

Bài thực hành số 1

Bài tập 1:

Viết chương trình minh họa các giải thuật tìm kiếm và sắp xếp trên mảng có kích thước n phần tử nguyên. Chương trình được mô tả với các yêu cầu như sau:

🔧 Cài đặt hàm tìm kiếm:

- Tìm kiếm tuần tự (tuyến tính) cho mảng bất kỳ
- Tìm kiếm nhị phân cho mảng dữ liệu được sắp tăng

🔧 Cài đặt các hàm sắp xếp (tăng) theo phương pháp:

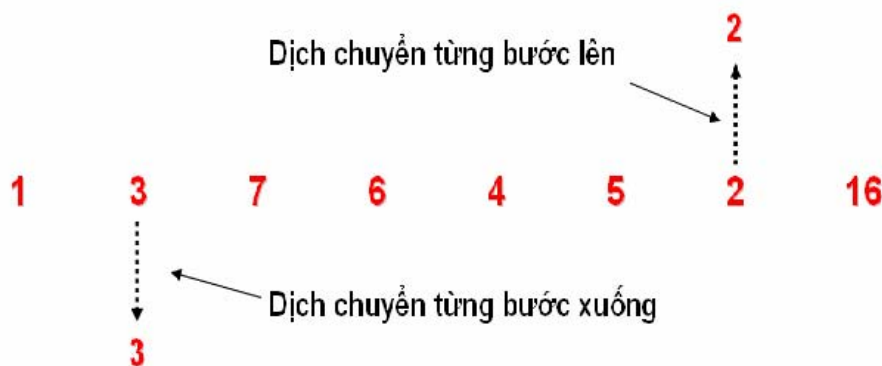
- Chọn trực tiếp
- Chèn trực tiếp
- Nổi bọt
- Đổi chỗ trực tiếp
- Shell Sort
- Quick Sort

Các hàm sắp xếp phải minh họa trực quan: tại mỗi bước hoán vị $a[i]$ và $a[j]$

Ví dụ: hoán vị $a[1] = 3$ và $a[6] = 2$

1 3 7 6 4 5 2 16

Bước 1: di chuyển 3 xuống dưới k dòng và 2 lên trên k dòng



Bước 2: di chuyển 3 qua vị trí cột của 2 và ngược lại

Dịch chuyển qua trái

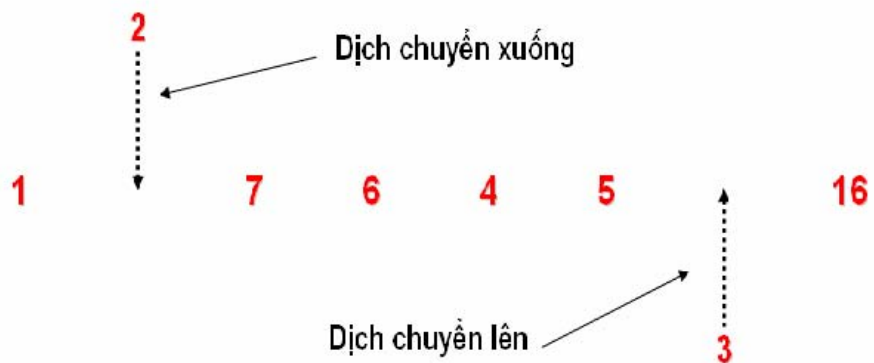
2 ← 2

1 7 6 4 5 16

3 3

Dịch chuyển qua phải

Bước 3: di chuyển 2 xuống và 3 lên đúng vị trí cuối cùng



Trường hợp sắp xếp theo kiểu chèn hơi khác, sinh viên tự tìm hiểu và cài đặt minh họa cho thuật toán sắp xếp chèn.

Lưu ý: mỗi hàm sẽ có tham số vào là mảng cần xử lý và kích thước của mảng, không dùng biến toàn cục.

- ✚ Trong hàm main xây dựng một menu chọn, cho phép nhập vào một số rồi thực hiện chức năng tương ứng, menu có mô tả như sau:
 - Khởi tạo mảng dữ liệu a, cho phép nhập vào n, sau đó chương trình phát sinh ngẫu nhiên các phần tử cho mảng a
 - Xem phần tử của mảng
 - Tìm kiếm tuần tự
 - Tìm kiếm nhị phân
 - Sắp xếp chọn
 - Sắp xếp chèn
 -
 - 11. Sắp xếp theo Radix Sort
 - 12. Thoát

VD: khi user nhập 1 thì tạo mảng, nhập 2 xem các phần tử của mảng, còn 11 thực hiện sắp xếp theo Radix Sort. Khi user nhập 12 thì kết thúc chương trình!

Lưu ý: khi chọn chức năng tìm kiếm nhị phân, thì phải kiểm tra xem mảng đã được sắp tăng chưa, SV tự cài đặt chức năng kiểm tra này.

✂ Hướng dẫn

- ✚ Phần định nghĩa các hằng số dùng trong chương trình

```
#define MAX 50
#define COL 10
#define ROW 10
#define VER 4
#define WAIT 100
#define SPACING 8
```

- ✚ Hàm xuất mảng

```
void ShowArray(int a[], int n)
{
    for(int i=0; i < n; i++)
    {
        gotoxy(i*SPACING+COL, ROW);
        cprintf("%d", a[i]);
    }
}
```

- ✚ Hàm hoán vị minh họa di chuyển từng bước

```

void Swap(int a[], int n, int i, int j)
{
    int ai = a[i], aj=a[j];

    int xi = i*SPACING+COL, xj=j*SPACING+COL;

    int d=0;
    // di chuyen ai xuong va aj len UER don vi
    while (d<UER)
    {
        gotoxy(xi, ROW+d); cprintf("    ");
        gotoxy(xj, ROW-d); cprintf("    ");
        d++;
        gotoxy(xi, ROW+d); cprintf("%d",ai);
        gotoxy(xj, ROW-d); cprintf("%d",aj);
        delay(WAIT);
    }
    // bat dau dich chuyen
    int xii = xi, xjj = xj;
    while (xj>xii || xi<xjj)
    {
        gotoxy(xi, ROW+d); cprintf("    ");
        gotoxy(xj, ROW-d); cprintf("    ");
        xj--; xi++;
        gotoxy(xi, ROW+d); cprintf("%d",ai);
        gotoxy(xj, ROW-d); cprintf("%d",aj);
        delay(WAIT);
    }
    // di chuyen ai len va aj xuong UER don vi
    while (d>0)
    {
        gotoxy(xi, ROW+d); cprintf("    ");
        gotoxy(xj, ROW-d); cprintf("    ");
        d--;
        gotoxy(xi, ROW+d); cprintf("%d",ai);
        gotoxy(xj, ROW-d); cprintf("%d",aj);
        delay(WAIT);
    }
    // hoan vi a[i] va a[j];
    a[i] = aj;
    a[j] = ai;
}

```

🔧 Hàm bubble sort sử dụng Swap trực quan trên

```

void BubbleSort(int a[], int n)
{
    for(int i=0; i < n-1; i++)
        for(int j=n-1; j>i; j--)
            if (a[j-1]> a[j])
                Swap(a,n,j-1,j);
}

```

🔧 Minh họa 1 phần của hàm main()

```

clrscr();
textcolor(CYAN);
_setcursortype(_NOCURSOR);

ShowArray(a, n);
BubbleSort(a,n);

getch();

```

Gợi ý chương trình:

// nạp thư viện

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

// định nghĩa hằng

#define MAX 100

#define MAXV 100

// định nghĩa các hàm

// các hàm có prototype như sau:

// hàm nhập dãy số ngẫu nhiên

void nhap(int a[], int &n);

// hàm hiển thị dãy số

void hienThi(int a[], int n);

```

//ham tim kiem su dung giai thuat tim kiem nhi phan
//tra lai ket qua -1 khong tim thay
//ngoai ra la vi tri tim duoc.
int timKiemTuyenTinh(int a[], int n, int x);
//ham tim kiem nhi phan
int timKiemNhiPhan(int a[], int n, int x);
//ham sap xep day so tang dung giai thuat sap xep noi bot
void sapXepNoiBot(int a[], int n);
//ham sap xep doi cho truc tiep
void sapXepDoiChoTrucTiep(int a[], int n);
//ham sap xep chen
void sapXepChen(int a[], int n);
//ham sap xep chon truc tiep
void sapXepChon(int a[], int n);
//ham sap xep shell sort
void shellSort(int a[], int n);
//ham sap xep nhanh
void quickSort(int a[], int n);

void main(){
    //khai bao mang a luu day so ngau nhien
    int a[MAX];
    in n;
    int chon;
    //hien thi menu
    do
    {
        //xoa man hinh
        clrscr();
        //hien thi menu
        printf("1: Tao day cac so ngau nhien");
        printf("2: Tim kiem tuyen tinh ");
        printf("3: Sap xep chon");
        printf("4: Sap xep doi cho truc tiep");
        printf("5: Sap xep noi bot");
        printf("6: Sap xep chen");
        printf("7: Shell Sort");
        printf("8: Quick Sort");
        printf("9: Tim kiem nhi phan");
        printf("10: Hien thi day so");
        printf("0 : Thoát");
        //doc chon lua cua nguoi dung
        scanf("%d",&chon);
        //xu ly lua chon tren menu
        switch (chon){
            case 1 :
                //goi ham nhap day so ngau nhien
                nhap(a,n);
                break ;
            case 2 :
                //thong bao nhap vao so can tim
                printf("Cho biet so can tim:");
                int x;
                scanf("%d", &x);
                //goi ham tim kiem tuyen tinh
                int i=timKiemTuyenTinh(a,n,x);
                //neu khong tim thay

```

```

        if (i== -1)
            printf("Khong tim thay %d trong day so",x);
        else
            printf("Tim thay %d o vi tri %d trong day so",x,i);
        //dung man hinh de xem ket qua
        getch();
        break ;
    case 3 :
        //goi ham sap xep chon
        break ;
    case 4 :
        //goi ham sap xep doi cho truc tiep
        break ;
    case 5 :
        //goi ham sap xep noi bot
        break ;
    case 6 :
        //goi ham sap xep chen
        break ;
    case 7 :
        //goi ham Shell Sort
        break ;
    case 8 :
        //goi ham Quick Sort
        break ;
    case 9:
        //goi ham tim kiem nhi phan
        break;
    case 10:
        //goi ham hien thi day so
        break;
    default :
        chon=0 ;
    }

} while (chon);

}

```

Phần còn lại sinh viên tự cài đặt! Mọi thắc mắc email về: vanthienhoang@yahoo.com.vn



Bài thực hành số 2

Singly Linked List – Danh sách liên kết đơn



Bài tập 2:

Minh họa chương trình quản lý sách đơn giản trong thư viện. Sử dụng cấu trúc dữ liệu danh sách liên kết đơn để cài đặt danh sách chứa nội dung các cuốn sách.

A. Thông tin liên quan đến một cuốn sách gồm:

- Mã số sách
- Tên sách
- Tên tác giả
- Nhà xuất bản
- Năm xuất bản
- Trạng thái sách: {TRUE: chưa mượn/ FALSE: đã mượn}

B. Các thao tác trên danh sách này:

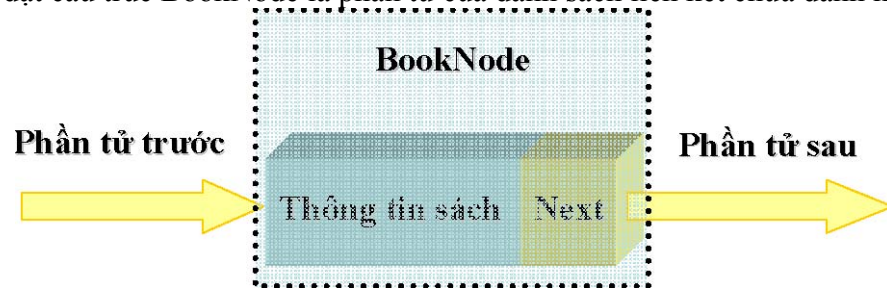
1. Khởi tạo danh sách
 - a. Khởi tạo danh mục sách rỗng (chưa có sách)
 - b. Đọc từ file: nhập vào tên file đã lưu danh mục sách ở lần làm việc trước đó.
2. Thêm một cuốn sách vào danh sách
 - a. Thêm vào đầu danh sách: InsertFirst
 - b. Thêm vào sau một cuốn sách nào đó: InsertAfter
 - c. Thêm vào cuối của danh sách: InsertLast
3. Xóa một cuốn sách khỏi danh sách theo
 - a. Mã số
 - b. Tên sách
 - c. Tên tác giả: (nếu tác giả có nhiều sách thì xóa hết)
 - d. Xóa cuốn sách ở đầu danh sách
 - e. Xóa cuốn sách ở sau cuốn sách có mã số nào đó
 - f. Xóa cuốn sách ở cuối danh sách
4. Tìm kiếm sách theo: lưu ý kết quả có bao nhiêu thì liệt kê ra hết, không phải chỉ liệt kê kết quả đầu tiên
 - a. Tên sách
 - b. Tên tác giả
 - c. Tên nhà xuất bản
5. Chức năng xem danh sách
 - a. Xem toàn bộ danh mục sách trong danh sách
 - b. Xem những cuốn sách đang cho mượn
 - c. Xem những cuốn sách chưa cho mượn
 - d. Xem danh sách theo thứ tự:
 - i. Theo vần alphabet của tên sách
 - ii. Theo vần alphabet của tên tác giả
 - iii. Theo vần alphabet của tên nhà xuất bản
 - iv. Sách được xuất bản mới nhất (theo năm)
6. Chức năng mượn/trả sách

- a. Mượn sách: liệt kê những cuốn sách chưa cho mượn, cho user chọn một cuốn sách \Rightarrow rồi cập nhật lại trạng thái cho mượn của sách.
 - b. Trả sách: nhập vào mã số sách được trả \Rightarrow cập nhật lại trạng thái đã trả sách cho cuốn sách đó.
7. Chức năng chỉnh sửa nội dung của sách: cho phép chọn các thông tin của sách để sửa và sau đó cập nhật lại.
 8. Chức năng lưu file: nhập vào một tên file rồi lưu toàn bộ trạng thái hiện tại của danh mục sách vào đó.

Yêu cầu:

Sinh viên phải thực hiện theo các yêu cầu sau:

1. Cài đặt cấu trúc dữ liệu Book theo mô tả như phần A
2. Cài đặt cấu trúc BookNode là phần tử của danh sách liên kết chứa danh mục sách.



Hình: Minh họa cấu trúc của phần tử trên danh sách liên kết

3. Cài đặt toàn bộ các chức năng mô tả trong phần B từ 1 \rightarrow 8: thể hiện các chức năng này theo menu chọn.

Ngoài ra sinh viên có thể bổ sung những chức năng mở rộng tùy ý. Tất cả các chức năng nâng cao này đều được đánh giá cao!

Mọi thắc mắc email về: vanthienhoang@yahoo.com.vn





Bài thực hành số 3

Stack - Queue



Bài tập 3.1:

Viết chương trình tính giá trị biểu thức trung tố theo các yêu cầu sau:

1. Nhập biểu thức trung tố: toán hạng, toán tử và dấu ngoặc

VD: $(20+5)/5+(7-3)*100$

2. Chuyển biểu thức trung tố thành hậu tố (xuất ra màn hình)

VD: $20\ 5\ +\ 5\ /\ 7\ 3\ -\ 100\ *\ +$

3. Tính giá trị của biểu thức hậu tố

VD: $(20+5)/5+(7-3)*100 = 405$

Yêu cầu:

Sinh viên cài đặt stack dùng danh sách liên kết:

1. Khai báo cấu trúc của phần tử trong DSLK dùng làm stack
2. Cài đặt các thao tác: IsEmpty, NewNode, FreeNode, **Pop**, **Push**... trên Stack.

Hướng dẫn:

1. Chuyển biểu thức trung tố thành hậu tố:

🚦 Duyệt biểu thức trung tố từ trái sang phải

- ❖ Nếu gặp toán hạng thì ghi vào chuỗi kết quả
- ❖ Nếu gặp dấu mở ngoặc thì push \Rightarrow stack
- ❖ Nếu gặp toán tử gọi là O_1 thực hiện các bước sau:
 - Chừng nào còn một toán tử O_2 ở đỉnh stack và độ ưu tiên của $O_1 \leq$ độ ưu tiên O_2 thì lấy O_2 ra khỏi stack và ghi vào chuỗi kết quả.
 - Push $O_1 \Rightarrow$ stack

- ❖ Nếu gặp dấu đóng ngoặc: thì lấy toán tử trong stack ra cho đến khi lấy được dấu mở ngoặc (lưu ý: pop dấu mở ngoặc ra, nhưng ko xuất ra chuỗi kết quả)
- ✚ Khi đã duyệt hết biểu thức trung tố, lấy tất cả toán tử trong stack và ghi vào chuỗi kết quả.

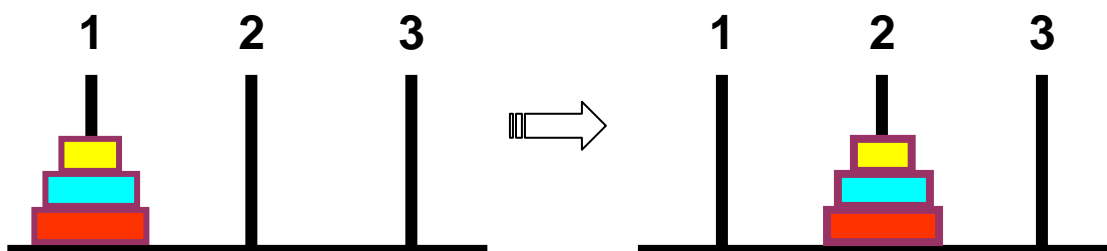
2. Tính giá trị biểu thức hậu tố:

- ✚ Đọc biểu thức từ trái sang phải
 - ❖ Nếu là toán hạng: Push \Rightarrow stack
 - ❖ Nếu gặp toán tử:
 - Lấy 2 toán hạng trong stack ra
 - Tính giá trị của 2 toán hạng đó theo toán tử
 - Push kết quả \Rightarrow stack
- ✚ Khi quá trình kết thúc thì con số cuối cùng còn lại trong stack chính là giá trị của biểu thức đó.

Bài tập 3.2:

Bài toán Tháp Hanoi được mô tả như sau: cho 3 cột được đánh số lần lượt là 1, 2 và 3. Có n đĩa được sắp theo thứ tự đĩa nhỏ ở bên trên đĩa lớn. Hãy liệt kê các bước thực hiện để chuyển tất cả các đĩa từ cột 1 sang cột 2. Quy luật di chuyển như sau:

1. Mỗi bước chỉ di chuyển 1 đĩa từ cột này sang cột khác.
2. Đĩa có bán kính nhỏ luôn sắp trên đĩa có bán kính lớn.



Yêu cầu:

Viết chương trình nhập vào số đĩa n, thực hiện các bước di chuyển các đĩa, mỗi bước di chuyển cho biết cột nguồn (cột lấy đĩa) và cột đích (cột đặt đĩa). Giải thuật di chuyển không đệ quy, dùng stack để chứa thông tin tạm thời trong quá trình di chuyển.

Sinh viên cài đặt stack dùng danh sách liên kết, mỗi node phần info chứa 3 thông tin {số đĩa di chuyển, cột nguồn, cột đích}.

Hướng dẫn:

Như chúng ta biết bài toán tháp Hanoi thường được giải bằng phương pháp đệ quy. Tuy nhiên có thể giải bằng cách dùng stack để khử đệ quy. Để thực hiện việc lưu trữ tạm trong quá trình di chuyển chúng ta dùng một stack. Trong đó mỗi phần tử của stack này chứa các thông tin gồm: số đĩa di chuyển (**N**), cột nguồn bắt đầu di chuyển (**Nguồn**) và cột đích là nơi cần di chuyển đến (**Đích**). Ở đây không cần lưu cột trung gian vì có 3 cột đánh số là 1, 2 và 3 thì cột trung gian để di chuyển là: $6 - (Nguồn + Đích)$.

Đầu tiên đưa vào stack thông tin di chuyển {n, 1, 2}, tức là di chuyển n đĩa từ cột 1 sang cột thứ 2 qua cột trung gian là $6 - (1 + 2) = 3$.

Tại mỗi bước khi **lấy trong stack ra một phần tử**, chúng ta thực hiện như sau:

- Nếu $N = 1$: \Rightarrow di chuyển đĩa từ cột Nguồn \rightarrow cột Đích
- Ngược lại (nếu $N > 1$):
 - Xác định cột trung gian $TrungGian = 6 - (Nguồn + Đích)$
 - Push \Rightarrow stack thông tin di chuyển {N-1, TrungGian, Đích}
 - Push \Rightarrow stack thông tin di chuyển {1, Nguồn, Đích}
 - Push \Rightarrow stack thông tin di chuyển {N-1, Nguồn, TrungGian}

Quá trình còn thực hiện khi stack khác rỗng.

Nhận xét: Lưu ý thứ tự khi đưa vào thông tin di chuyển vào stack. Trong phần trên thông tin {N-1, Nguồn, TrungGian} được đưa vào stack sau cùng nên chúng sẽ được lấy ra trước tiên, kế đến là thông tin di chuyển {1, Nguồn, Đích} và cuối cùng là thông tin di chuyển {N-1, TrungGian, Đích}.

Bài tập 3.3:

Viết chương trình quản lý kho đơn giản thực hiện các chức năng sau:

1. Cho phép thêm một mặt hàng vào kho
2. Xuất một mặt hàng ra khỏi kho

3. Xem tất cả hàng hoá trong kho
4. Xem mặt hàng nào kế tiếp sẽ được xuất kho

Yêu cầu

1. Cài đặt cấu trúc dữ liệu HàngHoá: có các dữ liệu nào liệt kê ra
2. Cài đặt một Queue chứa các hàng hoá trong kho
3. Cài đặt các thao tác trên Queue
4. Cài đặt các chức năng theo mô tả của bài tập.

🚦 Thời gian làm bài tập 3: từ

Ngoài ra sinh viên có thể bổ sung những chức năng mở rộng tùy ý. Tất cả các chức năng sáng tạo của sinh viên đều được đánh giá cao!

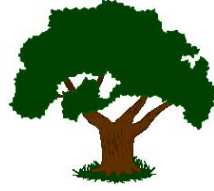
Mọi thắc mắc email về: vanthienhoang@yahoo.com.vn



Môn: CTDL & GT

Bài thực hành số 4

Cây nhị phân tìm kiếm




Bài tập

Viết chương trình quản lý lịch công tác trong tháng đơn giản: cho phép nhập vào nội dung công việc cần làm theo ngày, theo giờ. Trong một ngày có thể có nhiều công việc, mỗi công việc có **giờ bắt đầu**, tên công việc, nội dung công việc, tính chất công việc {rất quan trọng, quan trọng, bình thường, ko cần thiết}...

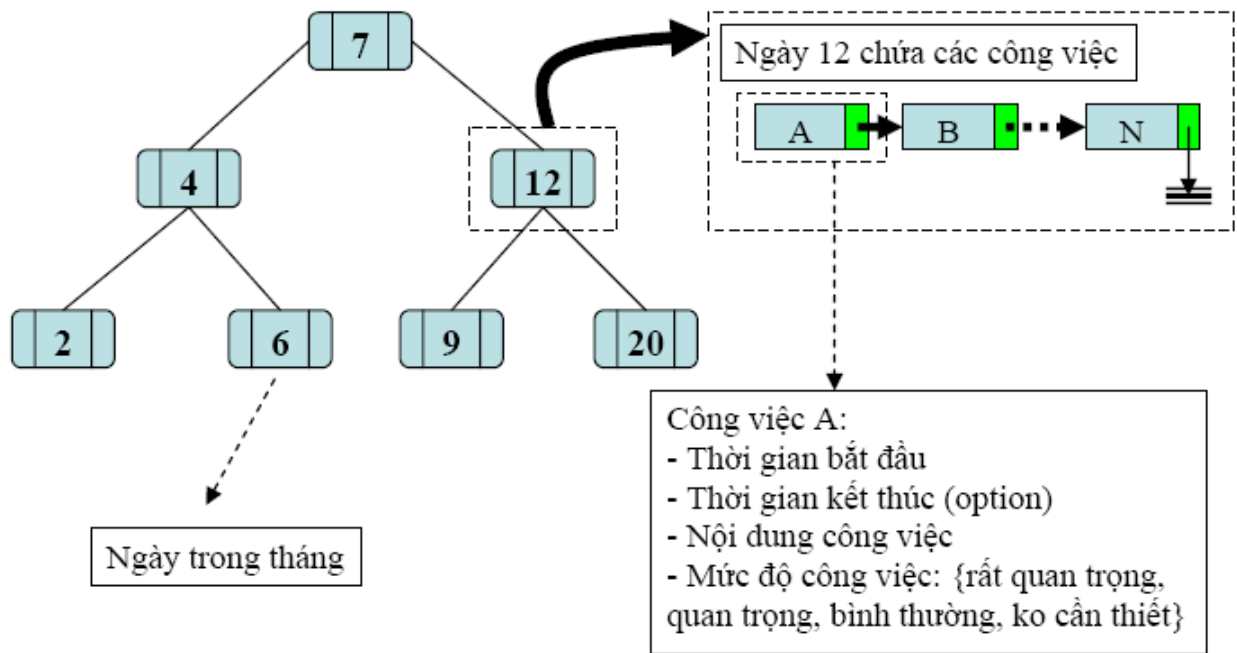
Chương trình có các chức năng chính như sau:

- Nhập nội dung công việc cần làm theo ngày, theo giờ
- Xem lịch công tác theo ngày yêu cầu
- Xem các công việc theo tính chất: rất quan trọng, quan trọng...
- Xem các công việc đã hoàn tất
- Xem các công việc chưa thực hiện
- Xem các công việc từ ngày a đến ngày b
- Xóa hay điều chỉnh lịch công tác. Nếu sau khi điều chỉnh, ngày nào không còn việc phải làm sẽ xóa khỏi lịch công tác.

 **Yêu cầu**: chương trình có cài đặt cây nhị phân tìm kiếm (BST):

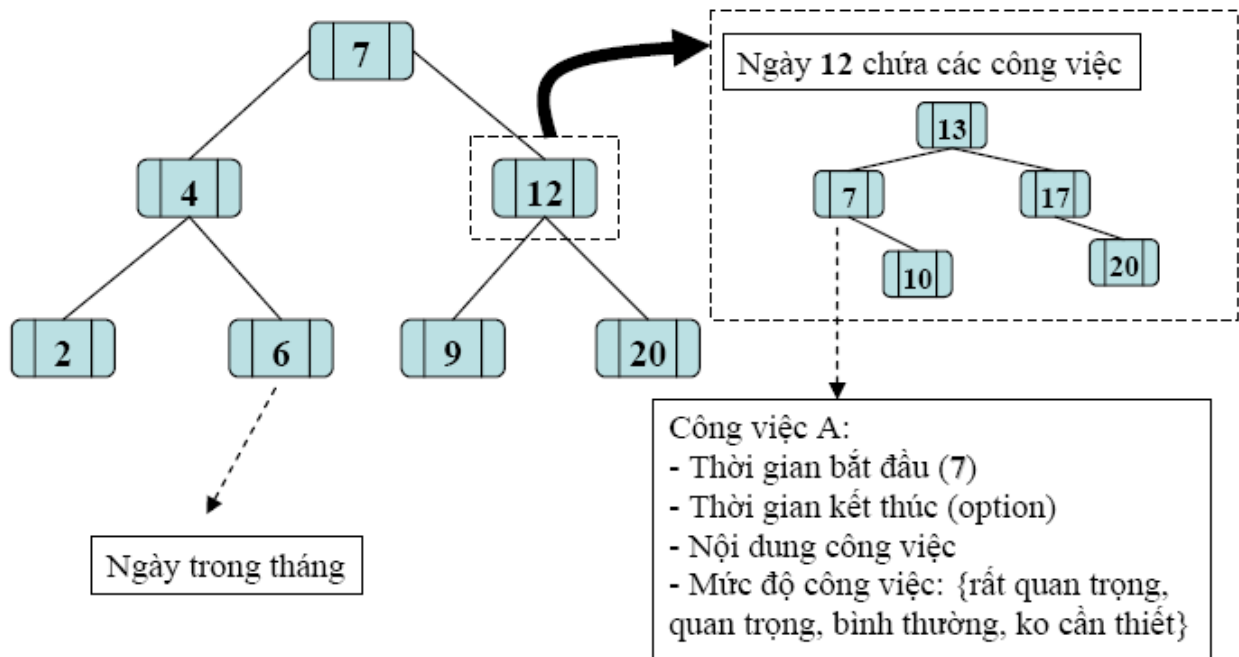
- Mỗi nút trên cây BST là một ngày của lịch công tác
- Trong mỗi nút ngày trên cây lại chứa một **danh sách liên kết** lưu thông tin các công việc.
- Khi thêm một công việc vào một ngày đã tồn tại trên cây, thì công việc này sẽ được đưa vào danh sách liên kết chứa các công việc theo thứ tự tăng dần của **giờ bắt đầu**.

Hình vẽ minh họa cấu trúc cây lịch công tác



Hình 1: Cấu trúc cây lịch công tác

🚦 Nâng cao (**không bắt buộc, dành cho sinh viên khá, giỏi**)
Thay danh sách liên kết chứa công việc trong ngày thành cây nhị phân tìm kiếm, khóa để xây dựng cây BST con là giờ bắt đầu!



Hình 2: Cấu trúc cây lịch công tác nâng cao

🚦 Tất cả những chức năng sáng tạo của SV đều được đánh giá cao!