

Ôn tập Cấu trúc dữ liệu và giải thuật

Danh sách liên kết

Tạo cấu trúc nút

```
struct Node {  
    int info;  
    Node *next;  
};  
Node *head;
```

Tạo một nút mới

```
Node *TaoNut(int x) {  
    Node *n = new Node;  
    n -> info = x;  
    n -> next = nullptr;  
    return n;  
}
```

Chèn cuối

```
void ChenCuoi(Node *&head, int x) {  
    Node *p = TaoNut(x);  
    if (head == nullptr)  
        head = p;  
    else {  
        Node *temp = head;  
        while (temp->next != nullptr)  
            temp = temp->next;  
        temp->next = p;  
    }  
}
```

Chèn đầu

```
void ChenDau(Node *&head, int x) {  
    Node *p = TaoNut(x);  
    p -> next = head;  
    head = p;  
}
```

Xoá phần tử bất kì trong danh sách liên kết

```
Node* XoaPhanTu(Node* head, int x) {
    Node* p = TimKiem(head, x);

    if (p == NULL) {
        printf("Khong tim thay phan tu %d trong danh sach.\n", x);
        return head;
    }

    if (p == head) {
        head = head->next;
        free(p);
        printf("Da xoa phan tu %d khoi danh sach.\n", x);
        return head;
    }

    Node* q = head;
    while (q->next != p)
        q = q->next;
    q->next = p->next;

    free(p);
    printf("Da xoa phan tu %d khoi danh sach.\n", x);
    return head;
}
```

Duyệt các phần tử có trong danh sách liên kết

```
void Duyet(Node *head) {
    Node *p = head;
    while (p != NULL)
    {
        cout << p -> info << " ";
        p = p -> next;
    }
}
```

Tìm kiếm

```
Node* TimKiem(Node* head, int x) {
    Node *p = head;
    while (p != NULL)
    {
        if (p -> info == x)
            return p;
        p = p -> next;
    }
}
```

```
    return NULL;
}
```

Ngăn xếp (Stack) và hàng đợi (Queue)

Khái niệm hàng đợi

Hàng đợi là danh sách đặc biệt: thêm và xóa phần tử được thực hiện theo nguyên tắc **vào trước ra trước** (FIFO – First In First Out).

Thao tác trên Queue


- **Push(x)** thêm một phần tử x vào Queue
- **Pop(x)** lấy ra phần tử x ra khỏi Queue
- **View(x)** xem phần tử kế tiếp.

Hàng đợi trong thư viện C++

```
#include <queue>          // sử dụng thư viện

queue<int> q;              // khai báo hàng đợi số nguyên

int main()
{
    q.push(10);           // thêm vào hàng đợi
    int x = q.front();    // xem phần tử tiếp theo sẽ lấy ra
    q.pop();              // lấy phần tử ra khỏi hàng đợi
    int n = q.size();     // số lượng phần tử hiện có trong hàng đợi
}
```

 **Bài tập:** Dùng stack, viết chương trình phân tích 1 số thành thừa số nguyên tố theo thứ tự lớn trước nhỏ sau. Ví dụ n = 90, hiển thị: 90= 533*2

```
#include <iostream>
#include <stack>
using namespace std;

void primeFactorization(int n)
{
    stack<int> factors;
    for (int i = 2; i <= n; i++) {
        while (n % i == 0) {
            factors.push(i);
            n /= i;
        }
    }
    cout << n << " = ";
    while (!factors.empty()) {
```

```

        cout << factors.top();
        factors.pop();
        if (!factors.empty()) {
            cout << " * ";
        }
    }

}

int main()
{
    int n;
    cout << "Nhập số nguyên n: ";
    cin >> n;
    primeFactorization(n);
    return 0;
}

```

Cây nhị phân

Định nghĩa

- Cây nhị phân là cây mà mỗi nút có tối đa 2 nút con.
- Hai nút con được phân biệt là nút bên trái và bên phải

Tính chất của cây nhị phân

- Gọi x^m là số lượng các nút ở mức m. Thì $x^m \leq 2^m$
- Gọi y_h là số lượng các nút của cây có chiều cao h . Thì: $y \leq 2^{h-1}$

Biểu diễn cây nhị phân trong máy tính

Dùng con trỏ để biểu diễn mối quan hệ cha - con

```

struct Node
{
    int info;
    Node *left;
    Node *right;
};
typedef Node* Tree;
Tree root;

```

Tương tự như danh sách liên kết

- Các nút của cây là biến cấp phát động.
- Dùng một con trỏ root để trỏ đến nút gốc của cây.

Ví dụ:

```

Node *TaoNut(int x)
{
    Node *n = new Node;
    n->info = x;
    n->left = n->right = NULL;
    return n;
}

void main()
{
    root = TaoNut(3);

    Node *p = TaoNut(6);
    root->left = p;

    p = TaoNut(8);
    root->right = p;

    p = TaoNut(7);
    root->right->right = p;
}

```

Duyệt cây nhị phân

- Duyệt cây là thăm tất cả các nút của cây.
- Các phép duyệt được thực hiện đệ quy

Duyệt theo thứ tự trước NLR (Node – Left – Right)

- Thăm nút gốc
- Duyệt cây con bên trái theo NLR
- Duyệt cây con bên phải theo NLR

```

void DuyetNLR(Tree t)
{
    if (t != NULL)
    {
        cout << t->info;
        DuyetNLR(t->left);
        DuyetNLR(t->right);
    }
}

void main()
{
    DuyetNLR(root);
}

```

Tính số nút

```
int SoNut(Tree t) {
    if (t == NULL)
        return 0;
    else
        return 1 + SoNut(t->left) + SoNut(t->right);
}
```

Tính chiều cao

```
int ChieuCao(Tree t) {
    if (t == NULL)
        return 0;
    int leftHeight = ChieuCao(t->left);
    int rightHeight = ChieuCao(t->right);
    return (leftHeight > rightHeight) ? leftHeight + 1 : rightHeight + 1; //tenary
}
```

Tính số nút lá

```
int SoNutLa(Tree t) {
    if (t == NULL)
        return 0;
    else if (t->left == NULL && t->right == NULL)
        return 1;
    else
        return SoNutLa(t->left) + SoNutLa(t->right);
}
```