Bài tập thực hành

Các thuật toán sắp xếp

1. MỤC TIÊU

Trong bài tập này, sinh viên thực hành cài đặt cách thuật toán liên quan đến danh sách liên kết (DSLK). Sẽ có hai bài tập:

- Cài đặt thuật toán Radix sort bằng DSLK
- Tính giá trị biểu thức

2. Radix sort

Sinh viên đã cài đặt thuật toán Radix sort trong bài tập trước bằng cách dùng mảng. Tuy nhiên, cách cài đặt bằng DSLK sẽ phản ánh tốt hơn ý tưởng thuật toán Radix Sort. Trong bài tập này, sinh viên sẽ cài đặt lại thuật toán Radix sort dùng DSLK để sắp xếp một mảng số nguyên (có thể có số âm)

- Yêu cầu người dùng nhập vào số lượng phần tử.
- Random dãy số nguyên có số lượng phần tử tương ứng
- Sắp xếp dãy số nguyên bằng phương pháp Radix Sort

3. Tính giá trị biểu thức.

Viết chương trình cho phép nhập vào một biểu thức toán học từ file input, tính giá trị của biểu thức này và xuất kết quả ra file output. Biểu thức có chứa các phép tính:

+	Cộng hai toán hạng
-	Trừ hai toán hạng
-	Số âm
*	Nhân hai toán hạng
/	Chia hai toán hạng
^	Lũy thừa
!	Giai thừa

(Mở ngoặc

) Đóng ngoặc

Giữa các token có thể có khoảng trắng hoặc không. VD: (- (5.5 + 8) * 4.9 + (11 - 3) * 4) * -9 + 12 ^ 2 + 3!

Lưu ý: chương trình phải xử lý được trong trường hợp toán hạng là ***số thực***

Nhắc lại:

- Biểu thức trung tố: Toán tử nằm giữa 2 toán hạng. VD: x + y
- Biểu thức hậu tố: Toán tử nằm sau 2 toán hạng. VD: x y +
- Trong bài viết này mỗi thành phần riêng lẻ trong biểu thức được gọi là Token.

Như trong ví dụ trên thì (, 5, +, 8,), *, 12, +, 11… là các Token..

Bước 1:

Dùng Stack và Queue để chứa các Token. Các thao tác:

- void Push(Token, Stack) //Thêm phần tử vào Stack
- void Push(Token, Queue) //Thêm phần tử vào Queue
- Token Pop(Stack) //Lấy phần tử ở đỉnh khỏi Stack
- Token Pop(Queue) //Lấy phần tử ở đỉnh ra khỏi Queue
- Token Top(Stack) //Xem giá trị phần tử ở đỉnh (không Pop).
- bool IsEmpty(Stack) //Kiểm tra Stack rỗng
- bool IsEmpty(Queue) //Kiểm tra Queue rỗng

```
struct Token
{
    int Kieu;
    int GiaTri;
};
```

Thuật toán chuyển từ dạng trung tố sang hậu tố

Input: Biểu thức

Output: Queue biểu diễn biểu thức ở dạng hậu tố

```
- Khởi động Stack rỗng
- Khởi động Queue rỗng
while (chưa hết biểu thức)
{
        - Đọc một Token
        - Nếu Token là:
                o Ngoặc trái: Push (Token, Stack).
                o Ngoặc phải: lặp lại thao tác Push(Pop(Stack), Queue) đến khi gặp
ngoặc trái. (pop cả ngoặc trái nhưng không push nó vào Queue).
                o Toán tử:
                        While (Token <= Top(Stack) or !IsEmpty(Stack))
                                Push(Pop(Stack), Queue)
                        Push(Token, Stack)
                o Toán hạng : Push(Token, Queue).
}
while (!isEmpty (Stack))
        Push(Pop(Stack), Queue))
```

Bước 2

Thuật toán tính giá trị biểu thức hậu tố:

Input: Queue biểu diễn biểu thức ở dạng hậu tố

Output: Giá trị biểu thức

```
- Khởi động Stack rỗng.
while (!isEmpty(Queue))
{
        x = Pop(Queue)
        Nếu x là :
                - Toán hạng: Push(X, Stack)
                - Toán tử:
        U = Pop(Stack)
        V = Pop(Stack)
                Y = X(V, U) // Thực hiện toán tử X cho 2 toán hạng trong Stack.
                Push(Y, Stack)
}
return Pop(Stack);
```

Ví dụ: Biểu thức: 11 – 9 + (5 - 8) * 12

Bước 1: chuyển sang dạng hậu tố

11: Toán hạng

Stack	Queue
	11

- : Toán tử

Stack	Queue
-	11

9: Toán hạng

Stack	Queue
-	11 9

+: Toán tử: độ ưu tiên bằng dấu -

Stack	Queue
+	11 9 -

(: dấu mở ngoặc

Stack	Queue
(+	11 9 -

5: toán hạng

Stack	Queue
(+	11 9 - 5

-: Toán tử: độ ưu tiên hơn dấu mở ngoặc

Stack	Queue
- (+	11 9 - 5

8: Toán hạng

Stack	Queue
- (+	11 9 - 5 8

): dấu đóng ngoặc

Stack	Queue
+	11 9 - 5 8 -

*: toán tử ưu tiên hơn dấu -

Stack	Queue
* +	11 9 - 5 8 -

12: Toán hạng

Stack	Queue
* +	11 9 - 5 8 - 12

Hết biểu thức

Stack	Queue
	11 9 - 5 8 -

	12 * +

Bước 2: Tính giá trị biểu thức:

Stack	Queue
	11 9 - 5 8 - 12 * +

Stack	Queue
11	9 - 5 8 - 12 * +

Stack	Queue
9 11	- 5 8 - 12 * +

Stack	Queue
2	5 8 - 12

	* +

Stack	Queue
5 2	8 - 12 * +

Stack	Queue
8 5 2	- 12 * +

Stack	Queue
-3 2	12 * +

Stack	Queue
12 -3 2	* +

Stack	Queue
-36 2	+

Stack	Queue
-34	

4. Qui định nộp

- Sinh viên nộp một tập tin nén, có tên là <***MSSV***>***.zip*** hoặc <***MSSV***>***.rar*** chứa source code và báo cáo của chương trình.

- File report định dạng pdf, trong đó trình bày rõ cấu trúc chương trình, các hàm được sử dụng trong chương trình
- Chương trình sẽ được chạy bằng tham số dòng lệnh, Ví dụ: balan.exe test1.txt, trong đó:

balan.exe là file thực thi của chương trình

test1.txt chứa bộ test. Mỗi bộ test chỉ gồm một dòng chứa biểu thức toán học cần tính giá trị bằng balan ngược.

Chương trình phải kiểm tra được biểu thức đầu vào có hợp lệ hay không, và báo lỗi nếu biểu thức không hợp lệ

Yêu cầu input và output của chương trình:

- Input: File chứa biểu thức toán học, như mô tả đã được đề cập ở phần mở đầu.
- Output: kết quả tính toán của biểu thức toán học.

Bài giống nhau hay nộp file rác sẽ 0 điểm MÔN HỌC.