Kỹ Thuật Lập Trình

Lab 7 – Con trỏ

1 MỤC TIÊU CỦA BÀI THỰC HÀNH

- Hiểu được bản chất hai ô nhớ độc lập của con trỏ
- Khai báo được biến con trỏ đến kiểu dữ liệu cơ bản, đến struct và đến con trỏ khác.
- Truy xuất vùng nhớ qua con trỏ
- Hiểu và sử dụng được toán tử: *, & và ->
- Sử dụng bộ nhớ cấp phát động

2 BÀI TẬP BẮT BUỘC

Câu 1: Viết chương trình theo các bước

a) Khai báo hai biến n và d có kiểu int và double tương ứng.
b) Khai báo biến con trỏ ptr có kiểu là con trỏ đến số nguyên: int*
c) Thực hiện 3 phép gán:
   - ✓ Địa chỉ biến n gán cho ptr
   - ✓ Địa chỉ biến d gán cho ptr
   - ✓ Số 0 gán cho ptr
   - ✓ Số 1000 gán cho ptr
d) Biên dịch chương trình, thảo luận về kết quả biên dịch

Câu 2: Viết chương trình: in ra kích thước (số byte) của các kiểu cơ bản theo từng cặp như ví dụ sau:

int, int*

long double, long double*

…

Lưu ý: nên chạy biên dịch và chạy chương trình trên cả hệ thống linux ở máy chủ (như hướng dẫn trong BTL01)

Câu 3: Viết chương trình theo các bước

a) Khai báo biến n khởi động bằng 100.
b) Khai báo ba biến con trỏ ptr1, ptr2, ptr3 có kiểu là con trỏ đến số nguyên: int*
c) Thực hiện 3 phép gán:
   - ✓ Địa chỉ biến n gán cho ptr1
   - ✓ Địa chỉ biến n gán cho ptr2
   - ✓ ptr1 gán vào ptr3
d) In ra giá trị sau trên các hàng
   - ✓ n, dùng "%d"
   - ✓ *ptr1, dùng "%d"
   - ✓ *ptr2, dùng "%d"
   - ✓ *ptr3, dùng "%d"
   - ✓ &n, dùng "%p"
   - ✓ &ptr1, dùng "%p"
   - ✓ &ptr2, dùng "%p"
   - ✓ &ptr3, dùng "%p"
e) Biên dịch, chạy và thảo luận kết quả

Câu 4: Viết chương trình theo các bước

a) Khai báo một mảng số nguyên có 10 phần tử tối đa và khởi động 5 phần tử đầu tiên là: 10, 20, 30, 40, 50. Tên mảng là arr.
b) Khai báo một con trỏ đến số nguyên: int*; tên con trỏ là ptr.
c) Thực hiện một trong các phép gán: (thử nghiệm lần lượt)
   - ✓ ptr = arr;
   - ✓ ptr = &arr[0];
   - ✓ ptr = &arr[2];
d) In ra các giá trị sau trên màn hình, các dòng khác nhau:
   - ✓ ptr[0]
   - ✓ ptr[1]
   - ✓ ptr[2]

Câu 5: Viết chương trình theo các bước

a) Khai báo một con trỏ đến số nguyên: int*; tên con trỏ là ptr.
b) Thực hiện một trong các phép gán: ptr = 0;
c) In ra giác trị sau trên màn hình:
   ✓ &ptr, dùng "%p"
   ✓ *ptr, dùng "%d"
d) Biên dịch, chạy và thảo luận kết quả

3 BÀI TẬP LÀM THÊM

Câu 1:

a) Hiện thực bài tập sau đây (đã cho trong phần mảng) theo cách dùng con trỏ. Cụ thể, vẫn khai báo mảng tĩnh chứa vector N chiều, gọi hai mảng này là vector_1 và vector_2.
b) Khai báo hai con trỏ: (giả sử chọn `double` là kiểu phần tử của mảng)

```
double *ptr1, *ptr2;
```

c) Gán ptr1 và ptr2 chỉ đến phần tử đầu tiên trong hai mảng.

```
ptr1 = vector_1;

ptr2 = vector_2;
```

d) Sử dụng ptr1 và ptr2 khi truy xuất mảng, thay cho sử dụng `vector_1` và `vector_2`.

Viết chương trình cho phép:

- Nhập vào hai vector trong không gian N chiều, N là hằng số định nghĩa bởi chỉ thị #define.
- 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)
  ✓ Tích vô hướng giữa hai vector. Ví dụ: vector a = [a1 a2 a3 a4] và b=[b1 b2 b3 b4]. Tích vô hướng là: a1*b1 + a2*b2 + a3*b3 + a4*b4

Hướng dẫn:

Khởi tạo con trỏ ptr1 với giá trị là địa chỉ của mảng vector_1. Điều này có thể thực hiện qua 2 cách:

- ptr1 = vector_1;
- ptr1 = &vector_1[0];

Tương tự với con trỏ ptr2.

Để sử dụng một con trỏ đã được gán đến địa chỉ của một mảng để truy xuất các phần tử trong mảng có 2 cách:

- Xem biến con trỏ như biến array, truy cập các phần tử thông phép lấy chỉ mục: ptr1[0], ptr1[1],…
- Di chuyển con trỏ từ phần tử thứ nhất sang phần tử thứ hai, và tương tự với phần còn lại của mảng, với mỗi lần như vậy dùng phép dereference (*) để lấy giá trị của mảng. ptr1 = vector_1 => *ptr1 == vector_1[0]; ptr1++ => *ptr1 == vector_1[1];
- Sử dụng biến con trỏ như địa chỉ nền, dùng phép toán cộng và chỉ mục i để tính ra địa chỉ thật của phần tử i trong mảng cần truy xuất. *(ptr1+0) == vector_1[0], *(ptr1+1) == vector_1[1], *(ptr1+i) == vector_1[i];

Sau khi đã truy cập được mảng đối với các vấn đề tính chiều dài và tích vô hướng, sử dụng kĩ thuật lặp cộng dồn giá trị để giải quyết.

Câu 2:

a) Khai báo một biến N nguyên dương. N cho biết chiều dài vector
b) Nhập giá trị N
c) Tạo mới hai mảng động, theo hướng dẫn:

```
double *vector_1 = (double*) malloc(N*sizeof(double));
double *vector_2 = (double*) malloc(N*sizeof(double));
hoặc
double *vector_1 = new double[N];
double *vector_2 = new double[N];
```

d) Sinh ngẫu nhiên các giá trị cho các vector, phạm vi dữ liệu: -10, đến 10
e) Tính chiều dài của hai vector và tích vô hướng giữa chúng
   - ✓ 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)
   - ✓ Tích vô hướng giữa hai vector. Ví dụ: vector a = [a1 a2 a3 a4] và b=[b1 b2 b3 b4]. Tích vô hướng là: a1*b1 + a2*b2 + a3*b3 + a4*b4

f) In ra kết quả.
g) Giải phóng bộ nhớ.

Hướng dẫn:

Để sinh ngẫu nhiên các giá trị, sinh viên có thể dùng thư viện <random> được hỗ trợ từ bản C++11. Cách dùng được cho ở ví dụ dưới

```
std::random_device rd;

std::mt19937 gen(rd)

std::uniform_real_distribution<double> dis(-10, 10);

std::cout << dis(gen) << ' '; //Each call to dis(gen) generates a new random double
```

Sau khi xin cấp phát một vùng nhớ một cách động (thông qua khai báo động) người dùng phải tự chịu trách nhiệm dọn dẹp vùng nhớ này sau khi dùng xong. Để xóa một vùng nhớ đã khai báo động sinh viên sử dụng các câu lệnh sau:

- free(vector_1); vector_1=NULL; // Nếu cấp phát bằng malloc
- delete[] vector_1; vector_1=NULL; // Nếu cấp phát bằng new

Sau viên chú ý: sau khi đã thu hồi lại vùng nhớ, hãy gán con trỏ bằng NULL để ngăn chặn việc truy xuất đến vùng nhớ không hợp lệ (đã bị thu hồi hoặc được cấp lại cho một vùng nhớ khác).

Câu 3:

Viết chương trình theo các bước

a) Khai báo biến N nguyên dương. N là kích thước ma trận.
b) Nhập N
c) **Cấp phát ma trận vuông có kích thước N*N phần tử số nguyên.**
d) Sinh ngẫu nhiên các số cho ma trận, phạm vi các số là -40 đến 50
e) In ra ma trận dưới (trên) từ ma trận đầu vào
f) In ra ma trận trước và sau khi đảo phần tử trên đường chéo chính
g) **Giải phóng bộ nhớ đã xin ở bước c)**

Hiện thực các bài tập khác của mảng theo hướng sử dụng con trỏ, theo hai cách:

a) **Cách 1:** Vẫn sử dụng mảng tĩnh để lưu trữ dữ liệu. Khai báo con trỏ chỉ đến phần tử đầu tiên của mảng. Truy xuất mảng qua con trỏ

b) **Cách 2:** Không sử dụng mảng tĩnh. Sử dụng con trỏ. Xin cấp bộ nhớ động. Truy xuất dữ liệu qua con trỏ thay cho sử dụng mảng tĩnh.

Hướng dẫn:

Như đã biết một con trỏ thông thường có thể dùng để trỏ đến một mảng một chiều. Vậy để xin cấp phát động một ma trận vuông, ta có thể sử dụng con trỏ 2 chiều, tức là con trỏ trỏ đến một mảng các con trỏ thông thường (con trỏ 1 chiều), và mỗi con trỏ 1 chiều đó sẽ biểu diễn một hàng của ma trận.

Ví dụ cấp phát ma trận 2x3: int **a = new int*[2]; a[0] = new int[3]; a[1] = new int[3];

Khi thu xóa vùng nhớ, chú ý thứ tự xóa sẽ ngược lại so với thứ tự tạo.

Ví dụ xóa vùng nhớ cấp phát ở trên: delete[] a[0]; delete[] a[1]; delete[] a;

Chú ý nhớ gán các biến con trỏ trỏ đến vùng nhớ đã xóa bằng giá trị NULL sau khi xóa xong.

Câu 4:

Viết chương trình cho phép:

a) 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
b) 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:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}.$$

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

Hướng dẫn:

✓ Câu a) và b) áp dụng kỹ thuật điểm danh như Câu 1.
✓ Câu c)
  - Gọi l và r là hai chỉ số, chỉ vào phần tử đầu và cuối của mảng.

- Trong khi mà l < r thì thực hiện phép hoán đổi hai phần tử a[l] và a[r] thông qua 1 biến trung gian. Gọi biến trung gian là t
  - t = a[l];
  - a[l] = a[r];
  - a[r] = t;

Hướng dẫn:

Sinh viên dựa vào những câu đã làm ở trên 1-3 để làm bài này.

Chú ý phải hiện thực bằng 2 cách.

Câu 5:

Viết chương trình cho phép:

a) Sinh ra ngẫu nhiên các điểm ảnh của một ảnh xám có kích thước R hàng và C cột. Ảnh xám là ảnh có các điểm ảnh có giá trị từ 0..255. Học viên từ đề xuất kiểu phần tử cho phù hợp.

b) Tính và in ra biểu đồ tần suất của các giá trị trong ảnh (như Biểu đồ tần suất trong BTL 01). Biểu đồ mức xám là một mảng 1 chiều, chứa 256 giá trị. Mỗi giá trị trong mảng tại chỉ số ID chính là số lượng điểm ảnh trong ảnh trên có giá trị bằng ID

Hướng dẫn:

a) Dữ liệu: Chương trình phải có các biến (ô nhớ) để chứa các kiểu dữ liệu:
- Biến chứa số hàng của ma trận (kiểu unsigned long)
- Biến chứa số cột của ma trận (kiểu unsigned long)
- Biến chứa địa chỉ của ma trận (kiểu con trỏ 2 chiều)
- Biến chứa biểu đồ tần suất (kiểu mảng 1 chiều)

b) Giải thuật:
- Nhập kích thước của ma trận (ảnh xám) từ bàn phím
- Dùng cơ chế cấp phát động bộ nhớ để khởi tạo ma trận
- Dùng các hàm thư viện srand, rand, time để tạo số ngẫu nhiên và gán giá trị cho từng phần tử (điểm ảnh) của ma trận
- Tính biểu đồ tần suất dựa trên công thức đã cho
- In biểu đồ tần suất ra màn hình

Câu 6:

a) Khai báo một ma trận vuông có kích thước **tối đa** là `100x100`. Định nghĩa một hằng có giá trị bằng 100 và sử dụng nó thay cho trực tiếp dùng 100 trong dòng khai báo

b) Nhập N là chiều kích thước thực tiễn của ma trận, phải kiểm tra khi nhập quan hệ:

```
1 <= N <= 100
```

c) Sinh ngẫu nhiên các số cho ma trận, phạm vi các số là -40 đến 50

d) Đảo các phần tử trên **đường chéo chính** của ma trận.

e) In ra ma trận trước và sau khi đảo phần tử

Hướng dẫn:

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

- Biến chứa số hàng (cột) của ma trận vuông (kiểu short)
- Biến chứa địa chỉ của ma trận (kiểu con trỏ 2 chiều)

b) Giải thuật:

- Dùng macro tiền xử lý #define để chỉ rõ kích thước tối đa của ma trận hợp lệ.
- Nhập kích thước của ma trận từ bàn phím, kiểm tra xem nó có hợp lệ không. Nếu không hợp lệ yêu cầu nhập lại
- Dùng cơ chế cấp phát động bộ nhớ để khởi tạo ma trận
- Dùng các hàm thư viện srand, rand, time để tạo số ngẫu nhiên và gán giá trị cho từng phần tử của ma trận
- In ma trận
- Đảo giá trị các phần tử trên đường chéo chính của ma trận
- In ma trận kết quả

Câu 7:

Viết chương trình cho phép:

a) Nhập vào một ma trận vuông có kích thước N. N là hằng số định nghĩa bởi #define. Nhập vào vector N chiều

b) Tính vector kết quả của phép nhân R = M x V. M và V là ma trận và vector nói trên

c) In ra M, V, và R

Hướng dẫn:

a) Sinh viên sử dụng kết quả câu số 3, thực hiện việc cấp phát động vùng nhớ cho matrix, vector và vector kết quả.
b) Sinh viên thực hiện việc phép nhân giữa matrix và vector sử dụng hai phương pháp truy xuất đến phần tử của mảng (sử dụng chỉ mục của mảng và con trỏ mảng)

Câu 8:

Gọi:

- ✓ M là ma trận tam giác dưới có kích thước N
- ✓ X và Y là hai vector có kích thước N

Viết chương trình theo các bước:

a) Sinh ngẫu nhiên các số cho ma trận tam giác dưới, phạm vi các số là -10 đến 10. Các phần tử trên đường chéo phải khác 0
b) Sinh ngẫu nhiên các phần tử cho vector Y
c) Xác định vector X sao cho M*X = Y

Hướng dẫn:

- Khai báo và cấp phát động cho ma trận tam giác M sử dụng con trỏ 2 chiều như hướng dẫn ở các bài tập trước. Lưu ý: ma trận tam giác dưới nên cần lưu ý có thể không cần cấp phát ô nhớ cho những phần tử có giá trị 0 để giảm tiêu tốn ô nhớ.
- Khi sinh phần tử ngẫu nhiên, cần kiểm tra để chắc chắn các giá trị của phần tử nằm trên đường chéo khác 0.
- Tương tự, sử dụng con trỏ một chiều để cấp phát động cho vector X, Y.
- Tham khảo Forward Substitution ở đây: https://en.wikipedia.org/wiki/Triangular_matrix để xác định vector X.
- Giải phóng bộ nhớ con trỏ lưu ma trận M, vector X, Y trước khi kết thúc chương trình

Câu 9:

Viết chương trình nhập 2 ma trận (sử dụng con trỏ và cấp phát động); nhân 2 ma trận nếu có thể và in ra kết quả. Nhớ giải phóng bộ nhớ trước khi kết thúc.

Hướng dẫn:

- Khai báo và cấp phát động cho các ma trận A, B và ma trận kết quả C sử dụng con trỏ 2 chiều như hướng dẫn ở các bài tập trước
- Tiến hành nhân ma trận như đã thực hành ở lab trước
- Giải phóng bộ nhớ con trỏ lưu ma trận A, B và C trước khi kết thúc chương trình