

# DISCRETE MATHEMATICS

Tài liệu thực hành toán rời rạc

*Phạm Phi Nhung*  
*phamphinhung2898@gmail.com*

Tháng 10 – 2020  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐẠI HỌC QUỐC GIA TP HCM  
KHOA TOÁN – TIN HỌC

# CHƯƠNG TRÌNH HỌC THỰC HÀNH

Ở phần thực hành môn Toán rời rạc, sinh viên sẽ lập trình 3-4 bài tập chính, ngoài ra sẽ có thêm các bài tập phụ để sinh viên làm quen với cách nộp bài tập, các quy định trong khi làm bài, hỗ trợ thêm các cú pháp/ kỹ thuật căn bản trong ngôn ngữ lập trình cũng như tính điểm thường trong thực hành. Các bài tập chính sẽ liên quan đến 3 chủ đề trong môn toán rời rạc gồm bảng chân trị, rút gọn biểu thức logic/ đa thức tối thiểu và đồ thị. (theo đề xuất có thể sẽ giảm tải nội dung rút gọn biểu thức logic và tăng cường thêm bài tập liên quan đến đồ thị).

Trong môn học này, ngôn ngữ chính được yêu cầu sử dụng là C/C++ để thực hành (nếu sinh viên không có nhu cầu nghiên cứu chuyên sâu về tin học thì có thể cài Code::Blocks (ưu điểm nhẹ máy hơn, cách thức tải và sử dụng có thể tham khảo trên google)).

Bên cạnh đó, nếu sinh viên có nhu cầu nghiên cứu chuyên sâu về tin thì khuyến khích có thể sử dụng Microsoft Visual Studio Community 2015 (bản miễn phí – đăng nhập từ tài khoản email sinh viên).

- Link đề xuất: <https://stackoverflow.com/questions/44290672/how-to-download-visual-studio-community-edition-2015-not-2017>
- Hướng dẫn cài đặt:
  - Chọn bản Community Edition, nhấp Web Installer (file được tải xuống)
  - Mở file vs\_community.exe, nhấp Run
  - Kiểm tra Location, chọn Default mode
  - Chọn Skip package (nhớ kiểm tra xem phiên bản đã tự chọn C++ chưa, nếu chưa bấm chọn)
  - Nhấp launch
  - (giao diện visual studio hỗ trợ viết C/C++ trong visual C+)

Đối với sinh viên có nhu cầu nghiên cứu về Tin cũng như muốn nâng cao kiến thức lập trình sử dụng các kỹ thuật trong môn Toán rời rạc, Cấu trúc dữ liệu và giải thuật, ... có thể lên trang Codeforce(<http://codeforces.com/>) hoặc HackerRank (<https://www.hackerrank.com/>) để thực hành. Trên các trang này cũng có những bài tập đòi hỏi sinh viên chắc các kiến thức Toán, lý thuyết đồ thị cũng như một số thuật toán khác để làm bài,

Nguồn thông tin chính của môn học này sẽ cập nhật chính trên hệ thống SAKAI:

- Link trang web: <http://learning.hvthao.com/portal/>
- Đăng nhập ở góc phía trên cùng bên phải với: User ID: MSSV – Password: MSSV (hoặc pass đã đổi tương ứng)
- Tại giao diện trang chủ: chọn môn học Discrete Mathematics
- Tại mục Overview (hoặc cột bên trái có các mục cần lưu ý sử dụng bao gồm:
  - Lesson: lưu trữ các bài học tóm tắt chính của môn học
  - Resources: Chứa các Books và Slide tham khảo chính cho môn học này (Slide và bài tập sẽ được update thường xuyên trong mục này)
  - Assignment List: Mục chứa các đề kiểm tra và nộp bài tập
  - Tests & Quizzes: Các bài tập trắc nghiệm liên quan đến môn học

# MỤC LỤC

DISCRETE MATHEMATICS.....	1
CHƯƠNG TRÌNH HỌC THỰC HÀNH .....	2
MỤC LỤC .....	3
YÊU CẦU THỰC HÀNH .....	4
CÁCH NỘP BÀI TẬP TRÊN SAKAI .....	5
CÁCH TÍNH ĐIỂM:.....	6
Thang điểm thực hành:.....	6
Trường hợp bị điểm 0:.....	6
Cách tính điểm thực hành .....	6
MỘT SỐ KỸ THUẬT CĂN BẢN.....	6

# YÊU CẦU THỰC HÀNH

Các bài tập chính của sinh viên tùy theo nội dung được yêu cầu sẽ nộp qua mục Assignment trên hệ thống SAKAI là chính. Bên cạnh đó, nhằm tạo thêm điểm cộng và khuyến khích ý thức tự giác làm bài của sinh viên sẽ có thêm các bài tập phụ (các bài tập này không bắt buộc và sẽ có thời gian hạn chót trong khoảng 1 tuần và các bài tập này sẽ nộp qua hình thức email cá nhân).

Lưu ý: đối với mỗi bài tập luôn đọc đúng và kỹ theo yêu cầu nộp bài, các bài tập nộp sai quy cách sẽ không được khiếu nại.

Ngoài ra, trong chương trình .cpp hoặc .c mà sinh viên chạy chương trình, hàm main() sinh viên viết theo kiểu **int main()** hoặc **void main()**, không sử dụng dạng khác nếu không được yêu cầu.

Với đầu mỗi bài thực hành (dù là bài chính hay phụ) đều phải khai báo các thông tin như sau:

```
/*
 * MSSV: 181101xxx
 * Ho va ten: Nguyen Van A
 * Assignment: bai1 / bt phụ 1
 * Created_at: 13/10/2020 (ghi ngay bat dau lam bai)
 * IDE: MS Visual Studio 2015 / Code
 */

#include <iostream>
using namespace std;

int main()
{
    cout<<"Hello World";
    return 0;
}
```

- Lưu ý khi nộp bài, chỉ nộp các file .cpp hoặc .c và .h (nếu có). Ngoài ra không nộp các file khác. (không nộp .sln, các folder debug,...)
- Đặt tên: Đối với bài chính (tùy theo yêu cầu trong mục Assignment của hệ thống và làm đúng yêu cầu); đối với bài phụ khi nộp qua mail cá nhân sẽ đặt tên file có lưu hàm main là **MSSVmain.c** hoặc **MSSVmain.cpp**
- Khi nộp bài tập phụ cần lưu ý:
  - \* Gửi bằng email cá nhân với tiêu đề **họ và tên\_MSSV\_TRR\_btphux** (ví dụ: NguyenVanA\_181101xxx\_TRR\_btphu1)
  - \* Nộp sai tiêu đề email hoặc sai file hoặc sai quy cách nộp được xem như không nộp bài.
- Ngoài ra, trong file code nộp khuyến khích có chú thích nội dung dòng code (ví dụ/ minh họa cụ thể ở phần Kỹ thuật căn bản)

# CÁCH NỘP BÀI TẬP TRÊN SAKAI

(Áp dụng đối với bài tập chính)

- Bước 1: đăng nhập hệ thống SAKAI và chọn đúng môn học.
- Bước 2: Trong mục ASSIGNMENT sẽ có các bài tập hoặc bài kiểm tra tương ứng. ví dụ:

Assignment Title	Status	Due
<a href="#">Assignment 0 - Testing</a>	Not Started	Jun 30, 2020 8:55 AM

Đây là Assignment 0 với due là hạn chót nộp bài tương ứng (hệ thống sẽ tự đóng sau khi hết thời gian, hãy cẩn thận và lưu ý về thời gian khi nộp bài => nên trừ hao ít nhất 5p để nộp bài trước deadline)

Khi chọn bài tập tương ứng:

**Instructions**  
Lab 04 - Ex 1.

**Additional resources for assignment**  
  
No attachments yet

Với instruction sẽ là nội dung đề bài hoặc các yêu cầu cần thực hiện. (ví dụ trên yêu cầu nộp bài tập 1 của lab 4)

- Bước 3: Nộp bài bằng submission (quan trọng, đây là bước bài sẽ được đăng trên hệ thống hay không). Click **chọn tệp** sau đó nhấn **submission**.

**Attachments**  
No attachments yet

Select a file from computer

Chọn tệp Không có tệp nào được chọn

or select files from 'Home' or site

Submit


Preview

Save Draft

Cancel

Nếu muốn cập nhật file mới để nộp: Chọn **remove** file sau đó **chọn tệp** mới rồi bấm **submission**

**Attachments**  

 [test.cpp](#) ( 1 KB; Jun 23, 2020 11:00 am ) [Remove](#)

Select more files from computer

Chọn tệp Không có tệp nào được chọn

or select more files from 'Home' or site

Submit

Preview

Save Draft

Cancel

# CÁCH TÍNH ĐIỂM:

## Thang điểm thực hành:

Với mỗi bài thực hành sẽ được tính trên thang điểm 10 và tổng tất cả các bài bonus tối đa 10 điểm

Điểm thực hành = SUM (tổng điểm bonus (nếu có) + điểm số 1 + điểm số 2 + .... + điểm số n)

Lưu ý: bonus từ các bài phụ và điểm số là từ các bài chính

Nghĩa là đối với điểm tổng môn toán rời rạc thì 70 Lý thuyết và 30 Thực hành, trong trường hợp có điểm Bonus sẽ là tối đa 40.

## Trường hợp bị điểm 0:

- Không nộp bài (sinh viên nộp qua hạn chót cũng được tính là không nộp bài).
- Có bài giống sinh viên khác (cho chép bài và chép bài hoặc chép bài trên mạng nhưng bị cùng nguồn,...)
- Không chạy đúng bất kỳ bộ test nào.

## Cách tính điểm thực hành

- Đối với các bài tập phụ (không bắt buộc):
  - Đây là các bài thực hành hỗ trợ giúp sinh viên làm quen/ ôn tập code, cũng như thực hành thêm một số kỹ thuật lập trình cho quen tay cũng như hỗ trợ phương pháp làm cho bài tập chính. Điểm của bài thực hành này được tính vào điểm thực hành như ở phần thang điểm thực hành.
  - Sinh viên đạt 0.25-2đ cho mỗi bài tập phụ, có ít nhất 5 bài tập phụ (tùy theo mức độ bài tập) – không tính trên thang 10 cho mỗi bài tập này mà sẽ cộng dồn, cộng điểm bài tập phụ tối đa 10 điểm.
- Đối với các bài tập chính:
  - Các bài tập sẽ được tính trên thang điểm 10

# MỘT SỐ KỸ THUẬT CĂN BẢN

*Phần này sẽ tập trung giới thiệu các kỹ thuật, hàm căn bản bằng cả 2 ngôn ngữ C và C++ (một số phần mình sẽ viết C++ nhiều hơn, bạn có thể sử dụng thêm google để hỗ trợ tìm hiểu thêm) nhằm hỗ trợ trong quá trình làm bài thực hành. Các hàm khác nâng cao hơn, sinh viên có thể tham khảo thêm trên các diễn đàn hoặc google để nâng cao kỹ năng.*

Trong môn học này không bắt buộc tất cả các sinh viên phải thành thạo cả 2 ngôn ngữ C và C++ nhưng vẫn khuyến khích sinh viên có thêm một ngôn ngữ mới làm hành trang trong quá trình chuẩn bị đi làm hoặc hiểu rõ hơn về bài tập.

Ngoài ra trong giai đoạn chuyên ngành cũng có thể có một số môn học yêu cầu các ngôn ngữ cụ thể, việc học thêm một ngôn ngữ mới tuy ban đầu hơi khó khăn nhưng vẫn sẽ có lợi ích hơn.

Do đó, trong phần tài liệu này sẽ tổng hợp các hàm và các kỹ thuật cơ bản từ hai ngôn ngữ C/C++ (có thể chọn một trong hai ngôn ngữ để thực hành)

## - Cấu trúc chương trình (chương trình Hello World)

Một chương trình cơ bản sẽ gồm các phần sau:

- Các lệnh tiền xử lý
- Các hàm
- Các biến
- Các lệnh và biểu thức
- Các comment

C		C++
<pre>#include &lt;stdio.h&gt;  int main(){      /* Đây là chương trình C */     printf("Hello, World! \n");     return 0; }</pre>	<pre>#include &lt;stdio.h&gt;  void main(){      // Chương trình C     printf("Hello, World! \n"); }</pre>	<pre>#include &lt;iostream&gt;  using namespace std;  /* Hàm main() */ int main(){      // In dòng chữ Hello, World!      cout &lt;&lt; "Hello, World!" &lt;&lt; endl;     return 0; }</pre>
<p>Giải thích:</p> <ol style="list-style-type: none"> <li>1. Dòng đầu tiên của chương trình <code>#include&lt;stdio.h&gt;</code> là lệnh khai báo thư viện, giúp nhắc nhở bộ biên dịch sử dụng thêm tệp <code>stdio.h</code> trước khi chạy biên dịch.</li> <li>2. Dòng tiếp theo <code>int main()</code> hoặc dòng <code>void main()</code> là nơi chương trình chính được bắt đầu.</li> <li>3. Dòng tiếp theo <code>/*...*/</code> hoặc <code>//</code> là dòng comment được bỏ qua bởi bộ biên dịch compiler và được dùng để thêm các chú thích cho chương trình. Đây được gọi là phần comment của chương trình.</li> <li>4. Dòng tiếp theo <code>printf(..)</code> là một hàm chức năng khác của ngôn ngữ C, in ra thông điệp nội dung "Hello, World!" hiển thị trên màn hình. Ký hiệu <code>\n</code> lúc này được xem như là một hiệu lệnh xuống dòng.</li> <li>5. Dòng <code>return 0</code> kết thúc hàm chính và trả về giá trị 0</li> </ol>		<p>Giải thích:</p> <ol style="list-style-type: none"> <li>1. Tương tự như C, dòng <code>#include &lt;iostream&gt;</code> là lệnh khai báo thư viện (hay có thể gọi như là header <code>&lt;iostream&gt;</code> cần thiết để sử dụng các hàm tương ứng).</li> <li>2. Dòng <code>using namespace std</code> nói cho compiler sử dụng <code>std namespace</code> (phần bổ sung của C++, sẽ hiểu rõ hơn khi học sâu về C++)</li> <li>3. Tương tự như C thì C++ cũng dùng <code>int main()</code> hoặc <code>void main()</code> cho chương trình chính và cách thức comment chú thích cũng tương tự.</li> <li>4. Dòng <code>cout &lt;&lt; ...</code>; là một hàm chức năng tương tự như <code>printf</code> trong C. và <code>&lt;&lt;endl</code> xem như là hiệu lệnh xuống dòng.</li> </ol>

## 2. Các kiểu dữ liệu trong C/C++

Có tất cả 7 kiểu dữ liệu cơ bản, ngoài ra còn có các kiểu dữ liệu khác được sửa đổi dựa trên một hoặc nhiều modifier như (*signed* (có dấu) ; *unsigned* (không dấu); *short*; *long*; ...)

Kiểu dữ liệu	Từ khóa
Boolean (kiểu true/false) (lưu ý trong C không có kiểu này, sẽ chuyển về giá trị int với 1 và 0)	bool
Ký tự	char
Số nguyên	int
Số thực	float
Số thực dạng double	double
Kiểu không có giá trị	void
Kiểu wide character (ít dùng trong C++ và trong C không thấy tài liệu ghi có)	wchar_t

Ngoài ra còn có các kỹ thuật để tạo một kiểu dữ liệu mới dựa trên dữ liệu đang tồn tại bằng *typedef* và *enum* – đây là kiểu dữ liệu liệt kê khai báo nhiều kiểu tùy ý và tập hợp nhiều identifier (định danh) và có thể sử dụng như là các giá trị của kiểu đó – có thể hiểu như là một kiểu liệt kê. Ví dụ về hai kiểu này như sau:

#### - Về typedef

```
typedef kieu_du_lieu ten_moi;

//ví dụ:

typedef float sothuc;

sothuc vantoc;
```

#### - Về enum

```
enum ten_cua_enum { danh_sach_cac_ten } danh_sach_bien;

// ví dụ:

enum hanghoa { sua, nuocngot, biachai } c;

c = nuocngot;

// hoặc:

enum hanghoa { sua, nuocngot=40, biachai };

// lúc này giá trị biachai sẽ có giá trị là 41 vì mỗi tên sẽ có giá trị lớn hơn của tên trước đó là 1
```

### 3. Mảng trong C/C++:

Mảng là một tập hợp tuần tự các phần tử có cùng kiểu dữ liệu và các phần tử được lưu trữ trong một dãy các ô nhớ liên tục trên bộ nhớ. Các phần tử của mảng được truy cập bằng cách sử dụng “chỉ số”. Mảng có kích thước N sẽ có chỉ số từ 0 tới N-1.

Ví dụ: với N = 5, khi đó chỉ số mảng (*index*) sẽ có giá trị từ 0 tới 4, tương ứng với 5 phần tử và các phần tử trong mảng được truy cập bằng cách sử dụng *<em>tênmảng[index]</em>*.

Cú pháp khai báo mảng trong C/C++ cần có 2 tham số:

- Kích thước của mảng (xác định số lượng phần tử có thể được lưu trữ trong mảng).
- Kiểu dữ liệu của mảng (chỉ định kiểu dữ liệu của phần tử trong mảng: có thể là số nguyên, số thực, ký tự hay là kiểu dữ liệu khác)



Ví dụ: trong C (có 2 cách)

```
#include <stdio.h>
const int MAX = 100;

void main(){
    int a[100]; // Khai báo mảng với kích thước lớn
    int na;     // Kích thước thật của mảng a

    printf("Nhap kích thước mảng: \n");
    scanf("%d",&na); //Cho phép nhập kích thước

    //Nhập mảng:
    for(int i=0; i<na; i++){
        printf("Phan tu [%d] \n",i);
        scanf("%d",&a[i]);
    }

    // Xuất mảng:
    printf("xuat mang a \n");
    for (int k=0; k<na; k++){
        printf("Phan tu [%d]: %d\n",k,a[k]);
    }
}

void NhapMang(int a[], int n){
    for(int i = 0; i < n; ++i){
        printf("\nNhap phan tu a[%d] = ", i);
        scanf("%d", &a[i]);
    }
}

void XuatMang(int a[], int n){
    for(int i = 0; i < n; ++i){
        printf("\nPhan tu a[%d] = %d", i, a[i]);
    }
}

int TimKiem(int a[], int n, int v){
    for(int i = 0; i < n; ++i){
        if(a[i] == v){
            return i;
        }
    }
    return -1;
}
```

Tham khảo C++: <http://www.cplusplus.com/doc/tutorial/arrays/>

Trong trường hợp sử dụng mảng 2 chiều, có thể hình dung mảng 2 chiều là mảng của các mảng 1 chiều, tương tự như bảng. Với các tham số sau:

- Khai báo số hàng của mảng 2 chiều (*row\_index*).
- Khai báo số cột của mảng 2 chiều (*column\_index*).
- Kiểu dữ liệu của mảng 2 chiều.

Hình dung qua mã giả như sau:

```
type arr[row_size][column_size] = { {elements}, {elements} ... }
for i from 0 to row_size
    for j from 0 to column_size
        print arr[i][j]
```

Ví dụ (trong C):

```
#include <stdio.h>

int main()
{
    // Khai báo mảng
    int arr[3][5] = {{5, 12, 17, 9, 3}, {13, 4, 8, 14, 1}, {9, 6, 3, 7, 21}};
    // Vòng lặp từng mảng (row_size)
    for(int i=0; i<3; i++) {
        for(int j=0; j<5; j++) {
            // Xuất từng phần tử
            printf("%5d", arr[i][j]);
        }
        // Xuất dòng trống dưới mỗi phần tử
        printf("\n");
    }
    return 0;
}
```

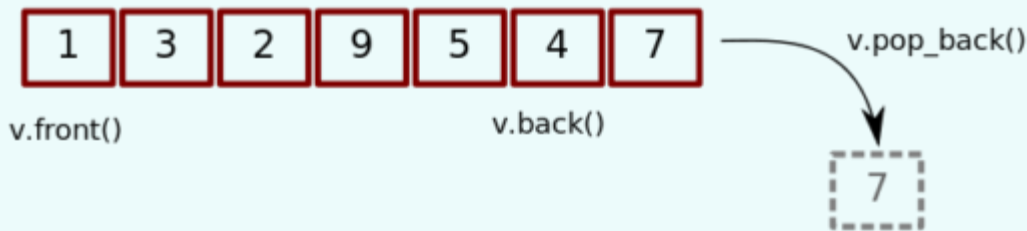
Ví dụ (trong C – chuyển sang hàm):

```
void NhapMaTran(int a[][100], int m, int n){
    for(int i = 0; i < m; i++)
        for(int j = 0; j < n; j++) {
            printf("A[%d][%d] = ", i, j);
            scanf("%d", &a[i][j]);
        }
}
```

```
void XuatMaTran(int a[][100], int m, int n){
    for(int i = 0; i < m; i++) {
        for(int j = 0; j < n; j++)
            printf("%d\t", a[i][j]);
        printf("\n");
    }
}
```

#### 4. Vector:

Tương tự với định nghĩa toán, trong ngôn ngữ C/C++ thì vector hình dung đơn giản là một chuỗi biểu diễn mảng nhưng có khả năng thay đổi kích thước (*dynamic array*). Đối với các bài toán chứa nhiều phần tử, việc chỉ sử dụng mảng thông thường sẽ gặp một số khó khăn trong việc lưu trữ, vì vậy vector lúc này được sử dụng như một loại mảng đặc biệt hơn. Ngoài ra, việc sử dụng vector cũng dễ dàng hơn trong việc quản lý và hình dung các giá trị cũng như các tác vụ thực thi, với việc lưu trữ của chúng được chứa tự động xử lý. Các phần tử vector được đặt trong các bộ nhớ liên kề (*contiguous storage*).

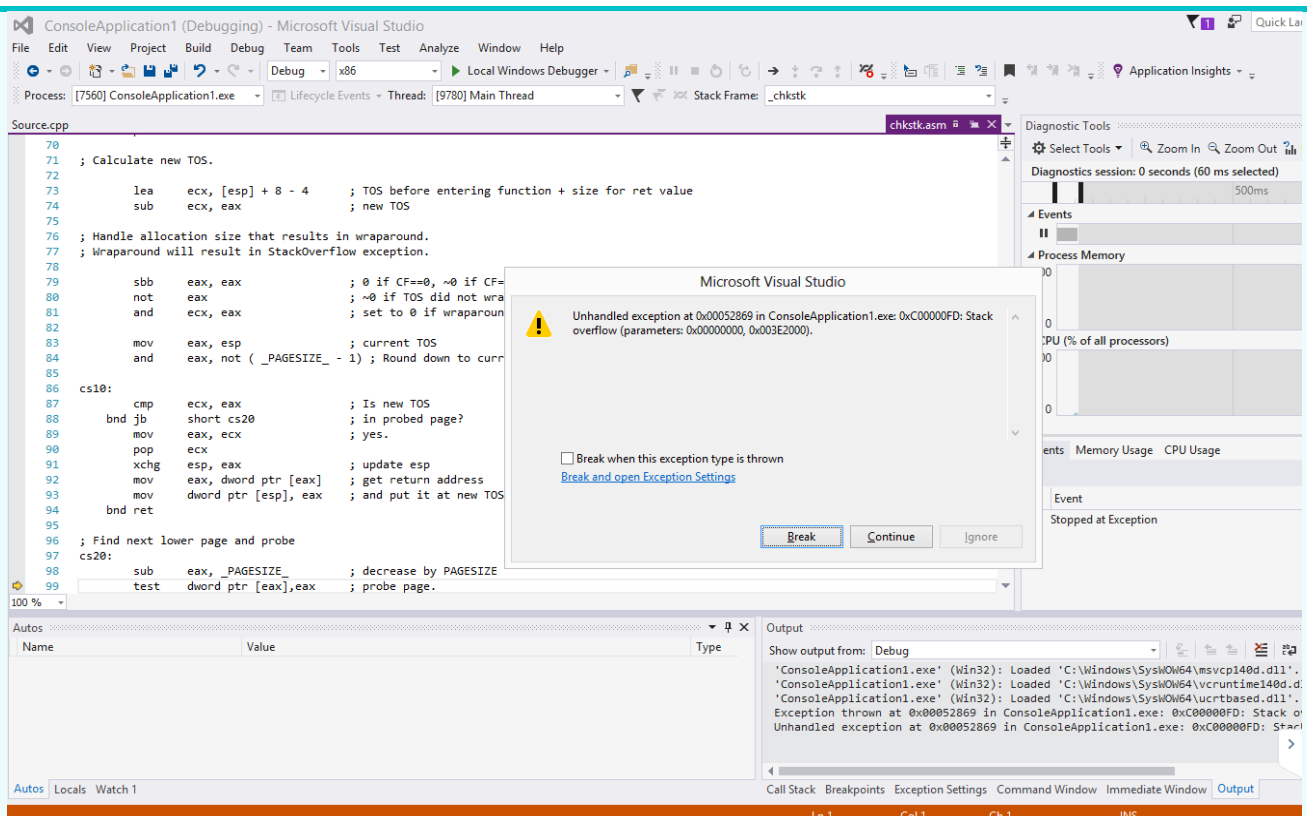


Trong C/C++, vector được biết đến thư viện `#include<vector>`, nếu muốn tạo mảng 1 chiều (tương tự như *array*) thì sẽ khai báo theo dạng `vector<kiểu_dữ_liệu> tên_biến`; trong đó *kiểu\_dữ\_liệu* có thể là *int*, *char*, *string*, ....; nếu muốn tạo mảng 2 chiều thì sẽ khai báo `vector<vector<kiểu_dữ_liệu>> tên_biến`; (với mảng vector 2 chiều trở lên, kích thước mỗi dòng có thể khác nhau tùy vào mục đích người sử dụng).

Ví dụ : tạo một mảng kiểu INT có kích thước 1.000.000

C++
<pre>#include &lt;iostream&gt; using namespace std; void main(){     int mang[1000000] = {0};     cout &lt;&lt; "hello"&lt;&lt; endl; }</pre>

Giải thích : khi chạy đoạn code trên, trong trường hợp này, việc sử dụng mảng có thể gây hiện tượng tràn bộ nhớ đệm (hay còn gọi là *Stack overflow*).



Ví dụ, ta có thể tạo 1 mảng tương tự có kích thước 1.000.000 với các giá trị từ 1 đến 1.000.000 với các giá trị từ 1 đến 1.000.000, sau đó xuất ra kết quả.

C++

```
#include <iostream>
#include <vector>

using namespace std;

int main(){

    vector<int> mang; // khai báo vector

    for (int i = 0; i < 1000000; i++) //nhập các giá trị của vector

        mang.push_back(i+1);

    int n = mang.size(); //lấy kích thước của vector

    for( int i = n-1; i>=0 ; i--) { //vòng lặp thực hiện n-1 lần tương ứng với số lượng giá trị
có trong vector để xuất ra màn hình

        int a = mang[i];

        cout << a << " ";

    }

    return 0;

}
```

## MỘT SỐ LỆNH VỀ VECTOR

**vector<kiểu\_dữ\_liệu> A;**

*A.size()*: Trả về kích thước vector

*A.empty()*: Kiểm tra vector có rỗng.

*A.front()*: Truy cập phần tử đầu tiên trong vector.

*A.back()*: Truy cập phần tử cuối cùng trong vector.

*A.push\_back()*: Đưa phần tử vào vị trí cuối cùng trong vector.

*A.pop\_back()*: Xóa phần tử cuối cùng trong vector.

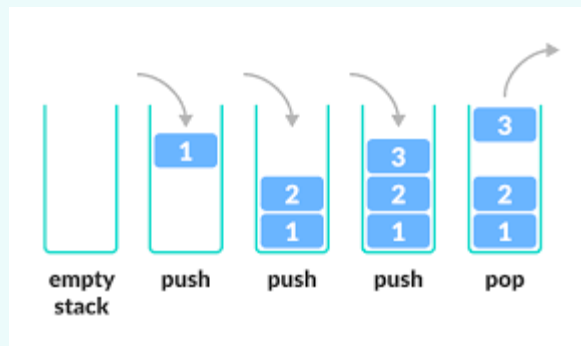
Tham khảo thêm:

<http://www.cplusplus.com/reference/vector/vector/>

<https://topdev.vn/blog/vector-trong-c/>

### 5. Stack (ngăn xếp):

Stack (ngăn xếp), là một cấu trúc thiết kế cho những quy trình mang yếu tố LIFO (Last In First Out → Vào trước ra sau). Các phần tử mới vào stack sẽ nằm ở vị trí cuối cùng, và cũng là phần tử được lấy ra đầu tiên.



Ví dụ bạn đưa lần lượt các giá trị 1 2 3 vào ngăn xếp, khi đó để lấy phần tử 2, bạn phải lấy 3, rồi mới tới 2.

Trước khi sử dụng stack, khai báo `#include<stack>`. Để sử dụng, khai báo `stack<kiểu_dữ_liệu> tên_biến`, trong đó kiểu\_dữ\_liệu có thể là int, char, string, ...

C++	C
<pre>#include &lt;iostream&gt; #include &lt;stack&gt; #include&lt;string&gt;  using namespace std;  int main() {     // Nhập Stack     stack&lt;string&gt; A;     A.push("xep");     A.push("ngan");     A.push("la");     A.push("Stack");      // Xuất Stack:     while ( !A.empty()) {</pre>	<p>Tham khảo cách cài đặt trong C: <a href="https://nguyenvanhieu.vn/ngan-xep-stack/">https://nguyenvanhieu.vn/ngan-xep-stack/</a></p>

```

string s = A.top();
cout << s << " ";
A.pop();
}
cout << endl;
return 0;
}

```

## MỘT SỐ LỆNH VỀ STACK

**stack<kiểu\_dữ\_liệu> A;**

*A.empty()*: Kiểm tra stack có rỗng.

*A.size()*: Trả về kích thước của stack.

*A.top()*: Trả về phần tử “cuối cùng” của stack.

*A.push(ele)*: Đưa phần tử *ele* vào stack.

*A.pop()*: Xóa phần tử *A.top()* ra khỏi stack.

Tham khảo thêm:

(Các hàm trong C++): <http://www.cplusplus.com/reference/stack/stack/>

## 6. Queue (hàng đợi):

Queue (hàng đợi), là một cấu trúc thiết kế cho những quy trình mang yếu tố FIFO (First In First Out → Vào trước ra trước). Các phần tử mới vào queue sẽ nằm ở vị trí đầu tiên, và cũng là phần tử được lấy ra đầu tiên.



Ví dụ bạn đưa lần lượt các giá trị 1 2 3 4 5 vào hàng đợi, khi đó để lấy phần tử 3, bạn phải lấy 1, rồi lấy 2, sau đó mới tới 3.

Trước khi sử dụng queue, khai báo `#include <queue>`. Để sử dụng, khai báo `queue<kiểu_dữ_liệu> tên_biến`, trong đó kiểu\_dữ\_liệu có thể là int, char, string, ...

C++	C
<pre> #include &lt;iostream&gt; #include &lt;queue&gt; using namespace std;  int main() {     // Nhập Queue     queue&lt;string&gt; A;     A.push('T');     A.push('o');     A.push('a');     A.push('n');     A.push('T');     A.push('i'); </pre>	<p>Tham khảo cách cài đặt trong C:  <a href="https://nguyenvanhieu.vn/hang-doi-queue/">https://nguyenvanhieu.vn/hang-doi-queue/</a></p>

```

A.push('n');

// Xuất Stack:
while ( !A.empty() ) {
    char c = A.front();
    cout << c << " ";
    A.pop();
}
cout << endl;
return 0;
}

```

## MỘT SỐ LỆNH VỀ QUEUE

**queue<kiểu\_dữ\_liệu> A;**

- A.empty()*: Kiểm tra queue có rỗng.
- A.size()*: Trả về kích thước của queue.
- A.front()*: Trả về phần tử đầu tiên trong queue.
- A.back()*: Trả về phần tử cuối cùng trong queue.
- A.push(ele)*: Đưa phần tử *ele* vào đầu queue.
- A.pop()*: Xóa phần tử *A.front()* ra khỏi queue.

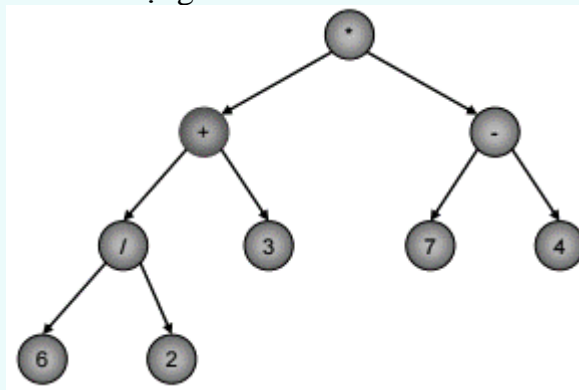
Tham khảo thêm:

<http://www.cplusplus.com/reference/queue/queue/>

## 7. Tiền tố, hậu tố và trung tố

Để hình dung rõ hơn về thuật toán này, đầu tiên cần nắm được 3 khái niệm cơ bản về tiền tố, hậu tố và trung tố. Hình dung qua ví dụ như sau: với input đầu vào là phép tính toán  $(6 / 2 + 3) * (7 - 4)$  sẽ được phân tích:

- (giải thích theo cấu trúc dữ liệu và giải thuật): hình dung phép tính trên khi phân tích thành dạng cây nhị phân sẽ có dạng



Sau đó từ dạng cây này sẽ có các cách duyệt từ left-node-right (LNR), từ left-right-note (LRN), từ note-left-right (NLR), thì ứng với mỗi cách duyệt có thể dễ dàng hình dung hơn cho các định nghĩa tiền tố, hậu tố và trung tố (nêu trong phần cách giải thích khác),...

- (cách giải thích khác):
  - Khi phân tích phép toán theo dạng  $* + / 6 2 3 - 7 4$  thì lúc này được gọi là dạng tiền tố (*prefix*) của biểu thức tính toán. Tên gọi khác của phương pháp

này chính là **Ký pháp Balan**. (trong CTDL thì đang được xem là duyệt NLR).

- Khi phân tích phép toán theo dạng  $6 / 2 + 3 * 7 - 4$  thì khuyết điểm lớn nhất là do sự mập mờ vì thiếu dấu ngoặc, người ta thường bổ sung thêm thủ tục duyệt inorder cho việc bổ sung các dấu ngoặc vào mỗi biểu thức con, kết quả lúc này sẽ thu được là  $((((6 / 2) + 3) * (7 - 4)))$ , đây là ký pháp dưới dạng trung tố (*infix*). (trong CTDL thì được xem như duyệt theo thứ tự LNR).
- Khi phân tích phép toán theo dạng  $6 2 / 3 + 7 4 - *$ , đây là dạng hậu tố (*postfix*). Trong ký pháp này khác với dạng tiền tố ở chỗ các toán tử phép tính sẽ được viết sau 2 toán hạng (phần tử phép tính). Tên gọi khác của phương pháp này chính là **Ký pháp nghịch đảo Balan** hay **Thuật toán Balan ngược**. (trong CTDL thì đây chính là duyệt (LRN))

Một số ví dụ khác: (trường hợp với 3 biến x, y, z)

Infix	Prefix	Postfix
$x+y$	$+xy$	$xy+$
$x+y-z$	$-+xyz$	$xy+z-$
$x+y*z$	$+x*yz$	$xyz*+$
$x+(y-z)$	$+x-yz$	$xyz-+$

Có ít nhất 2 phương pháp hiện tại cho việc chuyển tiền tố sang hậu tố đó là dùng Stack và Cây biểu thức (*Expression tree*), nhưng hầu hết sẽ sử dụng Stack phổ biến hơn, vì với ưu điểm dễ cài đặt và đơn giản hơn.

## 8. Ký pháp nghịch đảo Balan (Thuật toán Balan ngược) và tính kết quả

(đối với các trường hợp trong phần này là các phép toán tử  $+$   $-$   $*$   $/$  và dấu ngoặc, ngoài ra có thể có thể các toán tử khác như  $\%$ , ...)

Trong môn học này giới thiệu về ký pháp nghịch đảo Balan, nhằm hỗ trợ cho một số bài thực hành có liên quan, trong trường hợp nếu người học muốn làm các phương pháp khác, thì có thể làm thêm cách khác và ghi chú rõ vào trong báo cáo phần đã làm thêm.

Nguyên nhân hình thành các phương pháp này? Đối với con người thì việc dựa trên các phép toán để phân tích các thành phần là dễ dàng hơn so với máy tính, cụ thể: hình dung rằng khi cho biểu thức toán học:

$$1 * (2 + 3)$$

Thì đối với máy tính đây chỉ như là một chuỗi ký tự được nhập vào và việc làm sao để Máy tính hiểu rõ phải thực hiện phép tính toán nào trước và phép tính toán nào sau. Đối với trường hợp này, việc tính toán có thể phân biệt được dựa trên các phép tính dấu ngoặc, báo hiệu cho thấy một phép tính ưu tiên hơn cần được tính trước, nhưng một ví dụ khác cho thấy sự phức tạp trong việc ưu tiên các phép tính:

$$1 + 2 * 3$$



Đối với trường hợp này không có trường hợp dấu () làm dấu hiệu ưu tiên, thì vấn đề khó khăn tiếp theo đó là máy tính cần phải hiểu được là tính phép tính “nhân chia trước, cộng trừ sau”.

Có 2 bước thực hiện chính trong việc đề xuất phương pháp để tính toán các phép toán trên trong máy tính :

- Bước 1: Chuyển biểu thức về dạng hậu tố
- Bước 2: Tính kết quả

### *a. Chuyển biểu thức về dạng hậu tố (Balan ngược)*

Biểu thức tính toán người dùng nhập vào thường được viết dưới dạng trung tố (phép toán nằm giữa 2 giá trị) và mục tiêu cần thực hiện là chuyển phép toán về dạng hậu tố (phép toán nằm sau 2 giá trị tính toán), ví dụ:

Input (chuỗi biểu thức – string s)	Output (chuỗi biểu thức q dưới dạng hậu tố từ s)
$1 + 2 * 3$	$2\ 3\ * \ 1\ +$
$1 * (2 + 3)$	$2\ 3\ + \ 1\ *$

Các bước **thuật toán Balan ngược** thực hiện như sau:

Yêu cầu:

Input: chuỗi biểu thức – string s

Output: chuỗi biểu thức q dưới dạng hậu tố từ s

- Bước 1: Khởi tạo Stack rỗng để chứa phép toán,  $q = ""$  (chuỗi rỗng)
- Bước 2: Lặp cho đến khi kết thúc biểu thức s. Lần lượt đọc từng phần tử của biểu thức (có thể là hằng, biến, toán tử, dấu “)”) hay dấu “(“.

Nếu phần tử là:

- “(“ thì đưa vào Stack.
- “)” thì lấy các phần tử của Stack đưa vào chuỗi q cho đến khi gặp “(“ trong Stack
- Phép toán (3 TH)
- Nếu Stack rỗng thì đưa phép toán vào Stack.
- Ngược lại, nếu phép toán có độ ưu tiên cao hơn phần tử ở đầu Stack thì đưa vào Stack
- Ngược lại, lấy phần tử của Stack đưa vào chuỗi q, lặp lại so sánh với phần tử đầu Stack.
- Hằng số (hoặc biến số): lấy ra, đưa vào chuỗi q.
- Bước 3: Lấy hết các phần tử còn lại trong Stack ra và đưa vào chuỗi q.

Lưu ý rằng, với phép toán “(“ thì được xem là có độ ưu tiên thấp hơn độ ưu tiên của tất cả các phép toán.

Ví dụ minh họa:

Với input:  $7 * 8 - (2 + 3)$  và output là chuỗi biểu thức q viết dưới dạng hậu tố.

- Bước 1:
  - Khởi tạo string  $q = ""$  (chuỗi rỗng).
  - Tạo Stack rỗng




- Bước 2: Lặp đến hết biểu thức  $7 * 8 - (2 + 3)$

<div>Start</div> <div>q = ""</div> <div><div></div><div></div><div></div><div></div><div></div></div>	<div>2a. Xét phần tử 1:</div> <div>7 =&gt; là hằng số nên đưa vào q</div> <div>q = ""</div> <div><div></div><div></div><div></div><div></div><div></div></div>
<div>2b. Xét phần tử 2</div> <div>* =&gt; là phép toán, stack rỗng =&gt; đưa vào stack</div> <div>q = "7"</div> <div><div></div><div></div><div></div><div></div><div></div></div>	<div>2c. Xét phần tử 3</div> <div>8 =&gt; là hằng số, đưa vào q</div> <div>q = "7"</div> <div><div>*</div><div></div><div></div><div></div><div></div></div>
<div>2d. Xét phần tử 4</div> <div>- =&gt; là phép toán, stack khác rỗng, ưu tiên thấp hơn phần tử đầu stack (do * có độ ưu tiên cao hơn -) =&gt; lấy * ra khỏi stack đưa vào q</div> <div>q = "7 8"</div> <div><div>*</div><div></div><div></div><div></div><div></div></div>	<div>2e. Xét phần tử 4</div> <div>-</div> <div>q = "7 8 *"</div> <div><div></div><div></div><div></div><div></div><div></div></div>
<div>2f. Xét phần tử 4</div> <div>9. =&gt; là phép toán, stack rỗng =&gt; đưa vào stack</div> <div>q = "7 8 *"</div> <div><div></div><div></div><div></div><div></div><div></div></div>	<div>2g. Xét phần tử 5</div> <div>( =&gt; đưa vào stack</div> <div>q = "7 8 *"</div> <div><div>-</div><div></div><div></div><div></div><div></div></div>
<div>2h. Xét phần tử 6</div>	<div>2i. Xét phần tử 7</div>

<p>2 =&gt; là hằng số, đưa vào q</p> <p>q = “7 8 *”</p> <table><tr><td>(</td></tr><tr><td>-</td></tr><tr><td> </td></tr><tr><td> </td></tr><tr><td> </td></tr></table>	(	-				<p>+ =&gt; là phép toán, stack khác rỗng, độ ưu tiên cao hơn phép toán đầu stack (do + cao hơn () =&gt; đưa vào stack</p> <p>q = “7 8 * 2”</p> <table><tr><td>(</td></tr><tr><td>-</td></tr><tr><td> </td></tr><tr><td> </td></tr><tr><td> </td></tr></table>	(	-			
(											
-											
(											
-											
<p>2k. Xét phần tử 8</p> <p>3 =&gt; là hằng số, đưa vào q</p> <p>q = “7 8 * 2”</p> <table><tr><td>+</td></tr><tr><td>(</td></tr><tr><td>-</td></tr><tr><td> </td></tr><tr><td> </td></tr></table>	+	(	-			<p>2l. Xét phần tử 9</p> <p>) =&gt; lấy các phần tử của stack đưa vào q cho đến khi gặp “(“</p> <p>q = “7 8 * 2 3”</p> <table><tr><td>+</td></tr><tr><td>(</td></tr><tr><td>-</td></tr><tr><td> </td></tr><tr><td> </td></tr></table>	+	(	-		
+											
(											
-											
+											
(											
-											
<p>2m.</p> <p>q = “7 8 * 2 3 +”</p> <table><tr><td>-</td></tr><tr><td> </td></tr><tr><td> </td></tr><tr><td> </td></tr><tr><td> </td></tr></table>	-					<p>Kết thúc bước 2</p>					
-											

- Bước 3: Chuỗi phép toán kết thúc => lấy hết các phần tử còn lại của Stack đưa vào q
  - string q = "7 8 \* 2 3 + -".
  - Stack cuối cùng là rỗng


Vậy biểu thức tính toán  $7 * 8 - (2 + 3)$  có dạng hậu tố là  $7 8 * 2 3 + -$

Một ví dụ khác (cách viết khác), chuyển biểu thức  $A*B+C*((D-E)+F)/G$  sang hậu tố

Token	Stack	Output
A	{Empty}	A
*	*	A
B	*	A B
+	+	A B *
C	+	A B * C
*	+ *	A B * C
(	+ * (	A B * C
(	+ * ( (	A B * C
D	+ * ( (	A B * C D
-	+ * ( ( -	A B * C D
E	+ * ( ( -	A B * C D E
)	+ * (	A B * C D E -
+	+ * ( +	A B * C D E -
F	+ * ( +	A B * C D E - F
)	+ *	A B * C D E - F +
/	+ /	A B * C D E - F + *
G	+ /	A B * C D E - F + * G
	{Empty}	A B * C D E - F + * G / +

Vậy kết quả cuối cùng của biểu thức  $A*B+C*((D-E)+F)/G$  sang hậu tố là  
 $A B * C D E - F + * G / +$

### b. Tính kết quả

Các bước **tính kết quả từ hậu tố** thực hiện như sau:

Yêu cầu:

Input: chuỗi biểu thức q dưới dạng hậu tố

Output: kết quả chuỗi biểu thức

- Bước 1: Khởi tạo Stack rỗng để chứa hằng hoặc biến
- Bước 2: Lặp cho đến khi kết thúc biểu thức hậu tố. Lần lượt đọc từng phần tử của biểu thức (hằng, biến, phép toán).

Nếu phần tử là:

- Hằng hay biến: đưa vào Stack
- Ngược lại:

Lấy 2 phần tử của Stack.

Áp dụng phép toán cho 2 phần tử vừa lấy, phần tử đầu tiên đặt bên phải, phần tử thứ 2 đặt bên trái, phép toán đặt giữa 2 phần tử, tính kết quả.

- Đưa kết quả vào Stack
- Bước 3: Xuất phần tử cuối cùng của Stack chính là giá trị biểu thức.

Ví dụ minh họa: (từ biểu thức hậu tố được tính ở ví dụ minh họa trên)

Với input: 7 8 \* 2 3 + -

Output: Kết quả của biểu thức hậu tố input

- Bước 1: Tạo Stack rỗng


- Bước 2: Lặp đến hết biểu thức hậu tố 7 8 \* 2 3 + -

<p>Start</p> <table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr></table>				<p>2a. Xét phần tử 1: 7 =&gt; là hằng số nên đưa vào stack</p> <table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr></table>			
<p>2b. Xét phần tử 2 8 =&gt; là hằng số nên đưa vào stack</p> <table><tr><td>7</td></tr><tr><td></td></tr><tr><td></td></tr></table>	7			<p>2c. Xét phần tử 3 * =&gt; là toán tử =&gt; lấy 2 phần tử đầu tiên của stack =&gt; phần tử đầu bên phải, phần tử sau bên trái và phép toán ở giữa</p> <table><tr><td>8</td></tr><tr><td>7</td></tr><tr><td></td></tr></table>	8	7	
7							
8							
7							
<p>2d. Xét phần tử 3 * =&gt; là toán tử =&gt; lấy 2 phần tử đầu tiên của stack =&gt; phần tử đầu bên phải, phần tử sau bên trái và phép toán ở giữa (step 1: lấy phần tử đầu tiên của stack)</p> <p style="text-align: center;">8</p> <table><tr><td>7</td></tr><tr><td></td></tr><tr><td></td></tr></table>	7			<p>2e. Xét phần tử 3 * =&gt; là toán tử =&gt; lấy 2 phần tử đầu tiên của stack =&gt; phần tử đầu bên phải, phần tử sau bên trái và phép toán ở giữa (step 2: lấy phần tử tiếp theo của stack)</p> <p style="text-align: center;">7 8</p> <table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr></table>			
7							
<p>2f. (step 3: sắp xếp Phần tử đầu bên phải, phần tử sau bên trái và phép toán ở giữa)</p> <p style="text-align: center;"><math>7 * 8 = 56</math></p> <p>=&gt;Đưa kết quả vào stack</p>	<p>2g. Xét phần tử 4 2 =&gt; là hằng số nên đưa vào stack</p>						

<table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr></table>				<table><tr><td>56</td></tr><tr><td></td></tr><tr><td></td></tr></table>	56		
56							
<p>2h. Xét phần tử 5 3 =&gt; là hằng số nên đưa vào stack</p> <table><tr><td>2</td></tr><tr><td>56</td></tr><tr><td></td></tr></table>	2	56		<p>2i. Xét phần tử 6 + =&gt; là toán tử , lấy 2 phần tử đầu stack, phần tử đầu bên phải, phần tử thứ 2 bên trái, phép toán ở giữa</p> <table><tr><td>3</td></tr><tr><td>2</td></tr><tr><td>56</td></tr></table>	3	2	56
2							
56							
3							
2							
56							
<p>2k. Xét phần tử 6 + =&gt; là toán tử , lấy 2 phần tử đầu stack, phần tử đầu bên phải, phần tử thứ 2 bên trái, phép toán ở giữa (step 1: lấy phần tử đầu của stack)</p> <p style="text-align: center;">3</p> <table><tr><td>2</td></tr><tr><td>56</td></tr><tr><td></td></tr></table>	2	56		<p>2l. Xét phần tử 6 + =&gt; là toán tử , lấy 2 phần tử đầu stack, phần tử đầu bên phải, phần tử thứ 2 bên trái, phép toán ở giữa (step 2: lấy phần tử tiếp theo của stack )</p> <p style="text-align: center;">3</p> <table><tr><td>2</td></tr><tr><td>56</td></tr><tr><td></td></tr></table>	2	56	
2							
56							
2							
56							
<p>2m. (step 3: sắp xếp Phần tử đầu bên phải, phần tử sau bên trái và phép toán ở giữa)</p> <p style="text-align: center;"><math>2 + 3 = 5</math></p> <p>=&gt; Đưa kết quả vào stack</p> <table><tr><td>56</td></tr><tr><td></td></tr><tr><td></td></tr></table>	56			<p>2n. Xét phần tử thứ 7 (phần tử cuối của biểu thức hậu tố)</p> <p>- =&gt; là phép toán, lấy 2 phần tử đầu stack, phần tử đầu bên phải, phần tử thứ hai bên trái và phép toán ở giữa</p> <table><tr><td>5</td></tr><tr><td>56</td></tr><tr><td></td></tr></table>	5	56	
56							
5							
56							
<p>2o. Xét phần tử thứ 7 (phần tử cuối của biểu thức hậu tố)</p> <p>- =&gt; là phép toán, lấy 2 phần tử đầu stack, phần tử đầu bên phải, phần tử thứ hai bên trái và phép toán ở giữa (step 1: lấy phần tử đầu stack)</p> <p style="text-align: center;">5</p> <table><tr><td>56</td></tr><tr><td></td></tr><tr><td></td></tr></table>	56			<p>2p. Xét phần tử thứ 7 (phần tử cuối của biểu thức hậu tố)</p> <p>- =&gt; là phép toán, lấy 2 phần tử đầu stack, phần tử đầu bên phải, phần tử thứ hai bên trái và phép toán ở giữa (step 2: lấy phần tử tiếp theo của stack)</p> <p style="text-align: center;">56 5</p> <table><tr><td>56</td></tr><tr><td></td></tr><tr><td></td></tr></table>	56		
56							
56							
<p>2q. (step 3: sắp xếp Phần tử đầu bên phải, phần tử sau bên trái và phép toán ở giữa)</p> <p style="text-align: center;"><math>56 - 5 = 51</math></p>	<p>2r. Hết chuỗi biểu thức hậu tố =&gt;Kết thúc bước 2 Stack cuối bước 2</p>						

⇒ Đưa kết quả vào stack	51

- Bước 3: Chuỗi phép toán kết thúc => lấy hết các phần tử còn lại của Stack, đây chính là kết quả của chuỗi biểu thức hậu tố

- Kết quả: 51
- Stack cuối cùng là rỗng


Vậy biểu thức tính toán hậu tố  $7\ 8\ *\ 2\ 3\ +\ -$  có kết quả là 51

### c. Gợi ý về kỹ thuật lập trình (có thể có nhiều cách viết khác phù hợp cách viết hoặc chương trình khác)

Để xét độ ưu tiên của các phép toán thì có thể sử dụng hàm con như sau, ví dụ:

```
int getPriority (string op){
    if (op == "*" || op == "/" || op == "%")
        return 2;
    if (op == "+" || op == "-")
        return 1;
    return 0;
}
```

Giải thích: Do các phép toán multiply (+), subtract (-), multiply (\*), divide (/), Modulo (%) sẽ có cùng độ ưu tiên cao hơn các toán tử + - nên ứng với các giá trị trả về khi ở dạng số sẽ dễ so sánh mức độ cao thấp của phép toán.

Để chuyển từ trung tố sang hậu tố:

```
void infix2Postfix(char infix[], char postfix[]){
    Stack S;
    char x, token;
    int i = 0, j = 0;
    // i-index of infix, j-index of postfix

    init(&S);
    for (i = 0; infix[i] != '\0'; i++){
        token = infix[i];
        if (isalnum(token))
            postfix[j++] = token;
        else
            if (token == '(')
                Push(&S, '(');
            else
                if (token == ')')
                    while ((x = Pop(&S)) != '(')
                        postfix[j++] = x;
                else {
                    while (precedence(token) <= precedence(top(&S))
                        && !isEmpty(&S)){
```

Giải thích: Dựa theo thuật toán, nên đọc tương ứng

```

        x = Pop(&S);
        postfix[j++] = x;

    }

    Push(&S, token);

}

while (!isEmpty(&S)) {
    x = Pop(&S);
    postfix[j++] = x;
}

postfix[j] = '\0';
}

```

### Cách tính toán hậu tố:

```

float Evaluate(char *Postfix){
    struct Stack S;
    char *p;
    float op1, op2, result;
    S.TOP = -1;
    p = &Postfix[0];
    while (*p != '\0') {
        while (*p == ' ' || *p == '\t')
            p++;

        if (isdigit(*p)) {
            int num = 0;
            while (isdigit(*p)) {
                num = num * 10 + *p - 48;
                *p++;
            }
            Push(&S, num);
        }
        else {
            op1 = Pop(&S);
            op2 = Pop(&S);
            switch (*p) {
                case '+':
                    result = op2 + op1;
                    break;
                case '-':
                    result = op2 - op1;
                    break;
                case '/':
                    result = op2 / op1;
                    break;
                case '*':
                    result = op2 * op1;
                    break;
                default:
                    printf("\nInvalid Operator");
            }
            *p++;
        }
    }
    return result;
}

```

Giải thích: Dựa theo thuật toán, nên đọc tương ứng

```

        return 0;
    }
    Push(&S, result);
}
    p++;
}
result = Pop(&S);
return result;
}

```

(to be continue)

## TÀI LIỆU THAM KHẢO

10. Microsoft Visual Studio Community 2015
11. <http://www.cplusplus.com/>
12. Trần Đan Thư, *Nhập môn lập trình*, NXB khoa học và kỹ thuật.
13. Trần Đan Thư, *Kỹ thuật lập trình*, NXB khoa học và kỹ thuật.
14. Phạm Thế Bảo, *Cấu trúc dữ liệu và giải thuật*, NXB ĐHQG Tp. Hồ Chí Minh.
15. <http://vietjack.com/> , <https://nguyenvanhieu.vn/>
16. Nguyễn Hữu Anh, *Toán rời rạc*, NXB Giáo Dục.
17. Trần Đan Thư – Dương Anh Đức, *Lý thuyết đồ thị*, NXB ĐHQG Tp. Hồ Chí Minh.



**You should learn  
from your competitor,  
but never copy.  
Copy and you die.**

- JACK MA