Lab 01: Ôn tập

Thời gian dự kiến: 03 tuần

1 Con trỏ

Dùng các prototype cho trước, viết các hàm thực hiện các yêu cầu sau:

1. Nhập từ bàn phím một mảng số nguyên a có n phần tử (trong đó `int* a` là con trỏ trỏ tới vùng nhớ đã được cấp phát)
   ```
   void inputArray(int* & a, int& n);
   ```
2. Thu hồi một mảng a đã được cấp phát
   ```
   void deallocateArray(int* & a);
   ```
3. Tạo một mảng a ngẫu nhiên gồm n phần tử
   ```
   int* generateArray(int n);
   ```
4. In mảng a ra màn hình
   ```
   void printArray(int* a, int n);
   ```
5. Tìm phần tử nhỏ nhất trong mảng a
   ```
   int findMin(int* a, int n);
   ```
6. Tìm phần tử có giá trị tuyệt đối lớn nhất trong mảng a
   ```
   int findMaxAbsolute(int* a, int n);
   ```
7. Xác định mảng a đã được sắp xếp tăng dần hay chưa
   ```
   bool isAscending(int* a, int n);
   ```
8. Tính tổng các phần tử trong mảng a
   ```
   int sumOfArray(int* a, int n);
   ```
9. Đếm số lượng số nguyên tố có trong mảng a (gồm các số nguyên dương)
   ```
   int countPrimes(int* a, int n);
   ```
10. Đếm số lượng số chính phương có trong mảng a (gồm các số nguyên dương)
    ```
    int countSquareNumbers(int* a, int n);
    ```
11. Tạo mới một mảng động a gồm các phần tử của mảng b nhưng đảo ngược thứ tự (VD: $b = \{1, 2, 3, 4\}$ thì $a = \{4, 3, 2, 1\}$)
    ```
    void reverseArray(int* &a, int* b, int n);
    ```

12. Tạo mới ma trận chuyển vị $A^\intercal$ của ma trận A kích thước $n \times n$

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

13. Hoán vị 2 dòng / 2 cột i, j của ma trận A kích thước $n \times n$

```
void swapMatrixColumns(int** &a, int n, int i, int j);

void swapMatrixRows(int** &a, int n, int i, int j);
```

14. Nhân hai ma trận A (kích thước $m \times n$) và B (kích thước $n \times p$)

```
int** matrixMultiplication(int** a, int** b, int m, int n, int p);
```

15. Cho ma trận A kích thước $w \times h$. Tìm một ma trận con A' của A có kích thước $w_{sub} \times h_{sub}$ sao cho tổng các phần tử của A' là lớn nhất.

```
int** findLargestSubmatrix(int** a, int w, int h, int& w_sub, int& h_sub);
```

Từ câu 16 trở đi yêu cầu sinh viên cài đặt một số thuật toán tìm kiếm trên mảng số nguyên a gồm n phần tử. Nếu tìm thấy `key` trong a, trả về vị trí đầu tiên của `key`. Ngược lại, trả về -1.

16. Tìm kiếm tuần tự

```
int linearSearch(int* a, int n, int key);
```

17. Tìm kiếm tuần tự, sử dụng kỹ thuật lính canh

```
int sentinelLinearSearch(int* a, int n, int key);
```

18. Tìm kiếm nhị phân (không đệ quy) trên mảng a đã được sắp xếp giảm dần.

```
int binarySearch(int* a, int n, int key);
```

2 Đệ quy

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

1. Tính tổng bình phương các số nguyên nhỏ hơn hoặc bằng n

$$S(n) = 1^2 + 2^2 + .. + n^2$$

```
int sumOfSquares(int n);
```

2. Tìm ước chung lớn nhất (Greatest Common Divider - GCD) của 2 số nguyên a, b

```
int GCD(int a, int b);
```

3. Kiểm tra một mảng a có phải là mảng đối xứng (palindrome) hay không

```
bool isPalindrome(int* a, int left, int right);
```

4. Tính tổng các chữ số của một số nguyên n

```
int sumDigits(int n);
```

5. Số Fibonacci F_n được định nghĩa như sau:

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2}, \forall n \geq 2 \end{cases}$$

Viết hàm tìm số Fibonacci thứ n

```
int fibonacci(int n);
```

6. Tổ hợp chập k của n phần tử có công thức đệ quy:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Lưu ý: $\binom{n}{1} = n, \binom{n}{n} = 1$. Viết hàm tính tổ hợp chập k của n phần tử

```
int combination(int n, int k);
```

7. Tìm kiếm nhị phân (đệ quy) trên mảng a đã được sắp xếp tăng dần

```
int recursiveBinarySearch(int* a, int left, int right, int key);
```

3 Các thao tác với tập tin

3.1 Miêu tả dữ liệu

Dữ liệu được dùng trong bài tập thực hành là dữ liệu về điểm thi THPT của một tỉnh (thông tin thật của thí sinh đã được thay đổi). Tập tin *data.txt* được cung cấp có một phần nội dung như sau:

```
Số Báo Danh, Họ và Tên, Toán, Ngữ Văn, Vật Lý, Hóa Học, Sinh Học, Lịch Sử, Địa Lý, GDCD, KHTN, KHXH, Ngoại Ngữ, Ghi Chú, Tỉnh
BD1200000,,8.6,6.5,4.0,7.25,5.5,,,,,,8.4,N1,BinhDinh
BD1200001,,4.0,5.0,,,,4.25,7.0,7.75,,,2.0,N1,BinhDinh
BD1200002,,7.0,6.25,6.0,6.25,6.5,,,,,,5.2,N1,BinhDinh
BD1200003,,5.2,5.75,,,,5.75,7.25,9.25,,,4.6,N1,BinhDinh
BD1200004,,7.6,6.25,7.0,6.5,4.5,,,,,,6.2,N1,BinhDinh
BD1200005,,8.6,6.5,4.0,7.25,5.5,,,,,,8.4,N1,BinhDinh
```

Trong đó:

- Dòng đầu tiên thể hiện các trường thông tin có trong tập tin.
- Các dòng tiếp theo thể hiện thông tin của mỗi thí sinh, mỗi trường thông tin cách nhau bởi một dấu phẩy ",".
- Trường thông tin về *Họ và tên thí sinh* đã được bỏ. Những trường thông tin về điểm của thí sinh được bỏ trống nghĩa là thí sinh không tham gia thi môn đó. Để đơn giản hoá phần bài làm, khi đọc thông tin thí sinh, những trường thông tin về điểm được bỏ trống sẽ được lưu trữ trong struct mặc định là 0.
- Điểm ở trường KHTN và KHXH sẽ được hướng dẫn cụ thể ở phần 3.2

3.2 Yêu cầu

Cho struct `Examinee` định nghĩa thông tin của một thí sinh:

```
struct Examinee {
    std::string id;
    float math, physics, chemistry, biology, history, geography,
          civic_education, natural_science, social_science, foreign_language;
};
```

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

1. Đọc thông tin một thí sinh

   ```
   Examinee readExaminee(std::string lineInfo);
   ```

   - **Input:** `lineInfo` – một dòng dữ liệu đọc từ *data.txt* chứa thông tin của một thí sinh
   - **Output:** Biến dữ liệu kiểu `Examinee` lưu thông tin của thí sinh
   - **Note:** Do mặc định tập tin *data.txt* không có các trường dữ liệu KHTN và KHXH, vì vậy sinh viên cần tính các cột điểm này và lưu vào struct `Examinee`
     - Điểm tổ hợp KHTN = Lý + Hóa + Sinh
     - Điểm tổ hợp KHXH = Sử + Địa + GDCD

2. Đọc danh sách thông tin các thí sinh

   ```
   std::vector<Examinee> readExamineeList(std::string fileName);
   ```

   - Input: `fileName` – tên tập tin đầu vào, trong trường hợp này là "*data.txt*"
   - Output: Biến dữ liệu kiểu `std::vector<Examinee>` lưu danh sách thí sinh đọc được từ tập tin

3. Ghi thông tin điểm các thí sinh xuống tập tin

   ```
   void writeTotal(std::vector<Examinee> examineeList, std::string outFileName);
   ```

   - **Input:**
     - `examineeList` – danh sách các thí sinh
     - `outFileName` – tên tập tin ghi xuống

     Trong hàm, thực hiện tính tổng điểm thi của thí sinh và ghi xuống tập tin `outFileName` theo định dạng sau:
     - Mỗi thông tin của thí sinh ghi trên 1 dòng
     - Thông tin thí sinh bao gồm ID và điểm tổng, được cách nhau bởi 1 khoảng trắng " "
     - Ví dụ:
       XX001 42.0
       XX002 38.5
       ..
       XX999 23.25
     - Điểm tổng = Toán + Văn + Ngoại ngữ + KHTN + KHXH

4 Danh sách liên kết (DSLK)

4.1 DSLK đơn

Cho các struct định nghĩa một DSLK đơn và một node của DSLK đó:

```
struct Node {
    int key;
    Node* pNext;
};

struct List {
    Node* pHead;
    Node* pTail;
};
```

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

1. Tạo một Node từ một số nguyên `data` cho trước
   ```
   Node* createNode(int data);
   ```
2. Khởi tạo List từ một Node cho trước
   ```
   List* createList(Node* pNode);
   ```
3. Chèn một số nguyên `data` vào đầu List cho trước
   ```
   bool addHead(List*& L, int data);
   ```
4. Chèn một số nguyên `data` vào cuối List cho trước
   ```
   bool addTail(List*& L, int data);
   ```
5. Chèn một số nguyên `val` vào một vị trí `pos`
   ```
   bool addPos(List*& L, int val, int
       pos);
   ```
6. Chèn một số nguyên `data` vào trước vị trí của Node mang giá trị `val`
   ```
   bool addBefore(List*& L, int data,
       int val);
   ```
7. Chèn một số nguyên `data` vào sau vị trí của Node mang giá trị `val`
   ```
   bool addAfter(List*& L, int data,
       int val);
   ```
8. Xóa node đầu của List
   ```
   void removeHead(List*& L);
   ```
9. Xóa node cuối của List
   ```
   void removeTail(List*& L);
   ```
10. Xóa node ở vị trí `pos` trong List
    ```
    void removePos(List*& L, int pos);
    ```
11. Xóa tất cả node trong List
    ```
    void removeAll(List*& L);
    ```
12. Xóa node đứng trước node mang giá trị `val` trong List
    ```
    void removeBefore(List* L, int val);
    ```
13. Xóa node đứng sau node mang giá trị `val` trong List
    ```
    void removeAfter(List* L, int val);
    ```
14. In giá trị các node có trong list ra màn hình
    ```
    void printList(List* L);
    ```
15. Đếm số lượng node có trong list
    ```
    int countElements(List* L);
    ```
16. Đảo thứ tự các node trong list và lưu vào một list mới
    ```
    List* reverseList(List* L);
    ```
17. Xóa các node sao cho list không có node nào trùng nhau
    ```
    void removeDuplicate(List*& L);
    ```
18. Xóa các node có giá trị `val` trong list
    ```
    bool removeElement(List*& L, int val
        );
    ```

4.2 DSLK đôi

Cho các struct định nghĩa DSLK đôi và một Node của DSLK đó:

```
struct DNode {
    int key;
    DNode* pNext;
    DNode* pPrev;
};

struct DList {
    DNode* pHead;
    DNode* pTail;
};
```

Thực hiện các yêu cầu ở 4.1 với DSLK đôi trên.

5 Stack - Queue

Cho struct định nghĩa một Node của DSLK đơn:

```
struct Node {
    int key;
    Node* pNext;
};
```

Từ định nghĩa trên, cài đặt Stack và Queue bằng DSLK đơn, sau đó viết các hàm thực hiện các yêu cầu sau:

- Stack
    1. Khởi tạo Stack từ một `key` cho trước
    2. **Push** một `key` vào Stack
    3. **Pop** một Node ra khỏi Stack, trả về giá trị của Node này
    4. Đếm số lượng các Node trong Stack
    5. Kiểm tra Stack có rỗng hay không
- Queue
    1. Khởi tạo Queue từ một `key` cho trước
    2. **Enqueue** một `key` vào Queue
    3. **Dequeue** một Node ra khỏi Queue, trả về giá trị của Node này
    4. Đếm số lượng các Node trong Queue
    5. Kiểm tra Queue có rỗng hay không

———Hết———