**ỨNG DỤNG CỦA STACK:**

*Stack thích hợp lưu trữ các loại dữ liệu mà trình tự truy xuất ngược với trình tự lưu trữ*

**Một số ứng dụng của Stack:**

* Trong trình biên dịch (thông dịch), khi thực hiện các thủ tục, Stack được sử dụng để lưu môi trường của các thủ tục
* Lưu dữ liệu khi giải một số bài toán của lý thuyết đồ thị (như tìm đường đi)
* Khử đệ qui
* Ứng dụng trong các bài toán tính toán biểu thức

**Ví dụ: thủ tục Quick\_Sort dùng Stack để khử đệ qui:**

*Bước 1*. l=1; r=n;

*Bước 2.* Chọn phần tử giữa x=a[(l+r) / 2]

*Bước 3.* Phân hoạch (l, r) thành (l1, r1) và (l2, r2) bằng cách xét:

* y thuộc (l1, r1) nếu yx
* y thuộc (l2, r2) ngược lại

*Bước 4.* Nếu phân hoạch (l2, r2) có nhiều hơn 1 phần tử thì thực hiện:

Cất (l2, r2) vào Stack

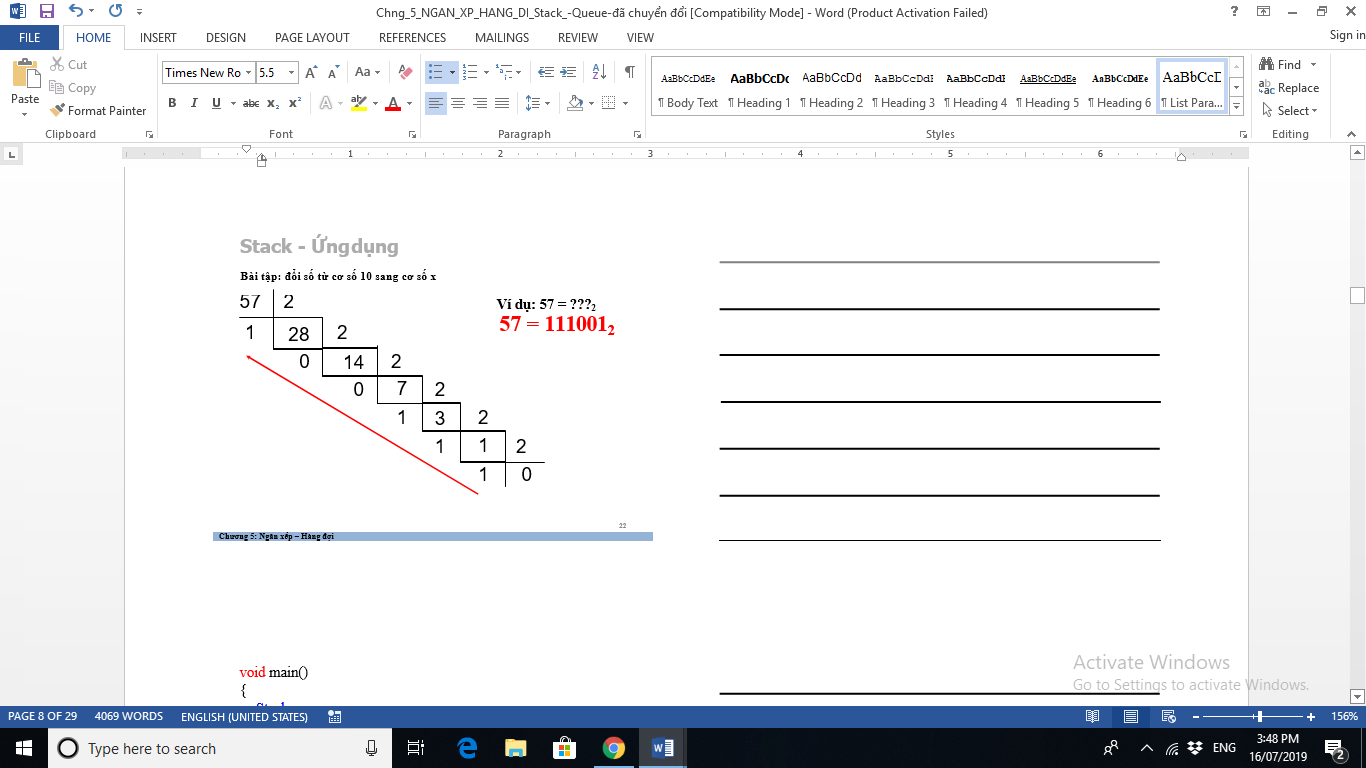
Nếu (l1, r1) có nhiều hơn 1 phần tử thì thực hiện:

* l = l1
* r = r1

Quay lên bước 2.

Ngược lại: Lấy (l, r) ra khỏi Stack, nếu Stack khác rỗng thì quay lên bước 2, ngược lại thì dừng.

**Ứng dụng: Chuyển cơ số 10 sang cơ số x nhập từ bàn phím**



***Gợi ý:***

void main()

{

Stack s;

int coso, so, sodu; Init(s);

// Nhập số cần chuyển vào so …

// Nhập cơ số cần chuyển vào coso…

while (so != 0)

{

sodu = so % coso;

**Push** (s, sodu); *// push so du vao stack*

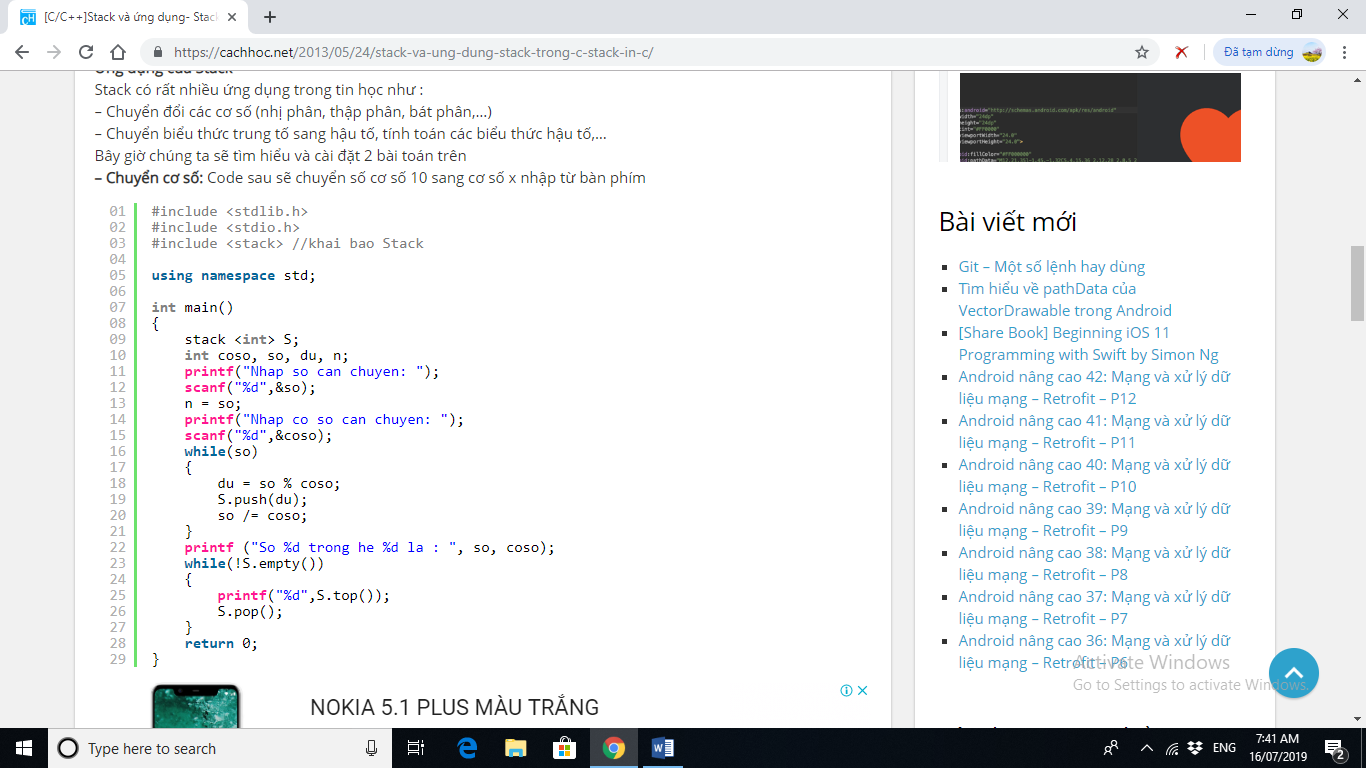
so = so/coso;

}

cout<<"Kết quả: "; while (!**isEmpty**(s))

cout<<**Pop**(s); *// pop so du ra khoi stack*

}



**Ứng dụng: Thuật toán Ba Lan ngược (Reverse Polish Notation – RPN)**

*Định nghĩa RPN:* Biểu thức toán học trong đó các toán tử được viết sau toán hạng và không dùng dấu ngoặc.

Phát minh bởi Jan Lukasiewics một nhà khoa học Ba Lan vào những năm 1950

#### Infix : toán tử viết giữa toán hạng

#### Postfix (RPN): toán tử viết sau toán hạng

#### Prefix : toán tử viết trước toán hạng

###### Ví dụ:

**INFIX RPN (POSTFIX) PREFIX**

|  |  |  |
| --- | --- | --- |
| **A + B**  **A \* B + C**  **A \*(B + C)** | **A B +**  **A B \* C +**  **A B C + \*** | **+ A B**  **+ \* A B C**  **\* A + B C** |
| **A - (B - (C - D))** | **A B C D - - -** | **- A - B - C D** |
| **A - B - C - D** | **A B - C - D -** | **- - - A B C D** |

***Kỹ thuật gạch dưới*:**

###### Duyệt từ trái sang phải của biểu thức cho đến khi gặp toán tử. Gạch dưới 2 toán hạng ngay trước toán tử và kết hợp chúng bằng toán tử trên

Lặp đi lặp lại cho đến hết biểu thức.

**Ví dụ 2\*((3+4)-(5-6))**

* **2 3 4 + 5 6 - - \***

** 2 3 4 + 5 6 - - \***

** 2 7 5 6 - - \***

** 2 7 5 6 - - \***

** 2 7 -1 -**

* **2 7 -1 - \***

** 2 8 \***

** 2 8 \***

** 16**

***Thuật toán: tính giá trị của biểu thức RPN/hậu tố***

Khởi tạo Stack rỗng *(chứa hằng hoặc biến).*

Lặp cho đến khi kết thúc biểu thức:

Đọc 01 phần tử của biếu thức *(hằng, biến, phép toán)*.

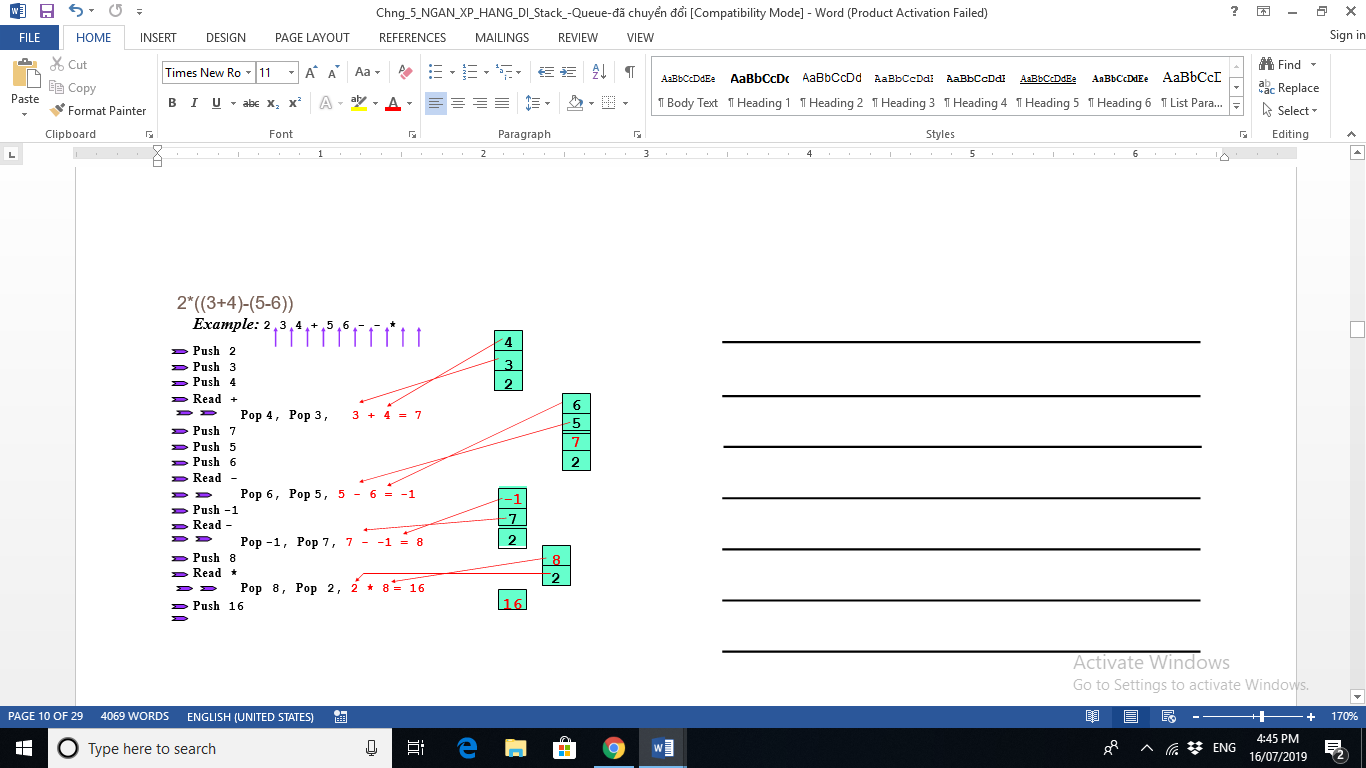
*Nếu phần tử là hằng hay biến: đưa vào Stack.*

*Ngược lại:*

*Lấy ra 02 phần tử của Stack.*

*Áp dụng phép toán cho 02 phần tử vừa lấy ra. Đưa kết quả vào Stack.*

Giá trị của biểu thức chính là phần tử cuối cùng của Stack.



**Ứng dụng: Chuyển infix thành postfix**

Khởi tạo Stack rỗng (chứa các phép toán).

Lặp cho đến khi kết thúc biểu thức:

Đọc 01 phần tử của biếu thức *(01 phần tử có thể là hằng, biến, phép toán, “)”*

*hay “(” ).*

Nếu phần tử là:

* + - **“(”**: đưa vào Stack.
    - **“)”**: lấy các phần tử của Stack ra cho đến khi gặp “(” trong Stack.

##### Một phép toán: + - \* / Nếu Stack rỗng: đưa vào Stack.

* + *Nếu Stack khác rỗng và* ***phép toán có độ ưu tiên cao hơn phần tử ở đầu Stack****: đưa vào Stack.*
  + *Nếu Stack khác rỗng và phép toán có độ ưu tiên thấp hơn hoặc bằng phần tử ở đầu Stack: lấy phần tử từ Stack ra; sau đó lặp lại việc so sánh với phần tử ở đầu Stack.*
    - Hằng hoặc biến: đưa vào kết quả.

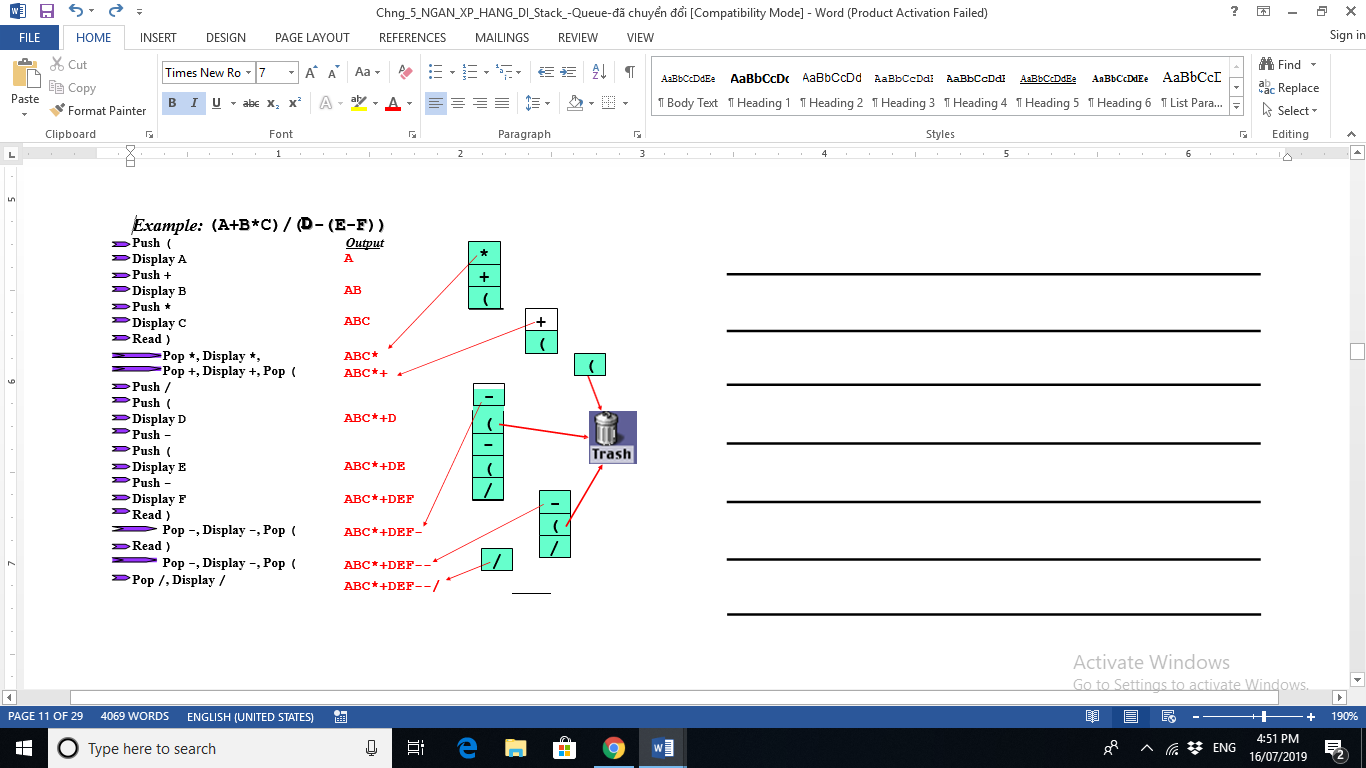
Lấy hết tất cả các phần tử của Stack ra.

**Độ ưu tiên**

+ , \_ 1

\*, / 2

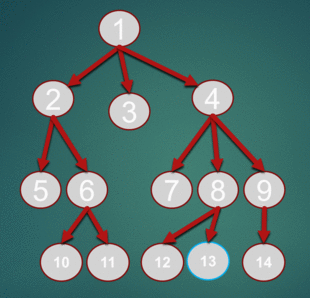
^ 3



**Queue - Ứng dụng**

Queue có thể được sử dụng trong một số bài toán:

* Bài toán “sản xuất và tiêu thụ” (ứng dụng trong các hệ điều hành song song)
* Bộ đệm (ví dụ: Nhấn phím 🡪 Bộ đệm 🡪 CPU xử lý)
* Xử lý các lệnh trong máy tính (ứng dụng trong HĐH, trình biên dịch), hàng đợi các tiến trình chờ được xử lý, ….
* Tìm kiếm theo chiều rộng BDF



**Bài tập thực hành 17/7/19:**

**Bài 1:** Chuyển đổi cơ số 10 sang cơ số x

**Bài 2:** Chuyển đổi biểu thức trung tố sang hậu tố

**Bài 3:** Tính giá trị của biểu thức hậu tố.

**Bài 4:** Duyệt đồ thị theo chiều rộng (xây dựng queue dùng DSLK đơn)

***Yêu cầu: nộp E. Hạn chót: 21/7/2019***