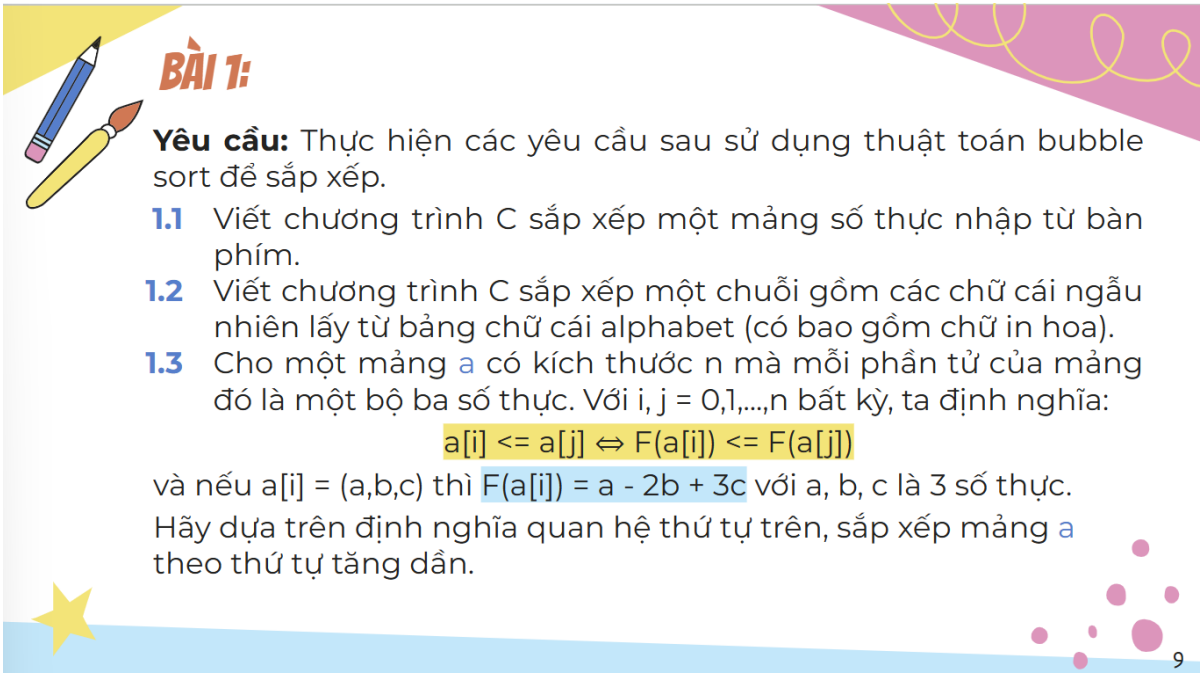
Bài 1:



BÀI 1:

Yêu cầu: Thực hiện các yêu cầu sau sử dụng thuật toán bubble sort để sắp xếp.

1.1 Viết chương trình C sắp xếp một mảng số thực nhập từ bàn phím.

1.2 Viết chương trình C sắp xếp một chuỗi gồm các chữ cái ngẫu nhiên lấy từ bảng chữ cái alphabet (có bao gồm chữ in hoa).

1.3 Cho một mảng a có kích thước n mà mỗi phần tử của mảng đó là một bộ ba số thực. Với i, j = 0,1,...,n bất kỳ, ta định nghĩa:

a[i] <= a[j] ⇔ F(a[i]) <= F(a[j])

và nếu a[i] = (a,b,c) thì F(a[i]) = a - 2b + 3c với a, b, c là 3 số thực.

Hãy dựa trên định nghĩa quan hệ thứ tự trên, sắp xếp mảng a theo thứ tự tăng dần.

9

1.1

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#define MAX 1001
//1.1
//hàm hoán đổi 2 số thực
void swap(float *a, float *b){
    float temp = *a;
    *a = *b;
    *b = temp;
}
//thuật toán sắp xếp bubble sort
void bubbleSort(float arr[], int n){
    for(int i = 0; i < n; i++){
```

```
        for(int j = 0; j < n - i - 1; j++){
            if(arr[j] > arr[j + 1]){
                swap(&arr[j], &arr[j + 1]);
            }
        }
    }
}
int main(){
    printf("Nhap kich thuoc mang: ");
    int n; scanf("%d", &n);
    float arr[MAX];
    printf("Nhap mang so thuc:\n");
    for(int i = 0; i < n; i++){
        scanf("%f", &arr[i]);
    }
    bubbleSort(arr, n);
    printf("Mang so thuc sau khi sap xep la:\n");
    for(int i = 0; i < n; i++){
        printf("%.2f ", arr[i]);
    }
    return 0;
}
```

TestCase:

```
Nhap kich thuoc mang: 5
Nhap mang so thuc:
4.5 3.6 8.9 2.1 1.4
Mang so thuc sau khi sap xep la:
1.40 2.10 3.60 4.50 8.90
```

1.2

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#define MAX 1001
//1.2
//thuật toán random chữ cái ngẫu nhiên
void initializeArrayStr(char str[], int n){
   char charset[] =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
   int size = strlen(charset);
   for(int i = 0; i < n; i++){
      str[i] = charset[rand() % size];
   }
   str[n] = '\0';
}
//hàm hoán đổi 2 chữ cái
void swap(char *c1, char *c2){
   char tempChar = *c1;
   *c1 = *c2;
   *c2 = tempChar;
}
//thuật toán sắp xếp bubble sort
void bubbleSort(char str[], int n){
   for(int i = 0; i < n; i++){
      for(int j = 0; j < n - i - 1; j++){
         if(str[j] > str[j + 1]){
            swap(&str[j], &str[j + 1]);
```

```
            }
        }
    }
}

int main(){
    time_t t;
    srand((unsigned)time(&t));
    //nhập độ dài của chuỗi
    int n;
    printf("Nhap do dai cua chuoi: ");
    scanf("%d", &n);
    char str[MAX];
    initializeArrayStr(str, n);
    printf("Chuoi duoc sinh ngau nhien la: \n%s", str);
    bubbleSort(str, n);
    printf("\nChuoi sau khi duoc sap xep: \n%s", str);
    return 0;
}
```

TestCase:

```
Nhap do dai cua chuoi: 4
Chuoi duoc sinh ngau nhien la:
Daki
Chuoi sau khi duoc sap xep:
Daik
```

1.3

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

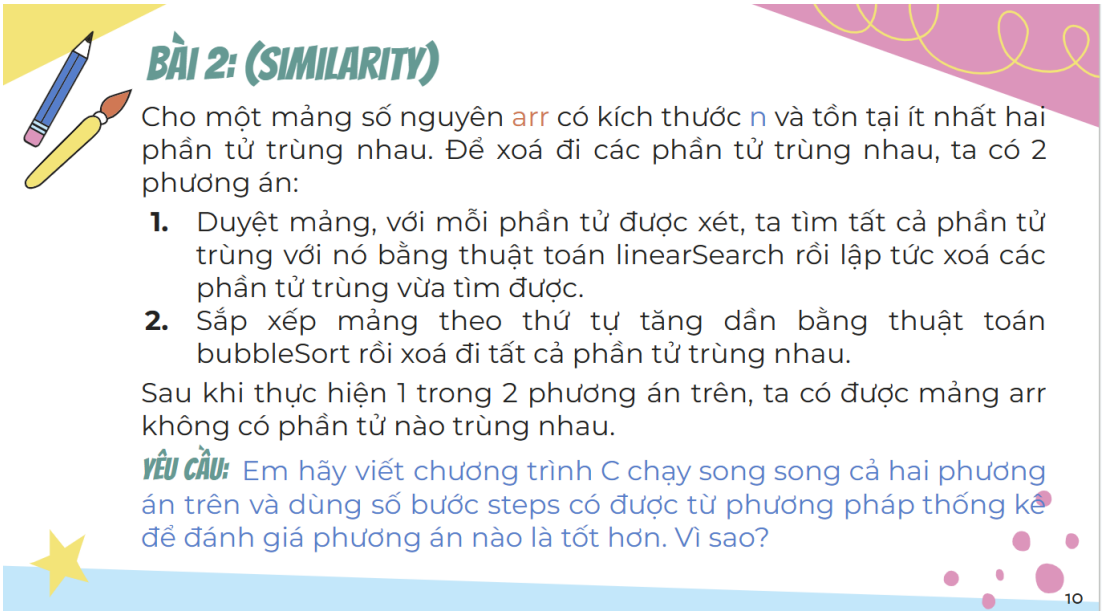
```
#include <string.h>
#define MAX 1001
//1.3
//hàm hoán đổi 2 số thực
void swap(float *a, float *b){
    float temp = *a;
    *a = *b;
    *b = temp;
}
//hàm hoán đổi 2 mảng số thực
void swapArr(float arr1[3], float arr2[3]){
    swap(&arr1[0], &arr2[0]);
    swap(&arr1[1], &arr2[1]);
    swap(&arr1[2], &arr2[2]);
}
//hàm tính giá trị F theo công thức
float F(float arr[]){
    return (float) arr[0] - 2*arr[1] + 3*arr[2];
}
//thuật toán sắp xếp bubble sort
void bubbleSort(float arr[][3], int n){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n - i - 1; j++){
            if(F(arr[j]) > F(arr[j + 1])){
                swapArr(arr[j], arr[j + 1]);
            }
        }
    }
```

```
}
int main(){
    //nhập kích thước n của mảng
    printf("nhap kich thuoc cua mang: ");
    int n; scanf("%d", &n);
    //khởi tạo mảng a với mỗi phần tử là bộ 3 số thực
    float arr[MAX][3];
    for(int i = 0; i < n; i++){
        for(int j = 0; j < 3; j++){
            scanf("%f", &arr[i][j]);
        }
    }
    //sắp xếp mảng theo thứ tự tăng dần
    bubbleSort(arr, n);
    printf("Mang sau khi sap xep la:\n");
    for(int i = 0; i < n; i++){
        for(int j = 0; j < 3; j++){
            printf("%.2f ", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

TestCase:

```
nhap kich thuoc cua mang: 5
1 2 3
4 5 6
7 8 9
0 1 0
1 0 1
Mang sau khi sap xep la:
0.00 1.00 0.00
1.00 0.00 1.00
1.00 2.00 3.00
4.00 5.00 6.00
7.00 8.00 9.00
```

Bài 2:



BÀI 2: (SIMILARITY)

Cho một mảng số nguyên arr có kích thước n và tồn tại ít nhất hai phần tử trùng nhau. Để xoá đi các phần tử trùng nhau, ta có 2 phương án:

1. Duyệt mảng, với mỗi phần tử được xét, ta tìm tất cả phần tử trùng với nó bằng thuật toán linearSearch rồi lập tức xoá các phần tử trùng vừa tìm được.
2. Sắp xếp mảng theo thứ tự tăng dần bằng thuật toán bubbleSort rồi xoá đi tất cả phần tử trùng nhau.

Sau khi thực hiện 1 trong 2 phương án trên, ta có được mảng arr không có phần tử nào trùng nhau.

YÊU CẦU: Em hãy viết chương trình C chạy song song cả hai phương án trên và dùng số bước steps có được từ phương pháp thống kê để đánh giá phương án nào là tốt hơn. Vì sao?

10

```
#include <stdio.h>
#define MAX 1000
//hàm hoán vị 2 số nguyên
void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
//cách 1
//hàm xóa phần tử trong mảng bằng chỉ số index
void deleteElements(int arr[], int *n, int index, int *assign){
    *assign = 0;
    for(int i = index; i < (*n) - 1; i++){
        arr[i] = arr[i + 1];
        (*assign)++;
    }
    (*n)--;
```

```
}
//hàm xóa phần tử trùng nhau bằng phương án 1
void deleteDuplicateElements1(int arr[], int *n, int *steps){
    int comps, assign;
    *steps = 0;
    for(int i = 0; i < *n - 1; i++){
        comps = 0;
        for(int j = i + 1; j < *n; j++){
            comps++;
            if(arr[j] == arr[i]){
                deleteElements(arr, n, j, &assign);
                *steps += assign;
                j--;
            }
        }
        *steps += comps;
    }
}
//cách 2
//thuật toán bubblesort
void bubbleSort(int arr[], int n, int *comps, int *swaps){
    *comps = 0, *swaps = 0;
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n - i - 1; j++){
            (*comps)++;
            if(arr[j] > arr[j + 1]){
                swap(&arr[j], &arr[j + 1]);
                (*swaps)++;
```

```
            }
        }
    }
}
//hàm xóa phần tử trùng nhau bằng phương án 2
void deleteDuplicateElements2(int arr[], int *n, int *steps){
    //đếm số comps và swaps của bubble sort
    int comps1, swaps1;
    bubbleSort(arr, *n, &comps1, &swaps1);
    //tính trước số steps của thuật toán bubble sort
    *steps = comps1 + swaps1;
    //tính số comps và assign của thuật toán ở dưới
    int j = 0, comps2 = 0, assign = 0;
    for(int i = 1; i < *n; i++){
        comps2++;
        if(arr[i] != arr[i - 1]){
            arr[j] = arr[i - 1];
            assign++;
            j++;
        }
    }
    *steps +=  comps2 + assign;
    arr[j] = arr[*n - 1];
    *n = j + 1;
}
int main(){
    int n; scanf("%d", &n);
    //khởi tạo kích thước 2 mảng giống nhau
```

```
    int n1 = n, n2 = n;
    //khởi tạo 2 mảng giống nhau
    int arr1[MAX], arr2[MAX];
    for(int i = 0; i < n; i++){
        scanf("%d", &arr1[i]);
        arr2[i] = arr1[i];
    }
    //khởi tạo steps1, steps2 để tính số bước của 2 phương án
    int steps1, steps2;
    deleteDuplicateElements1(arr1, &n1, &steps1);
    deleteDuplicateElements2(arr2, &n2, &steps2);
    printf("Steps of first method is: %d\n", steps1);
    printf("Steps of second method is: %d\n", steps2);
    return 0;
}
/*
TestCase1:
10
1 2 3 1 4 5 3 6 7 2
Steps of first method is: 35
Steps of second method is: 70
TestCase2:
10
5 5 6 7 8 8 9 5 10 7
Steps of first method is: 36
Steps of second method is: 68
TestCase3:
15
```

10 11 10 12 13 13 14 15 16 16 20 9 6 6 7

Steps of first method is: 104

Steps of second method is: 177

Ta có thể thấy số bước của phương án 2 nhiều hơn số bước của phương án 1.

Cho nên phương án 1 là tốt hơn.

*/

TestCase:

```
10
9 8 7 9 6 5 8 4 7 3
Steps of first method is: 37
Steps of second method is: 94
```