cua bieu thuc
evaluate:
    lb $t2, 0($s0)
    beq $t2, '\n', print_result
    beq $t2, $zero, print_result
    beq $t2, ' ', next_char
    move $a0, $t2
    jal isOperator

    beq $v0, $zero, operand
#day la toan tu
    addi $sp, $sp, 4
    lw $t4, 0($sp)
    addi $sp, $sp, 4
    lw $t3, 0($sp)

    move $a0, $t3
    move $a1, $t4
    move $a2, $t2
    jal compute

```

```
146 operand:
148     sub $s5, $t2, $s4
149     blt $s5, 0, error
150     bgt $s5, 9, error
151     li $t5, 0
152 #Luu tru cac ki tu co nhieu chu so
153 num_loop:
154     beq $t2, ' ', end_num
155     sub $t2, $t2, '0'
156     mul $t5, $t5, 10
157     add $t5, $t5, $t2
158     addi $s0, $s0, 1
159     lb $t2, 0($s0)
160     j num_loop
161 end_num:
162     sw $t5, 0($sp)
163     addi $sp, $sp, -4
164     addi $s0, $s0, -1
165
166 next_char:
167     addi $s0, $s0, 1
168     j evaluate
169
170 print_result:
171     addi $sp, $sp, 4
172     lw $v1, 0($sp)
173     li $v0, 4
174     la $a0, result_msg
175     syscall
176
177     li $v0, 1
178     move $a0, $v1
179     syscall
180
181     li $v0, 4
182     la $a0, nline
183     syscall
184
185     j main
186
187 error:
188     li $v0, 4
189     la $a0, error_msg
190     syscall
191     j main
192
193 exit:
194     li $v0, 10
```