**Bài tập về "Mảng một chiều"**

**Các bài tập cơ bản về "mảng một chiều" được sắp xếp, bố trí từ dễ đến khó dần nhằm giúp các bạn chưa vững (thậm chí mất căn bản) về cấu trúc dữ liệu kiểu mảng trong ngôn ngữ C có thể tiếp cận dễ dàng.**

**Khuyến cáo:** (1) những bạn mất căn bản nên coi, suy nghĩ và làm lần lượt các bài theo thứ tự (2) chỉ nên coi đáp án (code) khi đã cố gắng tự làm.

**MỤC LỤC**

[**Phần 1. Khởi động, làm quen (ý tưởng, thuật toán & code)**](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#p1)

1. [Nhập (biết n)](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#nhapn)
2. [Nhập (dừng khi -1)](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#nhapwhile)
3. [Tìm min](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#timmin)
4. [Tính mean](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#timmean)
5. [Đếm số phần tử bằng x](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#demx)
6. [Tìm phần tử bằng x](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#timx)
7. [Sắp xếp](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#sapxep)
8. [Độ lệch chuẩn](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#timsd)
9. [Xóa phần tử tại k](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#xoak)
10. [Xóa phần tử bằng x](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#xoax)
11. [Chèn phần tử tại k](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#chenk)
12. [Chèn phần tử x](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#chenx)

[**Phần 2. Thực tập (ý tưởng & thuật toán)**](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#p2)

1. [Nhập (ktra trùng)](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#nhapktratrung)
2. [Liệt kê ≤ x](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#lietkenhox)
3. [Liệt kê số nguyên tố](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#lietkesnt)
4. [Đảo ngược](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#daonguoc)
5. [Tìm ˜ dãy con tăng](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#daycontang)
6. [Trộn](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#tronmang)

[**Phần 3. Luyện tập (không lời giải, đơn giản)**](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#p3)

1. [Tìm max thứ k](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#timmaxk)
2. [Kiểm tra đối xứng](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#doixung)
3. [Tính mean (chẵn)](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#meanchan)
4. [Tìm SNT min max](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#sntminmax)
5. [Tách mảng](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#tachmang)
6. [Thông số thống kê](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#thongsothongke)

[**Phần 4. Thử thách (không lời giải, kho khó)**](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#p4)

1. [Quay vòng](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#quayvong)
2. [Phần tử xuất hiện max](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#xuathienmax)
3. [Tam giác Pascal](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#tamgiacpascal)
4. [Tổng hiệu số lớn](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#tonghieuab)
5. [Tích số lớn](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#tichab)

**Phần 1. Khởi động, làm quen (ý tưởng, thuật toán & code)**

1. **Nhập mảng (nhập trước n)**  
   Nhập số tự nhiên n (n≤ 99), nhập tiếp một mảng gồm n số nguyên sau đó hiển thị mảng vừa nhập (các phần tử cách nhau bởi một khoảng trắng).
2. Nhập vào số phần tử của mảng: 4
3. Nhập phần tử thứ 1: 1
4. Nhập phần tử thứ 2: 4
5. Nhập phần tử thứ 3: 2
6. Nhập phần tử thứ 4: 8
7. Mảng vừa nhập là: 1 4 2 8

**Ý tưởng:**

Sau khi nhập n, chương trình sẽ thực hiện 2 vòng lặp, một là để nhập mảng, hai là để hiển thị. Cả 2 vòng lặp này đều biết trước số lần lặp → vòng lặp for.

**Thuật toán:**

* + b1. Khai báo mảng avới số phần tử tối đa là 99, số phần tử thực sự là n và biến i để chạy.
  + b2. Nhập n.
  + b3. Nhập a[i] với i=0 → n-1.
  + b4. Hiển thị a[i] với i=0 → n-1.

**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99], i, n;                                //b1. Khai báo      printf("Nhap vao so phan tu cua mang: ");       //b2. Nhập n      scanf("%d", &n);      for (i = 0; i < n; i++)                         //b3. Nhập mảng      {          printf("Nhap phan tu thu %d: ", i + 1);          scanf("%d", &a[i]);      }      printf("Mang vua nhap la: ");                   //b4. Hiển thị mảng      for (i = 0; i < n; i++) printf("%d ", a[i]);      getch();                                        //Dừng màn hình để xem kết quả  } |

1. **Nhập mảng (dừng khi nhập -1)**  
   Nhập vào một mảng các số tự nhiên (có tối đa 99 phần tử), sau đó hiển thị mảng vừa nhập ra màn hình (các phần tử cách nhau bởi một khoảng trắng). Quá trình nhập sẽ dừng khi nhập vào giá trị -1.
2. Nhập phần tử thứ 1: 1
3. Nhập phần tử thứ 2: 4
4. Nhập phần tử thứ 3: 2
5. Nhập phần tử thứ 4: -1
6. Mảng vừa nhập là: 1 4 2

**Ý tưởng (cách 1):**

* + **[Cấu trúc điều khiển]** Sử dụng một vòng lặp để lặp lại việc nhập giá trị cho từng phần tử của mảng. Do chưa biết trước số phần tử của mảng mà người sử dụng sẽ nhập nên sẽ sử dụng vòng lặp while hoặc do..while. Số lần lặp tối thiểu là 1 (nghĩa là chương trình sẽ kiểm tra điều kiện lặp sau khi nhập x) → sử dụng vòng lặp do..while.
  + **[Điều kiện lặp]** Việc nhập sẽ lặp lại cho đến khi người sử dụng nhập x=-1, nghĩa là tiếp tục nhập trong khi x≠-1 → do..while (x≠-1).

**Thuật toán:**

* + b1. Khai báo mảng a, số phần tử n, giá trị x và biến i để chạy.
  + b2. **[Nhập mảng]** khởi tạo n=0;

lặp:

* + - Nhập x
    - Nếu x≠-1 thì gán x cho phần tử thứ n và tăng n.

trong khi x≠-1

* + b3. **[Hiển thị mảng]** xuất a[i] với i=0 → n-1.
  + b4. Dừng màn hình để xem kết quả.

**Code (cách 1):**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99], n = 0, x, i;                         //b1. Khai báo      do{                                             //b2. Nhập mảng          printf("Nhap phan tu thu %d: ", n + 1);          scanf("%d", &x);          if (x != -1)          {              a[n] = x;              n++;          }      } while (x != -1);      printf("Mang vua nhap la: ");                   //b3. Hiển thị mảng      for (i = 0; i < n; i++) printf("%d ", a[i]);      getch();                                        //b4. Dừng màn hình để xem kết quả  } |

**Ý tưởng cải tiến (cách 2):**

Cách làm ở trên sử dụng biến x để lưu giá trị mà NSD vừa nhập từ bàn phím. Tuy nhiên, ta hoàn toàn có thể lưu trực tiếp giá trị này vào mảng, ngay tại vị trí cuối cùng của mảng, tức vị trí n. Nói rõ ra, giả sử hiện tại mảng có n phần tử (a[0] → a[n-1]). Khi NSD nhập, ta lưu trực tiếp vào a[n] mà chưa tăng n vội (chú ý: a[n] lúc này chưa là phần tử của mảng). Sau khi kiểm tra nếu giá trị nhập này là hợp lệ (khác -1) thì mới tăng n, nghĩa là dung nạp phần tử a[n] vào mảng.

**Code (cách 2):**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99], n = 0, i;                            //khai báo      do{                                             //nhập mảng          printf("Nhap phan tu thu %d: ", n + 1);          scanf("%d", &a[n]);                         //lưu trực tiếp giá trị nhập vào a[n]          if (a[n] != -1) n++;                        //nếu giá trị hợp lệ thì dung nạp vào mảng      } while (a[n] != -1);      printf("Mang vua nhap la: ");                   //hiển thị mảng      for (i = 0; i < n; i++) printf("%d ", a[i]);      getch();                                        //dừng màn hình để xem kết quả  } |

1. **Tìm min (giá trị nhỏ nhất)**   
   Nhập vào một mảng gồm n (n≤ 99) số tự nhiên từ bàn phím (quá trình nhập sẽ dừng khi người sử dụng nhập vào giá trị -1) sau đó hiển thị mảng vừa nhập và giá trị nhỏ nhất của mảng ra màn hình.
2. Nhập phần tử thứ 1: 6
3. Nhập phần tử thứ 2: 5
4. Nhập phần tử thứ 3: 4
5. Nhập phần tử thứ 4: 8
6. Nhập phần tử thứ 4: -1
7. Mảng vừa nhập là: 6 5 4 8

Giá trị nhỏ nhất của mảng là: 4

**Ý tưởng:**

* + **[Lặp để tìm min]** Sau khi nhập mảng, ta sử dụng một biến min để lưu giá trị nhỏ nhất của mảng. Ban đầu, giả sử phần tử nhỏ nhất là a[0]. Sau đó, cứ duyệt từng phần từ a[i] từ đầu đến cuối mảng, nếu thấy a[i]<min thì cập nhật min=a[i].
  + **[Kết hợp 2 vòng lặp]** Do việc hiển thị mảng và tìm min đểu phải duyệt qua tất cả các phần tử của mảng, do vậy ta sử dụng một vòng lặp, ở đây là vòng lặp for do đã biết trước số lần lặp (là số phần tử của mảng, n).

**Thuật toán:**

* + b1. Khai báo các biến: mảng a, số phần tử n, biến chạy i, giá trị nhỏ nhất min.
  + b2. Nhập mảng (sử dụng vòng lặp while, xem [bài tập 2](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#nhapwhile))
  + b3. Tìm minvà hiển thị mảng:
    - Khởi gán min=a[0]
    - Duyệt qua các phần tử trong mảng (i=0→n-1):  
      - Hiển thị a[i] ra màn hình  
      - Nếu a[i]<min thì cập nhật min=a[i]
  + b4. Xuất kết quả (giá trị min) & dừng màn hình.

**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99], n = 0, i, min;                          //b1. Khai báo      do{                                                //b2. Nhập mảng          printf("Nhap phan tu thu %d: ", n + 1);          scanf("%d", &a[n]);          if (a[n] != -1) n++;      } while (a[n] != -1);      printf("Mang vua nhap la: ");      min = a[0];                                        //b3. Tìm min & hiển thị mảng      for (i = 0; i < n; i++)      {          printf("%d ", a[i]);          if (a[i] < min) min = a[i];      }      printf("\nGia tri nho nhat cua mang la: %d", min); //b4. Xuất kết quả & dừng màn hình      getch();  } |

1. **Tìm mean (giá trị trung bình)**  
   Khởi tạo một mảng gồm n (n≤ 99) số tự nhiên sau đó tính giá trị trung bình các phần tử của mảng.
2. Mảng được khởi tạo là: 6 5 4 8

Giá trị trung bình của mảng là: 5.75

**Ý tưởng:**

* + **[Cần tính gì]**Để tính giá trị trung bình, ta cần biết *tổng*giá trị các phần tử và *số phần tử*. Số phần tử thì đã rõ, chính là n, do vậy chỉ cần tính *tổng*giá trị các phần tử.
  + **[Tính tổng s]**Ta sử dụng biến s để lưu *tổng* giá trị các phần tử của mảng. Ban đầu, ta sẽ khởi gán s=0, sau đó cứ duyệt từng phần tử a[i] từ đầu đến cuối mảng (i=0→n-1). Với mỗi giá trị a[i] ta cộng dồn vào s (s=s+a[i]).
  + **[Kết hợp 2 vòng lặp]**Do việc hiển thị mảng và tính tổng s đểu phải duyệt qua tất cả các phần tử của mảng (mỗi phần tử một lần) nên ta sử dụng một vòng lặp, ở đây là vòng lặp for do đã biết trước số lần lặp (chính là số phần tử n của mảng).
  + **[Tính giá trị trung bình]**Sau khi đã có tổng s (là một số nguyên), giá trị trung bình sẽ là s/n (đây là một số *thực*). Tuy nhiên, do cả s và n đều có kiểu nguyên, nên phép chia cũng nguyên, nghĩa là kết quả là một số nguyên (ví dụ: 5/2=2). Để báo cho C biết rằng đây là phép chia thực thì ta cần ép biến s (hoặc n) sang kiểu thực bằng cách (float)s/n (hoặc s/(float)n). Một cách khác (không phải ép kiểu) là khai báo kiểu số thực (float hoặc double) cho biến s.

**Thuật toán:**

* + b1. Khai báo các biến: mảng a (và khởi gán giá trị), số phần tử n, biến chạy i, tổng s.
  + b2. Tính tổng svà hiển thị mảng:
    - Khởi gán s=0
    - Duyệt các phần tử a[i] trong mảng (i=0→n-1):
      * Hiển thị a[i]
      * Cộng dồn a[i] vào s.
  + b3. Xuất kết quả (giá trị trung bình, (float)s/n) & dừng màn hình.

**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99] = { 6, 5, 4, 8 };      int n = 4, i, s = 0;      printf("Mang duoc khoi tao la: ");      for (i = 0; i < n; i++) {          printf("%d ", a[i]);                            //hiển thị phần tử thứ i          s += a[i];                                      //cộng dồn a[i] vào tổng s      }      printf("\nGia tri trung binh cua mang la: %g", (float)s / n);      getch();  } |

1. **Đếm số phần tử bằng x**   
   Khởi tạo một mảng gồm n (n ≤ 99) số thực, nhập một số thực x từ bàn phím sau đó đếm số phần tử trong mảng có giá trị x.
2. Mảng được khởi tạo là: 6.0 5.5 4.3 8.8 5.5
3. Nhập một số thực: 5.5

Số phần tử trong mảng có giá trị 5.5 là: 2

**Ý tưởng cách 1 (không sử dụng hàm):**

* + **[Đếm]**Để đếm số phần tử trong mảng có giá trị bằng x, ta khai báo thêm biến dem (kiểu nguyên). Ban đầu ta khởi gán dem=0. Sau đó cứ duyệt từng phần tử a[i] từ đầu đến cuối mảng (i=0→n-1), với mỗi giá trị a[i] ta kiểm tra nếu a[i]=x thì tăng giá trị của biến dem lên 1. Để duyệt từng phần tử, ta sử dụng vòng lặp for do đã biết trước số lần lặp (chính là số phần tử n).

**Thuật toán cách 1 (không sử dụng hàm):**

* + b1. Khai báo các biến: mảng a (và khởi gán giá trị), số phần tử n, giá trị x, biến chạy i, biến đếm dem.
  + b2. Hiển thị mảng.
  + b3. Nhập giá trị x (từ bàn phím).
  + b4. Đếm:
    - Khởi gán dem=0
    - Duyệt qua các phần tử trong mảng (i=0→n-1):  
           - Nếu thỏa mãn điều kiện (a[i]=x) thì dem=dem+1.
  + b5. Xuất kết quả (dem) & dừng màn hình.

**Code cách 1 (không sử dụng hàm):**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #include <stdio.h>  #include <conio.h>  int main()  {      float a[99] = { 6, 5.5, 4.3, 8.8, 5.5 };          //b1. Khai báo biến      float x;      int n = 5, i, dem;      printf("Mang duoc khoi tao la: ");                //b2. Hiển thị mảng      for (i = 0; i < n; i++)          printf("%.1f ", a[i]);      printf("\nNhap mot so thuc: ");                   //b3. Nhập giá trị x (từ bàn phím)      scanf("%f", &x);      dem = 0;                                          //b4. Đếm      for (i = 0; i < n; i++)          if (a[i] == x) dem++;      printf("So phan tu trong mang co gia tri %g la: %d", x, dem);      getch();  } |

**Ý tưởng cách 2 (sử dụng hàm):**

* + **[Khai báo nguyên mẫu hàm]**Hàm này nhận nhiệm vụ đếm số phần tử bằng x trong mảng nên có thể đặt tên gợi nhớ là SoPhanTuBangX. Hàm nhận đầu vào là mảng (mảng b, số phần tử n) và giá trị x (kiểu số thực). Giá trị trả về của hàm là một số nguyên, chính là số phần tử trong mảng có giá trị x. Như vậy, nguyên mẫu hàm sẽ là: int SoPhanTuBangX(float b[], int n, float x).
  + **[Đếm]**Hàm SoPhanTuBangX sẽ phụ trách việc đếm. Cách đếm thì hoàn toàn tương tự như ở cách 1, tuy nhiên kết quả (giá trị biếm dem) thay vì hiển thị ra màn hình thì hàm này chỉ đơn giản đưa ở giá trị trả về, thông qua lời gọi return. Việc hiển thị kết quả ra màn hình hay có những tính toán gì khác là thuộc thẩm quyền của hàm gọi (ở đây là hàm main).

**Code (cách 2: sử dụng hàm):**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include <stdio.h>  #include <conio.h>  int SoPhanTuBangX(float b[], int n, float x)  {      int dem = 0, i;      for (i = 0; i < n; i++)          if (b[i] == x) dem++;      return dem;  }  int main()  {      float a[99] = { 6, 5.5, 4.3, 8.8, 5.5 };      int n = 5, i;      float x;      printf("Mang duoc khoi tao la: ");      for (i = 0; i < n; i++)          printf("%.1f ", a[i]);      printf("\nNhap mot so thuc: ");      scanf("%f", &x);      printf("So phan tu trong mang co gia tri %g la: %d", x, SoPhanTuBangX(a, n, x));      getch();  } |

1. **Kiểm tra sự tồn tại giá trị x trong mảng**  
   Khởi tạo một mảng gồm n (n ≤ 99) số thực, nhập một số thực x từ bàn phím sau đó tìm xem trong mảng có phần tử nào bằng xhay không.
2. Mảng được khởi tạo là: 6.0 5.5 4.3 8.8 5.5
3. Nhập một số thực: 5.5

Phần tử thứ 1 (tính từ 0) trong mảng có giá trị 5.5.

Mảng được khởi tạo là: 6.0 5.5 4.3 8.8 5.5

Nhập một số thực: 5.6

Không có phần tử nào trong mảng có giá trị 5.6.

**Thuật toán:**

* + b1. Khai báo các biến: mảng a (và khởi gán giá trị), số phần tử n, giá trị x, biến chạy i.
  + b2. Hiển thị mảng.
  + b3. Nhập giá trị x (từ bàn phím).
  + b4. Tìm kiến xtrong mảng:
    - Khởi gán i=0
    - Trong khi tìm chưa thấy (a[i]≠x) và tìm chưa hết (i<n) thì:  
                - Tìm tiếp i=i+1
    - Nếu i<n thì thông báo có x trong mảng
    - Ngược lại: thông báo không có x trong mảng

**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | #include <stdio.h>  #include <conio.h>  int main()  {      float a[99] = { 6, 5.5, 4.3, 8.8, 5.5 };      float x;      int n = 5, i, dem;      printf("Mang duoc khoi tao la: ");      for (i = 0; i < n; i++)          printf("%.1f ", a[i]);      printf("\nNhap mot so thuc: ");      scanf("%f", &x);      i = 0;      while (i < n && a[i] != x) i++;      if (i < n)          printf("Phan tu thu %d (tinh tu 0) trong mang co gia tri %g.", i, x);      else          printf("Khong co phan tu nao trong mang co gia tri %g.", x);      getch();  } |

1. **Sắp xếp**   
   Nhập vào một mảng A gồm n (n≤ 99) số tự nhiên từ bàn phím (quá trình nhập sẽ dừng khi người sử dụng nhập vào giá trị -1) sau đó sắp xếp mảng theo thứ tự không giảm.
2. Nhập phần tử thứ 1: 1
3. Nhập phần tử thứ 2: 4
4. Nhập phần tử thứ 3: 8
5. Nhập phần tử thứ 4: 2
6. Nhập phần tử thứ 4: -1
7. Mảng vừa nhập là: 1 4 8 2

Mảng sau khi sắp xếp là: 1 2 4 8

**Ý tưởng:**

* + **[Ý tưởng ban đầu]**Cho một biến chạy i ban đầu sẽ bằng 0. Chương trình sẽ cố gắng tìm phần tử nhỏ nhất để đưa về vị trí 0. Sau đó tăng i lên 1 (i++). Khi i=1, chương trình sẽ cố gắng tìm phần tử nhỏ nhì để đưa về vị trí 1. Và cứ như vậy, biến i sẽ chạy đến phần tử kế cuối (n-2). Lúc này mảng đã được sắp xếp. Chú ý, lúc này phần tử cuối cùng (n-1) chắc chắc là phần tử lớn nhất.
  + **[Cụ thể hóa]**Biến i chạy từ phần tử đầu tiên (0) đến phần tử kế cuối (n-2). Với mỗi i ta sẽ cố gắng tìm phần tử nhỏ nhất trong phần còn lại của mảng (a[i]..a[n-1]) bằng cách cho một biến j chạy từ phần tử ngay sau i (i+1) đến phần tử cuối cùng (n-1). Với mỗi giá trị của j, ta kiểm tra a[i]>a[j] hay không. Nếu đúng thì hoán đổi a[i] và a[j].

**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99], n = 0, i, j, tam;      do{                                                          //nhập mảng          printf("Nhap phan tu thu %d: ", n + 1);          scanf("%d", &a[n]);          if (a[n] != -1) n++;      } while (a[n] != -1);      printf("Mang vua nhap la: ");                      //hiển thị mảng      for (i = 0; i < n; i++) printf("%d ", a[i]);      for (i = 0; i < n - 1; i++)                        //sắp xếp mảng          for (j = i + 1; j < n; j++)              if (a[i]>a[j])              {                  tam = a[i];                  a[i] = a[j];                  a[j] = tam;              }      printf("\nMang sau khi sap xep la: ");             //hiển thị mảng      for (i = 0; i < n; i++) printf("%d ", a[i]);      getch();  } |

1. **Tính độ lệch chuẩn**   
   Khởi tạo một dãy gồm n (n ≤ 99) số tự nhiên sau đó tính độ lệch chuẩn của dãy, biết công thức tính độ lệch chuẩn là: http://www.forkosh.com/mimetex.cgi?%7b\sigma=\sqrt%7b\frac%7b1%7d%7bn%7d\sum_%7bi=0%7d%5e%7bn-1%7d(x_i-\bar%7bx%7d)%5e2%7d, với giá trị trung bình http://www.forkosh.com/mimetex.cgi?%7b\bar%7bx%7d=\frac%7b1%7d%7bn%7d\sum_%7bi=0%7d%5e%7bn-1%7dx_i%7d.
2. Dãy được khởi tạo là: 6 5 4 8

Độ lệch chuẩn của dãy là: 1.47902

**Ý tưởng:**

* + **[Trình tự tính toán]**Nhìn vào công thức tính độ lệch chuẩn ta thấy rõ ràng trước tiên phải tính giá trị trung bình (kỳ vọng) của dãy sau đó sẽ tính phương sai, nghĩa là biểu thức dưới dấu căn. Từ đây, độ lệch chuẩn sẽ đơn giản là căn bậc 2 của phương sai.

**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

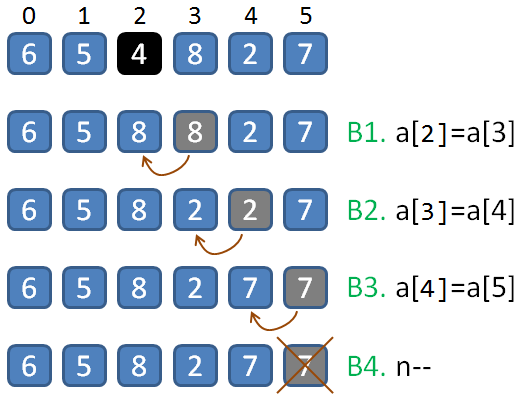
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | #include <stdio.h>  #include <conio.h>  #include <math.h>  int main()  {      int a[99] = { 6, 5, 4, 8 };      int n = 4, i;      float tb, s;      printf("Mang duoc khoi tao la: ");      s = 0;      for (i = 0; i < n; i++) {          printf("%d ", a[i]);          s += a[i];                                       //tính tổng s      }      tb = s / n;                                          //tính kỳ vọng      s = 0;                                               //tính phương sai      for (i = 0; i < n; i++)          s += (a[i] - tb)\*(a[i] - tb);      printf("\nDo lech chuan cua day la: %g", sqrt(s/n)); //tính độ lệch chuẩn & xuất kết quả      getch();  } |

1. **Xóa phần tử tại vị trí k**   
   Khởi tạo một mảng gồm n (n ≤ 99) số tự nhiên. Nhập một số tự nhiên k từ bàn phím sau đó xóa phần tử thứ k trong mảng (k được tính từ 0). **Chú ý:**thông báo ra mình nếu k không hợp lệ (vượt quá số phần tử của mảng).
2. Mảng được khởi tạo là: 6 5 4 8 2 7
3. Nhập vị trí phần tử cần xóa: 2

Mảng sau khi xóa phần tử thứ 2 là: 6 5 8 2 7

**Ý tưởng:**

* + **[Ý tưởng ban đầu]** Để xóa phần tử tại vị trí k, ta chỉ đơn giản lần lượt di chuyển các phần tử sau vị trí k sang trái 1 vị trí. Sau khi đã di chuyển hết (các phần tử từ k+1 đến n-1) thì lúc này mảng đã bị rút ngắn đi 1 vị trí, ta chỉ đơn giản kết thúc thao tác bằng việc giảm n đi 1.
  + **[Cụ thể hóa ý tưởng]** Ta sẽ sử dụng biến i để duyệt các phần tử cần di chuyển (sử dụng 1 vòng lặp). Số phần tử cần di chuyển là n-k-1, nghĩa là số lần lặp là biết trước ⇒ vòng lặp for. Tại mỗi vị trí của i, sẽ di chuyển phần tử a[i] sang trái bằng phép gán a[i-1]=a[i]. Sau khi di chuyển xong tất cả, chỉ đơn giản giảm n đi 1 để gút lại mảng, vì lúc này phần tử a[n-1] là thừa.
  + **[Ví dụ]** Quá trình xóa phần tử thứ 2 (giá trị 4) ở ví dụ trên được minh họa cụ thể như ở hình dưới. Như ví dụ ở hình dưới, bước đầu tiên (B1) ta di chuyển phần tử a[3] sang trái bằng lệnh gán a[2]=a[3]. Tiếp đến, bước 2 & 3, di chuyển phần tử a[4] và a[5]. Kết thúc bước 3, mảng đã thực sự loại đi phần tử có giá trị 4. Lúc này, bước 4, chỉ đơn giản giảm n đi 1 để chốt lại mảng.



**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99] = { 6, 5, 4, 8, 2, 7 };      int n = 6, i, k;      printf("Mang duoc khoi tao la: ");      for (i = 0; i < n; i++)          printf("%d ", a[i]);      printf("\nNhap vi tri phan tu can xoa: ");      scanf("%d", &k);      if (k < n)      {          for (i = k+1; i < n; i++)              a[i-1] = a[i];          n--;          printf("Mang sau khi xoa phan tu thu %d la: ", k);          for (i = 0; i < n; i++) printf("%d ", a[i]);      }      else          printf("Loi: vi tri can xoa khong hop le (vuot qua chi so cua phan tu cuoi cung).");      getch();  } |

1. **Xóa các phần tử có giá trị bằng x**   
   Khởi tạo một mảng gồm n (n ≤ 99) số tự nhiên. Nhập một số tự nhiên x từ bàn phím sau đó xóa tất cả các phần tử có giá trị x trong mảng. **Chú ý:**thông báo ra màn hình số phần tử bị xóa.
2. Mảng được khởi tạo là: 4 8 2 7 8 5
3. Nhập vào giá trị phần tử cần xóa: 8
4. Mảng sau khi xóa các phần tử có giá trị 8: 4 2 7 5

Có 2 phần tử bị xóa khỏi mảng.

**Ý tưởng:**

* + Duyệt từng phần tử từ cuối (n-1) trở về đầu (0). Tại mỗi phần tử a[i], nếu có giá trị bằng x thì xóa phần tử này khỏi mảng (cách xóa như ở câu trên).

**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

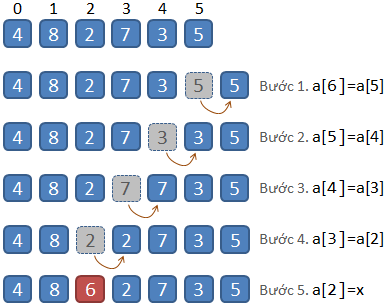
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99] = { 4, 8, 2, 7, 8, 5 };      int n = 6, i, j, x, dem;      printf("Mang duoc khoi tao la: ");      for (i = 0; i < n; i++)          printf("%d ", a[i]);      printf("\nNhap gia tri phan tu can xoa: ");      scanf("%d", &x);      i = n-1; dem = 0;      while (i >= 0)                                   //khi chưa hết mảng      {          if (a[i] == x)                              //nếu tìm được phần tử =x          {              for (j = i; j < n - 1; j++)             //xóa phần tử vị trí i                  a[j] = a[j + 1];              n--;              dem++;          }          else i--;                                   //nếu a[i]<>x thì cứ cho i chạy tiếp      }      printf("Mang sau khi xoa cac phan tu co gia tri %d la: ", x);      for (i = 0; i < n; i++) printf("%d ", a[i]);      printf("\nCo %d phan tu bi xoa khoi mang.", dem);      getch();  } |

1. **Chèn phần tử vào vị trí k**   
   Khởi tạo một mảng gồm n (n ≤ 99) số tự nhiên. Nhập hai số tự nhiên x và k từ bàn phím sau đó chèn phần tử có giá trị x vào mảng tại vị trí k (tính từ 0). **Chú ý:**thông báo ra màn hình nếu k không hợp lệ (vượt quá kích thước mảng).
2. Mảng được khởi tạo là: 4 8 2 7 3 5
3. Nhập giá trị cần chèn: 6
4. Nhập vị trí cần chèn: 2

Mảng sau khi chèn phần tử có giá trị 6 vào vị trí 2: 4 8 6 2 7 3 5

**Ý tưởng:**

* + **[Ý tưởng ban đầu]** Để chèn 1 phần tử tại vị trí k, ta chỉ đơn giản lần lượt di chuyển các phần tử từ a[k] đến a[n-1] sang phải 1 vị trí. Để tránh giẫm đạp lên nhau khi di chuyển, ta sẽ di chuyển các phần tử phía bên phải trước. Sau khi di chuyển xong n-k phần tử, ta đưa giá trị x và vị trí k bằng toán tử gán a[k]=x.
  + **[Ví dụ]** Quá trình chèn giá trị x=6 vào vị trí k=2 trong mảng được minh họa như ở hình dưới. Đầu tiên, phần tử a[5] được di chuyển sang phải (a[6]=a[5]). Tiếp đến là di chuyển các phần tử a[4], a[3] và a[2]. Cuối cùng mới đưa giá trị x=6 vào a[2].



**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

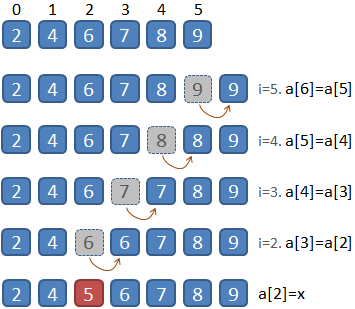
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99] = { 4, 8, 2, 7, 3, 5 };      int n = 6, x, k, i;      printf("Mang duoc khoi tao la: ");      for (i = 0; i < n; i++)          printf("%d ", a[i]);      printf("\nNhap gia tri can chen: ");      scanf("%d", &x);      printf("Nhap vi tri can chen: ");      scanf("%d", &k);      if (k >= n)          printf("Loi: vi tri can chen khong hop le (vuot qua chi so cua phan tu cuoi cung).");      else      {          for (i = n; i > k; i--)              a[i] = a[i - 1];          a[k] = x;          n++;          printf("Mang sau khi chen phan tu co gia tri %d vao vi tri %d la: ", x, k);          for (i = 0; i < n; i++) printf("%d ", a[i]);      }      getch();  } |

1. **Chèn phần tử có giá trị x vào mảng đã có thứ tự**   
   Khởi tạo một mảng gồm n (n≤ 99) số tự nhiên có thứ tự không giảm. Tiếp đến nhập một số tự nhiên x từ bàn phím và tìm cách chèn phần tử có giá trị x vào mảng sao cho mảng vẫn đảm bảo thứ tự không giảm.
2. Mảng được khởi tạo là: 2 4 6 7 8 9
3. Nhập giá trị cần chèn: 5

Mảng sau khi chèn phần tử có giá trị 5 là: 2 4 5 6 7 8 9

**Ý tưởng:**

* + **[Ý tưởng ban đầu]** Để chèn x vào mảng đã có thứ tự không giảm, trước tiên ta cần tìm vị trí thích hợp, sau đó mới thực hiện thao tác chèn. Để tìm vị trí chèn, ta có thể đi từ trái hoặc phải để tìm vị trí k sao cho a[k-1]≤x và x<a[k]. Sau đó di chuyển các phần tử từ a[k] đến a[n-1] sang phải 1 vị trí rồi đưa giá trị x vào a[k].
  + **[Gộp 2 vòng lặp]** Từ phân tích trên, ta cần 2 vòng lặp, một là để tìm vị trí i, hai là để di chuyển các phần tử. Để tối ưu, ta có thể kết hợp 2 công việc này trong một vòng lặp. Ban đầu, ta sẽ khởi tại i=n-1 và từng bước giảm i cho đến khi tìm đến vị trí thích hợp. Khi giảm i ta đồng thời di chuyển a[i] sang phải 1 vị trí. Như vậy, khi đến được vị trí thích hợp thì đã sẵn sàng đưa giá trị x vào a[i]. Rõ ràng, ban đầu chưa biết được vị trí i nên chưa biết sẽ lặp bao nhiêu lần ⇒ sử dụng vòng lặp while.
  + Chú ý: Nếu n=99 thì không thể chèn x vào mảng.



**Code:**

[?](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | #include <stdio.h>  #include <conio.h>  int main()  {      int a[99] = { 2, 4, 6, 7, 8, 9 };      int n = 6, x, i;      printf("Mang duoc khoi tao la: ");      for (i = 0; i < n; i++)          printf("%d ", a[i]);        if (n>99)      {          printf("Khong the chen them"); return 1;      }      printf("\nNhap gia tri can chen: ");      scanf("%d", &x);      i = n - 1;      while (i > 0 && a[i] > x)      {          a[i+1] = a[i];          i--;      }      a[i+1] = x;      n++;      printf("Mang sau khi chen phan tu co gia tri %d la: ", x);      for (i = 0; i < n; i++) printf("%d ", a[i]);      getch();  } |

**Phần 2. Thực tập (ý tưởng & thuật toán)**

1. **Nhập (có kiểm tra trùng)**   
   Nhập vào mảng gồm n (n≤ 99) phần tử là các số tự nhiên. Trong quá trình nhập phải kiểm tra các phần tử nhập vào không được trùng, nếu trùng thì thông báo và yêu cầu nhập lại. Quá trình nhập dừng lại khi nhập vào giá trị 0.
2. Nhập phần tử thứ 1: 5
3. Nhập phần tử thứ 2: 3
4. Nhập phần tử thứ 3: 5
5. Đã có phần tử này trong mảng. Vui lòng nhập lại!
6. Nhập phần tử thứ 3: 4
7. Nhập phần tử thứ 4: 0
8. Mảng vừa nhập là: 5 3 4

(có 3 phần tử)

**Ý tưởng:**

* + **[Chọn vòng lặp nhập]**Lý luận tương tự như ở [bài 2](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#nhapwhile), ta chọn vòng lặp do..while, mỗi lần lặp sẽ phụ trách việc nhập giá trị cho một phần tử của mảng (có kiểm tra trùng).
  + **[Lưu giá trị nhập vào đâu]**Như ở [bài 2](http://agreenet.vn/baiviet/laptrinh/c/?eq=Ak7wytsN1FjzuEkdqW1VKg==#nhapwhile), có hai cách lưu giá trị phần tử mà người sử dụng nhập vào. Ở đây ta chọn phương án 2: lưu trực tiếp vào mảng (tức phần tử a[n], không sử dụng biến phụ x).
  + **[Kiểm tra trùng]**Để kiểm tra phần tử vừa nhập (lưu tại a[n]) có tồn tại trong mảng chưa, ta sử dụng một biến i để duyệt tất cả các vị trí của mảng hiện tại (i=0→n-1) và kiểm tra xem a[i] có bằng a[n] không. Nếu bằng thì DỪNG việc duyệt i và kết luận giá trị này đã tồn tại đồng thời yêu cầu nhập lại giá trị khác. Đương nhiên, nếu a[i]≠a[n] thì cứ tiếp tục duyệt i. Nếu đã i đã duyệt hết (chạy vượt quá vị trí n-1) thì chứng tỏ "không trùng" (giá trị này chưa tồn tại trong mảng) ⇒ dung nạp giá trị mới này (a[n]) vào mảng bằng thao tác đơn giản là tăng n lên 1. Về vòng lặp thì do chưa biết sẽ lặp bao nhiêu lần và số lần lặp tối thiểu là 0 ⇒ sử dụng vòng lặp while.

**Thuật toán:**

* + b1. Khai báo mảng a, số phần tử n và biến chạy i.
  + b2. **[Nhập mảng]** khởi tạo n=0

lặp:

* + - Nhập giá trị từ bàn phím và lưu vào a[n]
    - **Kiểm tra trùng**:
      * Khởi gán i=0
      * Trong khi i<n và a[i]≠a[n] thì tăng i lên 1.
      * Nếu i<n (nghĩa là "trùng") thì yêu cầu nhập lại, ngược lại thì tăng n lên 1.

trong khi a[n]≠0

* + b3. **[Hiển thị mảng]** for i=0 → n-1: hiển thị phần tử thứ i.
  + b4. Dừng màn hình để xem kết quả.

1. **Liệt kê các phần tử có giá trị ≤ x**   
   Khởi tạo một mảng gồm n (n≤ 99) số thực sau đó liệt kê các phần tử có giá trị nhỏ hơn hoặc bằng x.
2. Mảng được khởi tạo là: 4.5 8.2 2 7.25 6
3. Nhập vào x: 5.3
4. Có 2 phần tử có giá trị nhỏ hơn hoặc bằng 5.3,

đó là: 4.5 2

**Ý tưởng:**

* + **[Duyệt mảng]**Bài toán cần 2 thao tác duyệt mảng: một là để *đếm* số phần tử ≤x, hai là để *liệt kê* các phần tử ≤x. Nhìn vào yêu cầu hiển thị của chương trình, do phải hiển thị *số phần tử* trước rồi mới đến *liệt kê* các phần tử nên ta không thể ghép 2 vòng lặp này lại với nhau. Mỗi vòng lặp đều cần phải duyệt tất cả các phần tử trong mảng (biết trước số lần lặp) nên ta sẽ sử dụng vòng lặp for.
  + **[Đếm số phần tử ≤x]**Để đếm số phần tử, ta chỉ cần sử dụng thêm biến dem, ban đầu được khởi tạo bằng 0, khi duyệt qua mỗi phần tử nếu a[i]≤x thì tăng dem lên 1.

**Thuật toán:**

* + b1. Khai báo mảng a (và khởi gán giá trị), số phần tử n, giá trị x, biến đếm dem và biến chạy i.
  + b2. **Đếm số phần tử ≤x**:
    - Khởi tạo bộ đếm: dem=0
    - for i=0→n-1: nếu a[i]≤x thì tăng dem lên 1
  + b3. **[Hiển thị các phần tử ≤x]** for i=0 → n-1: nếu a[i]≤x thì hiển thị a[i].
  + b4. Dừng màn hình để xem kết quả.

1. **Liệt kê các số nguyên tố**   
   Khởi tạo một mảng gồm n (n≤ 99) số tự nhiên sau đó liệt kê số nguyên tố trong mảng (kèm số thứ tự, tính từ 1).
2. Mảng được khởi tạo là: 4 5 8 3 7 6
3. Các số nguyên tố trong mảng là:
4. 5 (vị trí 2)
5. 3 (vị trí 4)
6. 7 (vị trí 5)
7. Mảng được khởi tạo là: 4 8 15
8. Các số nguyên tố trong mảng là:
9. không có :-)

**Ý tưởng:**

* + **[Hàm kiểm tra SNT]**Để chương trình rõ ràng, dễ kiểm tra ta nên xây dựng một hàm để đảm trách việc kiểm tra một số có phải là nguyên tố hay không. Điều này làm cho hàm main đỡ rắc rối. Việc tách hàm như thế này là một *thói quen tốt* khi lập trình, vì sau này có thể sẽ gặp nhiều bài toán phức tạp hơn. Về nguyên mẫu hàm, hàm nhận đầu vào là một số tự nhiên k nào đó và sau khi kiểm tra sẽ trả về một số nguyên có giá trị 1 nếu k là SNT và 0 nếu là hợp số.
  + **[Duyệt mảng]**Để liệt kê các SNT, ta cho biến i duyệt qua tất cả các phần tử của mảng. Tại mỗi phần tử, ta gọi hàm vừa xây đựng để kiểm tra SNT phần tử a[i], nếu giá trị trả về là 1 thì hiển thị a[i] ra màn hình.

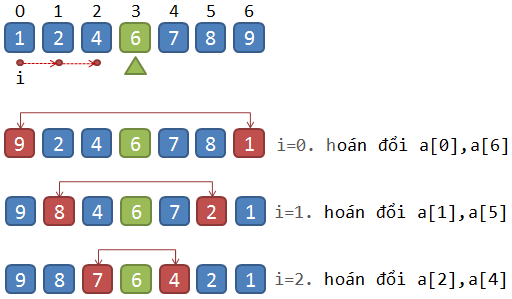
**Thuật toán:**

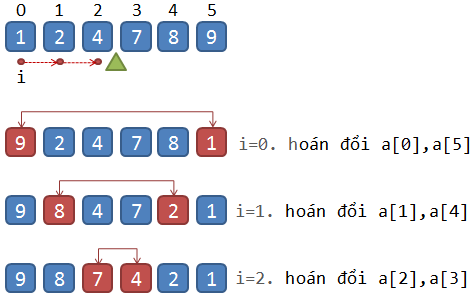
* + b1. Khai báo và cài đặt hàm kiểm tra SNT.
  + b2. Trong hàm main:
    - b21. Khai báo mảng a (và khởi gán giá trị), số phần tử n và biến chạy i.
    - b22. for i=0→n-1: nếu a[i] là SNT thì hiển thị a[i].
    - b23. Dừng màn hình để xem kết quả.

1. **Đảo ngược mảng**   
   Khởi tạo một mảng gồm n (n≤ 99) số tự nhiên sau đó đảo ngược các phần tử trong mảng.
2. Mảng được khởi tạo là: 1 2 4 6 7 8 9
3. Mảng sau khi đảo ngược là: 9 8 7 6 4 2 1

**Ý tưởng:**

* + **[Ý tưởng ban đầu]**Chia mảng thành hai phần bằng nhau sau đó lần lượt hoán đổi giá trị từng phần tử đối xứng nhau trong mỗi phần.
  + **[Cách chia mảng]**Nếu n lẻ, thì phần tử ở giữa (a[(n-1)/2] sẽ chia mảng thành 2 phần bằng nhau. Chú ý rằng khi lập trình ta tận dụng phép "chia nguyên" để đơn giản hóa công thức, thay vì a[(n-1)/2] ta sử dụng a[n/2] (C sẽ loại bỏ phần sau dấu chấm thập phân). Nếu n lẻ, thì ranh giới giữa 2 phần tử (a[n/2-1],a[n/2] sẽ chia mảng thành 2 phần bằng nhau.
  + **[Cách duyệt để tráo đổi]**Sau khi đã chia mảng thành 2 dãy con bằng nhau, ta cho một biến chạy i duyệt các phần tử ở dãy con bên trái. Với mỗi vị trí i, ta hoán đổi a[i] và phần tử đối xứng với nó trong dãy con bên phải (tức a[n-i-1]). Do biết trước số lần lặp (i=0→n/2-1) do vậy sử dụng vòng lặp for.

  
Trường hợp n lẻ.

  
Trường hợp n chẵn.

**Thuật toán:**

* + b1. Khai báo mảng a (và khởi gán giá trị), số phần tử n, biến tam t và biến chạy i.
  + b2. Hiển thị mảng.
  + b3. for i=0→n/2-1: hoán đổi a[i] và a[n-i-1].
  + b4. Hiển thị mảng.
  + b5. Dừng màn hình để xem kết quả.

1. **Tìm các dãy con tăng trong mảng**   
   Khởi tạo một mảng gồm n (n≤ 99) số tự nhiên sau đó hiển thị các dãy con tăng trong mảng.
2. Mảng được khởi tạo là: 1 2 8 4 7 10 9
3. Các dãy con tăng trong mảng là:
4. - Dãy con thứ 1: 1 2 8
5. - Dãy con thứ 2: 4 7 10

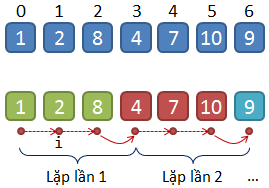
- Dãy con thứ 3: 9

**Ý tưởng:**

* + (đang cập nhật)

**Thuật toán:**

* + (đang cập nhật)



1. **Trộn mảng**   
   Khởi tạo hai mảng A và B có kích thước lần lượt là n và m phần tử (n, m ≤ 14) chứa các số nguyên trong phạm vi [1, 999]. Sắp xếp hai mảng này theo thứ tự không giảm, sau đó trộn 2 mảng lại với nhau sao cho mảng nhận được là một mảng không giảm.
2. Hai mảng ban đầu:
3. A: 4 9 7
4. B: 6 8 1 3
5. Hai mảng sau khi sắp xếp:
6. A: 4 7 9
7. B: 1 3 6 8

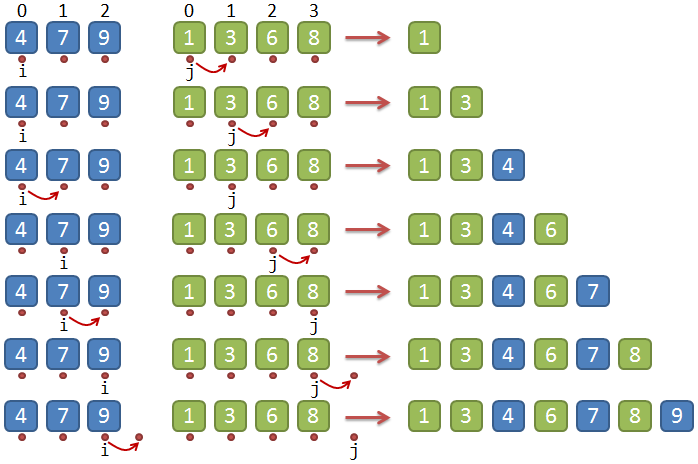
Trộn mảng A và B ta được: 1 3 4 6 7 8 9

**Ý tưởng:**

* + (đang cập nhật)

**Thuật toán:**

* + (đang cập nhật)



**Phần 3. Luyện tập (đơn giản)**

1. **Tìm giá trị phần tử lớn thứ k**   
   Nhập vào một mảng gồm n phần tử (n ≤ 99), mỗi phần tử là một số tự nhiên thuộc đoạn [0..30]. Quá trình nhập sẽ dừng khi người sử dụng nhập vào giá trị 0. Nhập tiếp một số tự nhiên k, sau đó tìm giá trị lớn thứ k trong mảng.
2. Nhập phần tử thứ 1: 1
3. Nhập phần tử thứ 2: 6
4. Nhập phần tử thứ 3: 3
5. Nhập phần tử thứ 4: 9
6. Nhập phần tử thứ 5: 7
7. Nhập phần tử thứ 6: 0
8. Mảng vừa nhập là: 1 6 3 9 7
9. Nhập k: 3

Phần tử lớn thứ 3 của mảng là: 6

1. **Kiểm tra tính đối xứng của mảng**   
   Khởi tạo một mảng số thực gồm n (n ≤ 30) phần tử, sau đó kiểm tra xem mảng này có đối xứng hay không. Một mảng http://www.forkosh.com/mimetex.cgi?%7b\%7ba_i\%7d_%7bi=0%7d%5e%7bn-1%7d%7d được gọi là đối xứng nếu http://www.forkosh.com/mimetex.cgi?%7ba_i=a_%7bn-i-1%7d,\forall%20i=\overline%7b0,n-1%7d%7d.
2. Mảng được khởi tạo là: 7.2 5 5 7.2

Mảng này đối xứng.

Mảng được khởi tạo là: 1 3.3 3.3 2

Mảng này không đối xứng.

1. **Tính giá trị trung bình các phần tử có giá trị chẵn**   
   Nhập vào một mảng gồm n ( n≤ 20) phần tử là các số tự nhiên sau đó tính trung bình cộng các phần tử có giá trị chẵn.
2. Nhập vào số phần tử của mảng: 4
3. Phần tử thứ 1: 3
4. Phần tử thứ 2: 2
5. Phần tử thứ 3: 5
6. Phần tử thứ 4: 8

Trung bình cộng các số chẵn là: 5

1. **Tìm số nguyên tố nhỏ nhất, lớn nhất**   
   Khởi tạo một mảng gồm n (n≤ 99) số tự nhiên sau đó tìm số nguyên tố lớn nhất và nhỏ nhất trong mảng (kèm số thứ tự, tính từ 1).
2. Mảng được khởi tạo là: 4 3 8 6 11 8
3. Số nguyên tố lớn nhất là: 3 (vị trí 2)

Số nguyên tố nhỏ nhất là: 11 (vị trí 5)

Mảng được khởi tạo là: 4 8 15

Các số nguyên tố trong mảng là:

:-)

1. **Tách mảng**   
   Khởi tạo mảng gồm n (n≤ 9) số tự nhiên sau đó tách mảng này thành 2 mảng A và B, sao cho mảng A chứa các số lẻ và mảng B chứa các số chẵn.
2. Mảng được khởi tạo là: 22 7 14 30 43 6 2
3. Sau khi tách, ta có:
4. - Mảng A: 7 43

- Mảng B: 22 14 30 6 2

1. **Tính các thông số thống kê (trung bình, trung vị, phương sai, độ lệch chuẩn,...)**   
   Khởi tạo dãy gồm n (n≤ 99) số tự nhiên sau đó hiển thị ra mình các thông số thống kê của dãy (nhỏ nhất, lớn nhất, trung bình, trung vị, phương sai, độ lệch chuẩn).
2. Mảng được khởi tạo là: 1 5 3 7 12
3. Giá trị nhỏ nhất (min) là: 1
4. Giá trị lớn nhất (max) là: 12
5. Giá trị trung bình (mean) là: 5.6
6. Giá trị trung vị (median) là: 5
7. Phương sai (variance) là: 14.24
8. Độ lệch chuẩn (standard deviation) là: 3.7735925

**Phần 4. Thử thách (kho khó)**

1. **Quay vòng mảng k vị trí**   
   Khởi tạo một mảng gồm n (n≤ 99) số thực và nhập một số tự nhiên k (k ≤ n) sau đó quay vòng mảng sang phải k vị trí.
2. Mảng được khởi tạo là: 1 2 3 4 6
3. Nhập k: 3

Mảng sau khi xoay sang phải 3 vị trí: 3 4 6 1 2

1. **Tìm phần tử xuất hiện nhiều nhất**   
   Khởi tạo một mảng gồm n phần tử (n ≤ 20), mỗi phần tử là một số tự nhiên thuộc đoạn [0..10] sau đó tìm giá trị xuất hiện nhiều nhất trong mảng.
2. Mảng A ban đầu: 1 4 5 4 5 4 7
3. Giá trị xuất hiện nhiều nhất là 4 (xuất hiện 3 lần).
4. **Vẽ tam giác Pascal**   
   Nhập một số tự nhiên h (h ≤ 20) từ bàn phím sau đó hiển thị ra màn hình tam giác Pascal với chiều cao h.
5. Nhập chiều cao: 5
6. 1
7. 1 1
8. 1 2 1
9. 1 3 3 1

1 4 6 4 1

1. **Tính tổng, hiệu hai số nguyên lớn**   
   Nhập vào hai số tự nhiên a, b (a, b có tối đa 50 chữ số và a > b). Tính tổng và hiệu của hai số trên.
2. Nhập a: 124356847384
3. Nhập b: 293847563
4. a + b = 124650694947

a – b = 124062999821

1. [hơi khó]**Tính tích hai số nguyên lớn**   
   Nhập vào hai số tự nhiên a, b (a, b có tối đa 50 chữ số). Tính tích của hai số trên.
2. Nhập a: 12435684
3. Nhập b: 2938

a \* b = 36536039592