Lab 01: Ôn tập & Con trỏ

Thời gian dự kiến: 03 tuần

1 Ôn tập

Dùng các nguyên mẫu hàm (prototype) cho trước, viết các hàm thực hiện các yêu cầu sau:

1.1 Hàm

1. Giải phương trình tuyến tính $ax + b = 0$
```
void linearEquation(int a, int b);
```

2. Giải phương trình bậc hai $ax^2 + bx + c = 0$ (trường hợp $\Delta < 0$ kết luận vô nghiệm)
```
void quadraticEquation(int a, int b, int c);
```

3. Kiểm tra số nguyên dương a có phải số chẵn hay không
(Optional: không dùng phép toán `mod`, gợi ý: số chẵn thì bit cuối là 0, lẻ thì bit cuối là 1)
```
bool isEven(int a);
```

4. Kiểm tra số nguyên dương a có phải là số nguyên tố hay không
```
bool isPrime(int a);
```

5. Tìm ước chung lớn nhất (greatest common divisor - GCD) của hai số nguyên dương a và b
(Gợi ý: giải thuật Euclid)
```
int GCD(int a, int b);
```

6. Đếm số lượng số nguyên tố trong khoảng (a, b)
```
int countPrimes(int a, int b);
```

7. Đếm số lượng số chia hết cho số nguyên dương k trong đoạn $[1, N]$
```
int countDivisible(int N, int k);
```

8. Hai số a và b được gọi là nguyên tố cùng nhau nếu $GCD(a, b) = 1$.
Đếm số lượng số nguyên tố cùng nhau với một số nguyên dương a trong khoảng $(1, a)$
```
int countCoprimes(int a);
```

9. Tính tổng các chữ số của số nguyên n
```
int sumDigits(int n);
```

10. Cho các số thực a, b, c, xác định xem a, b, c có phải độ dài 3 cạnh của một tam giác hay không? Nếu có, in ra màn hình loại tam giác (*tam giác thường, tam giác vuông, tam giác cân, tam giác vuông cân, tam giác đều*)
```
void triange(double a, double b, double c);
```

1.2 Mảng & Chuỗi

1.2.1 Mảng 1 chiều

1. Nhập mảng a có n phần tử từ bàn phím

```
void inputArray(int a[], int n);
```

2. In mảng a có n phần tử ra màn hình

```
void printArray(int a[], int n);
```

3. Đếm số lượng số nguyên tố trong mảng a

```
int countPrimesInArray(int[] a, int n);
```

4. Tính tổng các phần tử có trong mảng

```
double sumArray(double[] a, int n);
```

5. Kiểm tra mảng có được sắp xếp tăng/giảm dần hay không

```
bool isArraySortedAscending(int[] a, int n);
bool isArraySortedDescending(int[] a, int n);
```

1.2.2 Mảng 2 chiều

1. Nhập từ bàn phím mảng hai chiều kích thước $m \times n$ (m dòng, n cột)

```
void input2DArray(int A[][], int m, int n);
```

2. In mảng hai chiều A ra màn hình

```
void print2DArray(int A[][], int m, int n);
```

3. Tính tổng hai ma trận A và B cùng kích thước $m \times n$

```
void matrixSum(int A[][], int B[][], int result[][], int m, int n);
```

4. Nhân hai ma trận A và B lần lượt có kích thước $m \times n$ và $n \times p$

```
void matrixMultiplication(double A[][], double B[][], double result[][], int m
    , int n, int p);
```

5. Kiểm tra một ma trận vuông A kích thước $n \times n$ có phải là ma trận đường chéo / ma trận tam giác trên / ma trận tam giác dưới

```
bool isDiagonalMatrix(int A[][], int n);
bool isUpperTriangularMatrix(int A[][], int n);
bool isLowerTriangularlMatrix(int A[][], int n);
```

1.2.3 Chuỗi

1. Nhập một chuỗi từ bàn phím
```
void inputString(char C[100]);
```
2. In chuỗi ra màn hình
```
void printString(char C[100]);
```
3. Đếm số lượng chữ in hoa trong một chuỗi
```
int countCapitalInString(char C[100]);
```
4. Đếm số lần xuất hiện của một ký tự c trong một chuỗi
```
int countCharacterAppearance(char C[100], char c);
```
5. Đếm số từ trong một chuỗi
```
int countWords(char C[100]);
```

1.3 Struct

Định nghĩa các struct và các hàm chức năng của struct:

1. Struct lưu thông tin một mốc thời gian trong ngày (từ `00:00:00` tới `23:59:59`)
   - Nhập thông tin một mốc thời gian từ bàn phím, kiểm tra tính hợp lệ của mốc thời gian này
   - Cho một mốc thời gian, tính xem đã bao nhiêu phút/giây trôi qua kể từ nửa đêm (`00:00:00`)
   - Cho hai mốc thời gian (trong cùng một ngày), xác định xem mốc nào sớm hơn
   - Cho một mốc thời gian, tìm mốc thời gian sau đó đúng x phút
2. Struct lưu thông tin một phân số, với tử số và mẫu số là các số nguyên
   - Nhập một phân số từ bàn phím, kiểm tra tính hợp lệ của phân số
   - Rút gọn một phân số
   - Tính tổng, tích hai phân số
   - So sánh hai phân số
3. Struct lưu thông tin một điểm trong không gian hai chiều
   - Xác định góc phần tư của một điểm (góc phần tư thứ I, II, III, IV)
   - Tính khoảng cách (Euclide) giữa hai điểm cho trước
   - Tìm trung điểm của đoạn thẳng nối hai điểm cho trước
   - Kiểm tra ba điểm cho trước có thẳng hàng hay không

1.4 Tập tin

1. Cho tập tin *math.txt* chứa các phép toán hai ngôi trên tập số thực. Tập tin có cấu trúc như sau:

   - Dòng đầu tiên là số nguyên n – số lượng phép toán
   - n dòng tiếp theo, mỗi dòng là một phép toán hai ngôi trên tập số thực, VD:

   Yêu cầu:

   - Đọc các phép toán từ tập tin *math.txt*
   - Thực hiện các phép toán, kết quả lấy đến hai chữ số thập phân.
   - In kết quả ra tập tin *result.txt*, mỗi kết quả nằm trên một dòng.

   Ví dụ: với tập tin *math.txt* có nội dung

```
4
1.1 + 3.5
4.4 * 8
1.2 / 2.0
1.0 / 3.0
```

   Kết quả in ra tập tin *result.txt* như sau

```
4.60
35.20
0.60
0.33
```

2. Cho tập tin *keywords.txt* chứa danh sách các từ khóa tiếng Anh. Tập tin có cấu trúc như sau:

   - Dòng đầu tiên là số nguyên n – số lượng từ khóa
   - Dòng tiếp theo là n từ, phân cách với nhau bằng dấu cách " ". Mỗi từ gồm các chữ cái Latin viết thường, độ dài mỗi từ không vượt quá 20 kí tự.

   Yêu cầu:

   - Đọc danh sách từ khóa từ tập tin *keywords.txt*
   - Nhập từ bàn phím một từ khóa w, xác định và in ra màn hình số lần xuất hiện của w trong danh sách trên.

   Ví dụ: với tập tin *keywords.txt* có nội dung

```
11
natural language processing deep learning machine learning federated learning
   meta learning
```

   - Với từ khóa nhập vào $w =$ *"learning"*, in ra màn hình kết quả là 4.
   - Với từ khóa nhập vào $w =$ *"reinforcement"*, in ra màn hình kết quả là 0.

2 Con trỏ

Dùng các nguyên mẫu hàm (prototype) cho trước, viết các hàm thực hiện các yêu cầu sau:

1. Hoán đổi giá trị hai biến a và b

```
void swap(int* a, int* b);
```

2. Tính tổng hai số a và b

```
int* sum(int* a, int* b);
```

3. Nhập một mảng n số nguyên từ bàn phím

```
void inputArray(int* a, int& n);
```

4. In mảng a ra màn hình

```
void printArray(int* a, int n);
```

5. Tìm phần tử lớn nhất trong mảng a

```
int* findMax(int* a, int n);
```

6. Tạo một bản copy của mảng

```
int* copyArray(int* a, int n);
```

7. Đếm các sỗ chẵn có trong mảng, tạo một mảng mới gồm các số chẵn trong mảng ban đầu

```
int* countEvens(int* arr, int n, int* evens);
int* generateEvenArray(int* arr, int n, int* count);
```

8. Tìm dãy con có tổng lớn nhất trong một mảng

```
int* findLargestSumSubarray(int* a, int n, int& largestSum, int&
    subarrayLength);
```

9. Tìm dãy con tăng dài nhất trong một mảng

```
int* findLongestAscendingSubarray(int* a, int n, int& subarrayLength);
```

10. Hoán đổi giá trị hai mảng

```
void swapArrays(int* a, int* b, int& na, int& nb);
```

11. Nối hai mảng thành một mảng duy nhất

```
int* concatenateTwoArrays(int* a, int* b, int na, int nb);
```

12. Cho a và b là hai mảng với các phần tử phân biệt. Tạo một mảng có thứ tự tăng dần với các phần tử của hai mảng a và b

```
int* mergeTwoArrays(int* a, int* b, int na, int nb, int& nc);
```

13. Tạo một ma trận ngẫu nhiên m dòng và n cột, với m, n nhập từ bàn phím

```
void generateRandomMatrix(int** A, int& m, int& n);
```

14. Cho hai mảng (một chiều) a và b. Tạo ma trận c sao cho với mỗi vị trị $(i, j), c_{ij} = a_i \times b_j$

```
int** calculateProductMatrix(int* a, int* b, int na, int nb);
```

15. Hoán vị hai dòng/cột của một ma trận

```
void swapRows(int** a, int firstRow, int secondRow);
void swapColumns(int** a, int firstCol, int secondCol);
```

16. Tạo ma trận chuyển vị $A^\intercal$ từ ma trận A

```
int** transposeMatrix(int** a, int m, int n);
```

17. Nối hai ma trận A và B (cùng kích thước $m \times n$) theo dòng/cột

```
int** concatenateMatricesByRow(int** a, int** b, int m, int n);
int** concatenateMatricesByCol(int** a, int** b, int m, int n);
```

18. Nhân hai ma trận A (kích thước $m_a \times n_a$) và B (kích thước $m_b \times n_b$)

```
int** multiplyMatrices(int** a, int** b, int ma, int na, int mb, int nb);
```

19. Cho ma trận A kích thước $m \times n$. Tìm một ma trận con A' của A có kích thước $m_{sub} \times n_{sub}$ sao cho tổng các phần tử của A' là lớn nhất.

```
int** findLargestSubmatrix(int** a, int m, int n, int& m_sub, int& n_sub);
```

———Hết———