



BÀI TẬP THỰC HÀNH NHẬP MÔN LẬP TRÌNH

BIÊN SOẠN: HUỲNH THÀNH LỘC

MỤC LỤC

PHỤ LỤC 1 Một số phím tắt thường dùng trong Visual Studio	i
PHỤ LỤC 2 Xử lý một số lỗi biên dịch thường gặp	ii
PHỤ LỤC 3 Chạy chương trình ở chế độ dò và sửa lỗi	iv
BUỔI 01 Cấu trúc Tuần tự.....	1
BUỔI 02 Cấu trúc Lựa chọn	4
BUỔI 03 Cấu trúc Lặp.....	6
BUỔI 04 Bài tập tổng hợp 3 cấu trúc cơ bản	8
BUỔI 05 Hàm	10
BUỔI 06 Mảng một chiều	12
BUỔI 07 Mảng một chiều (tt)	14
BUỔI 08 Mảng 2 chiều	16
BUỔI 09 Kiểu dữ liệu Chuỗi và kỹ thuật đọc ghi file văn bản	25

PHỤ LỤC 1

MỘT SỐ PHÍM TẮT THƯỜNG DÙNG TRONG VISUAL STUDIO

Microsoft Visual Studio		DEFAULT KEYBOARD SHORTCUTS
SEARCH AND NAVIGATION		
Visual Studio search	Ctrl + Q	
Go to All	Ctrl + T or Ctrl + .	
Go to Type / File / Member / Symbol	Ctrl + T / F / M / S	
Navigate Backward / Forward	Ctrl + [/ Ctrl +]	
Go to Definition / Peek to Definition	F12 / Alt + F12	
Go to Implementation	Ctrl + F12	
Go to Next Error	Ctrl + Alt + F12	
Go to Next / Previous Result in List	F8 / Alt + F8	
EDITING AND REFACTORING		
Quick Actions / Refactoring Suggestions	Alt + J or Ctrl + .	
Method Info	Ctrl + K, Ctrl + I	
Comment / Uncomment	Ctrl + K, Ctrl + C / Ctrl + K, Ctrl + U	
Delete Line (without copying it)	Ctrl + Alt + L	
Paste from keyboard buffer ring	Ctrl + Alt + V	
Move Code Up / Down	Alt + Up / Down	
Format Document / Selection	Ctrl + K, Ctrl + D / Ctrl + K, Ctrl + F	
Surround with (... if/try/foreach)	Ctrl + K, Ctrl + S	
Rename	Ctrl + R, Ctrl + R	
Encapsulate Field	Ctrl + R, Ctrl + E	
Remove and Sort Usings	Ctrl + R, Ctrl + G	
Extract Method	Ctrl + R, Ctrl + M	
DEBUGGING AND TESTING		
Debug / Run / Stop	F5 / Ctrl + S / Alt + S	
Toggle Breakpoint	F9	
Step Over	F10	
Step Into	F11	
Step Out	Alt + F11	
Debug All Tests / Run All Tests	Ctrl + R, Ctrl + A / Ctrl + R, A	
WINDOW MANAGEMENT		
Select Active File in Solution Explorer	Ctrl + L, S	
Open Tool Windows		
Solution Explorer	Ctrl + Alt + L	
Output Window	Ctrl + Alt + O	
Error List	Ctrl + Alt + E	
Team Explorer	Ctrl + Alt + M	
Breakpoints	Ctrl + Alt + B	
Next / Previous Tool Window	Alt + F6 / Alt + F6	
Close Current Tool Window	Alt + Esc	
Go to Document to the Left / Right	Ctrl + Alt + Tab / Ctrl + Alt + Tab	
Most Recent / Least Recent Open Document	Ctrl + Tab / Ctrl + Alt + Tab	
Keep Preview Window Open	Ctrl + Alt + Tab	
Full Screen(max window size/reduced menus)	Alt + Alt + F11	
Configure keyboard shortcuts: Tools → Options; Environment → Keyboard		

<https://docs.microsoft.com/en-us/visualstudio/ide/default-keyboard-shortcuts-in-visual-studio>

Biên dịch chương trình (Building)

- **Ctrl + Shift + B:** Biên dịch chương trình nhưng không thực thi (*Build Solution*).

Chạy và dò lỗi chương trình (Debugging and Testing)

- **F5:** Biên dịch và chạy chương trình ở chế độ dò lỗi (*Start Debugging*).
- **Shift + F5:** Dừng quá trình chạy chế độ dò lỗi (*Stop Debugging*).
- **F9:** Đặt dấu ngắt để tạm dừng chương trình (*Toogle Breakpoint*).
- **F10:** Thực thi câu lệnh hiện tại và đi đến câu lệnh kế tiếp (*Step Over*).
- **Ctrl + F10:** Nhảy đến câu lệnh tại vị trí đang đặt con trỏ (*Run to Cursor*).
- **F11:** Nhảy vào bên trong chi tiết của một câu lệnh (*Step Into*).
- **Shift + F11:** Thoát ra bên ngoài phần chi tiết của một câu lệnh (*Step Out*).
- **Ctrl + F5:** Biên dịch và chạy chương trình ở chế độ không dò lỗi (*Start without Debugging*).

Soạn thảo mã nguồn chương trình (Editing and Refactoring)

- **Ctrl + Space:** Hiện thị gợi ý hoàn chỉnh câu lệnh từ Visual Studio (*IntelliSense*).
- **Ctrl + .** hoặc **Alt + Enter:** Chọn nhanh một số hành động (*Quick Actions*: 💡 🛠️ 🐞).
Ví dụ tự động thêm dòng khai báo sử dụng thư viện chứa phương thức hoặc một kiểu dữ liệu vừa sử dụng mà chưa khai báo trước đó.
- **Ctrl + G:** Di chuyển nhanh đến một dòng trong source code (*Go To*).
- **F12:** Di chuyển đến vị trí khai báo/định nghĩa Biến/Hàm (*Go To Definition*).
- **Ctrl + Alt + L:** Mở cửa sổ Solution Explorer (*View → Solution Explorer*).

PHỤ LỤC 2

XỬ LÝ MỘT SỐ LỖI BIÊN DỊCH THƯỜNG GẶP

Mở cửa sổ thông báo lỗi biên dịch: **Ctrl+** hoặc **Ctrl+E**: Để mở cửa sổ thông báo các lỗi biên dịch (**View** → **Error List**)

- **Errors**: trình biên dịch phát hiện lỗi trong source code → không thể biên dịch và thực thi chương trình
- **Warnings**: trình biên dịch cảnh báo một số vấn đề có thể chưa hợp lý trong source code (*khai báo biến nhưng trong chương trình không sử dụng đến, có một nhánh của câu lệnh lựa chọn sẽ không bao giờ được thực hiện,...*) → vẫn có thể được biên dịch và thực thi chương trình.

Một số lỗi biên dịch thường gặp:

- **“The name ‘x’ doesn’t exist in the current context”**: Sử dụng biến ‘x’ nhưng chưa được khai báo (hoặc được khai báo ở một nơi mà tại vị trí được báo lỗi không thể sử dụng được) → **Kiểm tra lại việc khai báo biến.**

```
static void Main(string[] args)
{
    int b = x + 7;
}
```

```
static void Main(string[] args)
{
    int x = 3;
    int b = x + 7;
}
```

- **“) expected”, “} expected”, or “; expected”**: Thiếu dấu ‘)’ , ‘}’, hoặc ‘;’ trong một câu trúc lệnh hoặc một khối lệnh → **Thông thường, chỉ cần thêm dấu ‘)’, ‘}’ hoặc ‘;’ tại vị trí được báo lỗi. Tuy nhiên, có trường trường hợp cần kiểm tra lại cú pháp sử dụng các câu lệnh, đóng mở các khối lệnh để sửa lỗi.**

```
namespace Example
{
    class Program
    {
        static void Main(string[] args)
        {
            for(int index = 0; index < 10; index++)
            {
                // missing curly brace here...
            }
        }
    }
} // error shows up here...
```

- **“Cannot convert type ‘x’ to ‘y’”**: Lỗi chuyển đổi từ kiểu dữ liệu ‘x’ sang kiểu dữ liệu ‘y’ → **Kiểm tra lại quy định về việc chuyển đổi từ kiểu dữ liệu ‘x’ sang kiểu dữ liệu ‘y’.**

```
static void Main(string[] args)
{
    double x = 1234.7;
    int a = x; // error show up here
}
```

```
static void Main(string[] args)
{
    double x = 1234.7;
    int a = (int)x; // The cast tells the compiler you know what's happening here.
}
```

- **“not all code paths return a value”**: hàm bắt buộc có dữ liệu trả về nhưng có khả năng thực hiện xong hàm vẫn không có giá trị nào được trả về (thường do câu lệnh return đặt trong câu lệnh rẽ nhánh không đầy đủ hoặc vòng lặp có khả năng không lặp lần nào cả,...)
→ Kiểm tra tính logic của source code, đảm bảo có lệnh return được thực hiện trong bất kỳ trường hợp nào.

```
public int DoSomething(int a)
{
    if (a < 10)
        return 0;
}
```

```
public int DoSomething(int a)
{
    if (a < 10)
        return 0;
}
```

- **“The type or namespace name ‘x’ could not be found”**: thiếu khai báo sử dụng thư viện
→ Sử dụng từ khóa “using” để khai báo thư viện cần thiết hoặc sử dụng Quick Actions như đã đề cập ở Phụ lục 1.

```
using System;

namespace Example
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> myList = new List<int>(); //Error here
        }
    }
}
```

```
using System;
using System.Collections.Generic; //Add the appropriate namespace here

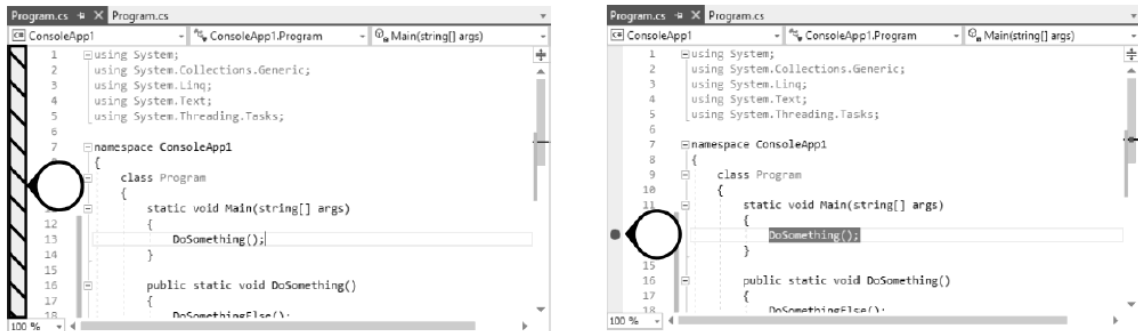
namespace Example
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> myList = new List<int>();
        }
    }
}
```

PHỤ LỤC 3

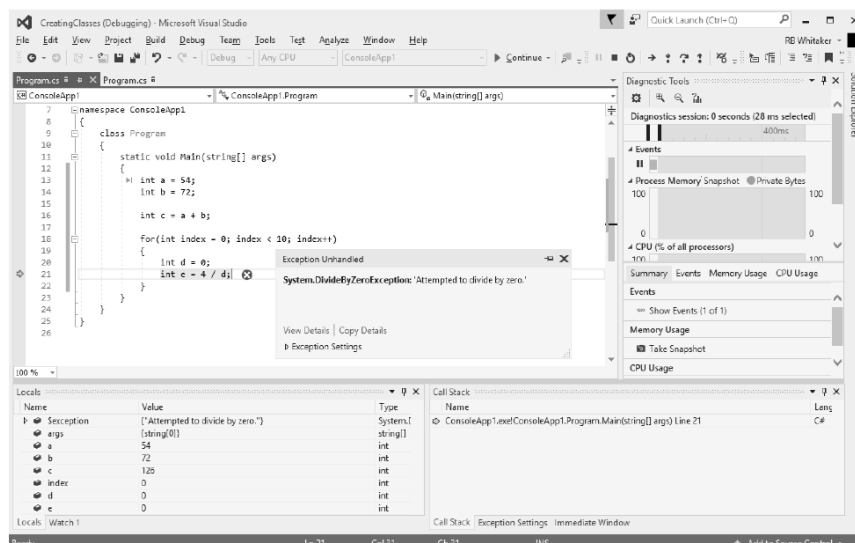
CHẠY CHƯƠNG TRÌNH Ở CHẾ ĐỘ DÒ VÀ SỬA LỖI

Chạy chương trình ở chế độ Debug:

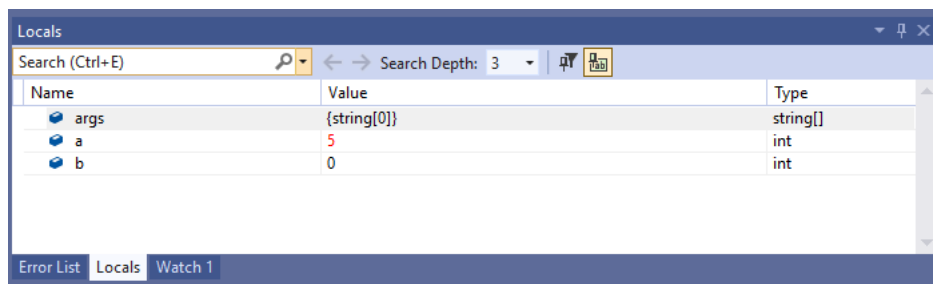
- **Đặt dấu ngắt (Breakpoint)** tạm dừng việc thực thi chương trình tại một vị trí cần kiểm tra trong source code: Nhấn **F9** (Chọn menu **Debug** → **Toggle Breakpoint**) hoặc nhấn chuột vào thanh dọc màu xám phía bên trái cửa sổ soạn thảo.



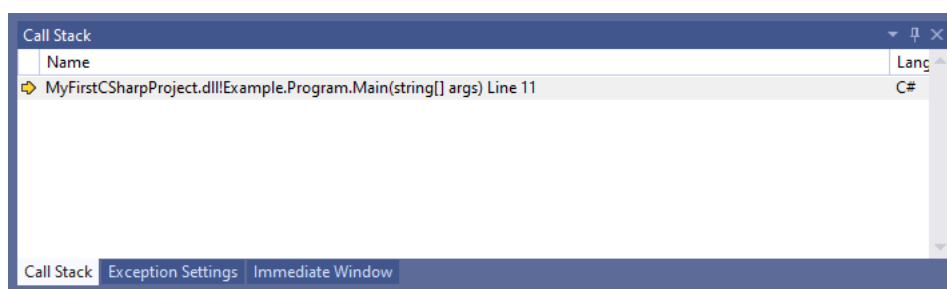
- **Bắt đầu chạy Debug:** Nhấn **F5** (Chọn menu **Debug** → **Start Debugging**), chương trình sẽ chạy và dừng ở vị trí dấu Breakpoint đầu tiên.
- **Tiếp tục thực hiện chạy Debug những câu lệnh tiếp theo:**
 - **Continue:** Nhấn **F5** (Chọn menu **Debug** → **Continue**) → chạy đến vị trí Breakpoint tiếp theo.
 - **Step Over:** Nhấn **F10** (Chọn menu **Debug** → **Step Over**) → Thực thi câu lệnh hiện tại và đi đến câu lệnh kế tiếp.
 - **Step Into:** Nhấn **F11** (Chọn menu **Debug** → **Step Into**) → Nhảy vào bên trong chi tiết của một câu lệnh (**Step Into**).
 - **Step Out:** Nhấn **Shift + F5** (Chọn menu **Debug** → **Step Out**) → Thoát ra bên ngoài phần chi tiết của một câu lệnh (**Step Out**).
 - Hoặc có thể mở thanh công cụ Debug và chọn các nút tương ứng: **View** → **Toolbars** → **Debug**
- **Xem các lỗi ngoại lệ (Exception):**



- **Xem sự thay đổi giá trị của biến qua từng bước thực thi chương trình:** Chọn menu *Debug* → *Windows* → *Locals*



- **Xem thứ tự thực thi các hàm trong chương trình:** Chọn menu *Debug* → *Windows* → *Call Stack*



- **Dừng quá trình chạy Debug:** Nhấn **Shift + F5** (Chọn menu *Debug* → *Stop Debugging*)

BUỔI 01

CẤU TRÚC TUẦN TỰ

MỤC TIÊU

- Tạo và hiểu cấu trúc **C# Console App** trên **Visual Studio**.
- Nắm vững cú pháp **khai báo biến** và nhớ các **kiểu dữ liệu cơ sở** trong C#.
- Viết được các chương trình C# đơn giản với các **câu lệnh Tuần tự**: *Khai báo biến, nhập và xuất dữ liệu, thực hiện các phép toán cơ bản trên các kiểu dữ liệu cơ sở trong C#.*

Lời nhắn:

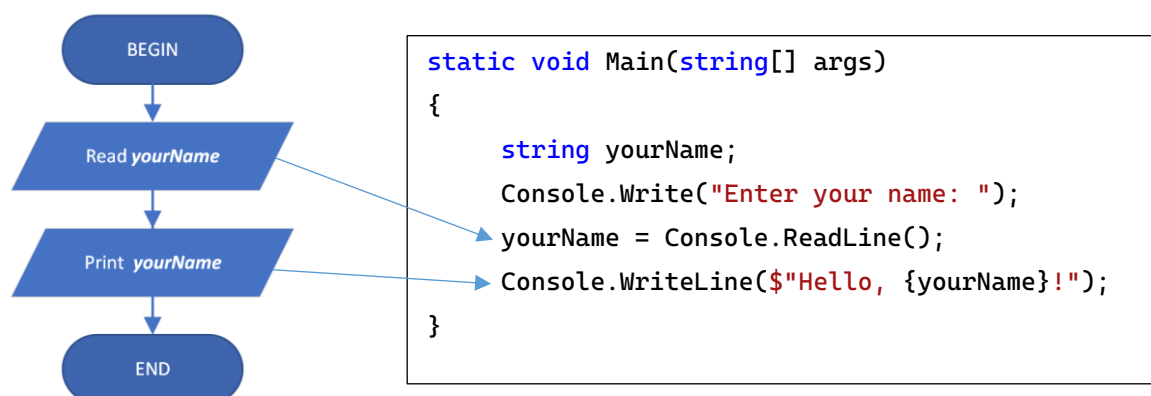
- Trong buổi thực hành đầu tiên, sinh viên bắt đầu làm quen với việc viết các C# Console App đơn giản. Với mỗi bài tập, sinh viên **không những cần cài đặt thuật toán để tìm ra lời giải đúng** cho bài toán **mà còn cần thực hiện định dạng dòng dữ liệu nhập/xuất sao cho giống với mô tả INPUT và OUTPUT**.
- Kể từ buổi thực hành thứ 2, **nếu bài toán không có mô tả cụ thể về định dạng dòng dữ liệu nhập/xuất thì sinh viên có thể tự quy định định dạng dòng dữ liệu nhập/xuất sao cho phù hợp với yêu cầu bài toán** (tạo các bộ dữ liệu mẫu bao gồm INPUT và OUTPUT tương ứng để chạy và kiểm thử chương trình).
- Để hạn chế sai sót khi viết thuật toán, **khuyến khích sinh viên thiết kế và mô tả thuật toán bằng sơ đồ khối trước khi cài đặt** thực tế trên máy tính.

Bài tập 1.

Viết chương trình yêu cầu nhập người dùng nhập vào một chuỗi họ tên (yourName). Sau đó in ra màn hình dòng chữ với nội dung: **"Hello, yourName!"**

INPUT	OUTPUT
Enter your name: <i>Loc Huynh</i>	Hello, Loc Huynh!

Ghi chú: Phần nội dung in nghiêng, màu cam *Loc Huynh* là do người dùng nhập vào. Tương tự, cho tất cả các bài tập sau này!



Bài tập 2.

Viết chương trình nhập vào một “chuỗi số nguyên” (stringNumber) sau đó chuyển chuỗi này về kiểu số nguyên (intNumber). In ra màn hình giá trị và kiểu dữ liệu của 2 biến stringNumber và intNumber. (Hướng dẫn: sử dụng hàm `a.GetType()` để lấy kiểu dữ liệu của biến `a`)

INPUT	OUTPUT
Input string: 123	“stringNumber” value & type: 123, System.String “intNumber” value & type: 123, System.Int32

Bài tập 3.

Viết chương trình yêu cầu người dùng nhập vào 2 số nguyên `a` và `b`. Tính và in ra màn hình kết quả các phép tính: `a + b`, `a - b`, `a * b`, `a / b`, `a % b`.

INPUT	OUTPUT
<code>a = 9</code> <code>b = 4</code>	<code>9 + 4 = 13</code> <code>9 - 4 = 5</code> <code>9 * 4 = 36</code> <code>9 / 4 = 2</code> <code>9 % 4 = 1</code>

Bài tập 4.

Viết chương trình yêu cầu người dùng nhập vào 2 số thực `a` và `b`. Thực hiện hoán đổi giá trị của `a` và `b`. In ra màn hình giá trị của `a`, `b` trước và sau khi hoán đổi.

INPUT	OUTPUT
<code>a = 8.53</code> <code>b = 4</code>	Before swapping: <code>a = 8.53, b = 4</code> After swapping: <code>a = 4, b = 8.53</code>

Bài tập 5.

Viết chương trình yêu cầu người dùng nhập 1 số thực `r` là bán kính của một hình tròn. Tính và in ra màn hình chu vi và diện tích của hình tròn đó (kết quả được làm tròn đến 2 chữ số thập phân). Biết rằng:

$$\text{Chu vi: } P = 2\pi r$$

$$\text{Diện tích: } A = \pi r^2$$

INPUT	OUTPUT
<code>r = 3</code>	<code>P = 18.85</code> <code>A = 28.27</code>

Hướng dẫn: sử dụng `Math.PI` để lấy giá trị số PI

Bài tập 6.

Viết chương trình yêu cầu người dùng nhập 1 số nguyên `n` ($1 \leq n \leq 10$). In ra màn hình bảng nhân của `n` (Yêu cầu: các số hạng và tích được in với độ rộng là 2, canh lề phải).

INPUT	OUTPUT
n = 7	7 x 1 = 7 7 x 2 = 14 7 x 3 = 21 7 x 4 = 28 7 x 5 = 35 7 x 6 = 42 7 x 7 = 49 7 x 8 = 56 7 x 9 = 63 7 x 10 = 70

Bài tập 7.

Viết chương trình nhập vào một số thực là số giờ làm việc của nhân viên và một số nguyên là số tiền công của mỗi giờ làm việc. **Tính và in ra màn hình lương tháng của nhân viên.**

INPUT	OUTPUT
Working hours = 7.5 Unit price = 15	Your Salary: 112.5

Bài tập 8.

Có một số tiền N và các tờ tiền với mệnh giá 100, 50, 20, 10, 5, 2, 1. Hãy **đổi N thành các tờ tiền sao cho tổng số tờ giấy bạc cần dùng là ít nhất.**

INPUT	OUTPUT
Amount of money: 266	100 : 2 50 : 1 20 : 0 10 : 1 5 : 1 2 : 0 1 : 1

Bài tập 9.

Cho một số nguyên là khoảng thời gian diễn ra của một sự kiện (tính bằng giây). Hãy **biểu diễn thời gian đó dưới dạng “hh : mm : ss”** (“giờ : phút : giây”).

INPUT	OUTPUT
Input time (s): 566	00 : 09 : 26

Bài tập 10.

Cho 3 số thực lần lượt là điểm quá trình, điểm thi thực hành và điểm thi cuối kỳ môn Nhập môn lập trình của một sinh viên có trọng số lần lượt là 20%, 30% và 50% (điểm có giá trị từ 0 đến 10). Hãy **tính điểm trung bình môn học (làm tròn đến 1 chữ số thập phân)** của sinh viên.

INPUT	OUTPUT
Attendant mark: 8.5 Practical mark: 7.5 Final exam: 9.0	Average mark: 8.5

BUỔI 02

CẤU TRÚC LỰA CHỌN

MỤC TIÊU

- Hiểu và vận dụng được **Cấu trúc lựa chọn** để thiết kế thuật toán cho chương trình.
- Biết cách sử dụng các dạng **câu lệnh lựa chọn** trong C#: “if...”, “if...else...”, “if...else if...”, “switch...case...”.
- Biết các sử dụng các **phép toán quan hệ**: ==, !=, >, <, >=, <= và các phép toán kết hợp: && (and), || (or) để đặt điều kiện cho câu lệnh chọn lựa.

Bài tập 1.

Viết chương trình nhập vào một số nguyên n. Cho biết n là **số âm hay số dương** và n là **số chẵn hay số lẻ**.

Bài tập 2.

Cho 3 số thực a, b, c ($a \neq 0$). **Giải phương trình bậc hai** $ax^2 + bx + c = 0$.

Bài tập 3.

Cho 3 số thực a, b, c. Hãy **tìm giá trị lớn nhất của 3 số** đó.

Bài tập 4.

Cho 3 số thực a, b, c. Hãy **hoán đổi giá trị của 3 số** sao cho giá trị của a, b, c có thứ tự **không giảm** (sắp xếp tăng dần).

Bài tập 5.

Nhập 3 số thực a, b, c. Hãy **kiểm tra 3 số vừa nhập có thể tạo thành một tam giác** với độ dài 3 cạnh lần lượt là a, b, c hay không?

- Nếu là tam giác thì tính chu vi của tam giác này.
- Nếu không là tam giác thì tính diện tích hình thang có 2 đáy là a, b; chiều cao là c.

Bài tập 6.

Nhập vào một số nguyên thuộc đoạn [1,12]. Hãy **in ra tên tháng (bằng tiếng Anh)** tương ứng với số đó.

Ví dụ: Nhập số **4**, tháng tương ứng là **April**.

Bài tập 7.

Viết chương trình nhập vào 2 số nguyên a, b và một phép toán op (+, -, *, / , %). Hãy tính và in ra màn hình **kết quả của phép toán tương ứng**.

Bài tập 8.

Cho số thực x ($0 \leq x \leq 100$). Hãy cho biết x **thuộc đoạn nào** trong các đoạn dưới đây:

[0,25]; (25,50]; (50,75]; (75,100]



Bài tập 9.

Cho bảng thông tin sản phẩm như sau:

CODE	PRODUCT NAME	PRICE
1	Cochorro Quente	R\$ 4.00
2	X-Salada	R\$ 4.50
3	X-Bacon	R\$ 5.00
4	Torrada simples	R\$ 2.00
5	Refrigerante	R\$ 1.00

Viết chương trình nhập vào **mã sản phẩm (code)** và **số lượng (amount)** cần mua. Hãy **tính và in ra màn hình thông tin thanh toán** như ví dụ bên dưới:

Ví dụ: Nhập code = 3, quantity = 2. Thông tin sẽ in ra màn hình như sau:

PRODUCT NAME	PRICE	QUANTITY	TOTAL (R\$)
X-Bacon	5.00	3	15.00

Hướng dẫn: Cột **PRODUCT NAME** có độ rộng 20 ký tự và canh lề trái; các cột **PRICE**, **QUANTITY**, **TOTAL** có độ rộng 12 ký tự và canh lề phải. Trước và sau mỗi ký tự ‘|’ đều có ký tự khoảng trắng (“PRODUCT NAME | PRICE | AMOUNT | TOTAL”)

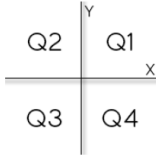
Bài tập 10. Công ty ABC quyết định tăng lương cho nhân viên theo bảng sau:

Salary	Increase Percentage
0 – 400.00	15%
400.01 – 800.00	12%
800.01 – 1200.00	10%
1200.01 – 2000.00	7%
>2000	4%

Viết chương trình nhập vào lương hiện tại của một nhân viên. **Tính lương mới cho nhân viên đó.** **Ví dụ:** Lương hiện tại của nhân viên là 400, thì lương mới là 460.

Bài tập 11.

Viết thuật toán **nhập vào 2 số thực x, y** là tọa độ của một điểm trong mặt phẳng. Hãy **cho biết điểm nằm ở vị trí nào?** (gốc tọa độ, trục Ox, trục Oy hay ở góc phần tư thứ mấy?) **Ví dụ:** x = 4.5, y = -2.2 thì điểm đó thuộc Q4.



BUỔI 03

CẤU TRÚC LẶP

MỤC TIÊU

- Hiểu và vận dụng được **Cấu trúc lặp** để thiết kế thuật toán cho chương trình.
- Biết cách sử dụng các dạng **câu lệnh lặp** trong C#: “while...”, “do...while”, “for...”.
- Biết các sử dụng câu lệnh **break** để thoát khỏi vòng lặp, câu lệnh **continue** để nhảy sang lần lặp tiếp theo.

Bài tập 1.

Viết chương trình yêu cầu người dùng nhập số nguyên dương n . In ra các số nguyên dương từ 1 và đến n (các số cách nhau bởi khoảng trắng).

Ví dụ: $n = 5 \rightarrow$ In ra màn hình 1 2 3 4 5

Bài tập 2.

Viết chương trình yêu cầu người dùng nhập số nguyên dương n . In ra các số nguyên lẻ từ 1 và đến n (các số cách nhau bởi khoảng trắng).

Ví dụ: $n = 10 \rightarrow$ In ra màn hình 1 3 5 9

Bài tập 3.

Viết chương trình yêu cầu người dùng nhập một số nguyên dương n ($n > 1$). Nếu người dùng nhập $n < 1$, yêu cầu nhập lại cho đến khi $n > 1$. In ra các số nguyên chẵn từ 1 và đến n (các số cách nhau bởi khoảng trắng).

Ví dụ: $n = 10 \rightarrow$ In ra màn hình 2 4 6 8 10

Bài tập 4.

Viết chương trình yêu cầu người dùng nhập số nguyên n . In ra tổng và trung bình cộng của các số nguyên từ 1 và đến n .

Ví dụ: $n = 10 \rightarrow$ Sum = 55, Avg = 5.5

Bài tập 5.

Viết chương trình nhập vào số nguyên n . Hãy in ra các ước số của n .

Bài tập 6.

Viết chương trình nhập vào số nguyên n . Hãy cho biết n có phải là số nguyên tố hay không?

Bài tập 7.

Viết chương trình in ra tổng lớn nhất của các số nguyên dương liên tiếp bắt đầu từ 1, sao cho tổng này nhỏ hơn 200.000.

Bài tập 8. Viết chương trình yêu cầu người dùng nhập số nguyên dương n . In ra tổng lớn nhất của các số nguyên dương liên tiếp bắt đầu từ 1, sao cho tổng này nhỏ hơn n .

Bài tập 9.

Viết chương trình yêu cầu người dùng nhập vào một số nguyên, sao cho số này phải **nằm trong khoảng từ 100 đến 500 và chia hết cho 2**, nếu không yêu cầu người dùng nhập lại.

Bài tập 10.

Viết chương trình nhập vào một số nguyên n . Tính $n!$ (nếu nhập $n < 0$ thì in kết quả là 0).

Bài tập 11.

Viết chương trình nhập vào số nguyên n . Tính và in ra màn hình tổng sau:

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

Bài tập 12.

Viết chương trình tính và in ra tổng sau:

$$S = 1 + \frac{3}{2} + \frac{5}{4} + \frac{7}{8} + \dots + \frac{39}{?}$$

Bài tập 13.

Viết chương trình yêu cầu người dùng nhập 1 số nguyên n ($1 \leq n \leq 10$). In ra màn hình bảng nhân của các số từ 1 đến n .

Ví dụ: $n = 3$

1 x 1 = 1		2 x 1 = 2		3 x 1 = 3
1 x 2 = 2		2 x 2 = 4		3 x 2 = 6
...				
1 x 10 = 10		2 x 10 = 20		3 x 10 = 30

Bài tập 14.

Nhân dịp năm mới, hãng nước ngọt K có chương trình khuyến mãi cho khách hàng. Nội dung của chương trình là nếu khách hàng đem đến cửa hàng 10 vỏ chai thì có thể đổi lấy 03 chai mới. Hỏi **nếu ban đầu anh A mua n chai thì tổng cộng anh có thể uống được tất cả bao nhiêu chai?**

Input: Số nguyên n ($1 \leq n \leq 1.000$) là số chai anh A mua ban đầu.

Output: Số nguyên m là tổng số chai mà anh A có thể uống.

Ví dụ: $n = 10 \rightarrow m = 13$;

$n = 24 \rightarrow m = 33$

Chú thích:

- Ban đầu có **24** chai;
- Lấy 20 vỏ chai ban đầu đổi được **thêm 6** chai mới;
- 6 chai mới + 4 chai ban đầu đổi được **thêm 3** chai mới;
- Tổng cộng uống được 33 chai.

BUỔI 04

BÀI TẬP TỔNG HỢP 3 CẤU TRÚC CƠ BẢN

MỤC TIÊU

- Hiểu rõ và vận dụng được **3 cấu trúc cơ bản**: **Tuần tự**, **Lựa chọn** và **Lặp** để giải các bài tập liên quan.

Bài tập 1.

Viết chương trình nhập vào 2 số nguyên x và y. Hãy **tính tổng các số lẻ** giữa chúng.

Ví dụ: x = 6, y = -5 thì tổng là 5

Bài tập 2.

Viết chương trình nhập số nguyên n. Hãy in ra **bình phương các số chẵn từ 1 đến n**.

Ví dụ: n = 6 → In ra màn hình 4 16 36

Bài tập 3.

Trong toán học, số n được gọi là **số hoàn hảo** nếu n bằng tổng các ước số của n (không kể số n). Viết chương trình nhập vào số nguyên dương n, **kiểm tra n có phải là số hoàn hảo hay không?**

Ví dụ: n = 6 là số hoàn hảo vì 6 có các ước số 1, 2, 3 và tổng các ước này $1 + 2 + 3 = 6$.

Bài tập 4.

Viết chương trình nhập vào một số nguyên n. In ra màn hình các hình tam giác có cạnh n như các mẫu dưới đây. **Ví dụ:** n = 4

Mẫu 1	Mẫu 2	Mẫu 3	Mẫu 4
* * * * * * * * * *	* * * * * * * * * *	* * * * * * * * * *	* * * * * * * * *
Mẫu 5	Mẫu 6	Mẫu 7	Mẫu 8
1 1 2 1 2 3 1 2 3 4	1 2 3 4 5 6 7 8 9 10	1 1 2 1 2 3 1 2 3 4	1 1 2 1 3 1 2 3 4

Bài tập 5. Cho bảng thông tin sản phẩm như sau:

CODE	PRODUCT NAME	PRICE
1	Cochorro Quente	R\$ 4.00
2	X-Salada	R\$ 4.50
3	X-Bacon	R\$ 5.00
4	Torrada simples	R\$ 2.00
5	Refrigerante	R\$ 1.00

Viết chương trình tính tiền cho một cửa hàng bán lẻ các sản phẩm trên với các chức năng sau đây:

1. Nhập số loại sản phẩm sẽ mua.
2. Lần lượt nhập mã (code) và số lượng cần mua cho mỗi loại sản phẩm.
3. Thực hiện tính và thông tin thanh toán ra màn hình
4. Nhập một lựa chọn (1: xóa nội dung trên màn hình và tiếp tục tính tiền cho một khách hàng khác, 0: thoát chương trình)

Ví dụ:

The number of products: 3

- Code of Product 1: 1

Quantity: 3

- Code of Product 2: 2

Quantity: 4

- Code of Product 3: 5

Quantity: 7

INVOICE

No	PRODUCT NAME	PRICE	QUANTITY	TOTAL (R\$)
1	Cochorro Quente	4.00	3	12.00
2	X-Salada	4.50	4	18.00
3	Refrigerante	1.00	7	7.00

TOTAL:

37.00

Enter your choice (1 - continue / 0 - exit): 0

Thank you & Good bye!

BUỔI 05

HÀM

MỤC TIÊU

- Nắm vững cú pháp **định nghĩa Hàm**.
- Phân biệt được 2 dạng tham số của Hàm: **tham trị** và **tham chiếu (ref và out)**.
- Có khả năng vận dụng để phân tích và chia nhỏ chương trình thành các hàm chức năng.

Bài tập 1.

Định nghĩa **các hàm** thực hiện chức năng tính toán sau đây:

- **Cộng** 2 số nguyên: $a + b$.
- **Trừ** 2 số nguyên: $a - b$
- **Nhân** 2 số nguyên: $a \times b$
- **Chia** 2 số nguyên: a / b (kết quả phép chia thực. Ví dụ: $5 / 2 = 2.5$)

Bài tập 2.

Định nghĩa hàm **hoán đổi giá trị** 2 của số nguyên a và b (kết quả hoán đổi phải được bảo lưu sau khi kết thúc hàm).

Bài tập 3.

Định nghĩa **một hàm duy nhất** thực hiện tính và trả về kết quả các phép toán Cộng, Trừ, Nhân, Chia (kết quả phép chia thực) của 2 số nguyên a và b .

Bài tập 4.

Định nghĩa hàm tìm giá trị nhỏ nhất trong 3 số nguyên a, b, c .

Bài tập 5.

Định nghĩa hàm tìm giá trị lớn nhất trong 3 số nguyên a, b, c .

Bài tập 6.

Định nghĩa **một hàm duy nhất** tìm giá trị nhỏ nhất và lớn nhất trong 3 số nguyên a, b, c .

Bài tập 7.

Định nghĩa hàm tính tổng các số nguyên từ 1 đến n .

Bài tập 8.

Định nghĩa hàm tính tổng các chữ số của một số nguyên dương n .

Ví dụ: $n = 168 \rightarrow S = 1 + 6 + 8 = 15$

Bài tập 9.

Định nghĩa hàm xác định số lớn nhất trong các chữ số của n .

Ví dụ: $n = 31685 \rightarrow \text{max_digit} = 8$

Bài tập 10.

Định nghĩa hàm **kiểm tra một số nguyên n có phải là số nguyên tố** hay không?

Bài tập 11.

Định nghĩa **hàm tính giá trị số Fibonacci thứ n (F_n)**. Biết rằng, Fibonacci là dãy số có dạng:

$$F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 5, \dots, F_n = F_{n-1} + F_{n-2}$$

Bài tập 12.

Định nghĩa **hàm tính giá trị a^n** (với a là một số thực, n là một số nguyên).

Bài tập 13.

Tổ chức lại chương trình ở Bài 5 - Buổi 4 thành các hàm sao cho hàm Main() có thể thực thi bằng cách gọi các hàm như sau:

```
static void Main()
{
    //Khai báo các biến
    ...
    int choice = 1;

    while (choice == 1)
    {
        //Thiết lập giá trị các biến về trạng thái ban đầu
        ResetAllVariables(...);

        //Nhập số Loại sản phẩm
        Console.Write("The number of products: ");
        n = int.Parse(Console.ReadLine());

        //Nhập mã code và số Lượng các sản phẩm đã mua
        InputPurchasedProducts(...);

        //Tính và in hóa đơn
        PrintInvoice(...);

        //Nhập Lựa chọn
        ...
    }
    Console.Write("Thank you & Good bye!");
}
```

BUỔI 06


MẢNG MỘT CHIỀU

MỤC TIÊU

- Biết cách **khai báo** và **khởi tạo mảng một chiều**.
- Thực hiện được các thao tác xử lý mảng 1 chiều: **nhập và xuất mảng, duyệt và tính toán trên mảng**.
- Vận dụng mảng một chiều để giải quyết các bài toán.

Bài tập 1. Nhập mảng 1 chiều

Định nghĩa các hàm sau đây:

- `static void` `NhapMang1(int[] a)`: Hàm có tham số là mảng 1 chiều chứa các số nguyên `a` đã được khởi tạo, yêu cầu người dùng nhập giá trị cho các phần tử của mảng `a`.
 - `static void` `NhapMang2(out int[] a)`: Hàm có tham số là mảng 1 chiều chứa các số nguyên `a` **không cần phải khởi tạo trước**, yêu cầu người dùng nhập kích thước, khởi tạo mảng `a` với kích thước vừa nhập, sau đó nhập giá trị cho các phần tử của `a`.
 - `static int[]` `NhapMang3()`: khởi tạo một mảng 1 chiều chứa các số nguyên `a` bên trong hàm, yêu cầu người dùng nhập kích thước, khởi tạo mảng `a` với kích thước vừa nhập, sau đó nhập giá trị cho các phần tử của `a`. Hàm có kết quả trả về là mảng `a` sau khi đã nhập xong.
 - Nhận xét sự khác nhau của 3 hàm trên.
-  Thông qua bài tập này, sinh viên cần hiểu và phân biệt được sự khác nhau của 3 hàm nhập mảng. Sau này, tùy vào yêu cầu cụ thể của từng bài toán, sinh viên có thể lựa chọn cách nhập mảng phù hợp chứ không nhất thiết phải cài đặt đầy đủ cả 3 hàm nhập mảng.

Bài tập 2. Xuất mảng 2 chiều

Định nghĩa hàm có tham số là một mảng 1 chiều chứa các số nguyên `a`. In các phần tử của mảng `a` ra màn hình, mỗi phần tử cách nhau bởi một ký tự khoảng trắng, sau khi in xong phần tử cuối cùng thì xuống hàng.

Bài tập 3. Tổng mảng

Định nghĩa hàm **tính và trả về tổng giá trị các phần tử** của một mảng số nguyên.

Bài tập 4. Tìm kiếm x

Định nghĩa hàm **tìm và trả về vị trí của phần tử có giá trị bằng x** trong mảng số nguyên.

- Nếu có nhiều phần tử có giá trị bằng `x` thì trả về vị trí đầu tiên tính từ đầu mảng;
- Nếu không có phần tử nào có giá trị bằng `x` thì trả về `-1`.

Bài tập 5. Tìm kiếm max

Định nghĩa hàm **tìm và trả về vị trí phần tử có giá trị lớn nhất** trong một mảng một chiều chứa các số thực. Nếu có nhiều phần tử có giá trị bằng giá trị lớn nhất thì trả về vị trí cuối cùng trong mảng.

Bài tập 6. Đếm số nguyên dương

Định nghĩa hàm **đếm và trả về số lượng phần tử có giá trị lớn hơn 0** trong một mảng một chiều chứa các số nguyên.

Bài tập 7. Bài toán Sắp xếp

Cho n số nguyên $a = (a_1, a_2, \dots, a_n)$ ($1 \leq n \leq 5000$). Hãy cài đặt các hàm sắp xếp dãy a tăng dần theo thuật toán **Bubble Sort** và **Interchange Sort**

Input

- Dòng đầu tiên chứa số nguyên n
- Dòng thứ hai chứa n số nguyên

Output

- Dòng duy nhất các số nguyên đã sắp xếp

Ví dụ

INPUT	OUTPUT
8 6 5 3 7 8 2 12 14	2 3 5 6 7 8 12 14

Bài tập 8.

Hiệu chỉnh 2 thuật toán ở Bài 1 để thực hiện việc **sắp xếp các phần tử trong đoạn $[k, l]$** ($0 \leq k \leq l < n$) **theo thứ tự tăng dần**.

BUỔI 07

MẢNG MỘT CHIỀU (TT)

MỤC TIÊU

- Vận dụng mảng một chiều để giải quyết các bài toán.

Bài tập 1. Đếm số lượng mỗi số dương

Cho n số nguyên dương $a = (a_1, a_2, \dots, a_n)$ ($1 \leq n \leq 10^8$ và $0 \leq a_i \leq 10^6$). Hãy cho biết mỗi số nguyên trong dãy a xuất hiện bao nhiêu lần?

Ví dụ: $a = (2, 2, 5, 5, 2, 3, 5, 4, 2, 4)$

- Số 2: xuất hiện 4 lần
- Số 3: xuất hiện 1 lần
- Số 4: xuất hiện 2 lần
- Số 5: xuất hiện 3 lần

Input

- Dòng đầu tiên chứa số nguyên n
- Dòng thứ hai chứa n số nguyên

Output

- Mỗi dòng xuất ra theo định dạng $i:k$ với ý nghĩa là số i xuất hiện k lần (các số i được xuất theo thứ tự từ nhỏ đến lớn)

Ví dụ

INPUT	OUTPUT
10 2 2 5 5 2 3 5 4 2 4	2:4 3:1 4:2 5:3

Hướng dẫn: Dùng kỹ thuật “mảng đếm”

Bài tập 2. Số nhỏ nhất

Cho n số nguyên dương $a = (a_1, a_2, \dots, a_n)$ ($1 \leq n \leq 10^8$ và $1 \leq a_i \leq 10^6$). Hãy tìm số nguyên dương nhỏ nhất không xuất hiện trong a .

Input

- Dòng đầu tiên chứa số nguyên n
- Dòng thứ hai chứa n số tự nhiên

Output

- Số tự nhiên nhỏ nhất không xuất hiện trong a

Ví dụ

INPUT	OUTPUT
8 6 7 1 2 5 3 2 6	4

Hướng dẫn: Dùng kỹ thuật “mảng đánh dấu trạng thái”

Bài tập 3. Sàng nguyên tố

Cho số nguyên n ($1 \leq n \leq 10^6$). Viết chương trình liệt kê các số nguyên tố nhỏ hơn hay bằng n .

Input

- Dòng duy nhất chứa số nguyên n

Output

- Dòng thứ nhất chứa số m là số lượng số nguyên tìm được
- Dòng thứ hai chứa m số nguyên tố nhỏ hơn n

Ví dụ

INPUT	OUTPUT
10	4 2 3 5 7

INPUT	OUTPUT
20	8 2 3 5 7 11 13 17 19

Hướng dẫn: Dùng kỹ thuật “mảng đánh dấu trạng thái”

Thuật toán “Eratosthene”

- Cấu trúc dữ liệu:** dùng một mảng a để đánh dấu số nào là số nguyên tố, số nào không phải là số nguyên tố.

$$a[i] = \begin{cases} true & \text{nếu } i \text{ là số nguyên tố} \\ false & \text{nếu } i \text{ không phải là số nguyên tố} \end{cases}$$

- Ý tưởng:**

- Ban đầu, chúng ta có tập các số $\{2, 3, \dots, n\}$
- Tại mỗi bước, chúng ta chọn số nhỏ nhất trong tập (số nhỏ nhất này là số nguyên tố) và bỏ đi các bội của số đó

- Cải tiến**

- Mọi số không nguyên tố có ước số $\leq \sqrt{n} \rightarrow$ chúng ta chỉ cần bỏ các bội của **số nguyên tố $\leq \sqrt{n}$**

BUỔI 08


MẢNG 2 CHIỀU

MỤC TIÊU

- Biết cách **khai báo** và **khởi tạo mảng 2 chiều**.
- Thực hiện được các thao tác xử lý mảng 2 chiều: **nhập và xuất mảng, duyệt và tính toán trên mảng**.
- Vận dụng mảng 2 chiều để giải quyết các bài toán.

Bài tập 1. Nhập mảng 2 chiều

Định nghĩa các hàm sau đây:

- `static void NhapMang1(int[,] a)`: Hàm có tham số là một mảng 2 chiều chứa các số nguyên a đã được khởi tạo, yêu cầu người dùng nhập giá trị cho các phần tử của mảng a .
 - `static void NhapMang2(out int[,] a)`: Hàm có tham số là một mảng 2 chiều chứa các số nguyên a **không cần phải khởi tạo trước**, yêu cầu người dùng nhập kích thước, khởi tạo mảng a với kích thước vừa nhập, sau đó nhập giá trị cho các phần tử của a .
 - `static int[,] NhapMang3()`: khởi tạo mảng 2 chiều chứa các số nguyên a bên trong hàm, yêu cầu người dùng nhập kích thước, khởi tạo mảng a với kích thước vừa nhập, sau đó nhập giá trị cho các phần tử của a . Hàm có kết quả trả về là mảng a sau khi đã nhập xong.
 - Nhận xét sự khác nhau của 3 hàm trên.
-  *Thông qua bài tập này, sinh viên cần hiểu và phân biệt được sự khác nhau của 3 hàm nhập mảng. Sau này, tùy vào yêu cầu cụ thể của từng bài toán, sinh viên có thể lựa chọn cách nhập mảng phù hợp chứ không nhất thiết phải cài đặt đầy đủ cả 3 hàm nhập mảng.*

Bài tập 2. Xuất mảng 2 chiều

Định nghĩa hàm có tham số là một mảng 2 chiều chứa các số nguyên a . In các phần tử của mảng a ra màn hình, các phần tử trên mỗi dòng các nhau bởi ký tự khoảng trắng.

Bài tập 3.

Viết chương trình nhập một bảng số nguyên $a[m \times n]$, ($1 \leq m, n \leq 100$) và một số nguyên k ($0 \leq k < m$). **In ra màn hình các phần tử nằm trên dòng thứ k của bảng a** (các phần tử nằm trên cùng một dòng và cách nhau bởi ký tự khoảng trắng). Trường hợp giá trị k không hợp lệ, in thông báo “Invalid k ”.

Input:

- Dòng đầu tiên: 3 số nguyên m, n, k .
- m dòng tiếp theo: mỗi dòng chứa n số nguyên cách nhau bởi dấu khoảng trắng là giá trị các phần tử của bảng số nguyên a .

Output:

- 1 dòng duy nhất chứa các phần tử trên dòng thứ k của bảng a hoặc thông báo “Invalid k ”

INPUT	OUTPUT
3 4 1	9 7 2 8
4 5 6 4	
9 7 2 8	
2 5 7 3	

Bài tập 4. Tính toán cơ bản trên mảng 2 chiều

Cho một bảng số nguyên $a[m \times n]$, ($1 \leq m, n \leq 100$). Định nghĩa các hàm chức năng sau:

- Tính tổng các phần tử nằm trên cột thứ k của bảng a , với k là tham số của hàm.
- Tính tổng tất cả các phần tử trong bảng a .
- Tính tổng các phần tử là số chẵn trong bảng a .
- Tính giá trị trung bình của tất cả các phần tử trong bảng a .

Bài tập 5. Tìm kiếm trên mảng 2 chiều

Cho một bảng số nguyên $a[m \times n]$, ($1 \leq m, n \leq 100$). Định nghĩa các hàm chức năng sau:

- Tìm giá trị phần tử lớn nhất trên dòng thứ k của bảng a , với k là tham số của hàm.
- Tìm giá trị phần tử nhỏ nhất trên cột thứ k của bảng a , với k là tham số của hàm.
- Tìm tất cả phần tử là số nguyên tố trong bảng a .

Bài tập 7.

Viết chương trình nhập một ma trận vuông $a[n \times n]$, ($1 \leq n \leq 100$) chứa các số nguyên. Định nghĩa các hàm chức năng sau:

- In ra màn hình các **phần tử nằm trên đường chéo chính**

INPUT	OUTPUT
3	5 7 6
5 1 3	
4 7 2	
4 2 6	

- In ra màn hình các **phần tử nằm trên đường chéo phụ**

INPUT	OUTPUT
3	3 7 4
5 1 3	
4 7 2	
4 2 6	

- In ra màn hình các **phần tử thuộc tam giác trên đường chéo chính**

INPUT	OUTPUT
3	5 1 3
5 1 3	7 2
4 7 2	6
4 2 6	

BÀI TẬP TỰ LUYỆN TẬP THÊM

Bài tập 1. Cộng 2 ma trận

Cho hai ma trận số nguyên $a[m \times n]$, $b[m \times n]$ với $(1 \leq m, n, p \leq 100)$. Phép cộng ma trận a với ma trận b được ma trận $c[m \times n]$ có các phần tử $c(i, j)$ được định nghĩa như sau:

$$c(i, j) = a(i, j) + b(i, j)$$

Ví dụ

$$a = \begin{pmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \end{pmatrix}$$

$$b = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Mảng kết quả phép cộng

$$c = \begin{pmatrix} 6 & 8 & 10 \\ 12 & 14 & 16 \end{pmatrix}$$

Input

- Dòng số đầu tiên chứa hai số nguyên: m, n
- m dòng tiếp theo, mỗi dòng chứa n số nguyên của ma trận a
- m dòng tiếp theo, mỗi dòng chứa n số nguyên là ma trận b

Output

- Gồm m dòng, mỗi dòng chứa n số nguyên là ma trận tổng của hai ma trận a và b

Ví dụ

INPUT	OUTPUT
2 3	6 8 10
2 3 4	12 14 16
5 6 7	
4 5 6	
7 8 9	

Bài tập 2. Nhân 2 ma trận

Cho hai ma trận số nguyên $a[m \times n]$ và $b[n \times p]$ với $(1 \leq m, n, p \leq 100)$. Phép nhân ma trận a với ma trận b được ma trận $c[m \times p]$ có các phần tử $c(i, j)$ được định nghĩa là tích vô hướng (dot product) của dòng i trong ma trận a với cột j trong ma trận b

$$c(i, j) = \sum_{k=0}^{n-1} a_{i,k} \times b_{k,j}$$

Ví dụ

Ma trận thứ nhất *firstMatrix*

$$\text{firstMatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Ma trận thứ hai *secondMatrix*

$$\text{secondMatrix} = \begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix}$$

Kết quả phép nhân 2 ma trận *productMatrix*

$$\text{productMatrix} = \begin{pmatrix} 21 & 24 & 27 \\ 47 & 54 & 61 \end{pmatrix}$$

$$\text{productMatrix}(0, 0) = 1 \times 5 + 2 \times 8 = 21$$

$$\text{productMatrix}(0, 1) = 1 \times 6 + 2 \times 9 = 24$$

$$\text{productMatrix}(0, 2) = 1 \times 7 + 2 \times 10 = 27$$

$$\text{productMatrix}(1, 0) = 3 \times 5 + 4 \times 8 = 47$$

$$\text{productMatrix}(1, 1) = 3 \times 6 + 4 \times 9 = 54$$

$$\text{productMatrix}(1, 2) = 3 \times 7 + 4 \times 10 = 61$$

Input

- Dòng số đầu tiên chứa ba số nguyên: m, n, p
- m dòng tiếp theo, mỗi dòng chứa n số nguyên của ma trận a
- n dòng tiếp theo, mỗi dòng chứa p số nguyên của ma trận b

Output

- Gồm m dòng, mỗi dòng chứa p số nguyên là ma trận tích của hai bảng trên

Ví dụ

INPUT	OUTPUT
2 2 3	21 24 27
1 2	47 54 61
3 4	
5 6 7	
8 9 10	

Gợi ý:

- Định nghĩa hàm ***DotProduct(a, b, i, j)*** để tính vô hướng của dòng i ma trận a và cột j ma trận b
- Định nghĩa hàm ***ProductMatrix(a, b)*** để tính tích 2 ma trận a và b . Sử dụng 2 vòng lặp:

Với mỗi dòng i ($0 \leq i \leq m$) của ma trận a

Lần lượt duyệt qua tất cả cột j ($0 \leq j \leq p$) của ma trận b

Tính ***DotProduct(a, b, i, j)*** chính là phần tử $[i, j]$ của ma trận tích

Bài tập 3. Tính doanh thu của cửa hàng

Một cửa hàng bán 3 loại trái cây với đơn giá như sau:

Tên trái cây	Đơn giá (\$/kg)
Táo	3
Cherry	4
Lê	2

Số lượng các loại trái cây đã bán từng ngày được thống kê thành bảng như sau:

	Ngày 1	Ngày 2	Ngày 3	Ngày 4	Ngày 5	Ngày 6	Ngày 7
Táo	34	12	15	11	10	2	15
Cherry	56	32	30	40	23	33	24
Lê	6	13	12	20	15	19	8

Doanh thu trong một ngày của cửa hàng được tính bằng tổng số tiền bán từng loại trái cây trong ngày đó. Ví dụ doanh thu của cửa hàng trong ngày 2 bằng:

$$(\$3 \times 12) + (\$4 \times 32) + (\$2 \times 13) = 190$$

Chúng ta có thể diễn đạt điều này bằng tích vô hướng của 2 vector (**dot product**) như sau:

$$(\$3, \$4, \$2) \cdot (12, 32, 13) = \$3 \times 12 + \$4 \times 32 + \$2 \times 13 = \$190$$

Chúng ta có thể mở rộng kết quả sang cả ma trận:

$$(\$3, \$4, \$2) \times \begin{pmatrix} 34 & 12 & 15 & 11 & 10 & 02 & 15 \\ 56 & 32 & 30 & 40 & 23 & 33 & 24 \\ 06 & 13 & 12 & 20 & 15 & 19 & 08 \end{pmatrix} = (338, 190, 189, 233, 152, 176, 157)$$

Tổng quát bài toán: một cửa hàng bán m sản phẩm, giá của m sản phẩm được cho trong mảng $a = (a_0, a_1, \dots, a_{m-1})$ với ($1 \leq m \leq 100$). Số lượng bán được của các sản phẩm trong từng ngày được cho trong bảng số nguyên $b[m \times n]$ với ($1 \leq n \leq 1000$). Trong đó $b_{i,j}$ ($0 \leq i < m, 1 \leq j \leq n$). cho biết số lượng sản phẩm i bán ra trong ngày j . Viết chương trình tính doanh thu từng ngày và tổng doanh thu tất cả các ngày của cửa hàng.

Input

- Dòng số đầu tiên chứa hai số nguyên: m, n
- Dòng thứ hai chứa m số nguyên a_0, a_1, \dots, a_{m-1}
- m dòng tiếp theo, mỗi dòng chứa n số nguyên của bảng b

Output

- Dòng thứ nhất là tổng doanh thu của cửa hàng
- Dòng thứ hai chứa n số, số thứ i là doanh thu của ngày i

Ví dụ

INPUT	OUTPUT
3 4	1435
3 4 2	338 190 189 233 152 176 157
34 12 15 11 10 2 5	
56 32 30 40 23 33 24	
6 13 12 20 15 19 8	

Bài tập 4. Chuyển vị ma trận

Cho ma trận số nguyên $a[m \times n]$ với $(1 \leq m, n \leq 100)$. Chuyển vị ma trận (transpose) là hoán vị các dòng và các cột. Chúng ta đặt ký hiệu T lên góc phía trên bên phải của ma trận với nghĩa là chuyển vị.

$$\begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix}^T = \begin{pmatrix} 5 & 8 \\ 6 & 9 \\ 7 & 10 \end{pmatrix}$$

Viết chương trình tạo ma trận chuyển vị của ma trận a

Input

- Dòng số đầu tiên chứa hai số nguyên: m, n
- m dòng tiếp theo, mỗi dòng chứa n số nguyên của bảng a

Output

- Gồm n dòng, mỗi dòng chứa m số nguyên là ma trận chuyển vị

Ví dụ

INPUT	OUTPUT
2 3	5 8
5 6 7	6 9
8 9 10	7 10

Bài tập 5. Khoảng cách Euclid giữa 2 ma trận

Cho hai ma trận số nguyên $a[m \times n]$, $b[m \times n]$ với $(1 \leq m, n \leq 100)$. Hãy tính khoảng cách Euclid của ma trận a và ma trận b . Biết rằng:

$$\text{dist}(a, b) = \sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (a_{i,j} - b_{i,j})^2}$$

Ví dụ 1

$$a = \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}$$

$$b = \begin{pmatrix} 2 & 5 \\ 8 & 1 \end{pmatrix}$$

$$\text{dist}(a, b) = \sqrt{(4-2)^2 + (5-5)^2 + (6-8)^2 + (7-1)^2} = \sqrt{4 + 0 + 4 + 36} = \sqrt{44} = 6.6332$$

Ví dụ 2

0	0	0	0	0	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	1	1	0	1	1	0	0	0
0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	1	0	0	0

Ma trận a

0	0	0	0	0	1	1	1	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	1	1	1	0	1	1	0	0	0
0	0	1	0	0	1	1	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	1	0	0	0

Ma trận b

Ta có khoảng cách giữa hai ma trận là

$$\text{dist}(a, b) = \sqrt{6} = 2.4495$$

Input

- Dòng số đầu tiên chứa hai số nguyên: m, n
- m dòng tiếp theo, mỗi dòng chứa n số nguyên của ma trận a
- m dòng tiếp theo, mỗi dòng chứa n số nguyên của ma trận b

Output

- Số thực là khoảng cách của 2 ma trận (lấy 2 số lẻ)

Ví dụ

INPUT	OUTPUT
2 2	6.63
4 5	
6 7	
2 5	
8 1	

Bài tập 6. Dot product của hai ma trận

Cho hai ma trận số nguyên $a[n \times n]$ và $b[n \times n]$ với $(1 \leq n \leq 100)$. Dot product của ma trận a với ma trận b là một giá trị được tính như sau

$$value = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{i,j} \times b_{i,j}$$

Ví dụ

Ma trận a

$$a = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 1 & 1 \\ 4 & 5 & 6 \end{pmatrix}$$

Ma trận b

$$b = \begin{pmatrix} 4 & 5 & 6 \\ 2 & 3 & 4 \\ 1 & 2 & 3 \end{pmatrix}$$

Kết quả dot product của a và b :

$$value = 2 \times 4 + 1 \times 5 + 2 \times 6 + 1 \times 2 + 1 \times 3 + 1 \times 4 + 4 \times 1 + 5 \times 2 + 6 \times 3 = 66$$

Input

- Dòng số đầu tiên chứa số nguyên: n
- n dòng tiếp theo, mỗi dòng chứa n số nguyên của ma trận a
- n dòng tiếp theo, mỗi dòng chứa n số nguyên của ma trận b

Output

- Một số là giá trị dot product của a và b

Ví dụ

INPUT	OUTPUT
3	66
2 1 2	
1 1 1	
4 5 6	
4 5 6	
2 3 4	
1 2 3	

Bài tập 7. Phép tích chập (Convolution)

Cho ma trận số nguyên $a[m \times n]$ và $b[k \times k]$ với $(1 \leq m, n, k \leq 100$ và $k < n; k < m)$. Ta gọi ma trận a là ma trận lớn (a còn gọi là image), ma trận b là ma trận nhỏ (b còn gọi là kernel).

Phép tích chập (convolution) của ma trận nhỏ b lên ma trận lớn a được tính bằng cách: trượt ma trận nhỏ b lên ma trận lớn a từ trên xuống dưới, từ trái sang phải (ma trận b phải nằm gọn trong ma trận a). Tại mỗi vị trí trượt chúng ta tính dot product giữa ma trận b với vùng của ma trận a mà b đang được đặt lên trên.

Ví dụ

Ma trận a

$$a = \begin{pmatrix} 4 & 2 & 2 & 4 \\ 1 & 9 & 5 & 3 \\ 1 & 4 & 2 & 4 \\ 0 & 9 & 8 & 1 \end{pmatrix}$$

Ma trận b

$$b = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Mảng kết quả phép tích chập c

$$c = \begin{pmatrix} 51 & 10 \\ -3 & -27 \end{pmatrix}$$

Input

- Dòng số đầu tiên chứa số nguyên: m, n, k
- m dòng tiếp theo, mỗi dòng chứa n số nguyên của ma trận a
- k dòng tiếp theo, mỗi dòng chứa k số nguyên của ma trận b

Output

- Chứa $(m - k + 1)$ dòng, mỗi dòng chứa $(n - k + 1)$ giá trị của ma trận c

Ví dụ

INPUT	OUTPUT
4 4 3	51 10
4 2 2 4	-3 -27
1 9 5 3	
1 4 2 4	
0 9 8 1	
-1 -1 -1	
-1 8 -1	
-1 -1 -1	

BUỔI 09

KIỂU DỮ LIỆU CHUỖI VÀ KỸ THUẬT ĐỌC GHI FILE VĂN BẢN

MỤC TIÊU

- **Nắm được các kỹ thuật xử lý chuỗi:** xét từng ký tự, chuyển đổi ký tự sang số, sử dụng một số hàm trên chuỗi,...
- Có kỹ năng đọc, ghi **File văn bản**

Bài tập 1. Đếm ký tự

Cho chuỗi s (có độ dài nhỏ hơn 10^6), chỉ gồm các ký tự từ 'a' đến 'z', 'A' đến 'Z' và khoảng trắng. Viết hàm thực hiện các chức năng sau:

- **Đếm số lần xuất hiện** của các ký tự khác nhau (không tính khoảng trắng, không phân biệt chữ in HOA và chữ thường).
- Cho biết **ký tự nào xuất hiện nhiều lần nhất** và **số lần xuất hiện là bao nhiêu**? Nếu có nhiều ký tự khác nhau có cùng số lượng xuất hiện nhiều nhất, **chọn ký tự nhỏ nhất** (theo thứ tự từ điển).

Input

- Dòng duy nhất chứa chuỗi s

Output

- Các dòng đầu tiên, mỗi dòng cho biết ký tự (in HOA) và số lần xuất hiện tương ứng (xuất theo thứ tự từ điển).
- Dòng cuối cùng, ký tự xuất hiện nhiều nhất và số lần xuất hiện (nếu có nhiều ký tự cùng số lần xuất hiện nhiều nhất, chọn ký tự nhỏ nhất theo thứ tự từ điển).

Ví dụ

INPUT	OUTPUT
<i>Khoa Công nghệ thông tin HUFLIT</i>	A : 1 C : 1 E : 1 F : 1 G : 3 H : 4 I : 2 K : 1 L : 1 N : 4 O : 3 T : 3 U : 1 First Max Freq: H,4

Hướng dẫn: Dùng mảng để đếm số lần xuất hiện của từng ký tự.

Bài tập 2. Kiểm tra Password

Viết chương trình đọc vào 1 file “*Password.txt*” chứa các chuỗi *s* là các mật khẩu. Hãy kiểm tra mỗi mật khẩu có thỏa các tất cả các ràng buộc sau hay không?

- Có ít nhất 8 ký tự
- Phải chứa ký tự thường, ký tự hoa và ký tự số

Chương trình cần đảm bảo được những yêu cầu sau:

- Yêu cầu người dùng nhập vào đường dẫn của thư mục chứa file “*Password.txt*”.
- Mở file “*Password.txt*” theo đường dẫn trên để đọc dữ liệu? Nếu *file không tồn tại* thông báo lỗi “*File not found!*” và yêu cầu nhập lại đường dẫn khác.
- Khi đọc file thành công, tiến hành kiểm tra các mật khẩu và ghi kết quả vào file “*PasswordValidation.txt*” đặt trong cùng thư mục với file “*Password.txt*”, dưới dạng:
Password : Yes/No (Yes: *hợp lệ*, No: *không hợp lệ*)
- Đóng các luồng đọc, ghi file đang mở.

Ví dụ

Password.txt	PasswordValidation.txt
abc123456	abc123456 : No
123456789	123456789 : No
Abcd12	Abcd12 : No
Abcd123456	Abcd123456 : Yes

Bài tập 3. Tách họ tên

Viết chương trình đọc vào 1 file “*Names.txt*” chứa các chuỗi Họ Tên (các từ cách nhau bởi 1 ký tự khoảng trắng). Hãy viết chương trình **tách chuỗi Họ Tên thành 3 phần: Họ, Tên lót và Tên.**

Yêu cầu về việc đọc, ghi file tương tự như Bài tập 1. Định dạng dữ liệu như ví dụ bên dưới:

Ví dụ

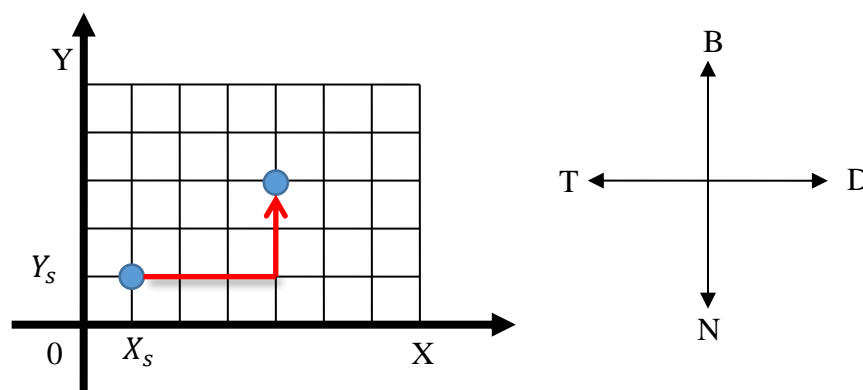
Names.txt	SplitNames.txt
Trần Thành Công	Trần Thành Công
Nguyễn Ngọc Quỳnh Hoa	- Họ: Trần
	- Tên lót: Thành
	- Tên: Công
	Nguyễn Ngọc Quỳnh Hoa
	- Họ: Nguyễn
	- Tên lót: Ngọc Quỳnh
	- Tên: Hoa

Bài tập 4.

Robot đi theo hướng đông, tây, nam, bắc

Một Robot xuất phát từ điểm có tọa độ (X_S, Y_S) trong mặt phẳng tọa độ nguyên OXY , mỗi lần Robot đi được một đơn vị độ dài theo một trong 4 hướng: Đông (*D*), Tây (*T*), Nam (*N*), Bắc (*B*).

Yêu cầu: Viết chương trình điều khiển Robot đi theo câu lệnh là một chuỗi *s* gồm các ký tự *D, T, N, B* ($1 \leq \text{length}(s) \leq 10000$). Hãy in vị trí của Robot (tọa độ) sau mỗi bước đi. (Đọc, ghi dữ liệu vào file).



Input

- Dòng thứ nhất chứa hai số nguyên X_s, Y_s là tọa độ ban đầu của Robot.
- Dòng thứ hai chứa câu lệnh s .

Output

- Nhiều dòng, mỗi dòng lần lượt là vị trí của Robot sau mỗi bước đi.

Ví dụ

Direction.txt	Moves.txt
1 1	1 1
DDDBB	2 1
	3 1
	4 1
	4 2
	4 3

Bài tập 5. Cộng hai số vô cùng lớn

Để tính toán (cộng, trừ, nhân, chia) các số có hàng ngàn, hàng triệu chữ số chúng ta không thể sử dụng các kiểu dữ liệu int, double. Một giải pháp khả thi là chúng ta lưu trữ các số lớn vào chuỗi, mỗi số là một chuỗi, sau đó viết thuật toán mô phỏng cách con người thực hiện các phép toán trên các chuỗi.

Yêu cầu: cho hai số nguyên dương lớn được lưu trong hai chuỗi s_1, s_2 . Hãy viết chương trình tính tổng hai số s_1, s_2

Input

- Dòng đầu tiên chứa chuỗi s_1
- Dòng thứ hai chứa chuỗi s_2

Output

- Dòng duy nhất chứa kết quả là tổng của hai số

Ví dụ

INPUT	OUTPUT
123456	124332
876	

Hướng dẫn:

- **Bước 1.** Thêm các số 0 vào đầu chuỗi của chuỗi có độ dài ngắn hơn sao cho 2 chuỗi có độ dài bằng nhau

123456	➔	123456
876		000876
-----		-----

- **Bước 2.** Thực hiện phép cộng từng chữ số từ phải sang trái, chú ý khi cộng có thể phát sinh ra giá trị nhớ cho số kế bên.

Ví dụ:

- $6 + 6 = 12$ viết 2 nhớ 1.
- $5 + 7 = 12$, cộng thêm nhớ 1 trước đó và được kết quả là 13. Viết 3 nhớ 1

123456	
000876	

2	Nhớ = 1

- **Bước 3.** Sau khi cộng nếu còn nhớ thì phải thêm nhớ vào đầu chuỗi kết quả

--- HẾT ---

