THỰC HÀNH 2

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Bài 2. GIẢI THUẬT SẮP XẾP

I. Mục tiêu

Sau khi thực hiện xong bài thực hành, sinh viên sẽ thực hiện được:

- Cài đặt được các giải thuật sắp xếp trên mảng một chiều các số nguyên.
- Vận dụng được các giải thuật sắp xếp vào một bài toán cụ thể.

II. Tóm tắt

1. Giải thuật Insertion Sort

Giải thuật	Cài đặt
InsertionSort(int A[], int n){ **// Bước 1:** **i = 2;** **// Bước 2:** Tìm vị trí **pos** thích hợp trong đoạn **A[1]** → **A[i-1]** để chèn **A[i]** vào **// Bước 3:** Dời chỗ các phần tử từ **A[pos]** → **A[i-1]** sang phải 1 vị trí để dành chỗ cho **A[i]** **// Bước 4:** **A[pos] = A[i];//Đoạn A[1] → A[i] đã được sắp** **// Bước 5:** i = i + 1; **// xét phần tử kế tiếp** Nếu i < n : Lặp lại bước 2 Ngược lại : Dừng. }	**void InsertionSort (int a[], int n) {** **int pos, i, x; // x lưu tạm a[i] để tránh ghi đè** **for (int i = 1; i < n; i++) {** **x = a[i];** **pos = i-1;** **while((pos > 0) && (a[pos] > x)) {** **a[pos+1] = a[pos]; //dời qua phải 1 vị trí** **poss--;** **}** **a[pos] = x; // chèn a[i] vào đúng vị trí** **}** **}**

2. Giải thuật Interchange Sort

Giải thuật	Cài đặt
Bước 1: i = 1; **//bắt đầu từ đầu dãy** **Bước 2**: j = i + 1;**//chuẩn bị so sánh a[i] với a[j]** **Bước 3**: (Lặp) Trong khi **j < n** thực hiện: Nếu a[i] < a[j] thì **hoán_vị**(a[i], a[j]); Nếu i < n: Lặp lại Bước 2 Ngược lại: Dừng j = j + 1; **// so sánh a[i] với phần tử kế** **Bước 4**: i = i + 1 ;	**void InterChangeSort (int a[], int n) {** **int x, tam; // x lưu tạm a[i]** **for (int i = 0; i < n-1; i++)** **for (int j = i+1; j < n; j++)** **if (a[i] > a[j]) { // hoán đổi a[i] với a[j]** **tam = a[i];** **a[i] = a[j];** **a[j] = tam;** **}** **}**

3. Giải thuật Bubble Sort

Giải thuật	Cài đặt
Bước 1: i = 1; // Lần xử lý đầu tiên **Bước 2**: j = n; // Duyệt từ cuối dãy về vị trí i Trong khi (j > i) Nếu a[j] < a[j-1] thì Hoán vị (a[j], a[j-1]) j = j-1; // Xét tiếp phần tử kế bên trái **Bước 3**: i = i + 1; // Xử lý lần kế tiếp Nếu i > n-1 : Hết dãy → Dừng. Ngược lại : Lặp lại bước 2.	void BubbleSort (int a[], int n) { int tam; // x lưu tạm a[i] for (int i = 0; i < n; i++) for (int j = n-1; j > i; j--) if (a[j] < a[j-1]) { //hoán đổi a[i] với a[j] tam = a[j]; a[j] = a[j-1]; a[j-1] = tam; } }

4. Giải thuật Merge Sort

Giải thuật: Cho A[n], B[m], trộn thành C[n+m]

Bước 1: bắt đầu: i=0, j=0, k=0;
// i:chỉ số mảng A, j:chỉ số mảng B,
// k:chỉ số mảng C

Bước 2: Trong khi i<m và j<n, so sánh:
if (A[i]<A[j])
C[k]=A[i];
i++; // Xét phần tử kế của A
k++; // Chuyển tới phần tử kế của C
Nếu i<m và j<n thì: Lặp lại bước 2.

Bước 3: // 1 trong 2 mảng A, B đã hết
Chép phần còn lại của mảng chưa hết vào cuối mảng C.

Cài đặt:

```
void  MergeSort (int a[], int m, int b[], int n) {
    int  i=0, j=0, k=0; // các chỉ số của A[], B[], C[]
    while((i<m) && (j<n)) {
        if(a[i] < b[j]) {
            c[k] = a[i];   //chuyển phần tử từ A vào C
            i++;           // Xét phần tử kế trong A
        }
        else {
            c[k] = b[j];   //chuyển phần tử từ B vào C
            j++;           // Xét phần tử kế trong B
        }
        k++;               // Đến phần tử kế trong C
    } // end while
```

```
    //if(i == m)  tiếp tục trang trước
    if (i == m)  // mảng A hết, chuyển mảng B vào C
         for(int x = j; x < n; x++) {
            c[k] = b[x];
            k++;
         }
    if (j == n) // mảng B hết, chuyển mảng A vào C
         for(int x = i; x < m; x++) {
            c[k] = a[x];
            k++;
         }
}
```

5. Giải thuật Quick Sort

Giải thuật	Cài đặt
Bước 1: chọn x=a[(n+1)/2] làm phần tử chốt. Hai bên x là 2 phân đoạn A1 và A2. **Bước 2**: Duyệt từ hai đầu của dãy: i++ và j--. Nếu gặp một cặp **a[i] ≥ x ≥ a[j]** thì hoán vị hai phần tử này. **Bước 3**: Tăng i=i+1, giảm j=j-1 **Bước 4**: Lặp lại bước 2 cho đến khi i > j **Bước 5**: Lặp lại từ bước 1 đến bước 4 với hai phân đoạn A1 và A2 Kết thúc khi tất cả các phân đoạn thu được có chiều dài là 1.	`void QuickSort (int a[], int l, int r) {` `int i, j, x, tam;` `x = a[(l+r)/2]; // phần tử chốt: giữa mảng` `i = l; j = r;` `do {` `while(a[i] < x) i++; // di chuyển từ trái qua` `while(a[j] > x) j--; // di chuyển từ phải qua` `if(i <= j) { // hoán vị a[i], a[j]` `tam = a[i];` `a[i] = a[j];` `a[j] = tam;` `i++;` `j--;` `}` `while(i < j);` `} // end do` `...` `if(l < j)` `QuickSort(a, l, j);` `if(i < r)` `QuickSort(a, i, r);` `} // END FUNCTION`

III. Nội dung thực hành trên lớp

Bài 1. Viết chương trình cho phép nhập vào một mảng các số nguyên (tối đa 30 phần tử) và thực hiện các yêu cầu sau:

1. Xuất mảng đã nhập ra màn hình.
2. Vận dụng giải thuật Insertion sort để sắp xếp mảng tăng dần.
3. Vận dụng giải thuật Interchange sort để sắp xếp mảng giảm dần.
4. Vận dụng giải thuật Bubble Sort để sắp xếp mảng tăng dần.
5. Vận dụng giải thuật Quick Sort để sắp xếp mảng tăng dần.
6. Vận dụng giải thuật Merge Sort để sắp xếp mảng giảm dần.

➢ Dưới đây là đoạn mã lệnh xây dựng menu chương trình cho phép nhập xuất một mảng. Sinh viên hãy xây dựng thêm các hàm cần thiết để thực thi các yêu cầu trên.

```
#include "conio.h"
#include "stdio.h"

const int KTM=30; //toi da 30 phan tu
//khai bao ham con
void NhapMang(int a[], int &n);
void XuatMang(int a[], int n);
void Sap_xep(int a[], int n);
int  LinearSearch(int a[], int n, int x) ;
void menu();

int main()
{
    int mang[KTM], sophantu;
    int chon;
    int x,vt;
    NhapMang(mang,sophantu);
    do{
        //hien thi menu
        menu();
        printf("\nNhap 1 so de chon...");
        scanf("%d",&chon);
       //xet gia tri cua bien chon de goi ham tuong ung
       switch(chon)
       {
       case 0:
           printf("\nThe end..... Good bye!!!!\n");
           break;
       case 1:
           NhapMang(mang,sophantu);
           break;
       case 2:
           printf("\nGia tri cua mang la:\n");
           XuatMang(mang,sophantu);
           break;
       case 3:
           printf("\nNhap gia tri can tim:");
           scanf("%d",&x);
           vt=LinearSearch(mang,sophantu,x);
           if(vt==-1)
               printf("\nKhong tim thay %d trong mang.\n",x);
           else
               printf("\nTim thay %d o vi tri %d trong mang.\n",x,vt);
```

```
            break;
        case 4:
            Sap_xep(mang,sophantu);
            printf("\nMang sau khi sx la: ");
            XuatMang(mang,sophantu);
            break;
        }
    }while(chon!=0);
    return 1;
}
void NhapMang(int a[], int &n)
{
    do{
        printf("\nNhap so phan tu mang :");
        scanf("%d",&n);
    }while(n<=0 ||n>KTM);
    //nhap gia tri cho cac phan tu mang
    for(int i=0;i<n;i++)
    {
        printf("\nNhap phan tu thu %d : ",i);
        scanf("%d",&a[i]);
    }
}
void XuatMang(int a[], int n)
{
    printf("\n");
    for(int i=0;i<n;i++)
        printf("%3d",a[i]);
    printf("\n");
}
void Sap_xep(int a[], int n)
{
    int  csmin;
    // chỉ số phần tử có giá trị nhỏ nhất
    for (int i = 0; i < n; i++)
    {
        csmin = i;
        for (int  j = i+1; j < n; j++)
            if (a[j] < a[csmin])
                csmin = j;
        // ghi nhận vị trí ptử nhỏ nhất
        // Hoán vị
        int tam = a[csmin];
```

```
87          a[csmin] = a[i];
88          a[i] = tam;
89      } //ket thuc for i
90  }//ket thuc ham
91
92  int  LinearSearch(int a[], int n, int x)
93  {
94      int  i = 0;
95      while ((i < n) && (a[i] != x))
96          i++;
97      if (i == n)
98          return -1; // Tìm hết mảng, không thấy
99      else
100         return i;  // a[i] là phần tử có khóa x
101 }
102
103 void menu()
104 {
105         printf("\n1. Nhap mang");
106         printf("\n2. Xuat mang");
107         printf("\n3. Tim kiem tuyen tinh");
108         printf("\n4. Sap xep chon");
109         printf("\n5. Sap xep chen");
110         printf("\n6. Quick Sort");
111         printf("\n0. Thoat");
112 }
```

Bài 2. Thông tin của một sản phẩm gồm có: mã sản phẩm (kiểu chuỗi 5 kí tự), tên sản phẩm (kiểu chuỗi 50 kí tự), đơn giá (số thực), số lượng (số nguyên). Hãy viết chương trình thực hiện các yêu cầu sau:

1. Nhập một danh sách các sản phẩm (tối đa 40 sản phẩm).
2. Xuất danh sách đã nhập.
3. Vận dụng giải thuật Selection Sort, sắp xếp danh sách sản phẩm tăng dần theo số lượng.
4. Vận dụng giải thuật Insertion Sort, sắp xếp danh sách sản phẩm tăng dần theo đơn giá.
5. Vận dụng giải thuật Buble Sort, sắp xếp danh sách sản phẩm tăng dần theo tên sản phẩm.

Yêu cầu: xây dựng menu chương trình.

IV. Bài tập về nhà

Bài 3. Thông tin của một đầu sách tại thư viện gồm có: mã sách (kiểu chuỗi 10 kí tự), tên sách (kiểu chuỗi 50 kí tự), đơn giá (số thực), số lượng (số nguyên), tên tác giả (kiểu chuỗi 50 kí tự), nhà xuất bản (kiểu chuỗi 50 kí tự). Hãy viết chương trình thực hiện các yêu cầu sau:

1. Nhập một danh sách các đầu sách (tối đa 30 đầu sách).
2. Xuất danh sách các đầu sách có trong danh sách.
3. Xuất danh sách các đầu sách khi biết tên tác giả.

4. Sắp xếp danh sách đầu sách tăng dần theo số lượng.
5. Sắp xếp danh sách đầu sách giảm dần theo đơn giá.
6. Sắp xếp danh sách đầu sách tăng dần theo tên tác giả.
7. Tính tổng giá trị của các đầu sách khi biết tên nhà xuất bản.

----------Hết----------