**Kỹ Thuật Lập Trình**

**Học kỳ 2 – Năm học 2017-2018**

**Lab 5 – Sử dụng mảng, chuỗi và cấu trúc**

# MỤC TIÊU CỦA BÀI THỰC HÀNH

* Định nghĩa được kiểu dữ liệu mới sử dụng typedef và struct.
* Khai báo được các biến có kiểu struct và array
* Ghi và đọc dữ liệu cho các thành phần của struct
* Ghi và đọc dữ liệu cho mảng
* Sử dụng struct và mảng để giải quyết một phần hoặc toàn bộ bài toán thực tế.
* Thực hiện các bài toán trên chuỗi

# BÀI TẬP BẮT BUỘC

**Câu 1:**

* Định nghĩa **các kiểu dữ liệu mới** để có thể lưu trữ được thông tin của các thực thể (đối tượng) như sau. Với mỗi loại đối tượng như vậy, hãy tự xác định ít nhất 4 thông tin của nó, tự xác định kiểu dữ liệu của các trường thông tin này.
  + Máy in
  + Môn học
  + Doanh nghiệp
  + Con chim
* Khai báo một mảng cho mỗi loại thực thể ở trên. Mỗi mảng có thể chứa 100 thực thể. Hãy sử dụng macro cho con số 100.
* Thực hiện việc gán giá trị (xác định thông tin) của 1 thực thể bất kỳ trong 100 thực thể đó (làm cho từng loại nói trên)
* In ra thông tin thực thể vừa gán, cũng làm cho từng loại
* Khai báo một biến có kiểu là “Máy in” nói trên, khởi động với các giá trị bất kỳ để mô tả một máy tin cụ thể. Gán thực thể máy tin vừa tạo ra vào một trong 100 thực thể máy in ở trên, vị trí bất kỳ. Làm lại các công việc trong mục này cho các kiểu dữ liệu khác như: “Môn học”, “Doanh nghiệp”, v.v.

**Hướng dẫn:**

Sinh viên có thể tham khảo ví dụ sau về khai báo kiểu dữ liệu mới Máy in với 2 thông tin: tên (chuỗi) và độ phân giải (số thực):

typedef struct {

string name;

float resolution[2];

} computer;

Kiểu dữ liệu mới của chúng ta được định nghĩa dựa trên struct, nên cách khởi tạo sẽ giống struct:

**Cách 1 (hợp lệ trong C):**

computer com = {

.name = "HP",

.resolution = { 1366, 768 },

};

Hoặc nếu ta chỉ muốn khởi tạo giá trị cho trường thông tin “resolution” (không thể khởi tạo theo cách này trong C++):

computer com = {

.resolution = { 1366, 768 },

};

**Cách 2 (hợp lệ trong C++):**

computer com = {

"HP",

{ 1366, 768 },

};

Sinh viên có thể thử bằng cách dùng trình biên dịch online như [CodeChef](https://www.codechef.com/ide) để thử các trường hợp:

* C (Gcc-4.9.2)
* C++ 4.9.2 (Gcc-4.9.2)
* C++ 14 (G++-4.9.2)

Cách gán cấu trúc tương tự như gán giá trị vào biến thông thường:

* com2 = com;
* com\_array[10] = com;

Thành phần trong cấu trúc (struct) có thể được gán bằng câu lệnh có dạng như sau:

com.name = "My new printer";

Lưu ý: giá trị của vế phải trong phép gán phải tương thích với kiểu dữ liệu của trường thông tin trong cấu trúc.

Truy xuất trường thông tin trong đối tượng bằng cách:

var\_struct.attr

với:

* var\_struct: tên biến có kiểu dữ liệu là struct
* attr: trường thông tin cần truy xuất

Khai báo mảng có kích thước MAX (với MAX là hằng số):

computer com\_array[MAX];

Ta có thể lấy thông tin của 1 đối tượng struct đang nằm trong 1 mảng bằng cách:

var\_array [index].attr

với:

* var\_array: tên biến kiểu mảng
* index: vị trí phần tử trong mảng

attr: trường thông tin cần truy xuất

**Câu 2:**

Viết chương trình cho phép:

* Đọc hai vector trong không gian N chiều từ tập tin đầu “input.txt”. Gọi đó là vector **a** và **b**. N là số chiều của hai vector. Sinh viên phải xác định N khi đọc dữ liệu từ hàng đầu tiên. N không thể lớn hơn 50. Tập tin “input.txt” có định dạng như sau cho các vector trong không gian 3 chiều. Nếu quá trình đọc có lỗi hay hai vector không cùng số chiều thì in ra thông báo và dừng chương trình.

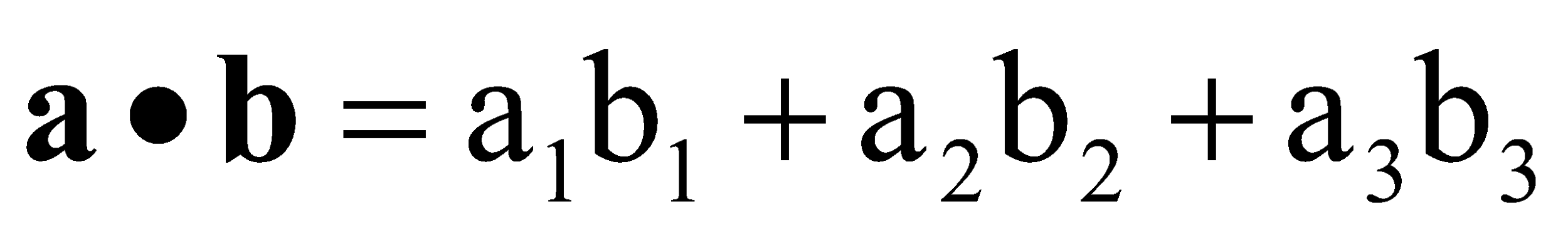
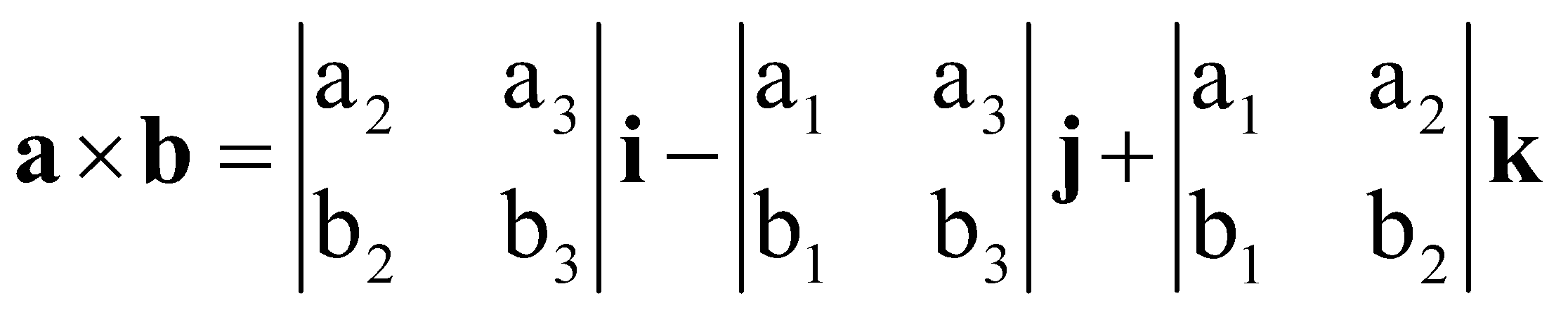
----------------------

----- Vectors ------

----------------------

12.4 34.2 44.2

3.5 2.21 21.56

* Tính và in ra:
  + Chiều dài của mỗi vector. Ví dụ: vetor a = [a1 a2 a3 a4] trong không gian 4 chiều có chiều dài là: sqrt(a1\*a1 + a2\*a2 + a3\*a3 + a4\*a4)
  + Tính và in ra **tích vô hướng** và **tích hữu hướng** (*chỉ tính tích hữu hướng cho trường hợp 3 chiều*) giữa hai vector.
* **Cho biết:** Nếu có 2 vector **a** và **b**: **a**=[a1, a2, a3]T và **b**=[b1, b2, b3]T, tích vô hướng và hữu hướng của **a** và **b** được tính như sau.
* 
* 
  + Tính và in ra góc tạo bởi hai vector.
  + Tính và in ra độ dài của hình chiếu vector **a** trên vector **b** và ngược lại
* Hai câu cuối sử dụng công thức **a**.**b** = |**a**| \* |**b**| \* cos(**alpha**). **alpha** là góc tạo bởi hai vector a và b. **alpha** = acos(**a**.**b**)/( |**a**| |**b**|). Độ dài hình chiếu **a** trên **b** là: **a**.**b**/|**b**|

**Yêu cầu cụ thể:**

* Sinh viên phải sử dụng macro cho hằng số 50 nói trên
* Sinh viên phải định nghĩa một cấu trúc vector. Hãy tự xác định cần phải lưu trữ gì cho một vector.

**Hướng dẫn:**

1. **Dữ liệu:**

* Có thể định nghĩa một struc VECTOR có các phần tử như sau: value, dimension. Sinh viên xác định kiểu các phần tử cho phù hợp và thực hiện việc đặt tên biến sao cho gợi nhớ.
* Define các hằng số MAX\_DIMENSION phục vụ cho việc nhập các phần tử từ input và PI, DEG dùng cho việc chuyển đổi góc từ radian sang độ.
* Khai báo các biến sử dụng cho đọc file input:
  + string str;
  + istringstream iss;
* Khai báo các biến liên quan đến tính toán vector:
  + double sum\_square, scalar\_p, angle …

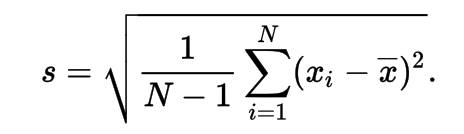
1. **Giải thuật:**

* Giải thuật đọc file input:
  + Sử dụng hàm getline (cin, str) để đọc một line từ luồng nhập (file) và gán vào biến str.
  + Thực hiện việc getline để bỏ qua 3 dòng đầu của file input
  + Đọc vector 1.
  + Sử dụng hàm istringstream::str() để gán lại content của iss. Việc này cho phép sử dụng lại được biến iss cho phép đọc các dòng khác trong file.
  + Sử dụng vòng lặp while để lần lượt dán giá trị cho vector. Điều kiện dừng while khi việc gán bị báo lỗi. ( tương tự lab3)
  + Clear cờ lỗi của istringstream
  + Thực hiện tiếp cho vector 2

**Câu 3:**

Viết chương trình cho phép:

1. Nhập vào một dãy số thực không âm và lưu các số này vào mảng 1 chiều. Việc nhập chỉ kết thúc khi người dùng nhập bất kỳ số âm nào hoặc số lượng phần tử đọc vào đã là MAX\_SIZE. Với MAX\_SIZE là số nguyên dương được định nghĩa bởi chỉ thị #define
2. Tính và in ra: trung bình cộng và độ lệch chuẩn của dãy số đã nhập ở trên. Giả sử, mảng đã nhập có N phần tử, độ lệch chuẩn được tính theo công thức sau:



1. Đảo (không dùng mảng tạm) và in ra dãy số nói trên:

**Hướng dẫn:**

Chương trình nên in hướng dẫn cách nhập trước khi người dùng bắt đầu nhập.

Một trong những cách để giải câu a đó là:

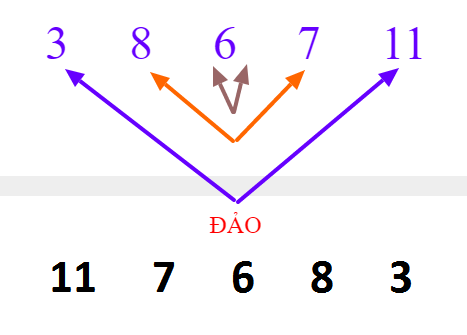
* Dùng vòng lặp for để lặp MAX\_SIZE lần, ta sẽ dừng vòng lặp for sớm khi người dùng nhập vào số âm.
* Ta có thể lồng ghép thêm 1 vòng lặp while ở trong for để bắt các trường hợp nhập vào không hợp lệ (nhập chữ thay vì số), đây là tùy chọn.

Trong câu b, sinh viên cần viết vòng lặp để:

* Tính tổng các số trong mảng, sau đó tính trung bình bằng cách chia cho số phần tử hợp lệ có trong mảng (N <= MAX\_SIZE).
* Tính tổng các bình phương như trong công thức độ lệch chuẩn đề cập, sau đó ta tiến hành tính độ lệch chuẩn.

Để đảo mảng mà không dùng mảng tạm, ta có thể thực hiện cách sau đây:

* Dùng vòng lặp *for* lặp N/2 lần (1 nửa số phần tử có trong mảng)
* Mỗi lần lặp ta đảo phần tử đang xét với phần tử tương ứng từ cuối đếm lên.



Ý tưởng tương tự có thể được hiện thực bằng vòng lặp *while*:

* Khởi tạo chỉ số **left** và **right** lần lượt là chỉ số của phần tử đầu tiên (tức là 0) và chỉ số của phần tử cuối cùng (tức là N-1) trong mảng.
* Mỗi lần lặp trong *while* ta đảo phần tử trong mảng có chỉ số **left** với phần tử có chỉ số **right**, sau đó **left** tiến 1 và **right** lùi 1.

Điều kiện dừng của vòng lặp là khi **left** và **right** vượt qua nhau, tức là **left > right**.

**Câu 4: Tính thứ từ ngày tháng năm**

1. Yêu cầu:
   * Nhập: Ngày, tháng, năm
   * Xuất: cho biết thứ tương ứng với ngày tháng năm đó

Ví dụ: Nhập 05/02/2018, chương trình xuất ra “Thứ hai”

1. Gợi ý: trong bài thực hành số 3, sinh viên dùng câu lệnh if…else để hiện thực chương trình. Trong bài thực hành này, sinh viên dùng mảng để lưu các bảng tra. Ví dụ, bảng tra năm có thể lưu trữ trong mảng như sau:

**int yearLookup[10]={4, 2, 0, 6, 4, 2, 0, 6, 4, 2};**

Để tra giá trị tương ứng của năm, sử dụng **yearLookup[(year / 100 - 17)]**

|  |  |
| --- | --- |
| Từ năm → đến năm | Giá trị |
| 1700 – 1799 | 4 |
| 1800 – 1899 | 2 |
| 1900 – 1999 | 0 |
| 2000 – 2099 | 6 |
| 2100 – 2199 | 4 |
| 2200 – 2299 | 2 |
| 2300 – 2399 | 0 |
| 2400 – 2499 | 6 |
| 2500 – 2599 | 4 |
| 2600 – 2699 | 2 |

*Bảng 1: Bảng tra năm*

|  |  |  |
| --- | --- | --- |
| Tháng | Giá trị  (năm thường) | Giá trị  (năm nhuận) |
| 1 | 0 | 6 |
| 2 | 3 | 2 |
| 3 | 3 | 3 |
| 4 | 6 | 6 |
| 5 | 1 | 1 |
| 6 | 4 | 4 |
| 7 | 6 | 6 |
| 8 | 2 | 2 |
| 9 | 5 | 5 |
| 10 | 0 | 0 |
| 11 | 3 | 3 |
| 12 | 5 | 5 |

*Bảng 2: Bảng tra tháng*

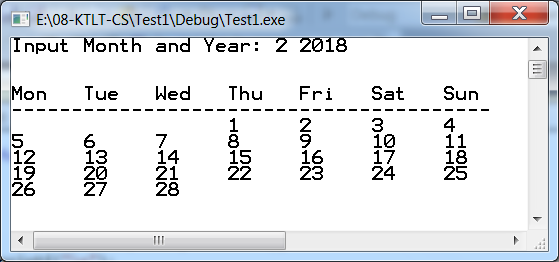
|  |  |
| --- | --- |
| Thứ | Giá trị |
| Chủ nhật | 0 |
| Thứ hai | 1 |
| Thứ ba | 2 |
| Thứ tư | 3 |
| Thứ năm | 4 |
| Thứ sáu | 5 |
| Thứ bảy | 6 |

*Bảng 3: Bảng tra thứ*

**Câu 5: In lịch tháng**

1. Yêu cầu:

* Nhập: tháng và năm.
* Xuất: lịch của tháng đó. (Tham khảo hình 1)



*Hình 1- Lịch tháng 2 năm 2018*

**Hướng dẫn:**

1. **Dữ liệu:**

* Biến *nMonth*, *nYear* chứa tháng và năm do người sử dụng nhập.
* Biến *bLeapYear* có giá trị **true** nếu là năm nhuận, có giá trị **false** nếu là năm thường.
* Biến *nNumOfDay* chứa số ngày trong tháng.
* Mảng *int arr[6][7]* dùng để chứa lịch. Một tháng nhiều nhất nằm trong 6 tuần. Tuần đầu có thể có từ 1 ~ 7 ngày. Tuần cuối có thể có từ 0 ~ 7 ngày. Các tuần giữa đều có 7 ngày. Mỗi hàng của mảng tương ứng với 1 tuần. Cột đầu tiên của mỗi hàng tương ứng với thứ hai, cột tiếp theo là thứ ba v.v. Nếu phần tử nào đó không thuộc về tháng thì giá trị tương ứng trong mảng bằng -1. Ví dụ, với hình vẽ 1;
  + **Tuần 1**: arr[0][0] = -1; arr[0][1] = -1; arr[0][2] = -1;

arr[0][3] = 1;arr[0][4] = 2; arr[0][5] = 3; arr[0][6] = 4;

Tức là, ngày 1 là thứ năm, ngày 2 là thứ 6 v.v

* + **Tuần 2**: arr[1][0] = 5, …, arr[1][6] = 11. Tức là, ngày 5 là thứ hai, ..., ngày 11 là chủ nhật.
  + **Tuần 3**: arr[2][0] = 12, …, arr[2][6] = 18. Tức là, ngày 12 là thứ hai, ..., ngày 18 là chủ nhật.
  + **Tuần 4**: arr[3][0] = 19, …, arr[3][6] = 25. Tức là, ngày 19 là thứ hai, ..., ngày 25 là chủ nhật.
  + **Tuần 5**: arr[4][0] = 26, arr[4][1] = 27, arr[4][2] = 28, arr[4][3] = -1, …, arr[4][6] = -1. Tức là, ngày 26 là thứ hai, …, ngày 28 là thứ tư.
  + **Tuần 6**: arr[5][0] = -1, …, arr[5][6] = -1, tháng 2 năm 2018 chỉ nằm trong 5 tuần.

1. **Giải thuật:**

* Người sử dụng nhập vào giá trị tháng và năm, được lưu lần lượt vào 2 biến *nMonth* và *nYear*.
* Dựa trên giá trị *nYear* để tính giá trị *bLeapYear*.
* Dựa trên giá trị *nMonth*, *nYear* và *bLeapYear* để tính giá trị *nNumOfDay* (số ngày trong tháng).
* Tính xem ngày 1 của tháng đó là thứ mấy.
* Gán giá trị -1 cho tất cả các phần tử của mảng *arr*.
* Gán giá trị từ 1 đến *nNumOfDay* cho các phần tử tương ứng của mảng *arr*.
* In ra lịch tháng.

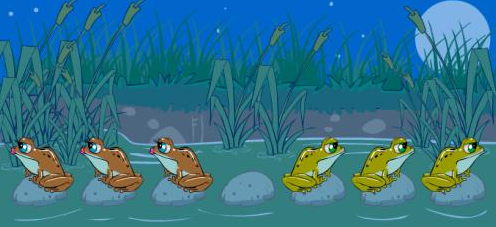
**Câu 6: Ếch qua sông**

Có 2 nhóm ếch muốn qua sông. Mỗi nhóm có 3 con, hãy viết chương trình, sao cho mỗi nhóm ếch đều đi sang được bờ sông đối diện với số bước ít nhất. Quy tắc di chuyển như sau:

* Mỗi con ếch chỉ được di chuyển về phía trước, không được di chuyển lùi về phía sau.
* Mỗi bước chỉ có 1 con ếch di chuyển.
* Ếch có thể di chuyển đến vị trí trống ngay trước mặt, cũng có thể nhảy qua đầu một con ếch của đội kia để đến vị trí trống ngay sau lưng con ếch đó.
* Mỗi lần nhảy, không thể nhảy quá hai vị trí.



*Hình 2 - Ếch qua sông (trạng thái ban đầu)*



*Hình 3- Trạng thái kết thúc*



*Hình 4 – Một ví dụ về hiện tượng kẹt*

**Hướng dẫn:**

Vì chỉ có 1 vị trí trống, nên mỗi lần di chuyển cần phải đề phòng hiện tượng bị kẹt. Hiện tượng bị kẹt là trong quá trình di chuyển, hai con ếch cùng nhóm nằm cạnh nhau (khi chưa nhảy đến vị trí cuối cùng), khiến cho những con ếch khác không thể di chuyển được.

Ví dụ như trong hình 4, con ếch ở vị trí số 3 và số 4 (cùng nhóm) nằm cạnh nhau, tạo nên hiện tượng kẹt. Lúc đó, cả hai nhóm đều không thể di chuyển đến bờ được đối diện. Từ hình 4 có thể thấy, chỉ có con ếch ở vị trí 7 có thể di chuyển sang trái, hoặc con ếch ở vị trí 5 di chuyển sang phải được. Tất cả những con ếch ở vị trí khác đều không thể di chuyển được.

Khi xuất hiện “phải, trái, trống, phải” hoặc “trái, trống, phải, trái” (trong đó “trái” tượng trưng cho con ếch di chuyển sang trái, “phải” tượng trưng cho con ếch di chuyển sang “phải”, “trống” tượng trưng cho vị trí trống), thì không thể di chuyển được con ếch nằm giữa, mà chỉ di chuyển được con ếch nằm ở hai đầu. Dựa vào quy tắc này, có thể đảm bảo trong quá trình di chuyển, không xuất hiện hiện tượng bị kẹt, hơn nữa có thể di chuyển hai nhóm ếch sang bờ đối diện với số lần di chuyển là ít nhất.

Trình tự các bước như sau:

1. Bước 1: Con ếch ở nhóm bên phải nhảy qua đầu con ếch nhóm bên trái để nhảy vào vị trí trống, rồi thực hiện bước 5. Nếu thao tác này không thực hiện được thì tiến hành bước 2.

Lưu ý: Gọi *i* là vị trí ô trống thì vị trí con ếch nhóm bên phải cần xét là *i-2*. Thao tác không thực hiện được khi không có con ếch nhóm bên phải nào tại vị trí đang xét hoặc con ếch ở giữa (*i-1*) không thuộc nhóm bên trái.

1. Bước 2: Con ếch ở nhóm bên trái nhảy qua đầu con ếch nhóm bên phải để nhảy vào vị trí trống, rồi thực hiện bước 5. Nếu thao tác này không thực hiện được thì tiến hành bước 3.

Lưu ý: Gọi *i* là vị trí ô trống thì vị trí con ếch nhóm bên trái cần xét là *i+2*. Thao tác không thực hiện được khi không có con ếch nhóm bên trái nào tại vị trí đang xét hoặc con ếch ở giữa (*i+1*) không thuộc nhóm bên phải.

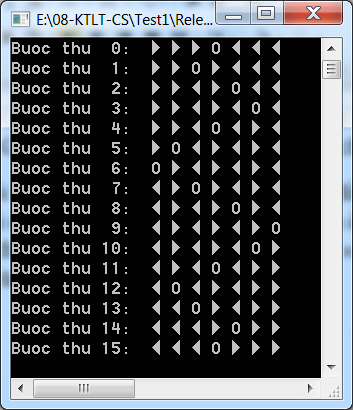
1. Bước 3: Con ếch ở nhóm bên phải di chuyển đến vị trí trống ngay trước mặt (không được tạo nên hiện tượng kẹt), rồi thực hiện bước 5. Nếu thao tác này không thực hiện được, thì thực hiện bước 4.
2. Bước 4: Con ếch ở nhóm bên trái di chuyển đến vị trí trống ngay trước mặt (không được tạo nên hiện tượng kẹt), rồi thực hiện bước 5. Nếu thao tác này không thực hiện được, thì thực hiện bước 5.
3. Bước 5: Kiểm tra xem hai nhóm ếch đã di chuyển sang bờ đối diện hay chưa. Nếu đúng, thì chương trình kết thúc. Nếu không thì quay trở lại bước 1.

Dữ liệu chính của bài toán là mảng có 7 phần tử, được khai báo như sau:

int frog[7] = {1, 1, 1, 0, -1, -1, -1};

* Giá trị 0 cho biết đó là vị trí trống
* Giá trị 1 cho biết đó là vị trí của con ếch thuộc nhóm ếch bên trái
* Giá trị -1 cho biết đó là vị trí của con ếch thuộc nhóm ếch bên phải

Mỗi khi thực hiện các bước di chuyển (1) đến (4) thì giá trị các phần của mảng sẽ thay đổi để phản ánh đúng vị trí của những con ếch. Khi thực hiện xong, thì giá trị của mảng frog sẽ như sau: frog[7] = {-1, -1, -1, 0, 1, 1, 1}. Việc kiểm tra kết thúc ở bước (5) cũng dựa trên mảng frog.



*Hình 5 – Kết quả của bài toán ếch qua sông*

**3 BÀI TẬP KHÔNG BẮT BUỘC**

**Câu 1:**

Viết chương trình nhập vào một chuỗi. Xác định chiều dài của chuỗi. Nghĩa là, hiện thực lại hàm strlen trong <string.h>.

**Hướng dẫn:**

1. **Dữ liệu:** Chương trình phải có các biến (ô nhớ) để chứa các kiểu dữ liệu:

* Biến chứa chuỗi nhập vào từ bàn phím: khai báo mảng kí tự
* Biến chứa độ dài chuỗi: kiểu dữ liệu size\_t

1. **Giải thuật:**

* Dùng macro tiền xử lý #define để chỉ rõ kích thước bộ đệm lưu chuỗi kí tự nhập vào từ bàn phím
* Dùng hàm cin.getline để lấy chuỗi kí tự nhập từ bàn phím (không dùng hàm cin vì cin không lấy được kí tự khoảng trắng từ bàn phím)
* Duyệt và đếm từng kí tự của chuỗi nhập vào cho đến khi bắt gặp kí tự ‘\0’ (kí tự kết thúc chuỗi, độ dài chuỗi không tính kí tự này)
* Số kí tự đếm được chính là độ dài của một chuỗi

**Câu 2:**

Viết chương trình nhập vào một chuỗi. Xoá khoảng trắng hai đầu và chỉ giữ lại 01 khoảng trắng giữa các từ.

**Hướng dẫn:**

1. **Dữ liệu:** Chương trình phải có các biến (ô nhớ) để chứa các kiểu dữ liệu:

* Biến chứa chuỗi nhập vào từ bàn phím: khai báo mảng kí tự

1. **Giải thuật:**

* Dùng macro tiền xử lý #define để chỉ rõ kích thước bộ đệm lưu chuỗi kí tự nhập vào từ bàn phím
* Dùng hàm cin.getline để lấy chuỗi kí tự nhập từ bàn phím (không dùng hàm cin vì cin không lấy được kí tự khoảng trắng từ bàn phím)
* Xóa khoảng trắng ở đầu và cuối chuỗi
* Duyệt qua các khoảng trắng ở giữa hai từ liên tiếp trong chuỗi và đếm số lượng khoảng trắng
* Xóa các khoảng trắng ở giữa hai từ liên tiếp vừa tìm được (giữ một khoảng trắng ở giữa hai từ liên tiếp).
* Lặp lại bước 3 và 4 cho đến khi kết thúc chuỗi.

Ví dụ: “ Lap trinh C++ “ -> “Lap trinh C++ “ ->

“Lap trinh C++ “ -> “Lap trinh C++ “ -> “Lap trinh C++“

**Câu 3:**

Viết chương trình nhập vào một chuỗi. Đảo các từ trong chuỗi. In ra chuỗi ban đầu và chuỗi đã thay đổi.

**Hướng dẫn:**

1. **Dữ liệu:** Chương trình phải có các biến (ô nhớ) để chứa các kiểu dữ liệu:

* Biến chứa chuỗi nhập vào từ bàn phím: khai báo mảng kí tự

1. **Giải thuật:**

* Dùng macro tiền xử lý #define để chỉ rõ kích thước bộ đệm lưu chuỗi kí tự nhập vào từ bàn phím.
* Dùng hàm cin.getline để lấy chuỗi kí tự nhập từ bàn phím (không dùng hàm cin vì cin không lấy được kí tự khoảng trắng từ bàn phím).
* Đảo ngược toàn bộ kí tự trong chuỗi

Ví dụ: “diep thanh dang” -> “gnad hnaht peid”

* Đảo ngược các kí tự trong từng từ trong chuỗi

Ví dụ: “gnad hnaht peid” -> “dang thanh diep”