

Bài 2: Xây dựng cấu trúc dữ liệu bằng C++

Câu 1. Thực hiện các yêu cầu sau đây:

- Trình bày khái niệm cấu trúc vector.
- Khi xây dựng lớp mẫu **Vector** có dữ liệu gồm **int n, cap; T*buffer;** hãy lập trình các phương thức sau của lớp Vector
 - + Hàm **void Vector<T>::erase(int k)** là xoá phần tử ở vị trí k
 - + Hàm **void Vector<T>::insert(int k, T x)** là thêm phần tử x vào vị trí k

Câu 2. Thực hiện các yêu cầu sau đây:

- Trình bày cấu trúc danh sách liên kết đơn (single list) (vẽ hình minh họa).
- Khi xây dựng lớp mẫu Node có dữ liệu gồm **public: T elem; Node *next;** và lớp mẫu Slist có dữ liệu gồm **int n; Node<T> *Head, *Tail;** hãy lập trình các phương thức sau trong (không sử dụng bộ lặp)
 - + Hàm **void Slist<T>::erase(T x)** xoá một phần tử x (nếu có) ra khỏi Slist.
 - + Hàm **void Slist<T>::push_front(T x)** thêm phần tử vào đầu Slist.

Câu 3. Thực hiện các yêu cầu sau đây:

- Trình bày cấu trúc danh sách liên kết kép (double list) (vẽ hình minh họa).
- Khi xây dựng lớp mẫu Node có dữ liệu gồm **public: T elem; Node *prev, *next;** và lớp mẫu Dlist có dữ liệu gồm **int n; Node<T> *Head, *Tail;** hãy lập trình các phương thức sau (không sử dụng bộ lặp)
 - + Hàm **void Dlist<T>::erase(T x)** xoá một phần tử x (nếu có) ra khỏi Dlist.
 - + Hàm **void Dlist<T>::push_front(T x)** thêm phần tử vào đầu Dlist.

Câu 4. Thực hiện các yêu cầu sau đây:

- Khái niệm cây, trình bày cấu trúc lớp cây tổng quát .
- Khi xây dựng lớp mẫu Node của cây tổng quát gồm **public: T elem; vector<Node<T>*> child;** hãy lập trình các hàm sau:
 - + Hàm **Node<T>*find(Node<T>*H, T x)** tìm giá trị x trong cây H trả về địa chỉ (con trỏ) của Node
 - + Hàm **void insert(Node<T>*H, T x, Node<T> *p)** thêm Node p vào cây H là con của 1 Node có giá trị x (có thể sử dụng hàm find ở trên).

Câu 5. Thực hiện các yêu cầu sau đây:

- Khái niệm cây, trình bày cấu trúc lớp cây tổng quát .
- Khi xây dựng lớp mẫu Node của cây tổng quát gồm **public: T elem; vector<Node<T>*> child;** hãy lập trình các hàm sau:
 - + Hàm **int count(Node<T>*H, T x)** đếm tất cả các Node của cây H chứa x.
 - + Hàm **void high(Node<T>*H)** xác định chiều cao của cây H.

Câu 6. Thực hiện các yêu cầu sau đây:

- Trình bày khái niệm cây nhị phân nói chung.
- Khi xây dựng lớp mẫu **Node** của cây nhị phân gồm **public: T elem;**
Node<T>*left,*right; hãy lập trình các hàm sau:
 - + Hàm **Node<T>*find(Node<T>*H, T x)** tìm giá trị x trong cây H trả về Node tìm được
 - + Hàm **int count(Node<T>*H, T x)** đếm số Node có giá trị bằng x trong cây H.

Câu 7. Thực hiện các yêu cầu sau đây:

- Trình bày khái niệm ngăn xếp (stack), trình bày cấu trúc dữ liệu stack, vẽ hình minh họa.
- Trình bày thuật toán (không viết code) chuyển đổi biểu thức dạng trung tố sang dạng hậu tố; mô phỏng chuyển sang hậu tố biểu thức $(3+5)*(4+(3*(6-(2+5)*3)+7)*2)$

Câu 8. Thực hiện các yêu cầu sau đây:

- Trình bày khái niệm hàng đợi (Queue), trình bày về cấu trúc dữ liệu queue, vẽ hình minh họa.
- Sử dụng cấu trúc dữ liệu **hàng đợi** trong thư viện STL có sẵn của C++ giải bài xếp búp bê Nga như sau: có n con búp bê có chiều cao là $a_1 \dots a_n$ đã sắp giảm dần nếu 2 con búp bê có chiều cao chênh ít nhất khoảng cách 3 thì sẽ nuốt nhau. Hãy tìm cách lồng nhiều búp bê nhất bằng cách duyệt lần lượt chiều cao đưa vào queue nếu con đứng đầu nuốt được con mới thêm thì bỏ con đứng đầu khỏi queue, sau khi duyệt hết size của queue chính là số búp bê ít nhất.