

DISCRETE MATHEMATICS

Tài liệu thực hành toán rời rạc

Phạm Phi Nhung
phamphinhung2898@gmail.com

Tháng 11 – 2021
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.

Các bài tập chính (có thể) 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.

Hoặc download **Visual Studio Code**

- Link đề xuất: <https://code.visualstudio.com/>
- Hướng dẫn cài đặt C/C++:
<https://code.visualstudio.com/docs/languages/cpp/>
https://www.youtube.com/watch?v=77v-Poud_io

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 tương ứng)
- Tại giao diện trang chủ: chọn môn học **MTH10406**
- 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 ôn tập liên quan đến môn học

MỤC LỤC

DISCRETE MATHEMATICS.....	1
CHƯƠNG TRÌNH HỌC THỰC HÀNH.....	2
MỤC LỤC	4
YÊU CẦU THỰC HÀNH	5
CÁCH NỘP BÀI TẬP TRÊN SAKAI	Error! Bookmark not defined.
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.....	7

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 (thường là trong **Tests & Quizzes**).

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 đều phải khai báo các thông tin như sau:

```
/*
 * MSSV: 181101xxx
 * Ho va ten: Nguyen Van A
 * Assignment: bai1 / bt phu 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); lưu ý file có hàm main đặt tên file có lưu hàm main là **MSSV_main.c** hoặc **MSSV_main.cpp**
- 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 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 chính:
 - Các bài tập sẽ được tính trên thang điểm 10
 - Điểm bonus (nếu có): tối đa 10
 - Tổng điểm thực hành = (bài 1 + bài 2 + bài 3 + bonus) x 30%

CÁC TÀI LIỆU THAM KHẢO

- www.cplusplus.com
- Cấu trúc dữ liệu và giải thuật (Phạm Thế Bảo)
https://ir.vnulib.edu.vn/flowpaper/simple_document.php?subfolder=91/03/13/&doc=91031388825966094679924945170048053863&bitsid=0712187e-72d3-46d6-a9d4-6efc1ffa816f&uid/
- Nhập môn lập trình
- Kỹ thuật lập trình

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 <stdio.h> int main(){ /* Đây là chương trình C*/ printf("Hello, World! \n"); return 0; }</pre>	<pre>#include <stdio.h> void main(){ // Chương trình C printf("Hello, World! \n"); }</pre>	<pre>#include <iostream> using namespace std; /* Hàm main()*/ int main(){ // In dòng chữ Hello, World! cout << "Hello, World!" << endl; return 0; }</pre>

Giải thích:

1. Dòng đầu tiên của chương trình `#include <stdio.h>` là lệnh khai báo thư viện, giúp nhắc nhở bộ biên dịch sử dụng thêm tệp `stdio.h` trước khi chạy biên dịch.
2. Dòng tiếp theo `int main()` hoặc dòng `void main()` là nơi chương trình chính được bắt đầu.
3. Dòng tiếp theo `/*...*/` hoặc `//` 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.
4. Dòng tiếp theo `printf(..)` 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 `\n` lúc này được xem như là một hiệu lệnh xuống dòng.
5. Dòng `return 0` kết thúc hàm chính và trả về giá trị 0

Giải thích:

1. Tương tự như C, dòng `#include <iostream>` là lệnh khai báo thư viện (hay có thể gọi như là header `<iostream>` cần thiết để sử dụng các hàm tương ứng).
2. Dòng `using namespace std` nói cho compiler sử dụng `std namespace` (phần bổ sung của C++, sẽ hiểu rõ hơn khi học sâu về C++)
3. Tương tự như C thì C++ cũng dùng `int main()` hoặc `void main()` cho chương trình chính và cách thức comment chú thích cũng tương tự.
4. Dòng `cout << ...`; là một hàm chức năng tương tự như `printf` trong C. và `<<endl` xem như là hiệu lệnh xuống dòng.

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;
```



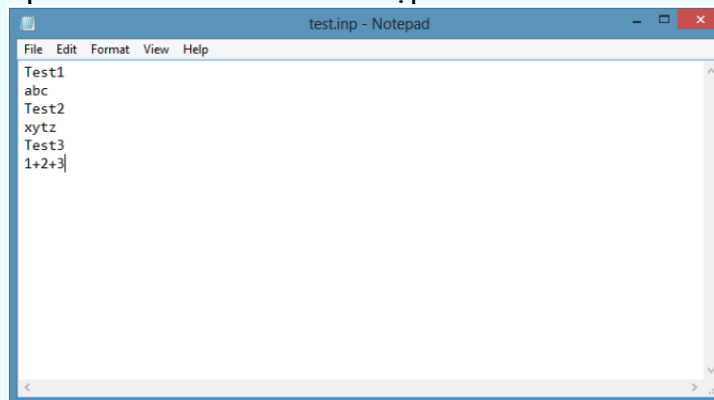
```
// hoặc:  
  
enum hanghoa { sua, nuocngot=40, biachai };  
  
// lúc này giá trị biachai sẽ có giá trị là 41 vì mỗi tên sẽ có giá trị lớn hơn của tên trước đó là 1
```

3. Đọc tập tin:

Với kỹ thuật thực hành trên tập tin, tức input từ một tập tin .txt nào đó chứ không pháp nhập trực tiếp từ màn hình. Do đó, sinh viên cần phải biết cách đọc dữ liệu từ tập tin. Dưới đây là 2 cách được khuyến nghị thực hiện vì cách làm đơn giản và phù hợp với môn học hơn.

a. Sử dụng kiểu ifstream, ofstream:

Đây là kiểu đọc tập tin cơ bản trong thư viện `<fstream>`, ifstream là viết tắt của input file stream, ofstream là output file stream. Giả sử có tập tin như sau :



Để thuận tiện cho việc đọc tập tin, thì đề xuất nên để tập tin vào thư mục chứa project đang sử dụng để làm bài

Name	Date modified	Type	Size
Debug	12/03/2017 11:44 ...	File folder	
ConsoleApplication1.vcxproj	12/03/2017 1:38 SA	VCXPROJ File	8 KB
ConsoleApplication1.vcxproj.filters	12/03/2017 1:05 SA	VC++ Project Filte...	1 KB
Source.cpp	18/03/2017 1:53 SA	C++ Source	1 KB
test.inp	18/03/2017 11:11 ...	INP File	0 KB

Ví dụ 1: (đọc tập tin –ifstream) Với yêu cầu đề bài là đọc tất cả nội dung có trong tập tin *test.inp* và xuất ra màn hình

```

1  /*
2   * Student ID: 1511000
3   * Name: Nguyen Van A
4   * Assignment: bail
5   * Created: 30/02/2017
6   * IDE: MS Visual Studio 2015
7   */
8
9  #include <iostream>
10 #include <fstream>
11 #include <string>
12 using namespace std;
13
14 int main()
15 {
16     ifstream fileIn;
17     fileIn.open("test.inp");
18
19     while (!fileIn.eof())
20     {
21         string s;
22         fileIn >> s;
23         cout << s << endl;
24     }
25
26     fileIn.close();
27
28     return 0;
29 }

```

Giải thích:

- Dòng 10: Khai báo thư viện , phải có thư viện này thì mới sử dụng được ifstream, ofstream.
- Dòng 16: Khai báo biến ifstream có tên là fileIn;
- Dòng 17: Ta mở tập tin test.txt, lưu ý rằng nếu tập tin này không nằm trong thư mục chứa project thì ta phải chèn đường dẫn đến tập tin đó, ví dụ như fileIn.open("E:\\toanroirac\\test.inp");
- Dòng 19: Ta tiến hành đọc các nội dung trong tập tin bằng cách đọc từng dòng, từ đầu đến cuối tập tin. Cú pháp !fileIn.eof() có nghĩa rằng nếu fileIn chưa đọc hết tập tin thì sẽ thực hiện nội dung trong vòng lặp (chữ eof là viết tắt của end of file).
- Dòng 22: Ta đưa nội dung dòng trong tập tin vào chuỗi s theo cú pháp fileIn >> s, cú pháp này khá giống với cách nhập dữ liệu từ bàn phím cin >> s.
- Dòng 26: Sau khi thực hiện xong, ta phải đóng tập tin lại bằng cú pháp fileIn.close().

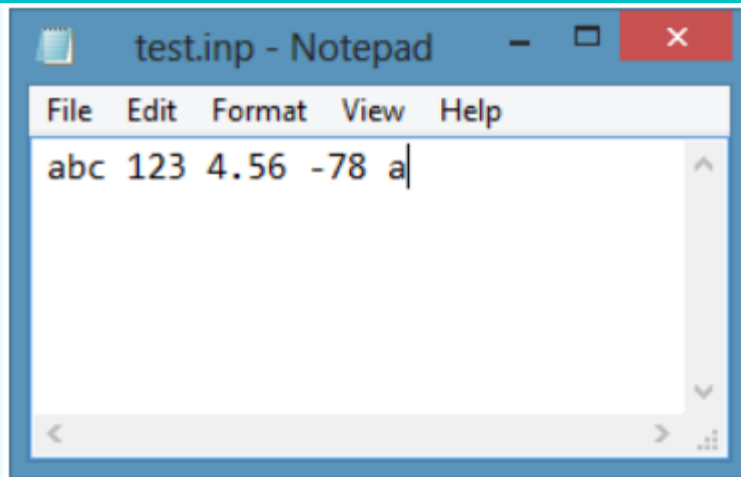
Kết quả sau khi chạy chương trình:

```

C:\Windows\system32\cmd.exe
Test1
abc
Test2
xyz
Test3
1+2+3
Press any key to continue . . .

```

Ví dụ 2 (đọc tập tin – ifstream): Nếu ta thay đổi cấu trúc và nội dung tệp như sau:

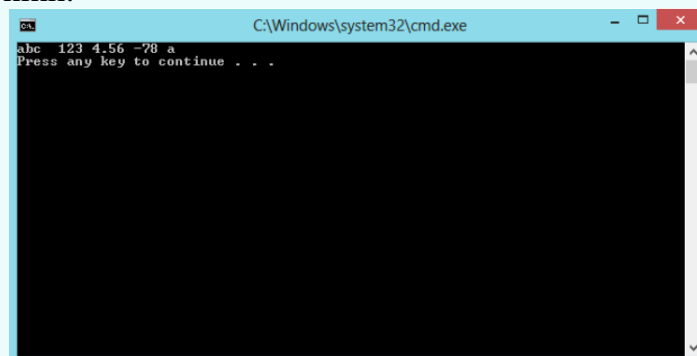


Ta thấy rằng cấu trúc nội dung này là chuỗi abc, số 123, số 4.56, số -78, ký tự a, do đó để hiển thị nội dung này ra màn hình, ta làm như sau:

```
19 while (!fileIn.eof())
20 {
21     string s;
22     int n1;
23     float n2, n3;
24     char c;
25     fileIn >> s >> n1 >> n2 >> n3 >> c;
26     if (fileIn.eof())
27         break;
28     cout << s << " " << n1 << " " << n2 << " " << n3 << " " << c << endl;
29 }
30
31 fileIn.close();
```

Ở dòng 21 đến 24, ta lần lượt khai báo chuỗi và kiểu số thực. Dòng 25 ta đẩy các giá trị trong tập tin vào các biến tương ứng.

Kết quả xuất ra màn hình:



Tại dòng 26, cần có điều kiện nếu hết tập tin thì thoát khỏi vòng lặp tại dòng 27. Nếu không có điều kiện này, chương trình sẽ xuất ra lỗi (lý do xem tại:

<http://stackoverflow.com/questions/21647/reading-from-text-file-until-eof-repeats-last-line>).

```
abc 123 4.56 -78 a
123 4.56 -78 a
Press any key to continue . . .
```

Ngoài ra ta cũng có thể viết đơn giản thành:

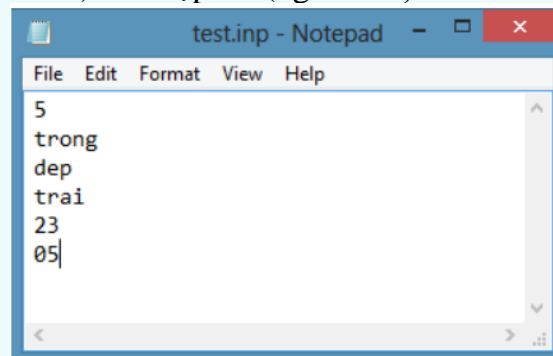
```

19     string s;
20     int n1;
21     float n2, n3;
22     char c;
23
24     while (fileIn >> s >> n1 >> n2 >> n3 >> c)
25     {
26         cout << s << " " << " " << n1 << " " << n2 << " " << n3 << " " << c << endl;
27     }
28
29     fileIn.close();

```

Ở đoạn chương trình này, dòng 24, thay vì sử dụng `!fileIn.eof()` thì ta để trực tiếp phép gán vào điều kiện vòng lặp, khi chương trình đã đọc đến hết tập tin, khi đó ta không thể gán được nữa, chương trình sẽ thoát.

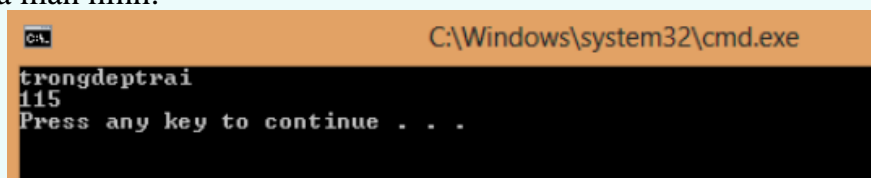
Ví dụ 3: (viết tập tin – ofstream) Cho tập tin (nguồn cũ)



trong đó dòng đầu tiên là số lượng các dòng bên dưới, dòng 2 đến dòng 4 là chuỗi, dòng 5 và 6 là số. Yêu cầu hãy nối dài chuỗi, còn số thì nhân lại, sau đó xuất kết quả ra tập tin tên “result.inp”, trong đó dòng đầu tiên là chuỗi nối dài, dòng thứ 2 là kết quả sau khi nhân. Đầu tiên, ta thực hiện các yêu cầu của bài, trừ việc in kết quả ra tập tin

<pre> 1 /* 2 * Student ID: 1511000 3 * Name: Nguyen Van A 4 * Assignment: bail 5 * Created: 30/02/2017 6 * IDE: MS Visual Studio 2015 7 */ 8 9 #include <iostream> 10 #include <fstream> 11 #include <string> 12 using namespace std; 13 14 int main() 15 { 16 ifstream fileIn; 17 fileIn.open("test.inp"); 18 string s = ""; 19 int k = 1; 20 21 if(fileIn.is_open()) 22 { 23 int n; </pre>	<pre> 24 fileIn >> n; 25 26 for (int i = 1; i <= n; i++) 27 { 28 if (i <= 3) 29 { 30 string s1; 31 fileIn >> s1; 32 s += s1; 33 } 34 else 35 { 36 int k1; 37 fileIn >> k1; 38 k *= k1; 39 } 40 } 41 42 cout << s << endl; 43 cout << k << endl; 44 } 45 else 46 cout << "Can't open file" << endl; 47 fileIn.close(); </pre>	<ul style="list-style-type: none"> - Dòng 21: Điều kiện này cho biết chương trình đã mở được tập tin chưa, nếu chưa sẽ chuyển đến dòng 46. - Dòng 24: Đưa giá trị 5, số ở dòng đầu tiên trong tập tin vào biến n. - Dòng 31: Đưa chuỗi trong tập tin vào biến chuỗi s1, sau đó ghép vào biến chuỗi s ở dòng 32. - Dòng 37: Đưa số trong tập tin vào biến số k1, sau đó nhân với biến k ở dòng 38 để ra kết quả.
--	--	---

Kết quả xuất ra màn hình:



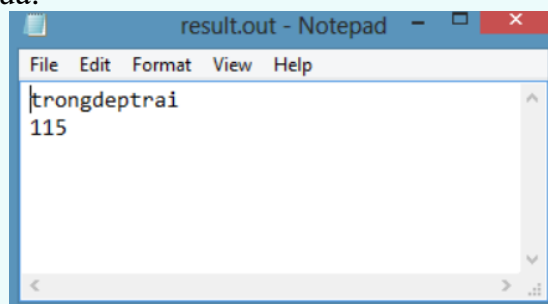
Bây giờ ta sẽ thực hiện việc đưa kết quả ra tập tin kết quả.

49	<code>ofstream fileOut;</code>	- Dòng 49: Ta khai báo fileOut là biến kiểu ofstream.
50	<code>fileOut.open("result.out", ofstream::out);</code>	- Dòng 50: Mở tập tin "result.out" để viết kết quả, nếu như tập tin này chưa tồn tại thì máy sẽ tự động tạo. Ta thấy có lệnh "ofstream::out", lệnh này cho biết tập tin đang mở ra dùng để viết.
51	<code>fileOut << s << endl;</code>	- Dòng 51: Đầu tiên, ta đưa chuỗi sau khi ghép vào tập tin trước bằng ký hiệu <<<).
52	<code>fileOut << k;</code>	- Dòng 52: Đưa số vào tập tin
53		
54	<code>return 0;</code>	

Sau khi chạy chương trình, ta vào thư mục project, sẽ thấy tập tin "result.txt"

Name	Date modified	Type	Size
Debug	19/03/2017 12:10 SA	File folder	
ConsoleApplication1.vcxproj	12/03/2017 1:38 SA	VCXPROJ File	8 KB
ConsoleApplication1.vcxproj.filters	12/03/2017 1:05 SA	VC++ Project Filte...	1 KB
result.out	19/03/2017 12:11 SA	OUT File	1 KB
Source.cpp	19/03/2017 12:10 SA	C++ Source	1 KB
test.inp	19/03/2017 12:11 SA	INP File	1 KB

Mở tập tin này sẽ có kết quả:



Nhận xét rằng ở dòng 50, lệnh "ofstream::out" cho biết đây là tập tin để viết, mỗi khi bạn chạy chương trình, toàn bộ thông tin trong tập tin "result.out" sẽ mất hết và nội dung mới sẽ được ghi vào. Nhưng nếu bạn muốn ghi nội dung mới nối tiếp nội dung cũ thì sao? Bạn chỉ cần thêm lệnh ofstream::app vào thành fileOut.open("result.out", ofstream::out | ofstream::app). Xem thêm về một số chế độ ghi tại <http://www.cplusplus.com/reference/fstream/ofstream/open/>, phần "Parameters", mục "mode".

Tham khảo thêm:

- fstream: <http://www.cplusplus.com/reference/fstream/fstream/>
- ifstream: <http://www.cplusplus.com/reference/fstream/ifstream/>
- ofstream: <http://www.cplusplus.com/reference/fstream/ofstream/>

b. Sử dụng kiểu freopen:

Một cách khác có thể sử dụng để đọc và viết tập tin là dùng hàm freopen, Hàm này có trong thư viện với cách sử dụng đơn giản, dùng trực tiếp cin, cout để xử lý dữ liệu trong tập tin thay vì trước đây ta thường dùng cin nhập dữ liệu từ bàn phím và cout để xuất kết quả ra màn hình console.

Ví dụ 1 (đọc tập tin): Cho tập tin sau, dòng đầu tiên chỉ số lượng các dòng, các dòng sau đó, mỗi dòng thứ i có cú pháp “Test i : a ”, với a là một con số, xác định xem số a này có chia hết cho 2 hay không. Nếu có, in kết quả ra màn hình là “Test i : YES”, nếu không, in ra “Test i : NO”.

```

test.inp - Notepad
File Edit Format View Help
4
Test1: 5
Test2: 6
Test3: 8
Test4: 1

```

```

1  /*
2   * Student ID: 1511000
3   * Name: Nguyen Van A
4   * Assignment: 1
5   * Created: 30/02/2017
6   * IDE: MS Visual Studio 2015
7   */
8
9  #include <iostream>
10 #include <cstdio>
11 #include <string>
12 using namespace std;
13
14 int main()
15 {
16     freopen("test.inp", "r", stdin);
17     int n;
18     cin >> n;
19     for (int i = 1; i <= n; i++)
20     {
21         string s;
22         int a;
23         cin >> s >> a;
24         cout << s << " ";
25         if (a % 2 == 0)
26             cout << "YES" << endl;
27         else
28             cout << "NO" << endl;
29     }
30     return 0;
31 }

```

- Dòng 10: Khai báo thư viện để sử dụng freopen.
- Dòng 16: Khai báo freopen, trong đó ta khai báo tên tập tin cần đọc “test.inp”, phần tiếp theo có chữ “r”, ký hiệu cho biết tập tin này chỉ dùng để đọc (read). Phần cuối cùng có chữ stdin, cho biết để nhập dữ liệu từ tập tin vào các biến, ta sử dụng cú pháp giống như nhập từ bàn phím.
- Dòng 18: Như đã khai báo ở dòng 16, do ta nhập dữ liệu vào các biến theo cú pháp nhập từ bàn phím nên lệnh cin >> n sẽ đưa con số đầu tiên trong tập tin vào biến n chứ không hiện ra màn hình console cho ta nhập số từ bàn phím. Tương tự với dòng 23.

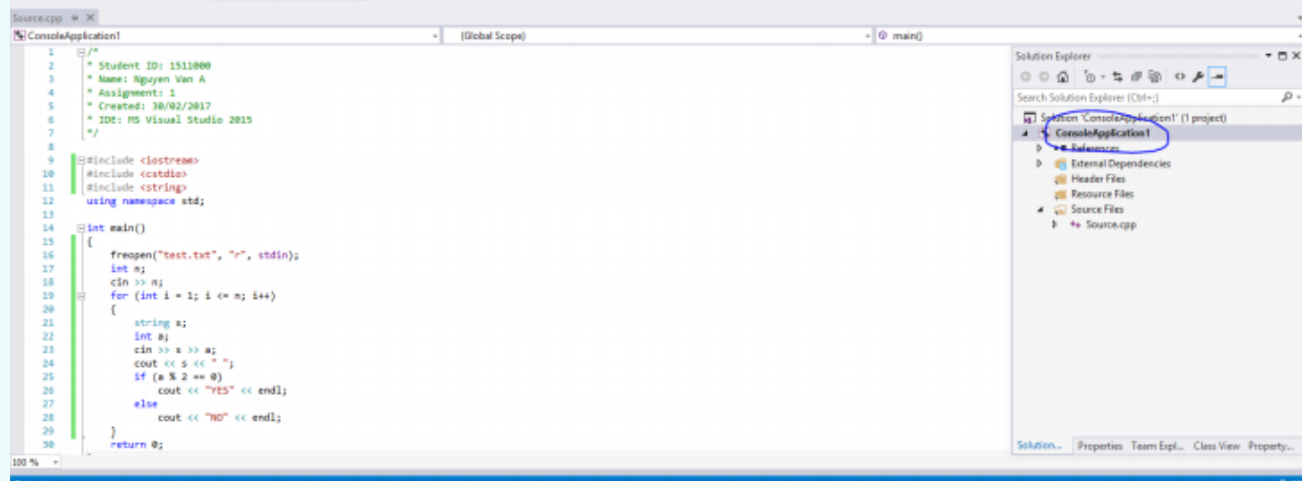
Kết quả tương ứng

```

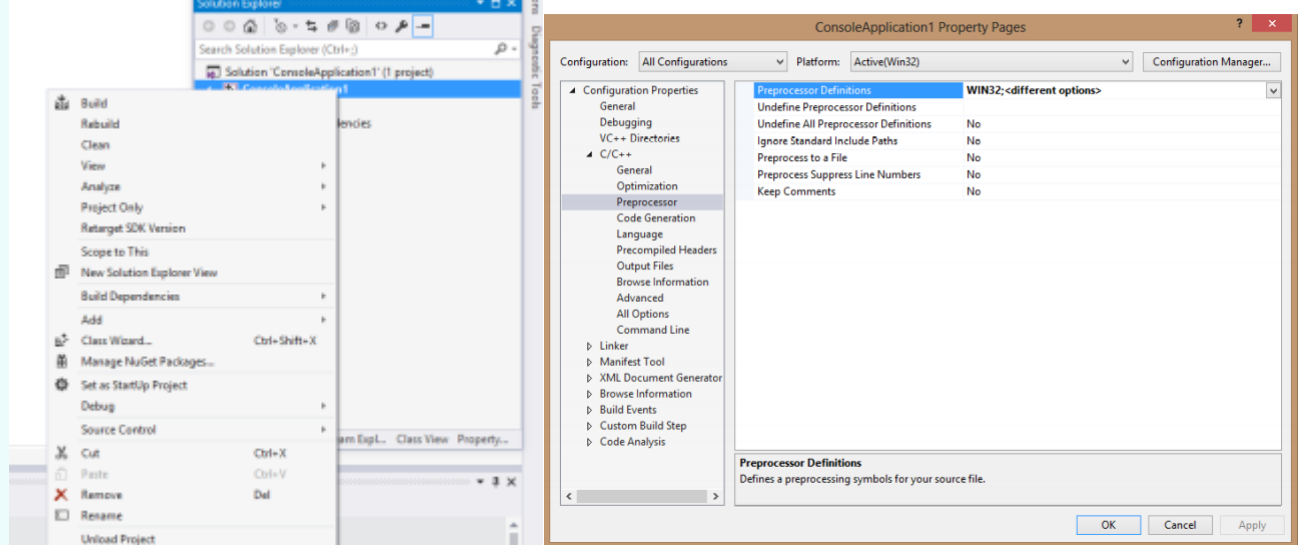
C:\Windows\system32\cmd.exe
Test1: NO
Test2: YES
Test3: YES
Test4: NO
Press any key to continue . . .

```

Lưu ý: Một số bạn khi sử dụng hàm freopen khi build chương trình sẽ bị lỗi “fopen: This function or variable may be unsafe. Consider using fopen_s instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS.” Để khắc phục lỗi này, bạn chuột phải vào tên project như hình dưới

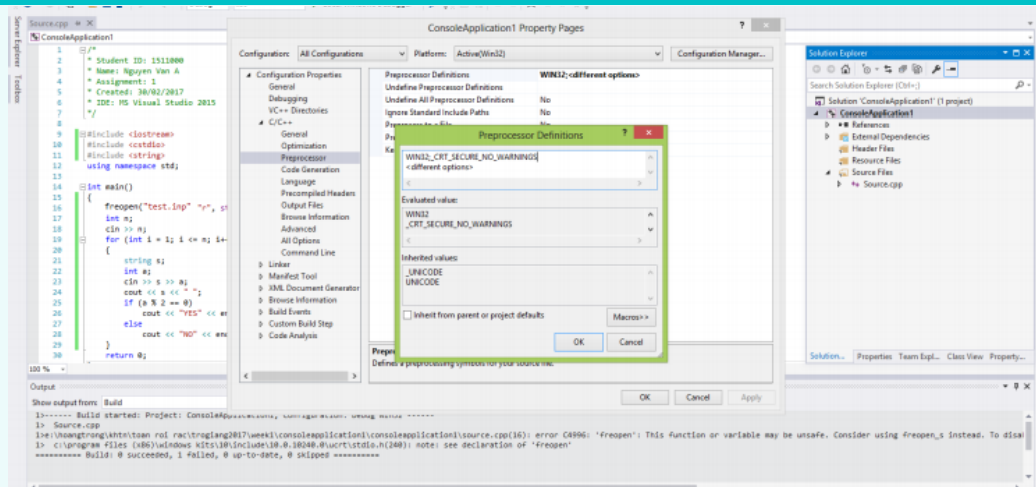


Ở mục “Configuration Properties”, chọn “C/C++”, chọn “Preprocessor”



Ở mục “Preprocessor Definition”, click vào dấu mũi tên, chọn “”, chèn lệnh

:_CRT_SECURE_NO_WARNINGS vào kế bên chữ “WIN32”, sau đó click “OK” và chạy chương trình.



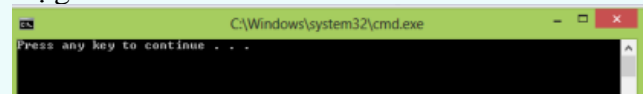
Ví dụ 2 (viết tập tin): để viết kết quả ra tập tin “result.out”, ta đơn giản khai báo như sau:

```

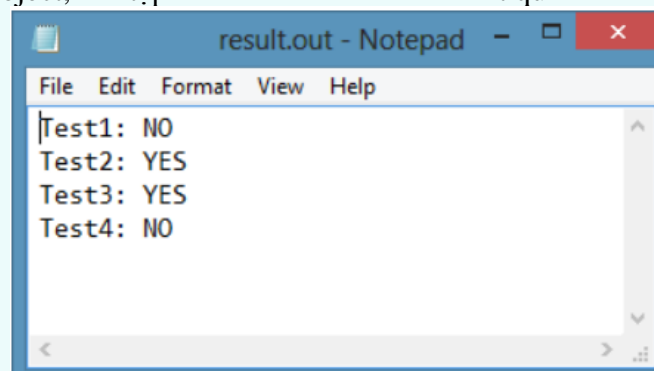
14 int main()
15 {
16     freopen("test.inp", "r", stdin);
17     freopen("result.out", "w", stdout);
18     int n;
19     cin >> n;
20     for (int i = 1; i <= n; i++)
21     {
22         string s;
23         int a;
24         cin >> s >> a;
25         cout << s << " ";
26         if (a % 2 == 0)
27             cout << "YES" << endl;
28         else
29             cout << "NO" << endl;
30     }
31     return 0;
32 }

```

Ở dòng 17 ta bổ sung thêm 1 hàm freopen chứa tập tin “result.out” ghi kết quả (“w”) với định dạng giống như ta in kết quả ra màn hình (“stdout”), do đó, ở dòng 25, 27 và 29, lệnh cout thay vì in ra màn hình, nó sẽ in thẳng kết quả ra tập tin và màn hình sau khi chạy sẽ không hiển thị gì.



Ta vào thư mục chứa project, mở tập tin “result.txt” và xem kết quả



Tham khảo thêm:

freopen: <http://www.cplusplus.com/reference/cstdio/freopen/>

c. Nên sử dụng ifstream/ofstream hay freopen?

Khi bạn chỉ đọc từ 1 tập tin và xuất kết quả ra một tập tin khác thì cả 2 cách đều dùng được, nhưng tôi khuyên nghị nên xài freopen vì cấu trúc đơn giản, sử dụng kiểu nhập xuất chuẩn cin, cout. Nếu bạn muốn nhập từ bàn phím, xuất ra màn hình, bạn chỉ việc comment lệnh freopen là xong. Khi bạn đọc từ nhiều tập tin hay xuất kết quả ra các tập tin khác nhau, bạn

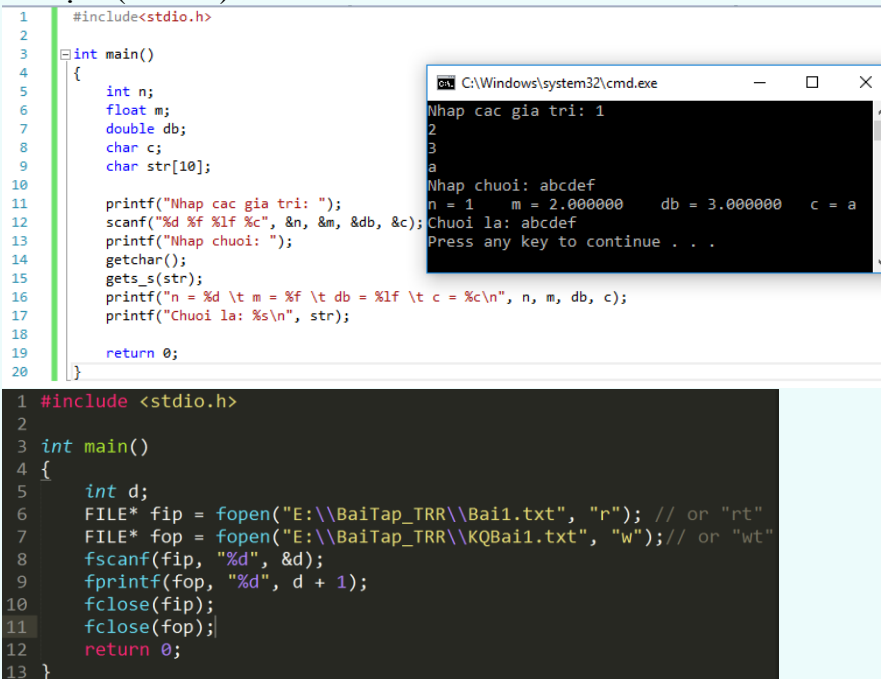
nên sử dụng ifstream/ofstream để dễ quản lý cách đọc, viết dữ liệu ra tập tin. Sau khi sử dụng xong nhớ đóng tập tin lại (ví dụ 1 phần ifstream/ofstream)

3. Một số hàm thư viện khác thường sử dụng trong C/C++:

- Thư viện nhập/ xuất - input/output library *stdio.h* / *cstudio.h* / *iostream*

C		C++	
Tên hàm	Công dụng	Tên hàm	Công dụng
scanf()	Nhận giá trị vào biến <i>scanf("%...%...", &var1, &var2,...);</i>	cin	Nhận giá trị vào <i>cin >> var1 >> var2;</i>
printf()	In thông báo/giá trị ra màn hình <i>printf("Van ban: %...", var);</i>	cout	In thông báo/giá trị ra màn hình <i>cout << "Văn bản" << var << endl</i>
gets()	Nhận vào một chuỗi <i>gets_s(var_str);</i>		

Ví dụ 1 (stdio.h)



```

1 #include<stdio.h>
2
3 int main()
4 {
5     int n;
6     float m;
7     double db;
8     char c;
9     char str[10];
10
11     printf("Nhap cac gia tri: ");
12     scanf("%d %f %lf %c", &n, &m, &db, &c);
13     printf("Nhap chuoi: ");
14     getchar();
15     gets_s(str);
16     printf("n = %d \t m = %f \t db = %lf \t c = %c\n", n, m, db, c);
17     printf("Chuoi la: %s\n", str);
18
19     return 0;
20 }

```

Output of the program:

```

Nhap cac gia tri: 1
2
3
a
Nhap chuoi: abcdef
n = 1      m = 2.000000    db = 3.000000    c = a
Chuoi la: abcdef
Press any key to continue . . .

```

```

1 #include <stdio.h>
2
3 int main()
4 {
5     int d;
6     FILE* fip = fopen("E:\\BaiTap_TRR\\Bai1.txt", "r"); // or "rt"
7     FILE* fop = fopen("E:\\BaiTap_TRR\\KQBai1.txt", "w"); // or "wt"
8     fscanf(fip, "%d", &d);
9     fprintf(fop, "%d", d + 1);
10    fclose(fip);
11    fclose(fop);
12    return 0;
13 }

```

Ví dụ 2 (cstudio.h)

```
1 #include<stdio>
2
3 int main()
4 {
5     int n;
6     float m;
7     double db;
8     char c;
9     char str[10];
10
11     printf("Nhap cac gia tri: ");
12     scanf("%d %f %lf %c", &n, &m, &db, &c);
13     printf("Nhap chuoi: ");
14     getchar();
15     gets_s(str);
16     printf("n = %d \t m = %f \t db = %lf \t c = %c\n", n, m, db, c);
17     printf("Chuoi la: %s\n", str);
18
19     return 0;
20 }
```

```
1 #include <iostream>
2 #include <cstdio>
3 using namespace std;
4 int main()
5 {
6     freopen("test.txt", "r", stdin);
7     freopen("test.txt", "w", stdout);
8     int a, b;
9     cin >> a >> b;
10    cout << a + b << endl;
11    cout << a * b << endl;
12    fclose(stdin);
13    fclose(stdout);
14    return 0;
15 }
```

Ví dụ 3: (iostream)

```
1 #include<iostream>
2 using namespace std;
3
4 int main()
5 {
6     int b;
7     cout << "Nhap b: " << endl;
8     cin >> b;
9     cout << "b + 1 = " << b + 1 << endl;
10    return 0;
11 }
```

Tham khảo thêm:

- https://www.tutorialspoint.com/c_standard_library/stdio_h.htm
- <http://www.cplusplus.com/reference/cstdio/>
- <http://www.cplusplus.com/reference/iostream/>

• Thư viện console input/output <conio.h>
Thư viện chứa các hàm điều khiển nhập xuất trên màn hình console.

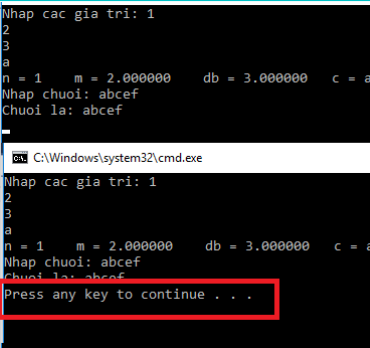
Tên hàm	Công dụng
clrscr()	Xóa thông báo cũ trên màn hình console
getch()	Đọc kí tự từ bàn phím

Ví dụ:

```

1 #include<stdio.h>
2 #include<conio.h>
3
4 int main()
5 {
6     int n;
7     float m;
8     double db;
9     char c;
10    char str[10];
11
12    printf("Nhap cac gia tri: ");
13    scanf("%d %f %lf %c", &n, &m, &db, &c);
14    printf("n = %d \t m = %f \t db = %lf \t c = %c\n", n, m, db, c);
15    printf("Nhap chuoi: ");
16    getchar();
17    gets_s(str);
18    printf("Chuoi la: %s\n", str);
19    getch();
20    return 0;
21 }

```



Tham khảo thêm:

<http://fresh2refresh.com/c-programming/c-function/conio-h-library-functions/>

- Thư viện tính toán **<math.h>/<c.math>**

Các hàm tiêu biểu:

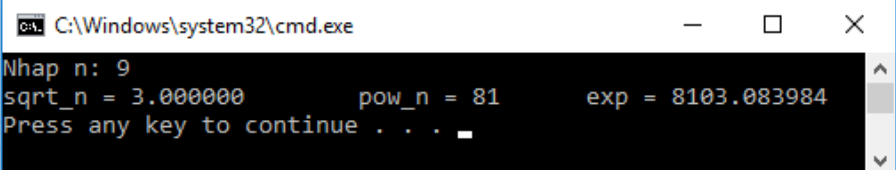
Tên hàm	Công dụng
sqrt()	Tính căn bậc 2 <i>sqrt(double Var);</i>
pow()	Tính lũy thừa <i>pow(double Var, double x);</i>
exp()	Tính lũy thừa của e <i>exp(double x);</i>

Ví dụ:

```

1 #include<stdio.h>
2 #include<math.h>
3
4 int main()
5 {
6     float n;
7
8     printf("Nhap n: ");
9     scanf("%f", &n);
10
11    float sqrt_n = sqrt(n);
12    int pow_n = pow(n, 2.0);
13    float exp_ = exp(n);
14    printf("sqrt_n = %f \t pow_n = %d \t exp = %f\n", sqrt_n, pow_n, exp_);
15
16    return 0;
17 }

```



Tham khảo thêm:

https://www.tutorialspoint.com/c_standard_library/math_h.htm

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

- Thư viện xử lý chuỗi **<string.h>**

Tên hàm	Công dụng
strlen()	Trả về kích thước của chuỗi

	<code>strlen(char str);</code>
<code>strcmp()</code>	So sánh 2 chuỗi <code>strcmp(char str1, char str2);</code>
<code>strcat()</code>	Nối 2 chuỗi <code>strcat(char destination, char source);</code>

Ví dụ 1:

```

1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char szInput[256];
7      printf("Enter a sentence: ");
8      gets_s(szInput);
9      printf("The sentence entered is %d characters.\n", strlen(szInput));
10     return 0;
11 }
```

C:\Windows\system32\cmd.exe

```
Enter a sentence: abcd
The sentence entered is 4 characters
Press any key to continue . . .
```

Ví dụ 2:

```

1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char key[] = "apple";
7      char buffer[80];
8      do {
9          printf("Guess my favorite fruit? ");
10         gets_s(buffer);
11     } while (strcmp(key, buffer) != 0);
12     printf("Correct Answer!\n");
13     return 0;
14 }
```

C:\Windows\system32\cmd.exe

```
Guess my favorite fruit? orange
Guess my favorite fruit? apple
Correct Answer!
Press any key to continue . . .
```

Ví dụ 3:

```

1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char str[80];
7      strcpy(str, "these ");
8      strcat(str, "strings ");
9      strcat(str, "are ");
10     strcat(str, "concatenated.");
11     puts(str);
12     return 0;
13 }
```

C:\Windows\system32\cmd.exe

```
these strings are concatenated.
Press any key to continue . . .
```

Tham khảo thêm:

https://www.tutorialspoint.com/c_standard_library/string_h.htm

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

- Thư viện `<stdlib.h>` / `<cstdlib>`

Một số hàm tiêu biểu:

Tên hàm	Công dụng
---------	-----------

abs()	Hàm tính trị tuyệt đối <i>abs(int n);</i>
atoi()	Chuyển chuỗi thành số nguyên <i>atoi(char str);</i>
atof()	Chuyển chuỗi thành số thực <i>atof(char str);</i>

Ví dụ:

```

1  #include<stdio.h>
2  #include<stdlib.h>
3
4  int main()
5  {
6      char m[] = "12";
7      int a;
8      int t = -3;
9      t = abs(t);
10     a = atoi(m);
11     printf("abs(t) = %d\n", t);
12     printf("Ket qua atoi\n");
13     printf("a + t = %d + %d = %d\n", a, t, a + t);
14
15     float b;
16     b = atof(m);
17     printf("Ket qua atof\n");
18     printf("b + t = %f + %d = %f\n", b, t, b + t);
19     return 0;
20 }

```

```

C:\Windows\system32\cmd.exe
abs(t) = 3
Ket qua atoi
a + t = 12 + 3 = 15
Ket qua atof
b + t = 12.000000 + 3 = 15.000000
Press any key to continue . . .

```

Tham khảo thêm:

https://www.tutorialspoint.com/c_standard_library/string_h.htm

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

•

4. 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 *tênmảng[index]*.

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>

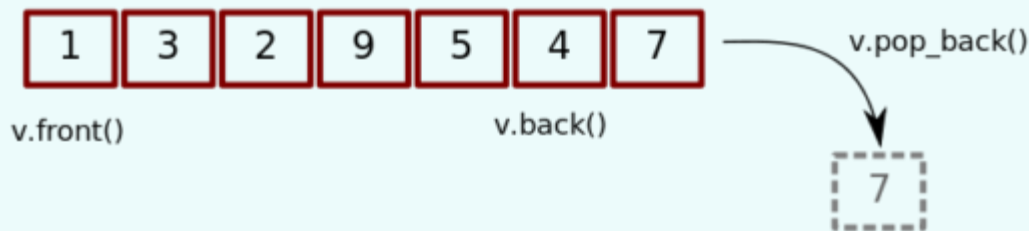
int main()
{
    // Khai báo mảng
    int arr[3][5] = {{5, 12, 17, 9, 3}, {13, 4, 8, 14, 1}, {9, 6, 3, 7, 21}};
    // Vòng lặp từng mảng (row_size)
    for(int i=0; i<3; i++) {
        for(int j=0; j<5; j++) {
            // Xuất từng phần tử
            printf("%5d", arr[i][j]);
        }
        //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

Ví dụ (trong C – chuyển sang hàm):

<pre>voidNhapMaTran(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]); } }</pre>	<pre>voidXuatMaTran(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"); } }</pre>
---	---

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*).

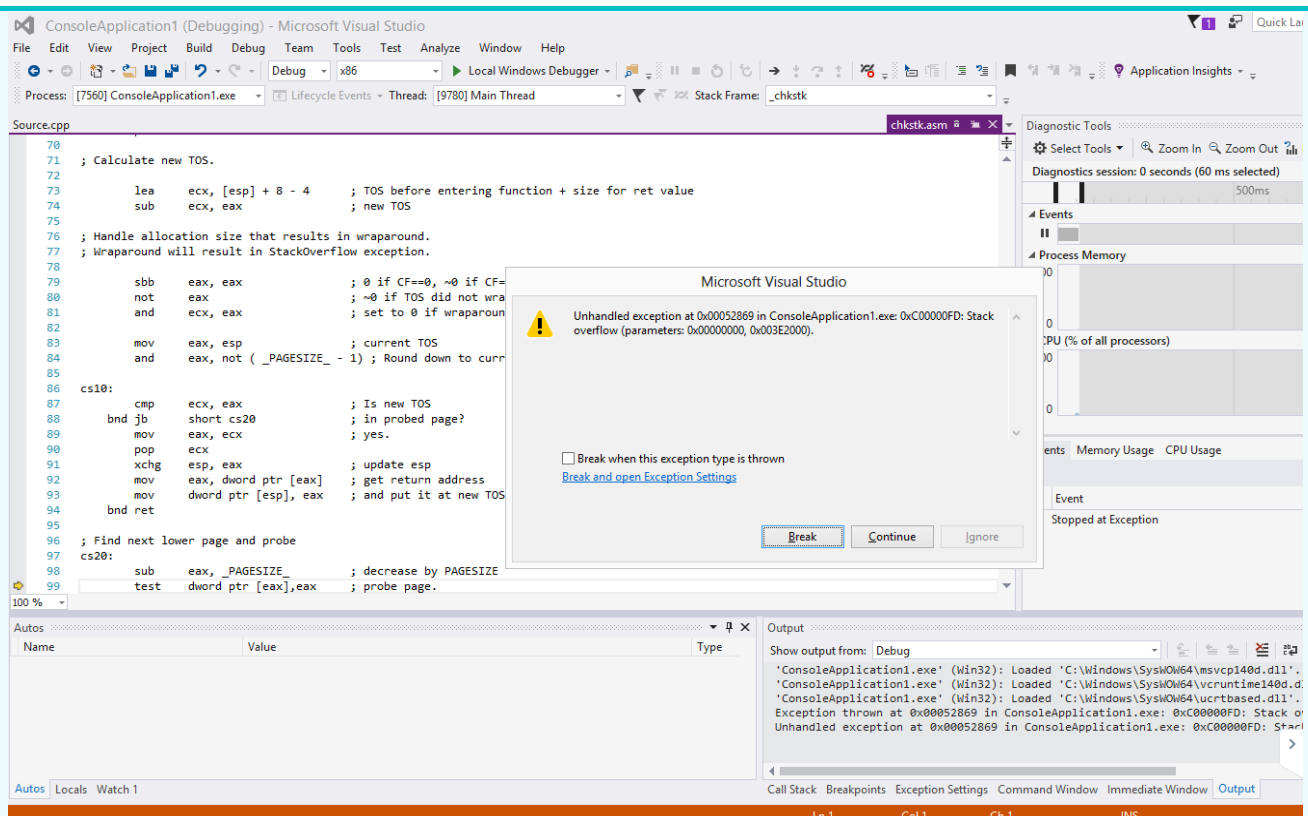


Trong C/C++, vector được biết đến thư viện `#include<vector>`, nếu muốn tạo mảng 1 chiều (tương tự như *array*) thì sẽ khai báo theo dạng `vector<kiểu_dữ_liệu> tên_biến`; trong đó *kiểu_dữ_liệu* có thể là *int*, *char*, *string*,; nếu muốn tạo mảng 2 chiều thì sẽ khai báo `vector<vector<kiểu_dữ_liệu>> 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 <iostream> using namespace std; void main(){ int mang[1000000] = {0}; cout << "hello"<< 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.

Ví dụ 1: (đưa vào vector, xuất ra tập tin)

Bây giờ ta sẽ thực hiện tạo một mảng có kích thước 1000000 chỗ gồm các giá trị từ 1 đến 1000000, sau đó xuất ra tập tin “result.txt” các phần tử trong mảng này từ 1000000 xuống 1.

```
1  /*
2   * Student ID: 1511000
3   * Name: Nguyen Van A
4   * Assignment: 1
5   * Created: 30/02/2017
6   * IDE: MS Visual Studio 2015
7   */
8
9  #include <iostream>
10 #include <vector>
11 #include <cstdio>
12 using namespace std;
13
14 int main()
15 {
16     vector<int> mang;
17     for (int i = 0; i < 1000000; i++)
18     {
19         mang.push_back(i + 1);
20     }
21
22     int n = mang.size();
23     freopen("result.txt", "w", stdout);
24     for (int i = n - 1; i >= 0; i--)
25     {
26         int a = mang[i];
27         cout << a << " ";
28     }
29     return 0;
30 }
```

- Dòng 10: Khai báo thư viện để dùng cấu trúc vector
- Dòng 11: Khai báo thư viện để viết kết quả ra tập tin.
- Dòng 16: Khai báo mảng theo cấu trúc vector, kiểu int.
- Dòng 19: Đưa lần lượt các giá trị vào vector bằng lệnh `push_back`(giá trị đưa vào mảng).
- Dòng 22: Xuất ra kích thước của mảng vector bằng lệnh `size()`.
- Dòng 23: Khai báo viết kết quả ra tập tin “result.txt” sử dụng cấu trúc `stdout`.
- Dòng 26: Lấy giá trị thứ `i` trong vector và gán vào biến `a`.
- Dòng 27: Xuất kết quả `a` ra tập tin, do ta khai báo cấu trúc xuất kết quả theo kiểu `stdout` nên lệnh `cout << a` sẽ đưa giá trị `a` vào tập tin chứ không có xuất ra màn hình. Bây giờ ta chạy chương trình, sau đó mở tập tin “result.txt” để xem kết quả

```
result.txt - Notepad
File Edit Format View Help
1000000 999999 999998 999997 999996 999995 999994 999993
99854 999853 999852 999851 999850 999849 999848 999847
708 999707 999706 999705 999704 999703 999702 999701 99
2 999561 999560 999559 999558 999557 999556 999555 9995
999415 999414 999413 999412 999411 999410 999409 999408
9269 999268 999267 999266 999265 999264 999263 999262 9
23 999122 999121 999120 999119 999118 999117 999116 999
998976 998975 998974 998973 998972 998971 998970 998969
8830 998829 998828 998827 998826 998825 998824 998823 9
84 998683 998682 998681 998680 998679 998678 998677 998
998537 998536 998535 998534 998533 998532 998531 998530
8391 998390 998389 998388 998387 998386 998385 998384 9
45 998244 998243 998242 998241 998240 998239 998238 998
```

Ta thấy rằng với kiểu vector, ta có thể đưa 1000000 phần tử vào mảng, trong khi khai báo mảng thông thường thì không thể. Do đó, kích thước vector phụ thuộc vào số lượng giá trị truyền vào. Ngoài ra, nếu ta muốn vector chứa ký tự, ta khai báo vector, còn nếu muốn chứa chuỗi, ta khai báo vector.

Ví dụ 2 (đưa vào vector, sắp xếp, ghép 2 vector thành 1)

Giả sử ta có một tập tin "test.txt", dòng 1 là số n , dòng 2 là n chuỗi, dòng 3 là số m , dòng 4 là m số âm, dòng 5 là số k , dòng 6 là k số dương.

Yêu cầu:

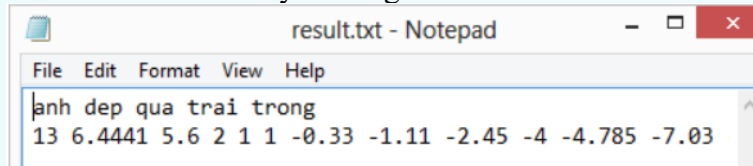
- Với mảng chuỗi, sắp xếp các phần tử tăng dần (từ a đến z).
- Với mảng số, ghép 2 mảng số lại thành 1, sau đó sắp xếp các phần tử mảng mới này giảm dần

```
1  /*
2   * Student ID: 1511000
3   * Name: Nguyen Van A
4   * Assignment: 1
5   * Created: 30/02/2017
6   * IDE: MS Visual Studio 2015
7   */
8
9  #include <iostream>
10 #include <vector>
11 #include <string>
12 #include <cstdio>
13 #include <algorithm>
14 using namespace std;
15
16 int main()
17 {
18     freopen("test.txt", "r", stdin);
19     freopen("result.txt", "w", stdout);
20     int n, m, k;
21     vector<string> chuoi;
22     vector<float> am;
23     vector<float> duong;
24
25     cin >> n;
26     for (int i = 0; i < n; i++)
27     {
28         string s;
29         cin >> s;
30         chuoi.push_back(s);
31     }
32
33     cin >> m;
34     for (int i = 0; i < m; i++)
35     {
36         float a;
37         cin >> a;
38         am.push_back(a);
39     }
40
41     cin >> k;
42     for (int i = 0; i < k; i++)
43     {
44         float d;
45         cin >> d;
46         duong.push_back(d);
47     }
48
49     duong.insert(duong.end(), am.begin(), am.end());
50     sort(chuoi.begin(), chuoi.end());
51     sort(duong.begin(), duong.end());
52     reverse(duong.begin(), duong.end());
53     for (int i = 0; i < n; i++)
54     {
55         cout << chuoi[i] << " ";
56     }
57     cout << endl;
58
59     int l = duong.size();
60     for (int i = 0; i < l; i++)
61     {
62         cout << duong[i] << " ";
63     }
64     return 0;
65 }
```

- Dòng 10: Khai báo thư viện vector để sử dụng cấu trúc vector.
- Dòng 12: Khai báo thư viện cstdio để sử dụng freopen đọc, viết tập tin.
- Dòng 13: Khai báo thư viện algorithm để dùng hàm sắp xếp.
- Dòng 21: Khai báo mảng vector kiểu string dùng để đưa dữ liệu chuỗi vào
- Dòng 22, 23: Khai báo mảng vector kiểu float dùng để đưa dữ liệu số âm, dương vào.

- Dòng 25 đến dòng 47: Lần lượt đưa các giá trị chuỗi, số vào vector tương ứng.
- Dòng 49: Ghép 2 vector am, duong vào vector duong. Trong cấu trúc insert này, lệnh `duong.end()` tức ta ghép vector mới vào cuối vector duong, lệnh `am.begin()` và `am.end()` có nghĩa ta lấy các giá trị từ đầu đến đuôi của vector am và gắn vào duong.
- Dòng 50: Sắp xếp các chuỗi trong vector chuoi từ đầu (`chuoi.begin()`) đến đuôi (`chuoi.end()`) tăng dần từ a đến z.
- Dòng 51: Sắp xếp các phần tử trong vector duong tăng dần. Tuy nhiên, đề yêu cầu ta sắp xếp giảm dần nên ta cần đảo ngược thứ tự trong vector duong.
- Dòng 52: Đảo ngược thứ tự các giá trị trong vector duong từ đầu đến đuôi.
- Dòng 53 đến dòng 63: Lần lượt đưa kết quả từ vector chuoi và vector duong ra tập tin “result.txt”.

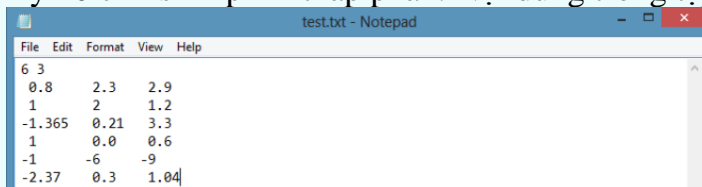
Kết quả trong tập tin result.txt sau khi chạy chương trình



Ví dụ 3 (vector 2 chiều)

Giống như trong mảng bình thường ta có mảng 1 chiều, 2 chiều, ... thì trong vector ta cũng có vector 1 chiều, 2 chiều, ... bằng cách khai báo `vector<vector>`, tức để 1 vector nằm trong 1 vector. Ta xem ví dụ sau.

Trong tập tin “test.txt” cho một mảng số thực 2 chiều gồm m dòng, n cột. Giả sử a là giá trị nhỏ nhất và b là giá trị lớn nhất trong mảng, tính b/a và xuất kết quả ra tập tin “result.txt”, lấy 10 chữ số ở phần thập phân. Nội dung trong tập tin “test.txt” như sau:



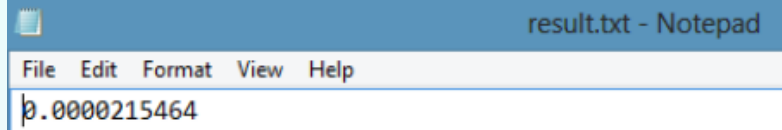
Dòng đầu tiên chứa 2 giá trị m và n . m dòng tiếp theo, mỗi dòng chứa n số thực.

```

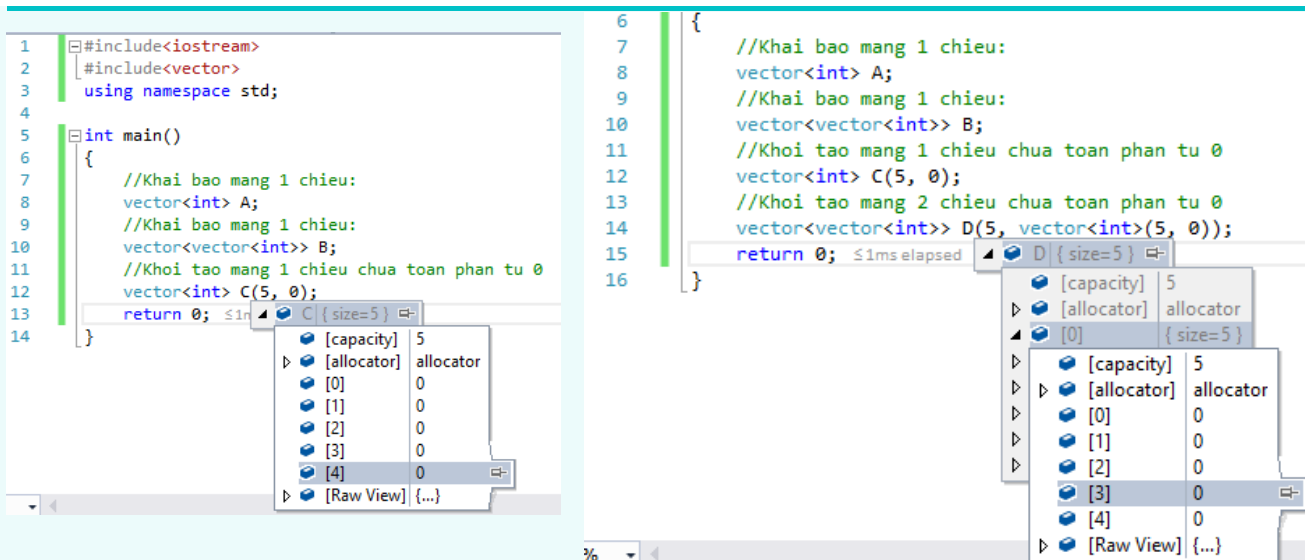
1  /*
2   * Student ID: 1511000
3   * Name: Nguyen Van A
4   * Assignment: 1
5   * Created: 30/02/2017
6   * IDE: MS Visual Studio 2015
7   */
8
9  #include <iostream>
10 #include <stdio.h>
11 #include <vector>
12 #include <cstdlib>
13 #include <cmath>
14 #include <iomanip>
15 using namespace std;
16
17 int main()
18 {
19     freopen("test.txt", "r", stdin);
20     freopen("result.txt", "w", stdout);
21     int m, n;
22     vector<vector<double>> mang2d;
23
24     cin >> m >> n;
25     for (int i = 0; i < m; i++)
26     {
27         vector<double> mang;
28         for (int j = 0; j < n; j++)
29         {
30             double a;
31             cin >> a;
32
33             mang.push_back(a);
34             mang2d.push_back(mang);
35         }
36     }
37     double a, b;
38     for (int i = 0; i < m; i++)
39     {
40         for (int j = 0; j < n; j++)
41         {
42             double c = mang2d[i][j];
43             if (i == 0 && j == 0)
44             {
45                 a = c;
46                 b = c;
47             }
48             else
49             {
50                 if (c < a)
51                     a = c;
52                 if (c > b)
53                     b = c;
54             }
55         }
56     }
57
58     double kq = pow(b, a);
59     cout << fixed << setprecision(10) << kq;
60
61     return 0;
62 }

```

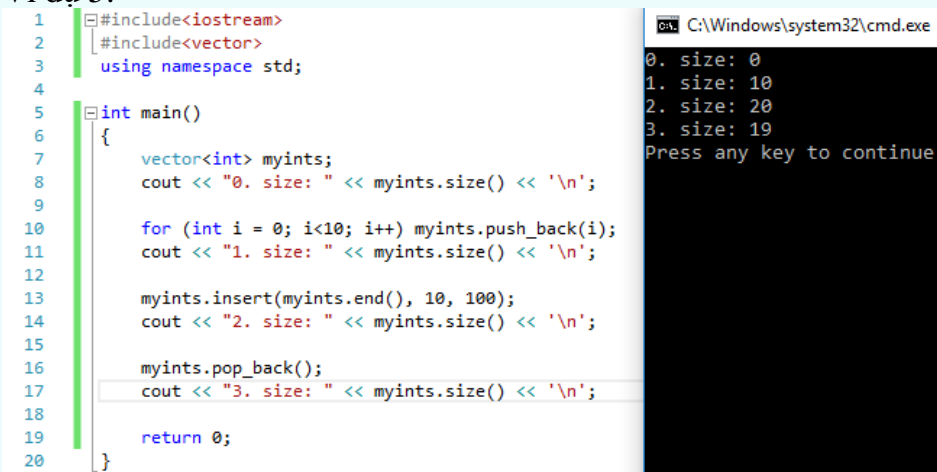
- Dòng 11: Khai báo thư viện để sử dụng cấu trúc vector.
 - Dòng 12: Khai báo thư viện để sử dụng freopen để đọc và viết tập tin.
 - Dòng 14: Khai báo thư viện để xuất ra 10 chữ số ở phần thập phân.
 - Dòng 22: Khai báo mảng vector 2 chiều kiểu double.
 - Dòng 25 đến dòng 35: Đưa các giá trị trong tập tin vào mảng 2 chiều bằng cách các giá trị ở mỗi dòng đều đưa vào một vector (dòng 27 đến dòng 33), sau đó ta đưa vector này vào vector mang2d (dòng 34)..
 - Dòng 37 đến dòng 56: Xác định giá trị nhỏ nhất a và giá trị lớn nhất b , xem vector này như là một mảng 2 chiều bình thường (dòng 42).
 - Dòng 58: Tính b^a .
 - Dòng 59: Xuất kết quả ra tập tin "result.txt", trong đó fix << setprecision(10) có chức năng lấy 10 giá trị ở phần thập phân.
- Kết quả trong tập tin "result.txt"



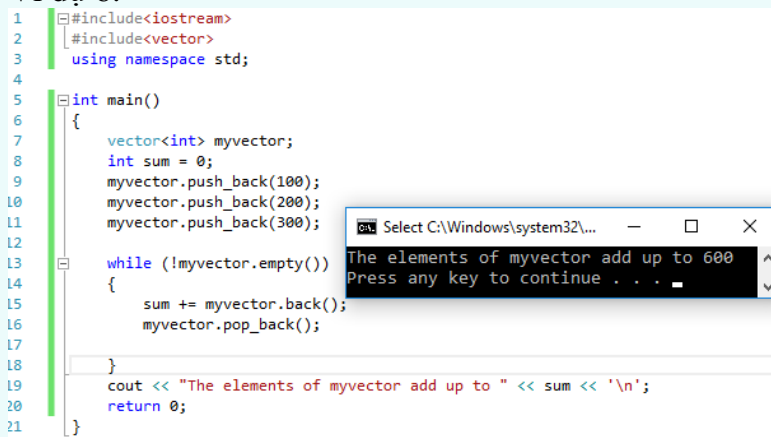
Ví dụ 4:



Ví dụ 5:



Ví dụ 6:



Tham khảo thêm:

- Vector :

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

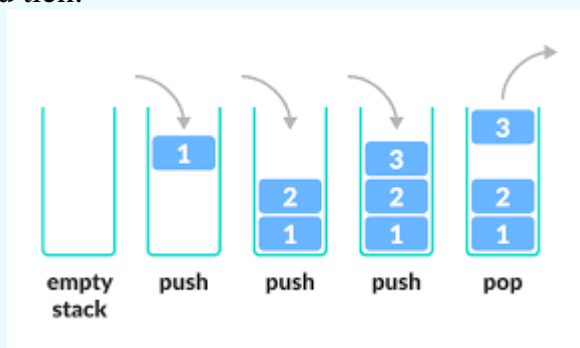
<https://topdev.vn/blog/vector-trong-c/>

- Sort: <http://www.cplusplus.com/reference/algorithm/sort/?kw=sort>

- Setprecision: <http://www.cplusplus.com/reference/iomanip/setprecision/?kw=setprecision>

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 <iostream> #include <stack> #include <string> using namespace std; int main() { // Nhập Stack stack<string> A; A.push("xep"); A.push("ngan"); A.push("la"); A.push("Stack"); // Xuất Stack: while (!A.empty()) { string s = A.top(); cout << s << " "; A.pop(); } cout << endl; return 0; }</pre>	<p>Tham khảo cách cài đặt trong C: https://nguyenvanhieu.vn/ngan-xep-stack/</p>

MỘT SỐ LỆNH VỀ STACK

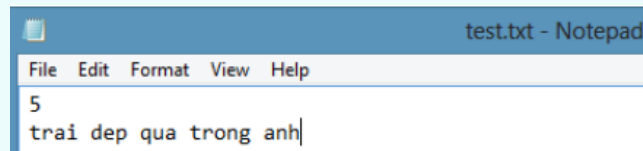
stack<kiểu_dữ_liệu> A;

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.

Với ví dụ sau, trong TH này được sử dụng freopen để đọc và viết kết quả từ tập tin .txt
Ta làm ví dụ sau: Trong tập tin “test.txt” có 2 dòng, dòng đầu tiên là giá trị *n*, dòng thứ 2 gồm *n* chuỗi. Đưa các chuỗi này vào ngăn xếp, sau đó lấy từng phần tử trong ngăn xếp này xuất ra màn hình, cho biết kích thước của ngăn xếp sau đó.

File input:

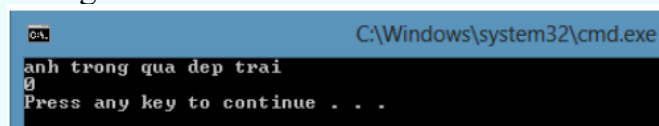


Chương trình mẫu:

```
1  /*
2   * Student ID: 1511000
3   * Name: Nguyen Van A
4   * Assignment: 1
5   * Created: 30/02/2017
6   * IDE: MS Visual Studio 2015
7   */
8
9  #include <iostream>
10 #include <cstdio>
11 #include <string>
12 #include <stack>
13 using namespace std;
14
15 int main()
16 {
17     freopen("test.txt", "r", stdin);
18     stack<string> str;
19     int n;
20     cin >> n;
21
22     for (int i = 1; i <= n; i++)
23     {
24         string s;
25         cin >> s;
26         str.push(s);
27     }
28     while (!str.empty())
29     {
30         string s = str.top();
31         cout << s;
32         str.pop();
33         if (!str.empty())
34             cout << " ";
35         else
36             cout << endl;
37     }
38     cout << str.size() << endl;
39     return 0;
40 }
```

- Dòng 12: Khai báo thư viện để xài ngăn xếp.
- Dòng 18: Khai báo một ngăn xếp gồm các chuỗi. Tương tự như vector, ta có thể khai báo ngăn xếp gồm các số nguyên, số thực, ... theo cú pháp stack.
- Dòng 21 đến dòng 26: Đưa các chuỗi trong tập tin vào ngăn xếp (dòng 25)
- Dòng 28 đến dòng 37: Lần lượt đưa các chuỗi trong ngăn xếp ra màn hình cho đến khi ngăn xếp rỗng (dòng 28), dòng 30 có ý nghĩa gán chuỗi *s* vào chuỗi đưa vào sau cùng của ngăn xếp, dòng 32 có ý nghĩa đưa chuỗi sau cùng đó ra ngoài ngăn xếp.
- Dòng 38: Xuất ra màn hình kích thước của ngăn xếp lúc này.

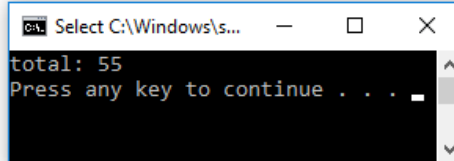
Kết quả sau khi chạy chương trình:



Nhận xét: Do hàng đợi thực hiện theo cấu trúc vào sau ra trước nên trong tập tin “test.txt”, chuỗi “anh” được đưa vào ngăn xếp sau cùng nên sẽ được đưa ra đầu tiên. Sau khi đưa hết tất cả các chuỗi ra khỏi ngăn xếp, hiển nhiên kích thước ngăn xếp lúc này bằng 0.

Ví dụ:

```
1 #include<iostream>
2 #include<stack>
3 using namespace std;
4
5 int main()
6 {
7     stack<int> mystack;
8     int sum = 0;
9
10    for (int i = 1; i <= 10; i++) mystack.push(i);
11
12    while (!mystack.empty())
13    {
14        sum += mystack.top();
15        mystack.pop();
16    }
17
18    cout << "total: " << sum << '\n';
19
20    return 0;
21 }
```



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


```

A.push('n');
A.push('T');
A.push('i');
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()*: Xoá phần tử *A.front()* ra khỏi queue.

Ví dụ:

```

1  /*
2   * Student ID: 1511000
3   * Name: Nguyen Van A
4   * Assignment: 1
5   * Created: 30/02/2017
6   * IDE: MS Visual Studio 2015
7   */
8
9  #include <iostream>
10 #include <cstdio>
11 #include <string>
12 #include <queue>
13 using namespace std;
14
15 int main()
16 {
17     freopen("test.txt", "r", stdin);
18     queue<string> str;
19     int n;
20     cin >> n;
21
22     for (int i = 1; i <= n; i++)
23     {
24         string s;
25         cin >> s;
26         str.push(s);
27     }
28
29     while (!str.empty())
30     {
31         string s = str.front();
32         cout << s;
33         str.pop();
34         if (!str.empty())
35             cout << " ";
36         else
37             cout << endl;
38     }
39     cout << str.size() << endl;
40     return 0;
}

```

Ta thấy rằng cấu trúc chương trình không khác biệt lắm so với ngăn xếp, chỉ khác vài chỗ

- Dòng 12: Khai báo thư viện để sử dụng hàng đợi.
- Dòng 30: Gán phần tử đầu tiên của hàng đợi vào biến s.

Kết quả sau khi chạy chương trình:

```
C:\Windows\system32\cmd.exe
traidepqua trong anh
0
Press any key to continue . . .
```

Ta thấy chuỗi kết quả giống với chuỗi trong tập tin do cơ chế FIFO của hàng đợi, chuỗi “traid” được đưa vào hàng đợi trước nên sẽ đưa ra khỏi hàng đợi đầu tiên.

Ví dụ:

```
1 #include<iostream>
2 #include<queue>
3 using namespace std;
4
5 int main()
6 {
7     queue<int> myqueue;
8     int sum = 0;
9
10    for (int i = 1; i <= 10; i++) myqueue.push(i);
11
12    while (!myqueue.empty())
13    {
14        sum += myqueue.front();
15        myqueue.pop();
16    }
17
18    cout << "total: " << sum << '\n';
19
20    return 0;
21 }
```

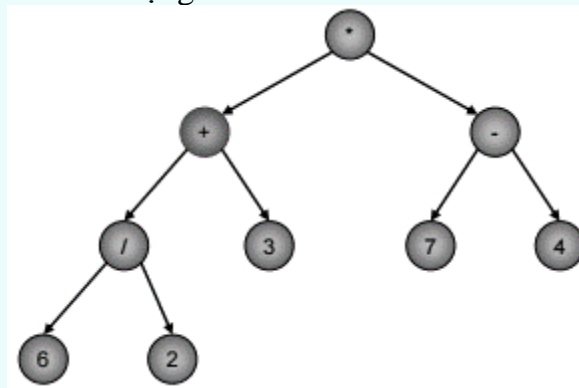
Tham khảo thêm:

<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 => 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>* => là phép toán, stack rỗng => 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 => 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>- => là phép toán, stack khác rỗng, ưu tiên thấp hơn phần tử đầu stack (do * có độ ưu tiên cao hơn -) => 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. => là phép toán, stack rỗng => đưa vào stack</div>	<div>2g. Xét phần tử 5</div> <div>(=> đưa vào stack</div>

$q = "7\ 8\ *"$ <div style="border: 1px solid black; height: 40px; width: 100%;"></div>	$q = "7\ 8\ *"$ <div style="border: 1px solid black; height: 40px; width: 100%;"></div>
<p>2h. Xét phần tử 6 $2 \Rightarrow$ là hằng số, đưa vào q</p> $q = "7\ 8\ *"$ <div style="border: 1px solid black; height: 40px; width: 100%;"></div>	<p>2i. Xét phần tử 7 $+$ \Rightarrow là phép toán, stack khác rỗng, độ ưu tiên cao hơn phép toán đầu stack (do $+$ cao hơn $() \Rightarrow$ đưa vào stack</p> $q = "7\ 8\ *\ 2"$ <div style="border: 1px solid black; height: 40px; width: 100%;"></div>
<p>2k. Xét phần tử 8 $3 \Rightarrow$ là hằng số, đưa vào q</p> $q = "7\ 8\ * 2"$ <div style="border: 1px solid black; height: 40px; width: 100%;"></div>	<p>2l. Xét phần tử 9 $) \Rightarrow$ lấy các phần tử của stack đưa vào q cho đến khi gặp "("</p> $q = "7\ 8\ * 2\ 3"$ <div style="border: 1px solid black; height: 40px; width: 100%;"></div>
<p>2m. $q = "7\ 8\ * 2\ 3\ +"$</p> <div style="border: 1px solid black; height: 40px; width: 100%;"></div>	<p>Kết thúc bước 2</p>

- Bước 3: Chuỗi phép toán kết thúc \Rightarrow 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 + -

Start	2a. Xét phần tử 1: 7 => là hằng số nên đưa vào stack						
<table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr></table>				<table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr></table>			
2b. Xét phần tử 2 8 => là hằng số nên đưa vào stack	2c. Xét phần tử 3 * => là toán tử => lấy 2 phần tử đầu tiên của stack => phần tử đầu bên phải, phần tử sau bên trái và phép toán ở giữa						
<table><tr><td>7</td></tr><tr><td></td></tr><tr><td></td></tr></table>	7			<table><tr><td>8</td></tr><tr><td>7</td></tr><tr><td></td></tr></table>	8	7	
7							
8							
7							
2d. Xét phần tử 3	2e. Xét phần tử 3						

<p>* => là toán tử => lấy 2 phần tử đầu tiên của stack => 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>* => là toán tử => lấy 2 phần tử đầu tiên của stack => 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;">$7 * 8 = 56$</p> <p>=>Đưa kết quả vào stack</p> <table><tr><td> </td></tr><tr><td> </td></tr><tr><td> </td></tr></table>				<p>2g. Xét phần tử 4</p> <p>2 => là hằng số nên đưa vào stack</p> <table><tr><td>56</td></tr><tr><td> </td></tr><tr><td> </td></tr></table>	56		
56							
<p>2h. Xét phần tử 5</p> <p>3 => 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</p> <p>+ => 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</p> <p>+ => 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</p> <p>+ => 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;">$2 + 3 = 5$</p> <p>⇒ 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>- => 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>- \Rightarrow 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 border="1" style="margin-left: auto; margin-right: auto;"><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>- \Rightarrow 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 border="1" style="margin-left: auto; margin-right: auto;"><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;">$56 - 5 = 51$</p> <p>\Rightarrow Đưa kết quả vào stack</p> <table border="1" style="margin-left: auto; margin-right: auto;"><tr><td> </td></tr><tr><td> </td></tr><tr><td> </td></tr></table>				<p>2r. Hết chuỗi biểu thức hậu tố \Rightarrow Kết thúc bước 2</p> <p>Stack cuối bước 2</p> <table border="1" style="margin-left: auto; margin-right: auto;"><tr><td>51</td></tr><tr><td> </td></tr><tr><td> </td></tr></table>	51		
51							

- Bước 3: Chuỗi phép toán kết thúc \Rightarrow 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:

<pre>int getPriority (string op){ if (op == "*" op == "/" op == "%") return 2; if (op == "+" op == "-") return 1; return 0; }</pre>	<p>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.</p>
--	--

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)){
                        x = Pop(&S);
                        postfix[j++] = x;
                    }
                    Push(&S, token);
                }
    }

    while (!isEmpty(&S)) {
        x = Pop(&S);
        postfix[j++] = x;
    }
    postfix[j] = '\0';
}

```

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

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)) {

```

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

```

        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");
    }
    return 0;
}
Push(&S, result);

}

p++;

}
result = Pop(&S);
return result;
}
}

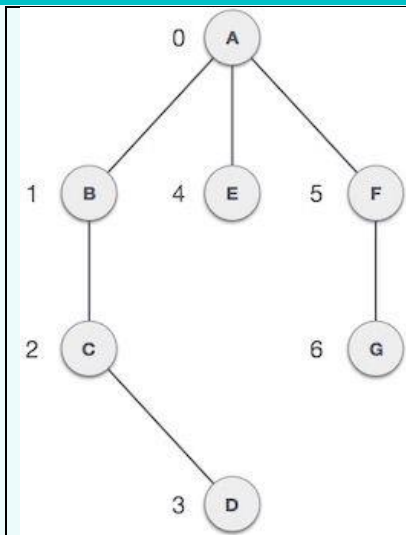
```

9. Biểu diễn đồ thị trong lập trình

a. Cấu trúc dữ liệu đồ thị:

Một đồ thị là một dạng biểu diễn hình ảnh của một tập các đối tượng, trong đó các cặp đối tượng được kết nối bởi các link. Các đối tượng được nối liền nhau được biểu diễn bởi các điểm được gọi là các đỉnh (vertices), và các link mà kết nối các đỉnh với nhau được gọi là các cạnh (edges).

Kiểu đồ thị tương tự khác có thể hình dung như sau:

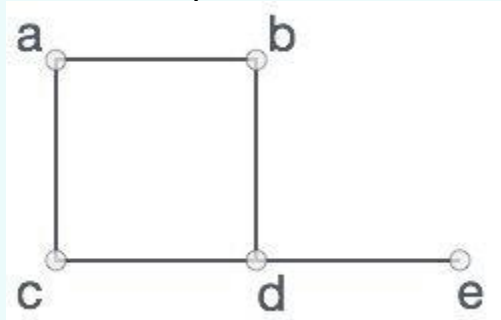


Với mỗi node (nút) của hình được biểu diễn như là 1 đỉnh, trong trường hợp này thì các điểm từ A tới G là đỉnh. Nếu lưu các đỉnh này theo mảng, thì có thể lưu theo các chỉ số của mảng với A tại chỉ mục 0, B tại chỉ mục 1,... Ngoài ra các đường nối giữa A và B, B và C,... là các cạnh => lúc này có thể sử dụng mảng 2 chiều để biểu diễn các cạnh này.

Một đỉnh nghĩa khác về 2 đỉnh được xem là kề nhau nếu chúng được kết nối với nhau trong TH này thì B kề với A, C kề với B, C không kề với A,...

Và đường đi được biểu diễn lúc này là một dãy các cạnh giữa 2 đỉnh, trong hình thì có thể hình dung ABCD với đường biểu diễn 1 đường từ A đến D.

Cụ thể hơn, một đồ thị có thể hình dung như một cặp các tập hợp (V, E) , trong đó V là tập các đỉnh và E là tập các cạnh mà kết nối với các cặp điểm, ví dụ trong trường hợp đồ thị sau:



Trong đồ thị trên:

$V = \{a, b, c, d, e\}$

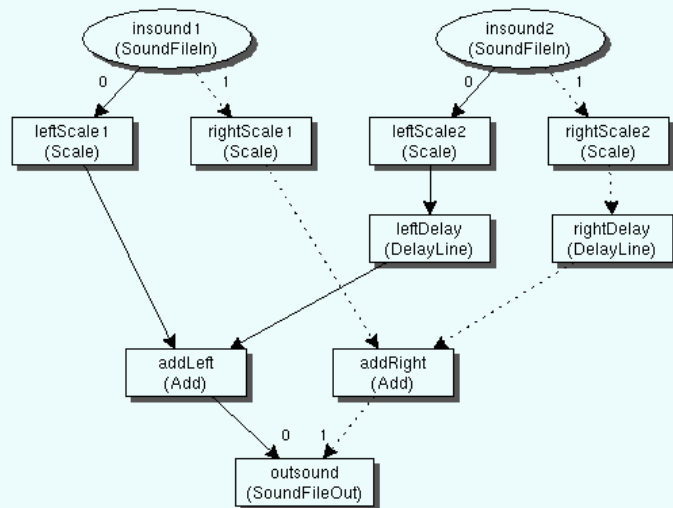
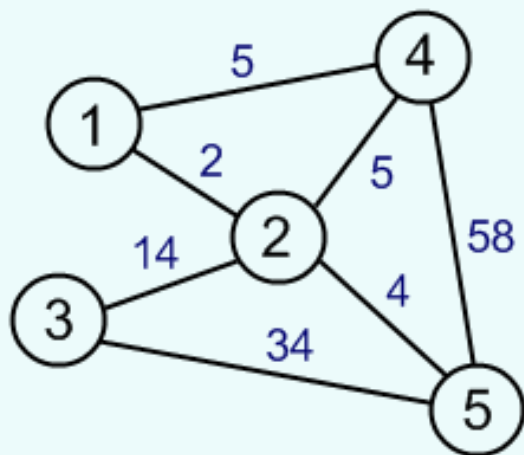
$E = \{ab, ac, bd, cd, de\}$

Tóm lại, đồ thị có thể hình dung là một cấu trúc dữ liệu gồm có 2 phần:

- Tập hữu hạn các đỉnh
- Tập hữu hạn các cạnh nối 2 đỉnh (u, v)
- Cạnh có thể có thêm trọng số (biểu diễn độ dài, chi phí,...)

Có 2 loại đồ thị:

- Đồ thị có hướng
- Đồ thị vô hướng



Có nhiều phương pháp giúp biểu diễn đồ thị khi lập trình, mỗi phương pháp đều có ưu và khuyết điểm riêng.

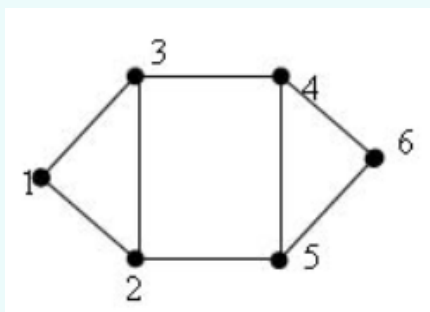
- Danh sách kề, nghĩa là: mỗi đỉnh A chứa danh sách các đỉnh có cạnh kề với đỉnh A
- Ma trận kề, nghĩa là: Ma trận 2 chiều với dòng là các đỉnh đích, cột là các đỉnh nguồn (hoặc ngược lại)
- Ma trận liên hợp, nghĩa là: ma trận bool 2 chiều với các dòng là các đỉnh và cột là các cạnh (hoặc ngược lại)

b. Danh sách kê

- Lý thuyết:

Đây là phương pháp thường được sử dụng, về lý thuyết thì trong cách biểu diễn này, với mỗi đỉnh v của đồ thị chúng ta lưu trữ danh sách các đỉnh kề với v . Trong trường hợp này, cấu trúc dữ liệu dùng theo dạng array sẽ không thuận tiện nên một số nguồn có thể sẽ đề xuất sử dụng danh sách liên kết (Linked List), vector,...

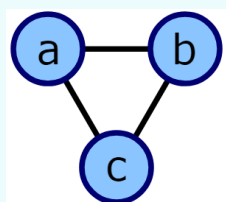
*** Ví dụ sơ về linked list (biểu diễn đồ thi)



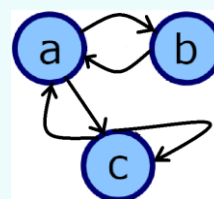
1		2		3		5	nil		
2		1		3		5	nil		
3		1		2		4	nil		
4		3		5		6	nil		
5		1		2		4		6	nil
6		4		5	nil				

- Tóm lại:

Danh sách kề bao gồm tập các danh sách các đỉnh, mỗi danh sách đỉnh chứa các đỉnh láng giềng



a: b, c
b: a, c
c: a, b



a: b, c
b: a
c: a, c

Khi lập trình, ta có thể dùng kiểu **vector<vector<string>>** để biểu diễn danh sách kề, ví dụ:

a: b, c b: a, c c: a, b	a: b, c b: a c: a, c
<pre>vector<vector<string>> dske; dske.push_back("a"); dske.push_back("b"); dske.push_back("c"); dske[0].push_back("b"); dske[0].push_back("c"); dske[1].push_back("a"); dske[1].push_back("c"); dske[2].push_back("a"); dske[2].push_back("b");</pre>	<pre>vector<vector<string>> dske; dske.push_back("a"); dske.push_back("b"); dske.push_back("c"); dske[0].push_back("b"); dske[0].push_back("c"); dske[1].push_back("a"); dske[2].push_back("a"); dske[2].push_back("c");</pre>

c. Ma trận kề

- Lý thuyết:

Xét đơn đồ thị vô hướng $G=(V,E)$, với tập đỉnh $V = \{1, 2, \dots, n\}$ và tập cạnh $E = \{e_1, e_2, \dots, e_m\}$. Ta gọi ma trận kề của đồ thị G có thể là ma trận

$A=\{a[i,j]$ trong đó $i,j= 1, 2, \dots, n\}$

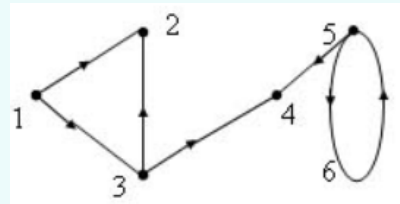
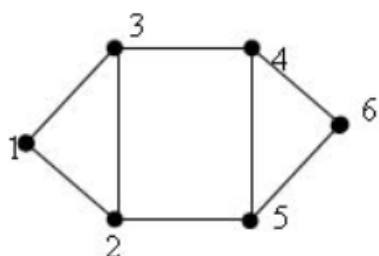
Trong đó các phần tử được xác định theo qui tắc sau đây:

$a[i,j]= 0$ nếu (i,j) không thuộc E

$a[i,j]= 1$ nếu (i,j) thuộc E

và $i,j= 1, 2, \dots, n$

Ví dụ 1: Đồ thị vô hướng và có hướng sau có thể được biểu diễn:

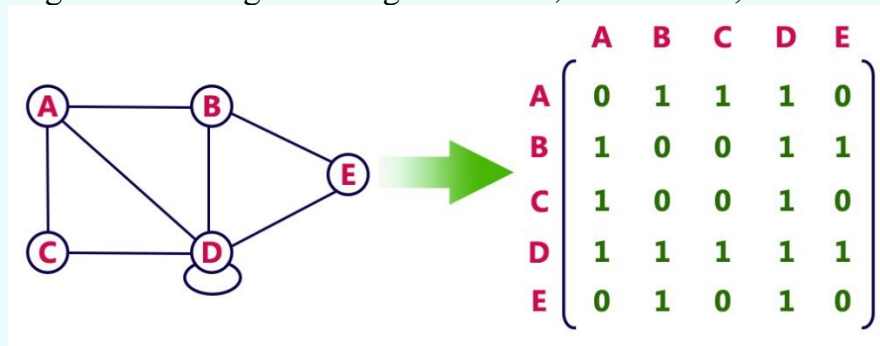


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	1	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	1	0	1	0	1
6	0	0	0	1	1	0

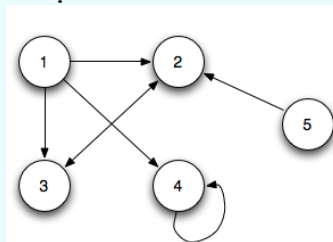
	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	0	0	0	0
3	0	1	0	1	0	0
4	0	0	0	0	0	0
5	0	0	0	1	0	1
6	0	0	0	0	1	0

- Ưu điểm của phương pháp biểu diễn đồ thị bằng ma trận kề (hoặc ma trận trọng số) là để trả lời câu hỏi “Hai đỉnh u, v có kề nhau trên đồ thị hay không”, chúng ta chỉ phải thực hiện một phép so sánh.
- Khuyết điểm của phương pháp này là không phụ thuộc vào số cạnh của đồ thị, ta luôn phải sử dụng n^2 đơn vị bộ nhớ để lưu trữ ma trận kề của nó.
- Tóm lại:

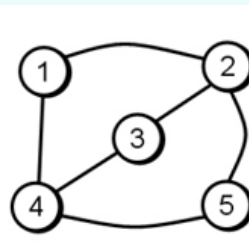
Ma trận kề là ma trận vuông với kích thước ma trận bằng với số đỉnh trong đồ thị, giá trị tại mỗi ô trong đồ thị có thể là độ dài cạnh nối đỉnh u và đỉnh v , số lượng đường đi nối đỉnh u và v , giá trị 0-1, trong đó 0 là không có đường nối u và v , còn 1 thì có,....



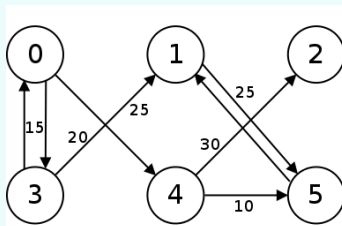
Ví dụ:



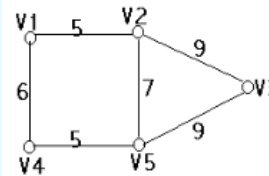
	1	2	3	4	5
1	0	1	1	1	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	0	0	1	0
5	0	1	0	0	0



	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0



	0	1	2	3	4	5
0				15	20	
1						25
2						
3	15	25				
4			30			10
5		25				



	V1	V2	V3	V4	V5
V1	0	5	0	6	0
V2	5	0	9	0	7
V3	0	9	0	0	9
V4	6	0	0	0	5
V5	0	7	9	5	0

Tham khảo thêm: <https://www.sciencedirect.com/topics/mathematics/adjacency-matrix>

d. Ma trận liên hợp / Danh sách cạnh

Giả sử đồ thị có n đỉnh và m cạnh thì ma trận liên hợp có kích thước $n \times m$, trong đó n dòng là đỉnh và m cột là cạnh.

Đồ thị có hướng: Mỗi ô có giá trị là -1, 0 hoặc 1, trong đó:

-1 Cạnh ở cột đi vào đỉnh ở dòng

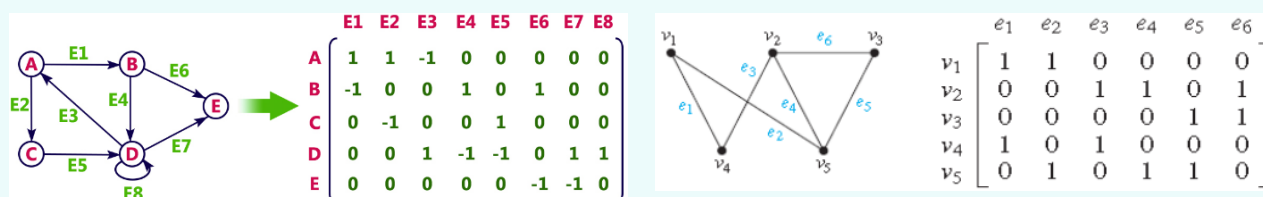
0 Cạnh không nối với đỉnh

1 Cạnh ở cột đi ra đỉnh ở dòng

Đồ thị vô hướng: mỗi ô có giá trị 0 hoặc 1, trong đó:

0 Cạnh không nối với đỉnh

1 Cạnh nối với đỉnh



Tham khảo thêm:

<https://voer.edu.vn/c/bieu-dien-do-thi-tren-may-vi-tinh/9c021e14/a2d8be8f>

(giải thích những thuật toán khác cho đồ thị: BFS, DFS) <https://viblo.asia/p/cau-truc-du-lieu-va-giai-thuat-cau-truc-du-lieu-do-thi-graph-djeZ1V6YIWz>

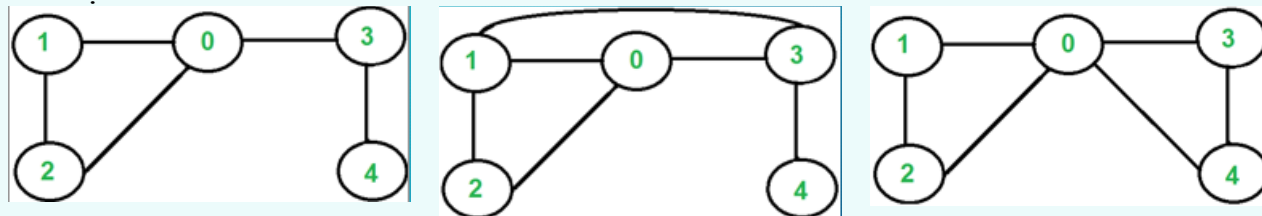
(Danh sách cạnh) <https://kienthuc24h.com/danh-sach-can-h-c-ly-thuyet-do-thi/>

10. Đường đi Euler – Chu trình Euler

Lý thuyết : <https://cuuduongthancong.com/d2h/giao-trinh-toan-roi-rac/giao-trinh-toan-roi-rac-ch4-do-thi-euler-va-do-thi-hamilton.html?src=detail>

Đặt vấn đề / bài toán : Cho một đồ thị bất kỳ, liệu :

- Có thể đi qua tất cả các cạnh của đồ thị, mỗi cạnh chỉ đi qua 1 lần ?
- Ứng với đồ thị, xuất phát từ 1 đỉnh, đi qua tất cả các cạnh, mỗi cạnh chỉ được đi qua 1 lần và về lại đỉnh đó ?

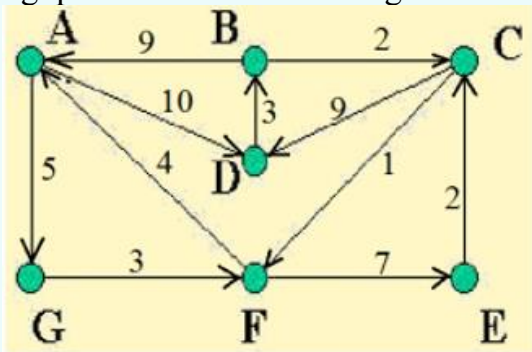


Có nhiều phương pháp được đề xuất, nhưng trong phần này sẽ đề xuất 2 phương pháp, bao gồm : ứng với Vấn đề 1 : nên sử dụng đường đi Euler và Vấn đề 2 thì nên sử dụng chu trình Euler

a. Đường đi Euler

Với bài toán này được mô tả : theo dạng cho đồ thị vô hướng $G = (V, E)$. Hãy xác định mọi đường đi qua tất cả các cạnh chỉ qua duy nhất 1 lần.

Ý tưởng bài toán này có thể sử dụng kỹ thuật tìm kiếm theo chiều sâu bằng cách xóa cạnh đã đi qua trong quá trình tìm kiếm đường đi



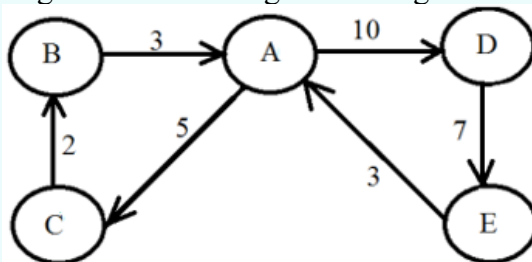
A: 2 cạnh vào, 2 cạnh ra
 B: 1 cạnh vào, 2 cạnh ra
 C: 2 cạnh vào, 2 cạnh ra
 D: 2 cạnh vào, 1 cạnh ra
 E: 1 cạnh vào, 1 cạnh ra
 F: 2 cạnh vào, 2 cạnh ra
 G: 1 cạnh vào, 1 cạnh ra
 → Có đường đi Euler

Ta có thể lưu sơ đồ theo bảng như sau :

→	A	B	C	D	E	F	G
A				1			1
B	1		1				
C				1		1	
D		1					
E			1				
F	1				1		
G						1	

b. Chu trình Euler

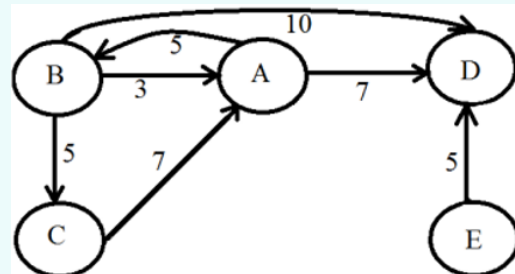
- Đây là thuật toán thường được sử dụng cho việc xây dựng các ứng dụng hỗ trợ tìm đường đi cho người đưa thư và người đi thu gom rác.



A: 2 cạnh vào, 2 cạnh ra
 B: 1 cạnh vào, 1 cạnh ra
 C: 1 cạnh vào, 1 cạnh ra
 D: 1 cạnh vào, 1 cạnh ra
 E: 1 cạnh vào, 1 cạnh ra
 → Có chu trình Euler

Ta có thể lưu sơ đồ theo bảng như sau :

→	A	B	C	D	E
A			1	1	
B	1				
C		1			
D					1
E	1				



- Các bước đề xuất thực hiện:

Bước 1: Kiểm tra đồ thị liên thông hay không. (ví dụ hàm `int lienthong()`)

Bước 2: Kiểm tra đồ thị có phải là đồ thị Euler hay không phải là đồ thị Euler (dựa vào bậc của mỗi đỉnh) (ví dụ hàm `int kiểmtraEuler()`).

Bước 3: Nếu là đồ thị Euler thì ta đi tìm chu trình Euler (ví dụ hàm `void Euler (dothi G)`)

Đối với bài toán tìm đường đi sẽ có thêm bước sử dụng định lý Goodman để chuyển đồ thị thành Euler

c. Tóm lại (Đường đi Euler – Chu trình Euler)

	Đường đi Euler	Chu trình Euler
Đồ thị vô hướng	<ul style="list-style-type: none">• Đồ thị phải liên thông, tức từ một đỉnh bất kỳ, ta phải tìm được đường đi đến các đỉnh còn lại.• Có 0 hoặc 2 đỉnh bậc lẻ.	<ul style="list-style-type: none">• Đồ thị phải liên thông, tức từ một đỉnh bất kỳ, ta phải tìm được đường đi đến các đỉnh còn lại.• Tất cả các đỉnh đều là đỉnh bậc chẵn.
Đồ thị có hướng	<ul style="list-style-type: none">• Đồ thị phải liên thông ở dạng vô hướng, tức xem đồ thị như đồ thị vô hướng, từ một đỉnh bất kỳ, ta phải tìm được đường đi đến các đỉnh còn lại.• Ở mỗi đỉnh, số cạnh vào phải bằng với số cạnh ra, hoặc• Có 1 đỉnh mà số cạnh vào – số cạnh ra = 1, có 1 đỉnh mà số cạnh ra – số cạnh vào = 1, các đỉnh còn lại số cạnh vào bằng số cạnh ra	<ul style="list-style-type: none">• Đồ thị phải liên thông, tức từ một đỉnh bất kỳ, ta phải tìm được đường đi đến các đỉnh còn lại.• Ở mỗi đỉnh, số cạnh vào phải bằng với số cạnh ra

d. Ví dụ về cài đặt chương trình (Đường đi Euler – Chu trình Euler)

(Bài toán tìm đường đi Euler của đồ thị)

<http://www.hoclaptrinh.xyz/2014/07/code-cc-tim-uong-i-euler-cua-o-thi-bai.html>

<http://ngoton.it/tim-chu-trinh-duong-di-euler/>

(Chu trình Euler và chu trình Hamilton)

<https://sites.google.com/site/ngo2uochung/courses/ltdt-tim-chu-trinh-euler-chu-trinh-hamilton>

<https://voer.edu.vn/c/do-thi-euler-va-do-thi-hamilton/9c021e14/86e257a7>

(Thuật toán về tìm đường đi và chu trình Euler)

<https://expressmagazine.net/development/4012/thuat-toan-ve-tim-duong-di-va-chu-trinh-euler-bang-cc>

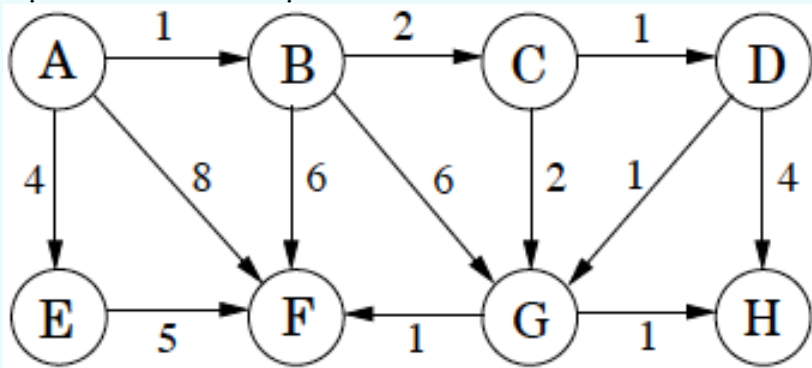
Tham khảo thêm:

(bài toán người đưa thư và đi xe rác)

http://lrc.quangbinhuni.edu.vn:8181/dspace/bitstream/DHQB_123456789/3042/1/B_I%20T_O_N%20NG_I%20_A%20TH_%20V_%20_NG%20D_NG%20T_M%20L_%20TR_NH

11. Thuật giải Disjkstra (bài toán tìm đường đi ngắn nhất)

Đặt vấn đề: Cho đồ thị:



Hỏi đường đi ngắn nhất từ A đến H?

Hình dung thuật giải như sau:

Bước 1:

- Ta có thể biểu diễn đồ thị dưới dạng ma trận kề có trọng số
- Tạo bảng, gán A = 0 và các đỉnh còn lại là một giá trị rất lớn (∞)
- A = 0 tức là con đường ngắn nhất từ A đến A có độ dài = 0

A	B	C	D	E	F	G	H
0	∞	∞	∞	∞	∞	∞	∞

Bước 2:

TÀI LIỆU THAM KHẢO

1. Microsoft Visual Studio Community 2015
2. <http://www.cplusplus.com/>
3. Trần Đan Thư, *Nhập môn lập trình*, NXB khoa học và kỹ thuật.
4. Trần Đan Thư, *Kỹ thuật lập trình*, NXB khoa học và kỹ thuật.
5. Phạm Thế Bảo, *Cấu trúc dữ liệu và giải thuật*, NXB ĐHQG Tp. Hồ Chí Minh.
6. <http://vietjack.com/> , <https://nguyenvanhieu.vn/>
7. Nguyễn Hữu Anh, *Toán rời rạc*, NXB Giáo Dục.
8. Trần Đan Thư – Dương Anh Đức, *Lý thuyết đồ thị*, NXB ĐHQG Tp. Hồ Chí Minh.
9. Science Direct <https://www.sciencedirect.com/>

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

- JACK MA