ÔN THI CUỐI KÌ CƠ SỞ LẬP TRÌNH

MỘT SỐ ĐOẠN CODE THÔNG DỤNG

PHẦN 1 — MẢNG 1 CHIỀU

1.1. Mảng tĩnh

- **Thêm phần tử vào vị trí bất kỳ**

Code

```
void insertAt(int a[], int &n, int pos, int x) {
    for (int i = n; i > pos; i--) a[i] = a[i-1];
    a[pos] = x;
    n++;
}
```

Input

```
Mảng: 1 3 7
pos = 1, x = 5
```

Output

```
1 5 3 7
```

- **Xóa phần tử**

Code

```
void deleteAt(int a[], int &n, int pos) {
    for (int i = pos; i < n - 1; i++) a[i] = a[i + 1];
    n--;
}
```

Input

```
Mảng: 2 4 6
pos = 1
```

Output

```
2 6
```

- **Reverse**

Code

```
void reverseArr(int a[], int n) {
    for(int i = 0; i < n/2; i++) swap(a[i], a[n-1-i]);
}
```

Input

```
1 2 3 4 5
```

Output

```
5 4 3 2 1
```

- **Tìm giá trị x trong đoạn [l r]**

Code

```
int Find(int a[], int l, int r, int x) {
    for (int i = l; i <= r; i++) if (a[i] == x) return i;
    return -1;
}
```

❖ **Unique**

Code

```
void uniqueArr(int a[], int &n) {
    for (int i = 0; i < n; i++) {
        while (Find(a, 0, i-1, a[i]) != -1) deleteAt(a, n, i);
    }
}
```

Input

```
1 2 2 3 1 4
```

Output

```
1 2 3 4
```

❖ **Hợp 2 mảng**

Code

```
void mergeArr(int a[], int n, int b[], int m, int c[], int &lenC) {
    for(int i=0;i<n;i++) c[i] = a[i];
    for(int i=0;i<m;i++) c[n+i] = b[i];
    lenC = n + m;
}
```

Input

```
a: 1 2 3
b: 9 8
```

Output

```
1 2 3 9 8
```

1.2. Mảng động

- **Cấp phát**

Code

```
int* allocArr(int n) {
    return new int[n];
}
```

Input

```
n = 5
```

Output

```
Mảng gồm 5 ô nhớ được cấp phát động
```

- **Thu hồi**

Code

```
void freeArr(int* &a) {
    delete[] a;
    a = nullptr;
}
```

Input

```
Mảng động bất kỳ
```

Output

```
Bộ nhớ được giải phóng
```

❖ **Thêm vào vị trí bất kỳ**

Code

```
void insertAt(int* &a, int &n, int pos, int x) {
    int* b = new int[n + 1];
    for(int i=0;i<pos;i++) b[i] = a[i];
    b[pos] = x;
    for(int i=pos;i<n;i++) b[i+1] = a[i];
    delete[] a;
    a = b;
    n++;
}
```

Input

```
Mảng: 1 3 7 9
pos = 2, x = 5
```

Output

```
1 3 5 7 9
```

❖ **Xóa phần tử**

Code

```
void eraseAt(int* &a, int &n, int pos) {
    int* b = new int[n - 1];
    for(int i=0, j=0;i<n;i++) if(i != pos) b[j++] = a[i];
    delete[] a;
    a = b;
    n--;
}
```

Input

```
Mảng: 10 20 30 40
pos = 1
```

Output

```
10 30 40
```

- ❖ **Hợp 2 mảng động**

Code

```
int* mergeDyn(int a[], int n, int b[], int m, int &lenC) {
    int* c = new int[n+m];
    for(int i=0;i<n;i++) c[i] = a[i];
    for(int i=0;i<m;i++) c[n+i] = b[i];
    lenC = n + m;
    return c;
}
```

Input

```
a: 1 2 3
b: 4 5
```

Output

```
1 2 3 4 5
```

- ❖ **Reverse**

Code

```
void reverseDyn(int* a, int n) {
    for(int i = 0; i < n/2; i++) swap(a[i], a[n-1-i]);
}
```

Input

```
1 2 3 4 5
```

Output

```
5 4 3 2 1
```

❖ **Tìm giá trị**

```
int Find(int *a, int l, int r, int x) {
      for (int i = l; i <= r; i++) if (a[i] == x) return i;
      return -1;
   }
```

❖ **Unique**

Code

```
void uniqueDyn(int* a, int &n) {
    for (int i = 0; i < n; i++) {
        while (Find(a, 0, i-1, a[i]) != -1) eraseAt(a, n, i);
    }
}
```

Input

```
1 2 2 3 1 4
```

Output

```
1 2 3 4
```

PHẦN 2 — MẢNG 2 CHIỀU

2.1. Mảng tĩnh

❖ **Nhập mảng 2D**

Code

```
void input(int a[][100], int r, int c) {
    for(int i=0;i<r;i++)
        for(int j=0;j<c;j++)
            cin >> a[i][j];
}
```

Input

```
2 3
1 2 3
4 5 6
```

Output

```
Mảng được nhập thành công
```

❖ **Xuất mảng 2D**

Code

```
void output(int a[][100], int r, int c) {
    for(int i=0;i<r;i++){
        for(int j=0;j<c;j++) cout << a[i][j] << " ";
        cout << endl;
    }
}
```

Input

```
1 2 3
4 5 6
```

Output

```
1 2 3
4 5 6
```

2.2. Mảng động

- ❖ **Cấp phát**

Code

```
int** alloc2D(int r, int c) {
    int** a = new int*[r];
    for(int i=0;i<r;i++) a[i] = new int[c];
    return a;
}
```

Input

```
r = 3, c = 4
```

Output

```
Ma trận 3x4 được cấp phát động
```

- ❖ **Thu hồi**

Code

```
void free2D(int** &a, int r) {
    for(int i=0;i<r;i++) delete[] a[i];
    delete[] a;
    a = nullptr;
}
```

Input

```
Ma trận động bất kỳ
```

Output

```
Bộ nhớ ma trận đã được giải phóng
```

- ❖ **Nhập**

Code

```
void input2D(int** a, int r, int c) {
    for(int i=0;i<r;i++)
        for(int j=0;j<c;j++)
            cin >> a[i][j];
}
```

Input

```
3 3
1 2 3
4 5 6
7 8 9
```

Output

```
Ma trận được nhập thành công
```

❖ **Xuất**

Code

```
void output2D(int** a, int r, int c) {
    for(int i=0;i<r;i++){
        for(int j=0;j<c;j++) cout << a[i][j] << " ";
        cout << endl;
    }
}
```

Input

```
1 2 3
4 5 6
```

Output

```
1 2 3
4 5 6
```

❖ **Xoay ma trận 90°**

Code

```
int** rotate90(int** a, int n) {
    int** b = alloc2D(n, n);
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            b[j][n-1-i] = a[i][j];
    return b;
}
```

Input

```
1 2 3
4 5 6
7 8 9
```

Output

```
7 4 1
8 5 2
9 6 3
```

PHẦN 3 — LINKED LIST

- ❖ **Khai báo Node**

Code

```
struct Node {
    int val;
    Node* next;
    Node(int v) : val(v), next(nullptr) {}
};
```

- ❖ **Thêm đầu**

Code

```
void pushFront(Node* &head, int x) {
    Node *newNode = new Node(x);
    newNode->next = head;
    head = newNode;
}
```

Input

```
Danh sách: 5 -> 7
Thêm đầu: x = 3
```

Output

```
3 -> 5 -> 7
```

- **Thêm cuối**

Code

```
void pushBack(Node* &head, int x) {
    Node *newNode = new Node(x);
    if (head == nullptr) {
        head = newNode;
        return;
    }
    Node *p = head;
    while (p->next != nullptr) p = p->next;
    p->next = newNode;
}
```

Input

```
5 -> 7
Thêm cuối: x = 10
```

Output

```
5 -> 7 -> 10
```

❖ **Xóa đầu**

Code

```
void popFront(Node* &head) {
   if (head == nullptr) return;
   Node* tmp = head;
   head = head->next;
   delete tmp;
}
```

Input

```
3 -> 5 -> 7
```

Output

```
5 -> 7
```

❖ **Xóa cuối**

Code

```
void popBack(Node* &head) {
    if (!head) return;
    if (!head->next) { delete head; head = nullptr; return; }

    Node *p = head, *pre = nullptr;
    while (p->next) {
        pre = p;
        p = p->next;
    }
    pre->next = nullptr;
    delete p;
}
```

Input

```
5 -> 7 -> 10
```

Output

```
5 -> 7
```

- ❖ **Kích thước**

Code

```
int Size(Node* head) {
    int sz = 0;
    for (Node* p = head; p; p = p->next)
        sz++;
    return sz;
}
```

Input

```
1 -> 2 -> 3
```

Output

```
Size = 3
```

❖ **Thêm vào vị trí K**

Code

```
// Nếu k = 0 -> Thêm đầu
// Nếu k >= size -> Thêm cuối
void InsertAtK(Node* &head, int x, int k) {
    if (k == 0) {
        pushFront(head, x);
        return;
    }
    if (k >= Size(head)) {
        pushBack(head, x);
        return;
    }

    Node *newNode = new Node(x);
    Node *p = head;
    for (int i = 1; i < k; i++) p = p->next;

    newNode->next = p->next;
    p->next = newNode;
}
```

Input

```
5 -> 7 -> 10
k = 1, x = 99
```

Output

```
5 -> 99 -> 7 -> 10
```

❖ **Xóa tại vị trí K**

Code

```
void deleteAtK(Node* &head, int k) {
    if (k < 0 || k >= Size(head)) return;

    if (k == 0) { popFront(head); return; }

    Node *p = head, *pre = nullptr;
    for (int i = 0; i < k; i++) {
        pre = p;
        p = p->next;
    }

    pre->next = p->next;
    delete p;
    p = nullptr;
}
```

Input

```
5 -> 99 -> 7 -> 10
k = 1
```

Output

```
5 -> 7 -> 10
```

❖ **Đảo ngược danh sách**

Code

```
void Reverse(Node* &head) {
    Node *prev = nullptr, *cur = head;
    while (cur) {
        Node *nxt = cur->next;
        cur->next = prev;
        prev = cur;
        cur = nxt;
    }
    head = prev;
}
```

Input

```
1 -> 2 -> 3
```

Output

```
3 -> 2 -> 1
```

❖ **In danh sách**

Code

```
void printList(Node* head) {
    for (Node* p = head; p; p = p->next) cout << p->val << " ";
}
```

Input

```
5 -> 99 -> 7
```

Output

```
5 99 7
```

❖ **Giải phóng danh sách**

Code

```
void clearList(Node* &head) {
    while (head != nullptr) popFront(head);
}
```

Input

```
Danh sách bất kỳ
```

Output

```
Danh sách rỗng, không còn rò rỉ bộ nhớ
```

PHẦN 4 — CSTRING

❖ **strlen**

```
strlen("Hello");
```

Output:

```
5
```

❖ **strcpy**

```
strcpy(dest, "Hi");
```

Output:

```
dest = "Hi"
```

❖ **strcat**

```
strcat(a, "world");
```

Input:

```
a = "Hello "
```

Output:

```
"Hello world"
```

❖ **strcmp**

```
strcmp("abc","abc") → 0
strcmp("ab","ac") → <0
strcmp("ac","ab") → >0
```

❖ **strtok**

```
strtok(s, ",");
```

Input:

```
"a,b,c"
```

Output:

```
a
b
c
```

PHẦN 5 — ĐỌC / GHI FILE TEXT

❖ **Đọc số**

```
int x;
while(fin >> x) ...
```

Input:

```
1 2 3
```

Output:

```
1 2 3
```

❖ **Đọc dòng**

```
getline(fin, s);
```

Input:

```
Hello world
```

Output:

```
Hello world
```

❖ **Ghi file**

```
fout << "Hello file!";
```

Output file:

```
Hello file!
```

--- HẾT ---