ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

CHUYÊN ĐỀ
MẢNG MỘT CHIỀU

(Bản thảo ngày 18/05/2022)

Nguyễn Tấn Trần Minh Khang
Võ Duy Nguyên

THÀNH PHỐ HỒ CHÍ MINH 2022

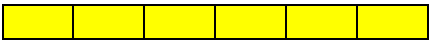
MỤC LỤC

CHƯƠNG 01. MẢNG MỘT CHIỀU CƠ BẢN

01.01 KHÁI NIỆM

- Mảng (array) một chiều là một cấu trúc dữ liệu bao gồm một dãy các phần tử có cùng cấu trúc được lưu trữ liên tiếp trong bộ nhớ.
- Mảng (array) là một tập hợp các biến có cùng kiểu và cùng tên.
- Việc truy xuất đến một phần tử trong mảng được thực hiện thông qua một biến chỉ số (index).

01.02 HÌNH ẢNH MẢNG MỘT CHIỀU



01.03 CHỈ SỐ CÁC PHẦN TỬ TRONG MẢNG MỘT CHIỀU

01.04 KHAI BÁO MẢNG MỘT CHIỀU

- Để khai báo biến có kiểu là mảng ta khai báo sau:

```
00001.  <KDL> <Tênbiến>[<hằngsố>];
```

- Trong đó:
  - KDL: là KDL của 1 phần tử trong mảng.
  - Hằng số: là 1 số nguyên dương chỉ số lượng phần tử tối đa có thể có trên một chiều trong mảng.
- Ví dụ 01:

```
00002.  int a[100];
```

- Trong ví dụ trên ta có các thông tin:
  - Tên biến mảng a.
  - Số phần tử tối đa của mảng a là 100.
  - Kiểu dữ liệu của các phần tử trong mảng là int.
  - Kích thước của biến a là 400 byte (do biến kiểu int chiếm 4 byte, điều này còn tùy thuộc vào môi trường).
- Ví dụ 02:

```
00003.  float b[50];
```

- Trong ví dụ trên ta có các thông tin:

+ Tên biến mảng `b`.
+ Số phần tử tối đa của mảng `b` là `50`.
+ Kiểu dữ liệu của các phần tử trong mảng là `float`.

01.05 KỸ THUẬT NHẬP MẢNG

01.05.01 Nhập mảng một chiều từ bàn phím

Bài cơ sở 001. Định nghĩa hàm nhập mảng một chiều các số nguyên.

– Khai báo hàm.

```
00004. void Nhap(int [],int &);
```

– Nhìn khai báo hàm trên và trả lời các câu hỏi
  + Tên hàm là gì?
  + Kiểu dữ liệu trả về?
  + Mấy tham số đầu vào?
  + Tham số thứ nhất có kiểu dữ liệu là gì?
  + Tham số thứ hai có kiểu dữ liệu là gì?
  + Loại tham số thứ nhất là gì? Tại sao?
  + Loại tham số thứ hai là gì? Tại sao?
– Các trả lời
  + Tên hàm?
    - Tên hàm là Nhap.
  + Kiểu dữ liệu trả về?
    - Không có.
  + Mấy tham số đầu vào?
    - Hai tham số đầu vào.
  + Tham số thứ nhất có kiểu dữ liệu là gì?
    - Tham số thứ nhất là mảng một chiều các số nguyên,
  + Tham số thứ hai có kiểu dữ liệu là gì?
    - Tham số thứ hai có kiểu dữ liệu là số nguyên.
  + Loại tham số thứ nhất là gì? Tại sao?
    - Tham số thứ nhất là tham biến.
    - Trước khi gọi hàm nhập ta chưa có mảng một chiều các số nguyên, sau khi gọi hàm nhập xong ta đã có mảng, như vậy tham số này phải được thiết kế là tham biến.
    - Hơn nữa trong C/C++ hàm nhận tham số đầu vào là mảng (một chiều, hai chiều, ba chiều,…) thì tham số đó là tham biến.
  + Loại tham số thứ hai là gì? Tại sao?

- Tham số thứ hai là tham biến, tham số thứ hai giúp chúng ta kiểm soát số lượng phần tử trong mảng.
- Trước khi gọi hàm nhập ta chưa số lượng phần tử trong mảng, sau khi gọi hàm nhập xong ta đã phải biết số lượng phần tử của mảng, như vậy tham số này phải được thiết kế là tham biến.

– Định nghĩa hàm.

```
00005.  void Nhap(int a[],int &n)
00006.  {
00007.     cout << "Nhap n: ";
00008.     cin >> n;
00009.     for (int i = 0; i < n; i++)
00010.     {
00011.        cout << "Nhap a[" << i << "]: ";
00012.        cin >> a[i];
00013.     }
00014.  }
```

Bài cơ sở 002. Định nghĩa hàm nhập mảng một chiều các số thực.

– Khai báo hàm.

```
00015.  void Nhap(float [],int &);
```

– Định nghĩa hàm.

```
00016.  void Nhap(float a[],int &n)
00017.  {
00018.     cout << “Nhap n: ";
00019.     cin >> n;
00020.     for (int i = 0; i < n; i++)
00021.     {
00022.        cout << “Nhap a["<<i<<"]: ";
00023.        cin >> a[i];
00024.     }
00025.  }
```

01.05.01 Tạo ngẫu nhiên mảng một chiều

Bài cơ sở 003. Định nghĩa hàm tạo ngẫu nhiên mảng một chiều các số nguyên.

– Khai báo hàm.

```
00026.  void Nhap(int [],int &);
```

– Định nghĩa hàm.

```
00027.  void Nhap(int a[],int &n)
00028.  {
00029.     cout << "Nhap n: ";
00030.     cin >> n;
00031.     srand(time(NULL));
00032.     for (int i = 0; i < n; i++)
00033.        a[i] = rand()%(200+1) - 100;
00034.  }
```

Bài cơ sở 004. Định nghĩa hàm tạo ngẫu nhiên mảng một chiều các số thực.

– Khai báo hàm.

```
00035.  void Nhap(float [],int &);
```

– Định nghĩa hàm.

```
00036.  void Nhap(float a[],int &n)
00037.  {
00038.     cout << "Nhap n: ";
00039.     cin >> n;
00040.     srand(time(NULL));
00041.     for (int i = 0; i < n; i++)
00042.        a[i] = -100.0 +
00043.           (rand()/(RAND_MAX/(100.0-(-100.0))));
00044.  }
```

01.05.02 Nhập mảng một chiều từ tập tin

Bài cơ sở 005. Định nghĩa hàm nhập mảng một chiều các số nguyên từ file.

– Khai báo hàm.

```
00045.  void Nhap(int [], int &, string);
```

– Định nghĩa hàm.

```
00046.  void Nhap(int a[], int &n, string filename)
00047.  {
00048.     ifstream fi(filename);
00049.     fi >> n;
00050.     for(int i=0; i<=n-1; i++)
00051.     {
00052.        fi >> a[i];
00053.     }
00054.  }
```

Bài cơ sở 006. Định nghĩa hàm nhập mảng một chiều các số thực từ file.

- Khai báo hàm.

```
00055. void Nhap(float [], int &, string);
```

- Định nghĩa hàm.

```
00056. void Nhap(float a[], int &n, string filename)
00057. {
00058.    ifstream fi(filename);
00059.    fi >> n;
00060.    for(int i=0; i<=n-1; i++)
00061.    {
00062.       fi >> a[i];
00063.    }
00064. }
```

01.06 KỸ THUẬT XUẤT MẢNG

01.06.01 Xuất mảng một chiều ra màn hình

Bài cơ sở 007. Định nghĩa hàm xuất mảng một chiều các số nguyên ra màn hình.

- Khai báo hàm.

```
00065. void Xuat(int [],int);
```

- Nhìn khai báo hàm trên và trả lời các câu hỏi
  - + Tên hàm là gì?
  - + Kiểu dữ liệu trả về?
  - + Mấy tham số đầu vào?
  - + Tham số thứ nhất có kiểu dữ liệu là gì?
  - + Tham số thứ hai có kiểu dữ liệu là gì?
  - + Loại tham số thứ nhất là gì? Tại sao?
  - + Loại tham số thứ hai là gì? Tại sao?
- Các trả lời
  - + Tên hàm?
    - Tên hàm là Xuat.
  - + Kiểu dữ liệu trả về?
    - Không có.
  - + Mấy tham số đầu vào?
    - Hai tham số đầu vào.
  - + Tham số thứ nhất có kiểu dữ liệu là gì?

- Tham số thứ nhất là mảng một chiều các số nguyên.

+ Tham số thứ hai có kiểu dữ liệu là gì?
  - Tham số thứ hai có kiểu dữ liệu là số nguyên.
+ Loại tham số thứ nhất là gì? Tại sao?
  - Trong C/C++ hàm nhận tham số đầu vào là mảng (một chiều, hai chiều, ba chiều,…) thì tham số đó là tham biến.
  - Tham số thứ nhất là tham biến. Tham số này tuy là tham biến nhưng được sử dụng trong hàm như là tham trị.
  - Trước khi gọi hàm xuất ta đã có mảng một chiều các số nguyên
  - Sau khi hàm Xuat được gọi thực hiện xong giá các phần tử trong mảng không thay đổi.
+ Loại tham số thứ hai là gì? Tại sao?
  - Tham số thứ hai là tham trị, tham số thứ hai giữ thông tin về số lượng phần tử trong mảng.
  - Trước khi gọi hàm xuất ta đã có thông tin về số lượng các phần tử trong mảng.
  - Sau khi hàm Xuat được gọi thực hiện xong các số lượng các phần tử trong mảng không thay đổi.

– Định nghĩa hàm.

```
00066.  void Xuat(int a[],int n)
00067.  {
00068.     for(int i=0; i<=n-1; i++)
00069.        cout << setw(8) << a[i];
00070.  }
```

Bài cơ sở 008. Định nghĩa hàm xuất mảng một chiều các số thực ra màn hình.

– Khai báo hàm.

```
00071.  void Xuat(float [],int);
```

– Định nghĩa hàm.

```
00072.  void Xuat(float a[], int n)
00073.  {
00074.     for(int i=0; i<=n-1; i++)
00075.        cout<<setw(8)<<setprecision(5)<<a[i];
00076.  }
```

01.06.02 Xuất mảng một chiều ra tập tin

> Bài cơ sở 009. Định nghĩa hàm xuất mảng một chiều các số nguyên ra tập tin.

– Khai báo hàm.

```
00077. void Xuat(int [], int, string);
```

– Định nghĩa hàm.

```
00078. void Xuat(int a[], int n, string filename)
00079. {
00080.    ofstream fo(filename);
00081.
00082.    fo<<n<<endl;
00083.    for(int i=0; i<=n-1; i++)
00084.       fo << setw(8) << a[i];
00085. }
```

> Bài cơ sở 010. Định nghĩa hàm xuất mảng một chiều các số thực ra tập tin.

– Khai báo hàm.

```
00086. void Xuat(float [], int, string);
```

– Định nghĩa hàm.

```
00087. void Xuat(float a[], int n, string filename)
00088. {
00089.    ofstream fo(filename);
00090.
00091.    fo << n << endl;
00092.    for(int i=0; i<=n-1; i++)
00093.       fo<<setw(8)<<setprecision(5)<<a[i];
00094. }
```

01.07 CÁC HÀM CƠ BẢN THƯỜNG SỬ DỤNG

01.07.01 Hoán vị

> Bài cơ sở 011. Định nghĩa hàm hoán vị giá trị của hai biến số nguyên.

– Khai báo hàm.

```
00095. void HoanVi(int &,int &);
```

– Định nghĩa hàm.

```
00096.  void HoanVi(int &a,int &b)
00097.  {
00098.     int temp = a;
00099.     a = b;
00100.     b = temp;
00101.  }
```

– Cách 02: Khai báo hàm.

```
00102.  void HoanVi(int *,int *);
```

– Cách 02: Định nghĩa hàm.

```
00103.  void HoanVi(int *a,int *b)
00104.  {
00105.     int temp = *a;
00106.     *a = *b;
00107.     *b = temp;
00108.  }
```

– Lưu ý: Từ đây về sau chúng ta không sử dụng cách 02 trong lập trình.

Bài cơ sở 012. Định nghĩa hàm hoán vị giá trị của hai biến số thực.

– Khai báo hàm.

```
00109.  void HoanVi(float &, float &);
```

– Định nghĩa hàm.

```
00110.  void HoanVi(float &a, float &b)
00111.  {
00112.     float temp = a;
00113.     a = b;
00114.     b = temp;
00115.  }
```

01.07.02 Kiểm tra nguyên tố

Bài cơ sở 013. Định nghĩa hàm kiểm tra một số nguyên *k* có phải số nguyên tố hay không?

– Khai báo hàm.

```
00116.  bool ktNguyenTo(int);
```

– Cách 01: Định nghĩa hàm.

```
00117.  bool ktNguyenTo(int k)
00118.  {
```

```
00119.      int dem = 0;
00120.      for(int i=1; i<=k; i++)
00121.          if(k%i==0)
00122.              dem++;
00123.      if(dem==2)
00124.          return true;
00125.      else
00126.          return false;
00127.  }
```

– Cách 02: Định nghĩa hàm.

```
00128.  bool ktNguyenTo(int k)
00129.  {
00130.      int dem = 0;
00131.      for(int i=1; i<=k; i++)
00132.          if(k%i==0)
00133.              dem++;
00134.      if(dem==2)
00135.          return true;
00136.      return false;
00137.  }
```

– Cách 03: Định nghĩa hàm.

```
00138.  bool ktNguyenTo(int k)
00139.  {
00140.      int dem = 0;
00141.      for(int i=1; i<=k; i++)
00142.          if(k%i==0)
00143.              dem++;
00144.      return (dem==2);
00145.  }
```

– Câu hỏi: Còn cách nào tốt hơn không?
– Trả lời: Còn nhiều cách lắm, người ơi!!!!!

01.07.03 Kiểm tra chính phương

> Bài cơ sở 014. Định nghĩa hàm kiểm tra một số nguyên *k* có phải số chính phương hay không?

– Khai báo hàm.

```
00146.  bool ktChinhPhuong(int);
```

– Cách 01: Định nghĩa hàm.

```
00147.  bool ktChinhPhuong(int k)
```

```
00148. {
00149.     bool flag = false;
00150.     for(int i=0; i<=k;i++)
00151.         if(i*i==k)
00152.             flag = true;
00153.     return flag;
00154. }
```

01.07.04 Kiểm tra hoàn thiện

Bài cơ sở 015. Định nghĩa hàm kiểm tra một số nguyên *k* có phải là số hoàn thiện hay không?

– Khai báo hàm.

```
00155. bool ktHoanThien(int);
```

– Cách 01: Định nghĩa hàm.

```
00156. bool ktHoanThien(int k)
00157. {
00158.     int s = 0;
00159.     for(int i=1; i<k; i++)
00160.         if(k%i==0)
00161.             s = s + i;
00162.     if(s==k)
00163.         return true;
00164.     return false;
00165. }
```

– Cách 02: Định nghĩa hàm.

```
00166. bool ktHoanThien(int k)
00167. {
00168.     int s = 0;
00169.     for(int i=1; i<k; i++)
00170.         if(k%i==0)
00171.             s = s + i;
00172.     return (s==k);
00173. }
```

– Cách 03: Định nghĩa hàm.

```
00174. bool ktHoanThien(int k)
00175. {
00176.     int s = 0;
00177.     for(int i=1;i<=k/2;i++)
00178.         if(k%i==0)
```

```
00179.              s = s + i;
00180.      return (s==k);
00181.  }
```

01.07.05 Kiểm tra đối xứng

Bài cơ sở 016. Định nghĩa hàm kiểm tra một số nguyên *k* có đối xứng hay không?

– Khai báo hàm.

```
00182.  bool ktDoiXung(int);
```

– Cách 01: Định nghĩa hàm.

```
00183.  bool ktDoiXung(int k)
00184.  {
00185.      k = abs(k);
00186.      int dn = 0;
00187.      for(int t=k; t!=0; t=t/10)
00188.      {
00189.          int dv = t%10;
00190.          dn = dn * 10 + dv;
00191.      }
00192.      if(dn==k)
00193.          return true;
00194.      return false;
00195.  }
```

– Cách 02: Định nghĩa hàm.

```
00196.  bool ktDoiXung(int k)
00197.  {
00198.      k = abs(k);
00199.      int dn = 0;
00200.      for(int t=k; t!=0; t=t/10)
00201.      {
00202.          int dv = t%10;
00203.          dn = dn * 10 + dv;
00204.      }
00205.      return (dn==k);
00206.  }
```

– Cách 03: Định nghĩa hàm.

```
00207.  bool ktDoiXung(int k)
00208.  {
00209.      k = abs(k);
```

```
00210.      int dn = 0;
00211.      for(int t=k; t!=0; t=t/10)
00212.          dn = dn * 10 + t%10;
00213.      return (dn==k);
00214.  }
```

– Cách 04: Định nghĩa hàm.

```
00215.  bool ktDoiXung(int k)
00216.  {
00217.      int dn = 0;
00218.      for(int t=k=abs(k); t!=0; t=t/10)
00219.          dn = dn * 10 + t%10;
00220.      return (dn==k);
00221.  }
```

01.07.06 Kiểm tra toàn chẵn – toàn lẻ

Bài cơ sở 017. Định nghĩa hàm kiểm tra một số nguyên *k* có toàn chữ số lẻ hay không?

– Khai báo hàm.

```
00222.  bool ktToanLe(int);
```

– Cách 01: Định nghĩa hàm.

```
00223.  bool ktToanLe(int k)
00224.  {
00225.      if(k==0)
00226.          return false;
00227.      bool flag = true;
00228.      for(int t=k=abs(k); t!=0; t=t/10)
00229.      {
00230.          int dv = t%10;
00231.          if(dv%2==0)
00232.              flag = false;
00233.      }
00234.      return flag;
00235.  }
```

– Cách 02: Định nghĩa hàm.

```
00236.  bool ktToanLe(int k)
00237.  {
00238.      if(k==0)
00239.          return false;
00240.      bool flag = true;
```

```
00241.      for(int t=k=abs(k);t!=0;t=t/10)
00242.          if((t%10)%2==0)
00243.              flag = false;
00244.      return flag;
00245.  }
```

– Cách 03: Định nghĩa hàm.

```
00246.  bool ktToanLe(int k)
00247.  {
00248.      bool flag = true;
00249.      for(int t=k=abs(k); t!=0; t=t/10)
00250.          if(t%2==0)
00251.              flag = false;
00252.      return flag;
00253.  }
```

Bài cơ sở 018. Định nghĩa hàm kiểm tra một số nguyên *k* có toàn chữ số chẵn hay không?

– Khai báo hàm.

```
00254.  bool ktToanChan(int);
```

– Cách 01: Định nghĩa hàm.

```
00255.  bool ktToanChan(int k)
00256.  {
00257.      bool flag = true;
00258.      for(int t=k=abs(k); t!=0; t=t/10)
00259.      {
00260.          int dv = t%10;
00261.          if(dv%2!=0)
00262.              flag = false;
00263.      }
00264.      return flag;
00265.  }
```

– Cách 02: Định nghĩa hàm.

```
00266.  bool ktToanChan(int k)
00267.  {
00268.      bool flag = true;
00269.      for(int t=k=abs(k); t!=0; t=t/10)
00270.          if((t%10)%2!=0)
00271.              flag = false;
00272.      return flag;
00273.  }
```

– Cách 02: Định nghĩa hàm.

```
00274.  bool ktToanLe(int k)
00275.  {
00276.      bool flag = true;
00277.      for(int t=k=abs(k); t!=0; t=t/10)
00278.          if(t%2!=0)
00279.              flag = false;
00280.      return flag;
00281.  }
```

01.07.07 Kiểm tra số nguyên có dạng 2^m, 3^m, 5^m

Bài cơ sở 019. Định nghĩa hàm kiểm tra số nguyên k có dạng 2^m ($m \in N$ và $m \geq 0$) hay không?

– Khai báo hàm.

```
00282.  bool ktDang2m(int);
```

– Cách 01: Định nghĩa hàm.

```
00283.  bool ktDang2m(int k)
00284.  {
00285.      if(k<1)
00286.          return false;
00287.      bool flag = true;
00288.      for(int t=k;t>1;t=t/2)
00289.      {
00290.         int du = t%2;
00291.         if(du!=0)
00292.            flag = false;
00293.      }
00294.      return flag;
00295.  }
```

– Cách 02: Định nghĩa hàm.

```
00296.  bool ktDang2m(int k)
00297.  {
00298.      if(k<1)
00299.          return false;
00300.      bool flag = true;
00301.      for(int t=k;t>1;t=t/2)
00302.         if(t%2!=0)
00303.            flag = false;
00304.      return flag;
00305.  }
```

Bài cơ sở 020. Định nghĩa hàm kiểm tra số nguyên k có dạng 3^m (m∈N và m ≥ 0) hay không?

– Khai báo hàm.

```
00306. bool ktDang3m(int);
```

– Định nghĩa hàm.

```
00307. bool ktDang3m(int k)
00308. {
00309.     if(k<1)
00310.         return false;
00311.     bool flag = true;
00312.     for(int t=k;t>1;t=t/3)
00313.        if(t%3!=0)
00314.           flag = false;
00315.     return flag;
00316. }
```

Bài cơ sở 021. Định nghĩa hàm kiểm tra số nguyên k có dạng 5^m (m∈N và m ≥ 0) hay không?

– Khai báo hàm.

```
00317. bool ktDang5m(int);
```

– Định nghĩa hàm.

```
00318. bool ktDang5m(int k)
00319. {
00320.     if(k<1)
00321.         return false;
00322.     bool flag = true;
00323.     for(int t=k;t>1;t=t/5)
00324.        if(t%5!=0)
00325.           flag = false;
00326.     return flag;
00327. }
```

01.07.08 Tìm chữ số đầu tiên của một số nguyên

Bài cơ sở 022. Định nghĩa hàm tìm chữ số đầu tiên của số nguyên k?

– Khai báo hàm.

```
00328. int ChuSoDau(int);
```

– Cách 01: Định nghĩa hàm.

```
00329.  int ChuSoDau(int k)
00330.  {
00331.      int dt = abs(k);
00332.      while(dt>=10)
00333.          dt = dt / 10;
00334.      return dt;
00335.  }
```

– Cách 02: Định nghĩa hàm.

```
00336.  int ChuSoDau(int k)
00337.  {
00338.      for(int dt = abs(k);dt>=10;dt = dt/10);
00339.      return dt;
00340.  }
```

01.07.09 Tìm giá trị đảo ngược của một số nguyên

Bài cơ sở 023. Định nghĩa hàm tìm giá trị đảo ngược của số nguyên k?

– Khai báo hàm.

```
00341.  int DaoNguoc(int);
```

– Định nghĩa hàm (cách 01)

```
00342.  int DaoNguoc(int k)
00343.  {
00344.      int dn = 0;
00345.      for(int t=k;t!=0;t=t/10)
00346.      {
00347.          int dv = t%10;
00348.          dn = dn * 10 + dv;
00349.      }
00350.      return dn;
00351.  }
```

– Hàm trên đúng cho các trường hợp:

+ k là số dương
+ k là số âm
+ k bằng không

– Cách 02: Gọn gàng, súc tích, cải tiến từ cách trên..

```
00352.  int DaoNguoc(int k)
00353.  {
00354.      int dn = 0;
```

```
00355.      for(int t=k;t!=0;t=t/10)
00356.          dn = dn * 10 + t%10;
00357.      return dn;
00358.  }
```

01.07.10 Đếm số lượng chữ số của một số nguyên

> Bài cơ sở 024. Định nghĩa hàm đếm số lượng chữ số của một số nguyên?

– Khai báo hàm.

```
00359.  int DemChuSo(int);
```

– Định nghĩa hàm.

```
00360.  int DemChuSo(int k)
00361.  {
00362.     if(k==0)
00363.        return 1;
00364.     int dem = 0;
00365.     for(int t=k;t!=0;t=t/10)
00366.        dem++;
00367.     return dem;
00368.  }
```

– Hàm trên đúng cho các trường hợp:

+ k là số dương
+ k là số âm
+ k bằng không

01.07.11 Tính tổng các chữ số của một số nguyên

– Khai báo hàm.

```
00369.  int TongChuSo(int);
```

– Định nghĩa hàm.

```
00370.  int TongChuSo(int k)
00371.  {
00372.     int S = 0;
00373.     for(int t=abs(k);t!=0;t=t/10)
00374.        S = S + t%10;
00375.     return S;
00376.  }
```

01.07.12 Tìm ước chung lớn nhất của hai số nguyên

– Khai báo hàm.

```
00377.  int UocChungLonNhat(int,int);
```

– Định nghĩa hàm.

```
00378.  int UocChungLonNhat(int a,int b)
00379.  {
00380.     a = abs(a);
00381.     b = abs(b);
00382.     while(a*b!=0)
00383.        if(a>b)
00384.           a=a-b;
00385.        else
00386.           b=b-a;
00387.     return (a+b);
00388.  }
```

– Cách 2

```
00389.  int UocChungLonNhat(int a,int b)
00390.  {
00391.     a = abs(a);
00392.     b = abs(b);
00393.     while(a*b!=0)
00394.        if(a>b)
00395.           a = a % b;
00396.        else
00397.           b = b % a;
00398.     return (a+b);
00399.  }
```

01.07.13 Tìm bội chung nhỏ nhất của hai số nguyên

– Khai báo hàm.

```
00400.  int BoiChungLonNhat(int,int);
```

– Định nghĩa hàm.

```
00401.  int BoiChungLonNhat(int a,int b)
00402.  {
00403.     return abs(a*b)/UocChungLonNhat(a,b);
00404.  }
```

01.08 KỸ THUẬT LIỆT KÊ

Bài cơ sở 025. Định nghĩa hàm liệt kê các giá trị lẻ trong mảng một chiều các số nguyên?

- Ví dụ:

19	29	62	76	99	23	12	98	88	11

- Các giá trị lẻ có trong mảng

19	29	62	76	99	23	12	98	88	11

- Kết quả: 19, 29, 99, 23, 11.
- Khai báo hàm.

```
00405. void LietKe(int [],int);
```

- Định nghĩa hàm.

```
00406. void LietKe(int a[],int n)
00407. {
00408.    for(int i=0;i<=n-1;i++)
00409.       if(a[i]%2!=0)
00410.          cout << setw(8) << a[i];
00411. }
```

Chương trình 001. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều.
b. Xuất mảng.
c. Liệt kê các giá trị lẻ có trong mảng.

- Chương trình.

```
00412. #include <iostream>
00413. #include <iomanip>
00414. #include <cstdlib>
00415. #include <ctime>
00416. using namespace std;
00417.
00418. void Nhap(int[],int&);
00419. void Xuat(int[],int);
00420. void LietKe(int [],int);
00421.
00422. int main()
00423. {
00424.    int b[100];
00425.    int k;
00426.
00427.    Nhap(b, k);
00428.    cout << "\nMang ban dau:";
```

```
00429.    Xuat(b, k);
00430.
00431.    cout << "\nCac gia tri le:";
00432.    LietKe(b, k);
00433.    return 0;
00434. }
00435.
00436. void Nhap(int a[],int &n)
00437. {
00438.    cout << "Nhap n: ";
00439.    cin >> n;
00440.    srand(time(NULL));
00441.    for (int i = 0; i < n; i++)
00442.       a[i] = rand()%(200+1) - 100;
00443. }
00444.
00445. void Xuat(int a[],int n)
00446. {
00447.    for (int i = 0; i < n; i++)
00448.       cout<<setw(8)<<a[i];
00449. }
00450.
00451. void LietKe(int a[],int n)
00452. {
00453.    for (int i = 0; i < n; i++)
00454.       if(a[i]%2!=0)
00455.          cout<<setw(8)<<a[i];
00456. }
```

Chương trình 002. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ các file có tên intdata01.inp; intdata02.inp; intdata03.inp; intdata04.inp; intdata05.inp; intdata06.inp; intdata07.inp; intdata08.inp; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
b. Xuất mảng.
c. Liệt kê các giá trị lẻ có trong mảng.

– Chương trình.

```
00457. #include <iostream>
00458. #include <iomanip>
00459. #include <fstream>
00460. #include <string>
00461. using namespace std;
```

```
00462.
00463.  string getInputFileName(int);
00464.  string getOutputFileName(int);
00465.  void Nhap(int[], int &, string);
00466.  void Xuat(int[], int, string);
00467.
00468.  void Xuat(int[], int);
00469.  void LietKeLe(int[], int);
00470.
00471.  int main()
00472.  {
00473.     int b[100000];
00474.     int n;
00475.     for (int i = 1; i <= 13; i++)
00476.     {
00477.        string FileName = getInputFileName(i);
00478.        Nhap(b, n, FileName);
00479.
00480.        cout << "\n" << FileName << endl;
00481.        LietKeLe(b, n);
00482.
00483.        FileName = getOutputFileName(i);
00484.        Xuat(b, n, FileName);
00485.     }
00486.     return 1;
00487.  }
00488.
00489.  string getInputFileName(int n)
00490.  {
00491.     string fileName = string("intdata");
00492.     if (n < 10)
00493.        fileName += "0" + to_string(n) + ".inp";
00494.     else
00495.        fileName += to_string(n) + ".inp";
00496.     return fileName;
00497.  }
00498.
00499.  string getOutputFileName(int n)
00500.  {
00501.     string fileName = string("intdata");
00502.     if (n < 10)
00503.        fileName += "0" + to_string(n) + ".out";
00504.     else
```

```
00505.       fileName += to_string(n) + ".out";
00506.    return fileName;
00507. }
00508.
00509. void Nhap(int a[], int &n, string filename)
00510. {
00511.    ifstream fi(filename);
00512.    fi >> n;
00513.    for (int i = 0; i < n; i++)
00514.       fi >> a[i];
00515. }
00516.
00517. void Xuat(int a[], int n, string filename)
00518. {
00519.    ofstream fo(filename);
00520.    fo << n << endl;
00521.    for (int i = 0; i < n; i++)
00522.       fo << setw(8) <<  a[i];
00523. }
00524.
00525. void Xuat(int a[], int n)
00526. {
00527.    for (int i = 0; i < n; i++)
00528.       cout << setw(8) << a[i];
00529. }
00530.
00531. void LietKeLe(int a[], int n)
00532. {
00533.    for (int i = 0; i < n; i++)
00534.       if(a[i]%2!=0)
00535.          cout << setw(8) << a[i];
00536. }
```

01.09 KỸ THUẬT TÍNH TOÁN

Bài cơ sở 026. Định nghĩa hàm tính tổng các giá trị âm trong mảng một chiều các số thực?

- Ví dụ:

19.87	-28.72	-12.78	11.23	-4.36	87.25

- Các giá trị âm có trong mảng

19.87	-28.72	-12.78	11.23	-4.36	87.25

- Kết quả: $(-28.72) + (-12.78) + (-4.36) = -45.86$.

– Khai báo hàm.

```
00537. float TongAm(float [],int);
```

– Định nghĩa hàm.

```
00538. float TongAm(float a[],int n)
00539. {
00540.    float s = 0;
00541.    for (int i = 0; i < n; i++)
00542.       if(a[i]<0)
00543.          s = s + a[i];
00544.    return s;
00545. }
```

Chương trình 003. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều các số thực.
b. Xuất mảng.
c. Tính tổng các giá trị âm có trong mảng.

– Chương trình.

```
00546. #include <iostream>
00547. #include <iomanip>
00548. #include <cstdlib>
00549. #include <ctime>
00550. using namespace std;
00551.
00552. void Nhap(float[],int &);
00553. void Xuat(float[],int);
00554. float TongAm(float[],int);
00555.
00556. int main()
00557. {
00558.    float b[100];
00559.    int k;
00560.
00561.    Nhap(b, k);
00562.
00563.    cout << “\nMang ban dau:";
00564.    Xuat(b, k);
00565.
00566.    float kq = TongAm(b, k);
00567.    cout << “\nTong cac gia tri am: "<<kq;
00568.    return 0;
00569. }
```

```
00570.
00571.  void Nhap(float a[],int &n)
00572.  {
00573.     cout << “Nhap n: ";
00574.     cin >> n;
00575.     srand(time(NULL));
00576.     for (int i = 0; i < n; i++)
00577.        a[i] = -100.0 +
00578.           (rand()/(RAND_MAX/(100.0-(-100.0))));
00579.  }
00580.
00581.  void Xuat(float a[],int n)
00582.  {
00583.     for (int i = 0; i < n; i++)
00584.        cout<<setw(8)<<setprecision(5)<<a[i];
00585.  }
00586.
00587.  float TongAm(float a[],int n)
00588.  {
00589.     float s = 0;
00590.     for (int i = 0; i < n; i++)
00591.        if(a[i]<0)
00592.           s = s + a[i];
00593.     return s;
00594.  }
```

Chương trình 004. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ các file có tên floatdata01.inp; floatdata02.inp; floatdata03.inp; floatdata04.inp; floatdata05.inp; floatdata06.inp; floatdata07.inp; floatdata08.inp; floatdata09.inp; floatdata10.inp; floatdata11.inp; floatdata12.inp; floatdata13.inp;
b. Xuất mảng.
c. Tính tổng các giá trị âm có trong mảng.

– Chương trình.

```
00595.  #include <iostream>
00596.  #include <iomanip>
00597.  #include <fstream>
00598.  #include <string>
00599.  using namespace std;
00600.
00601.  string getInputFileName(int);
```

```
00602.  string getOutputFileName(int);
00603.  void Nhap(float[], int&, string);
00604.  void Xuat(float[], int, string);
00605.
00606.  void Xuat(float[], int);
00607.  float TongAm(float[], int);
00608.
00609.  int main()
00610.  {
00611.     float b[100000];
00612.     int n;
00613.     for (int i = 1; i <= 13; i++)
00614.     {
00615.        string FileName = getInputFileName(i);
00616.        Nhap(b, n, FileName);
00617.
00618.        cout << "\n" << FileName << endl;
00619.        Xuat(b, n);
00620.
00621.        FileName = getOutputFileName(i);
00622.        Xuat(b, n, FileName);
00623.
00624.        float kq;
00625.        kq = TongAm(b, n);
00626.        cout << "\nTong cac gia tri am trong mang =
" << setprecision(5) << kq;
00627.     }
00628.     return 1;
00629.  }
00630.
00631.  string getInputFileName(int n)
00632.  {
00633.     string fileName = string("floatdata");
00634.     if (n < 10)
00635.        fileName += "0" + to_string(n) + ".inp";
00636.     else
00637.        fileName += to_string(n) + ".inp";
00638.     return fileName;
00639.  }
00640.
00641.  string getOutputFileName(int n)
00642.  {
00643.     string fileName = string("floatdata");
```

```
00644.    if (n < 10)
00645.       fileName += "0" + to_string(n) + ".out";
00646.    else
00647.       fileName += to_string(n) + ".out";
00648.    return fileName;
00649. }
00650.
00651. void Nhap(float a[], int& n, string filename)
00652. {
00653.    ifstream fi(filename);
00654.    fi >> n;
00655.    for (int i = 0; i < n; i++)
00656.       fi >> a[i];
00657. }
00658.
00659. void Xuat(float a[], int n, string filename)
00660. {
00661.    ofstream fo(filename);
00662.    fo << n << endl;
00663.    for (int i = 0; i < n; i++)
00664.       fo<<setw(8)<<setprecision(5)<<a[i];
00665. }
00666.
00667. void Xuat(float a[], int n)
00668. {
00669.    for (int i = 0; i < n; i++)
00670.       cout<<setw(8)<<setprecision(5)<<a[i];
00671. }
00672.
00673. float TongAm(float a[], int n)
00674. {
00675.    float s = 0;
00676.    for (int i = 0; i < n; i++)
00677.       if (a[i] < 0)
00678.          s = s + a[i];
00679.    return s;
00680. }
```

01.10 KỸ THUẬT ĐẾM

Bài cơ sở 027. Định nghĩa hàm đếm số lượng số nguyên tố nhỏ hơn 100 trong mảng?

– Ví dụ:

19	29	62	76	223	23	227	98	88	53

– Các số nguyên tố có trong mảng

19	29	62	76	223	23	227	98	88	53

– Kết quả: 4 (19, 29, 23, 53).

– Khai báo hàm.

```
00681. int DemNguyenTo(int [],int);
```

– Cách 01: định nghĩa hàm.

```
00682. int DemNguyenTo(int a[],int n)
00683. {
00684.    int dem = 0;
00685.    for (int i = 0; i < n; i++)
00686.       if(ktNguyenTo(a[i])==true && a[i]<100)
00687.          dem = dem + 1;
00688.    return dem;
00689. }
```

– Cách 02: Cải tiến.

```
00690. int DemNguyenTo(int a[],int n)
00691. {
00692.    int dem = 0;
00693.    for (int i = 0; i < n; i++)
00694.       if(a[i]<100 && ktNguyenTo(a[i])==true)
00695.          dem = dem + 1;
00696.    return dem;
00697. }
```

Chương trình 005. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều các số nguyên.
b. Xuất mảng.
c. Đếm số lượng số nguyên tố nhỏ hơn 100 có trong mảng.

– Chương trình.

```
00698. #include <iostream>
00699. #include <iomanip>
00700. #include <cstdlib>
00701. #include <ctime>
00702. using namespace std;
00703.
00704. void Nhap(int[],int&);
00705. void Xuat(int[],int);
00706. bool ktNguyenTo(int);
00707. int DemNguyenTo(int[],int);
```

```
00708.
00709.  int main()
00710.  {
00711.     int b[100];
00712.     int k;
00713.
00714.     Nhap(b, k);
00715.     cout << “\nMang ban dau:";
00716.     Xuat(b, k);
00717.
00718.     int kq = DemNguyenTo(b,k);
00719.     cout << “\nKet qua: "<<kq;
00720.     return 0;
00721.  }
00722.
00723.  void Nhap(int a[],int &n)
00724.  {
00725.     cout << “Nhap n: ";
00726.     cin >> n;
00727.     srand(time(NULL));
00728.     for (int i = 0; i < n; i++)
00729.        a[i] = rand()%(200+1) - 100;
00730.  }
00731.
00732.  bool ktNguyenTo(int k)
00733.  {
00734.     int dem = 0;
00735.     for(int i=1; i<=k; i++)
00736.        if(k%i==0)
00737.           dem++;
00738.     if(dem==2)
00739.        return true;
00740.     return false;
00741.  }
00742.
00743.  void Xuat(int a[],int n)
00744.  {
00745.     for (int i = 0; i < n; i++)
00746.        cout<<setw(8)<<a[i];
00747.  }
00748.
00749.  int DemNguyenTo(int a[],int n)
00750.  {
```

```
00751.     int dem = 0;
00752.     for (int i = 0; i < n; i++)
00753.        if(a[i]<100 && ktNguyenTo(a[i]))
00754.           dem = dem + 1;
00755.     return dem;
00756.  }
```

Chương trình 006. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ các file có tên intdata01.inp; intdata02.inp; intdata03.inp; intdata04.inp; intdata05.inp; intdata06.inp; intdata07.inp; intdata08.inp; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
b. Xuất mảng.
c. Đếm số lượng số nguyên tố nhỏ hơn 100 có trong mảng.

– Chương trình.

```
00757.  #include <iostream>
00758.  #include <iomanip>
00759.  #include <fstream>
00760.  #include <string>
00761.  using namespace std;
00762.
00763.  string getInputFileName(int);
00764.  string getOutputFileName(int);
00765.  void Nhap(int[], int&, string);
00766.  void Xuat(int[], int, string);
00767.
00768.  void Xuat(int[], int);
00769.  bool ktNguyenTo(int);
00770.  int DemNguyenTo(int[], int);
00771.
00772.  int main()
00773.  {
00774.     int b[100000];
00775.     int n;
00776.     for (int i = 1; i <= 13; i++)
00777.     {
00778.        string FileName = getInputFileName(i);
00779.        Nhap(b, n, FileName);
00780.
00781.        cout << "\n" << FileName << endl;
00782.        Xuat(b, n);
00783.
```

```
00784.        FileName = getOutputFileName(i);
00785.        Xuat(b, n, FileName);
00786.
00787.        int dem;
00788.        dem = DemNguyenTo(b, n);
00789.        cout << "\nSo luong so nguyen to nho hon
100 trong mang = " << dem;
00790.     }
00791.     return 1;
00792.  }
00793.
00794.  string getInputFileName(int n)
00795.  {
00796.     string fileName = string("intdata");
00797.     if (n < 10)
00798.        fileName += "0" + to_string(n) + ".inp";
00799.     else
00800.        fileName += to_string(n) + ".inp";
00801.     return fileName;
00802.  }
00803.
00804.  string getOutputFileName(int n)
00805.  {
00806.     string fileName = string("intdata");
00807.     if (n < 10)
00808.        fileName += "0" + to_string(n) + ".out";
00809.     else
00810.        fileName += to_string(n) + ".out";
00811.     return fileName;
00812.  }
00813.
00814.  void Nhap(int a[], int& n, string filename)
00815.  {
00816.     ifstream fi(filename);
00817.     fi >> n;
00818.     for (int i = 0; i < n; i++)
00819.        fi >> a[i];
00820.  }
00821.
00822.  void Xuat(int a[], int n, string filename)
00823.  {
00824.     ofstream fo(filename);
00825.     fo << n << endl;
```

```
00826.    for (int i = 0; i < n; i++)
00827.       fo << setw(8) << a[i];
00828. }
00829.
00830. void Xuat(int a[], int n)
00831. {
00832.    for (int i = 0; i < n; i++)
00833.       cout << setw(8) << a[i];
00834. }
00835.
00836. bool ktNguyenTo(int k)
00837. {
00838.    int dem = 0;
00839.    for (int i = 1; i <= k; i++)
00840.       if (k % i == 0)
00841.          dem++;
00842.    if (dem == 2)
00843.       return true;
00844.    return false;
00845. }
00846.
00847. int DemNguyenTo(int a[], int n)
00848. {
00849.    int dem = 0;
00850.    for (int i = 0; i < n; i++)
00851.       if (a[i] < 100 && ktNguyenTo(a[i]))
00852.          dem = dem + 1;
00853.    return dem;
00854. }
```

01.11 KỸ THUẬT TÌM KIẾM – KỸ THUẬT ĐẶT LÍNH CANH

Bài cơ sở 028. Định nghĩa hàm tìm giá trị lớn nhất trong mảng một chiều các số thực.

- Ví dụ:

19	29	62	76	223	23	227	98	88	53

- Kết quả: 227.
- Khai báo hàm.

```
00855. float LonNhat(float [],int);
```

- Định nghĩa hàm.

```
00856. float LonNhat(float a[],int n)
```

```
{
   float lc = a[0];
   for (int i = 0; i < n; i++)
      if(a[i]>lc)
         lc = a[i];
   return lc;
}
```

Chương trình 007. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều các số thực.
b. Xuất mảng.
c. Tìm giá trị lớn nhất trong mảng một chiều các số thực.

– Chương trình

```
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
using namespace std;

void Nhap(float[],int &);
void Xuat(float[],int);
float LonNhat(float[],int);

int main()
{
   float b[100];
   int k;

   Nhap(b, k);
   cout << "\nMang ban dau:";
   Xuat(b, k);

   float kq = LonNhat(b, k);
   cout << "\nGia tri lon nhat la: "<<kq;
   return 0;
}

void Nhap(float a[],int &n)
{
   cout << "Nhap n: ";
   cin >> n;
   srand(time(NULL));
```

```
00893.    for (int i = 0; i < n; i++)
00894.       a[i] = -100.0 +
00895.          (rand()/(RAND_MAX/(100.0-(-100.0))));
00896. }
00897.
00898. void Xuat(float a[],int n)
00899. {
00900.    for (int i = 0; i < n; i++)
00901.       cout<<setw(8)<<a[i];
00902. }
00903.
00904. float LonNhat(float a[],int n)
00905. {
00906.    float lc = a[0];
00907.    for (int i = 0; i < n; i++)
00908.       if(a[i] > lc)
00909.          lc = a[i];
00910.    return lc;
00911. }
```

Chương trình 008. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ các file có tên floatdata01.inp; floatdata02.inp; floatdata03.inp; floatdata04.inp; floatdata05.inp; floatdata06.inp; floatdata07.inp; floatdata08.inp; floatdata09.inp; floatdata10.inp; floatdata11.inp; floatdata12.inp; floatdata13.inp;
b. Xuất mảng.
c. Tìm giá trị lớn nhất trong mảng một chiều các số thực.

– Chương trình.

```
00912. #include <iostream>
00913. #include <iomanip>
00914. #include <fstream>
00915. #include <string>
00916. using namespace std;
00917.
00918. string getInputFileName(int);
00919. string getOutputFileName(int);
00920. void Nhap(float[], int&, string);
00921. void Xuat(float[], int, string);
00922.
00923. void Xuat(float[], int);
00924. float LonNhat(float[], int);
00925.
```

```
00926.  int main()
00927.  {
00928.     float b[100000];
00929.     int n;
00930.     for (int i = 1; i <= 13; i++)
00931.     {
00932.        string FileName = getInputFileName(i);
00933.        Nhap(b, n, FileName);
00934.
00935.        cout << "\n" << FileName << endl;
00936.        Xuat(b, n);
00937.
00938.        FileName = getOutputFileName(i);
00939.        Xuat(b, n, FileName);
00940.
00941.        float kq;
00942.        kq = LonNhat(b, n);
00943.        cout << "\nGia tri lon nhat trong la: " <<
setprecision(5) << kq;
00944.     }
00945.     return 1;
00946.  }
00947.
00948.  string getInputFileName(int n)
00949.  {
00950.     string fileName = string("floatdata");
00951.     if (n < 10)
00952.        fileName += "0" + to_string(n) + ".inp";
00953.     else
00954.        fileName += to_string(n) + ".inp";
00955.     return fileName;
00956.  }
00957.
00958.  string getOutputFileName(int n)
00959.  {
00960.     string fileName = string("floatdata");
00961.     if (n < 10)
00962.        fileName += "0" + to_string(n) + ".out";
00963.     else
00964.        fileName += to_string(n) + ".out";
00965.     return fileName;
00966.  }
00967.
```

```
00968. void Nhap(float a[], int& n, string filename)
00969. {
00970.    ifstream fi(filename);
00971.    fi >> n;
00972.    for (int i = 0; i < n; i++)
00973.       fi >> a[i];
00974. }
00975.
00976. void Xuat(float a[], int n, string filename)
00977. {
00978.    ofstream fo(filename);
00979.    fo << n << endl;
00980.    for (int i = 0; i < n; i++)
00981.       fo<<setw(8)<<setprecision(5)<< a[i];
00982. }
00983.
00984. void Xuat(float a[], int n)
00985. {
00986.    for (int i = 0; i < n; i++)
00987.       cout<<setw(8)<<setprecision(5)<<a[i];
00988. }
00989.
00990. float LonNhat(float a[], int n)
00991. {
00992.    float lc = a[0];
00993.    for (int i = 0; i < n; i++)
00994.       if (a[i] > lc)
00995.          lc = a[i];
00996.    return lc;
00997. }
```

01.12 KỸ THUẬT ĐẶT CỜ HIỆU

Bài cơ sở 029. Định nghĩa hàm kiểm tra trong mảng các số nguyên có tồn tại giá trị chẵn nhỏ hơn 2004 hay không?

- Ví dụ 01:

19	29	67	761	2230	23	227	981	87	53

- Các giá trị chẵn có trong mảng

19	29	67	761	2230	23	227	981	87	53

- Kết quả: không tồn tại.

– Ví dụ 02:

19	29	68	761	2230	23	227	982	87	53

– Các giá trị chẵn có trong mảng

19	29	68	761	2230	23	227	982	87	53

– Kết quả: tồn tại.

– Khai báo hàm.

```
00998.  bool KiemTra(int [],int);
```

– Cách 01: định nghĩa hàm.

```
00999.  bool KiemTra(int a[],int n)
01000.  {
01001.     bool flag = false;
01002.     for (int i = 0; i < n; i++)
01003.        if(a[i]%2==0 && a[i]<2004)
01004.           flag = true;
01005.     return flag;
01006.  }
```

– Cách 02: định nghĩa hàm.

```
01007.  bool KiemTra(int a[],int n)
01008.  {
01009.     bool flag = false;
01010.     for(int i=0;i<n && flag==false;i++)
01011.        if(a[i]%2==0 && a[i]<2004)
01012.           flag = true;
01013.     return flag;
01014.  }
```

Chương trình 009. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều nguyên.
b. Xuất mảng.
c. Kiểm tra trong mảng có tồn tại giá trị chẵn bé hơn 2004.

– Chương trình

```
01015.  #include <iostream>
01016.  #include <iomanip>
01017.  #include <cstdlib>
01018.  #include <ctime>
01019.  #include <cmath>
01020.  using namespace std;
01021.
01022.  void Nhap(int[],int&);
01023.  void Xuat(int[],int);
01024.  bool KiemTra(int[],int);
```

```
01025.
01026.  int main()
01027.  {
01028.     int b[100];
01029.     int k;
01030.
01031.     Nhap(b, k);
01032.     cout << "\nMang ban dau:";
01033.     Xuat(b, k);
01034.
01035.     if(KiemTra(b, k))
01036.        cout << "\nMang co.";
01037.     else
01038.        cout << "\nMang khong co ";
01039.     return 0;
01040.  }
01041.
01042.  void Nhap(int a[],int &n)
01043.  {
01044.     cout << "Nhap n: ";
01045.     cin >> n;
01046.     srand(time(NULL));
01047.     for (int i = 0; i < n; i++)
01048.        a[i] = rand()%(200+1) - 100;
01049.  }
01050.
01051.  void Xuat(int a[],int n)
01052.  {
01053.     for (int i = 0; i < n; i++)
01054.        cout<<setw(8)<<a[i];
01055.  }
01056.
01057.  bool KiemTra(int a[],int n)
01058.  {
01059.     bool flag = false;
01060.     for (int i = 0; i < n; i++)
01061.        if(a[i]%2==0 && a[i]<2004)
01062.           flag = true;
01063.     return flag;
01064.  }
```

Chương trình 010. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ các file có tên intdata01.inp;

intdata02.inp; intdata03.inp; intdata04.inp; intdata05.inp;
intdata06.inp; intdata07.inp; intdata08.inp; intdata09.inp;
intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
b. Xuất mảng.
c. Kiểm tra trong mảng có tồn tại giá trị chẵn bé hơn 2004.

– Chương trình.

```
01065. #include <iostream>
01066. #include <iomanip>
01067. #include <fstream>
01068. #include <string>
01069. using namespace std;
01070.
01071. string getInputFileName(int);
01072. string getOutputFileName(int);
01073. void Nhap(int[], int&, string);
01074. void Xuat(int[], int, string);
01075.
01076. void Xuat(int[], int);
01077. bool KiemTra(int[], int);
01078.
01079. int main()
01080. {
01081.    int b[100000];
01082.    int n;
01083.    for (int i = 1; i <= 13; i++)
01084.    {
01085.       string FileName = getInputFileName(i);
01086.       Nhap(b, n, FileName);
01087.
01088.       cout << "\n" << FileName << endl;
01089.       Xuat(b, n);
01090.
01091.       FileName = getOutputFileName(i);
01092.       Xuat(b, n, FileName);
01093.
01094.       if (KiemTra(b, n))
01095.          cout << "\nMang co.";
01096.       else
01097.          cout << "\nMang khong co ";
01098.    }
01099.    return 1;
01100. }
```

```
01101.
01102.  string getInputFileName(int n)
01103.  {
01104.     string fileName = string("intdata");
01105.     if (n < 10)
01106.        fileName += "0" + to_string(n) + ".inp";
01107.     else
01108.        fileName += to_string(n) + ".inp";
01109.     return fileName;
01110.  }
01111.
01112.  string getOutputFileName(int n)
01113.  {
01114.     string fileName = string("intdata");
01115.     if (n < 10)
01116.        fileName += "0" + to_string(n) + ".out";
01117.     else
01118.        fileName += to_string(n) + ".out";
01119.     return fileName;
01120.  }
01121.
01122.  void Nhap(int a[], int& n, string filename)
01123.  {
01124.     ifstream fi(filename);
01125.     fi >> n;
01126.     for (int i = 0; i < n; i++)
01127.        fi >> a[i];
01128.  }
01129.
01130.  void Xuat(int a[], int n, string filename)
01131.  {
01132.     ofstream fo(filename);
01133.     fo << n << endl;
01134.     for (int i = 0; i < n; i++)
01135.        fo << setw(8) << a[i];
01136.  }
01137.
01138.  void Xuat(int a[], int n)
01139.  {
01140.     for (int i = 0; i < n; i++)
01141.        cout << setw(8) << a[i];
01142.  }
01143.
```

```
01144.  bool KiemTra(int a[], int n)
01145.  {
01146.     bool flag = false;
01147.     for (int i = 0; i < n; i++)
01148.        if (a[i] % 2 == 0 && a[i] < 2004)
01149.           flag = true;
01150.     return flag;
01151.  }
```

01.13 KỸ THUẬT XÂY DỰNG MẢNG

Bài cơ sở 030. Định nghĩa hàm xây dựng mảng *b* từ mảng *a* các số nguyên sao cho mảng *b* chỉ chứa các giá trị đối xứng trong mảng.

- Ví dụ:

19	29	7	761	2230	232	227	9889	87	53

- Các giá trị đối xứng trong mảng:

19	29	7	761	2230	232	227	9889	87	53

- Kết quả:

7	232	9889

- Khai báo hàm.

```
01152.  void XayDung(int [],int,int [],int&);
```

- Cách 01: định nghĩa hàm.

```
01153.  void XayDung(int a[],int n, int b[],int &k)
01154.  {
01155.     k = 0;
01156.     for (int i = 0; i < n; i++)
01157.        if(ktDoiXung(a[i])==true)
01158.        {
01159.           b[k] = a[i];
01160.          k++;
01161.        }
01162.  }
```

- Cách 02: định nghĩa hàm.

```
01163.  void XayDung(int a[],int n, int b[],int &k)
01164.  {
01165.     k = 0;
01166.     for (int i = 0; i < n; i++)
01167.        if(ktDoiXung(a[i]))
01168.           b[k++] = a[i];
01169.  }
```

Chương trình 011. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều.
b. Xuất mảng.
c. Xây dựng mảng b từ mảng a sao cho mảng b chỉ chứa các giá trị đối xứng trong mảng.

– Chương trình.

```
01170. #include <iostream>
01171. #include <iomanip>
01172. #include <cstdlib>
01173. #include <ctime>
01174. #include <cmath>
01175. using namespace std;
01176.
01177. void Nhap(int[],int&);
01178. void Xuat(int[],int);
01179. bool ktDoiXung(int);
01180. void XayDung(int[],int,int[],int&);
01181.
01182. int main()
01183. {
01184.    int b[100];
01185.    int k;
01186.    Nhap(b, k);
01187.    cout << "\nMang ban dau:";
01188.    Xuat(b, k);
01189.
01190.    int c[100];
01191.    int l;
01192.    XayDung(b, k, c, l);
01193.    cout << "\nMang ket qua:";
01194.    Xuat(c, l);
01195.    return 0;
01196. }
01197.
01198. void Nhap(int a[],int &n)
01199. {
01200.    cout << "Nhap n: ";
01201.    cin >> n;
01202.    srand(time(NULL));
01203.    for (int i = 0; i < n; i++)
01204.       a[i] = rand()%(200+1) - 100;
01205. }
```

```
01206.
01207.  bool ktDoiXung(int k)
01208.  {
01209.     int dn = 0;
01210.     for(int t=k=abs(k);t!=0;t=t/10)
01211.        dn = dn*10 + t%10;
01212.     return (dn==k);
01213.  }
01214.
01215.  void Xuat(int a[],int n)
01216.  {
01217.     for (int i = 0; i < n; i++)
01218.        cout<<setw(8)<<a[i];
01219.  }
01220.
01221.  void XayDung(int a[],int n,int b[],int &k)
01222.  {
01223.     k = 0;
01224.     for (int i = 0; i < n; i++)
01225.        if(ktDoiXung(a[i]))
01226.           b[k++] = a[i];
01227.  }
```

Chương trình 012. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ các file có tên intdata01.inp; intdata02.inp; intdata03.inp; intdata04.inp; intdata05.inp; intdata06.inp; intdata07.inp; intdata08.inp; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
b. Xuất mảng.
c. Xây dựng mảng b từ mảng a sao cho mảng b chỉ chứa các giá trị đối xứng trong mảng.

– Chương trình.

```
01228.  #include <iostream>
01229.  #include <iomanip>
01230.  #include <fstream>
01231.  #include <string>
01232.  using namespace std;
01233.
01234.  string getInputFileName(int);
01235.  string getOutputFileName(int);
01236.  void Nhap(int[], int&, string);
01237.  void Xuat(int[], int, string);
```

```
01238.
01239.  void Xuat(int[], int);
01240.  bool ktDoiXung(int);
01241.  void XayDung(int[], int, int[], int&);
01242.
01243.  int main()
01244.  {
01245.     int b[100000];
01246.     int n;
01247.     for (int i = 1; i <= 13; i++)
01248.     {
01249.        string FileName = getInputFileName(i);
01250.        Nhap(b, n, FileName);
01251.
01252.        cout << "\n" << FileName << endl;
01253.        Xuat(b, n);
01254.
01255.        int c[100000];
01256.        int l;
01257.        XayDung(b, n, c, l);
01258.        cout << "\nMang ket qua:";
01259.        Xuat(c, l);
01260.     }
01261.     return 1;
01262.  }
01263.
01264.  string getInputFileName(int n)
01265.  {
01266.     string fileName = string("intdata");
01267.     if (n < 10)
01268.        fileName += "0" + to_string(n) + ".inp";
01269.     else
01270.        fileName += to_string(n) + ".inp";
01271.     return fileName;
01272.  }
01273.
01274.  string getOutputFileName(int n)
01275.  {
01276.     string fileName = string("intdata");
01277.     if (n < 10)
01278.        fileName += "0" + to_string(n) + ".out";
01279.     else
01280.        fileName += to_string(n) + ".out";
```

```
01281.    return fileName;
01282. }
01283.
01284. void Nhap(int a[], int& n, string filename)
01285. {
01286.    ifstream fi(filename);
01287.    fi >> n;
01288.    for (int i = 0; i < n; i++)
01289.       fi >> a[i];
01290. }
01291.
01292. void Xuat(int a[], int n, string filename)
01293. {
01294.    ofstream fo(filename);
01295.    fo << n << endl;
01296.    for (int i = 0; i < n; i++)
01297.       fo << setw(8) << a[i];
01298. }
01299.
01300. void Xuat(int a[], int n)
01301. {
01302.    for (int i = 0; i < n; i++)
01303.       cout << setw(8) << a[i];
01304. }
01305.
01306. bool ktDoiXung(int k)
01307. {
01308.    int dn = 0;
01309.    for (int t = k = abs(k); t != 0; t = t / 10)
01310.       dn = dn * 10 + t % 10;
01311.    return (dn == k);
01312.
01313. }
01314.
01315. void XayDung(int a[], int n, int b[], int& k)
01316. {
01317.    k = 0;
01318.    for (int i = 0; i < n; i++)
01319.       if (ktDoiXung(a[i]))
01320.          b[k++] = a[i];
01321. }
```

01.14 KỸ THUẬT SẮP XẾP

Bài cơ sở 031. Định nghĩa hàm sắp xếp mảng một chiều các số thực tăng dần?

– Ví dụ:

19	29	7	761	2230	232	227	9889	87	53

– Mảng sau khi sắp tăng:

7	19	29	53	87	227	232	761	2230	9889

– Khai báo hàm.

```
01322. void SapTang(float [],int);
```

– Định nghĩa hàm.

```
01323. void SapTang(float a[],int n)
01324. {
01325.    for(int i=0;i<=n-2;i++)
01326.       for(int j=i+1;j<=n-1;j++)
01327.          if(a[i]>a[j])
01328.             HoanVi(a[i], a[j]);
01329. }
```

Chương trình 013. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều thực.
b. Xuất mảng.
c. Sắp mảng tăng dần.

– Chương trình

```
01330. #include <iostream>
01331. #include <iomanip>
01332. #include <cstdlib>
01333. #include <ctime>
01334. using namespace std;
01335.
01336. void Nhap(float[],int &);
01337. void Xuat(float[],int);
01338. void HoanVi(float &,float &);
01339. void SapTang(float[],int);
01340.
01341. int main()
01342. {
01343.    float b[100];
01344.    int k;
01345.
01346.    Nhap(b, k);
```

```
01347.     cout << “\nMang ban dau:";
01348.     Xuat(b, k);
01349.
01350.     SapTang(b, k);
01351.     cout << “\nMang sau khi sap tang:";
01352.     Xuat(b, k);
01353.     return 0;
01354.  }
01355.
01356.  void Nhap(float a[],int &n)
01357.  {
01358.     cout << “Nhap n: ";
01359.     cin >> n;
01360.     srand(time(NULL));
01361.     for (int i = 0; i < n; i++)
01362.        a[i] = -100.0 +
01363.           (rand()/(RAND_MAX/(100.0-(-100.0))));
01364.  }
01365.
01366.  void Xuat(float a[],int n)
01367.  {
01368.     for (int i = 0; i < n; i++)
01369.        cout<<setw(8)<<a[i];
01370.  }
01371.
01372.  void HoanVi(float &a,float &b)
01373.  {
01374.     float temp = a;
01375.     a = b;
01376.     b = temp;
01377.  }
01378.
01379.  void SapTang(float a[],int n)
01380.  {
01381.     for(int i=0;i<=n-2;i++)
01382.        for(int j=i+1;j<=n-1;j++)
01383.           if(a[i] > a[j])
01384.              HoanVi(a[i], a[j]);
01385.  }
```

Chương trình 014. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ các file có tên floatdata01.inp; floatdata02.inp; floatdata03.inp; floatdata04.inp;

floatdata05.inp; floatdata06.inp; floatdata07.inp;
floatdata08.inp; floatdata09.inp; floatdata10.inp;
floatdata11.inp; floatdata12.inp; floatdata13.inp;
b. Xuất mảng.
c. Sắp mảng tăng dần.
d. Xuất mảng sau khi sắp xếp ra file floatdata01.out;
floatdata02.out; floatdata03.out; floatdata04.out;
floatdata05.out; floatdata06.out; floatdata07.out;
floatdata08.out; floatdata09.out; floatdata10.out;
floatdata11.out; floatdata12.out; floatdata13.out;

– Chương trình.

```
01386. #include <iostream>
01387. #include <iomanip>
01388. #include <fstream>
01389. #include <string>
01390. using namespace std;
01391.
01392. string getInputFileName(int);
01393. string getOutputFileName(int);
01394. void Nhap(float[], int&, string);
01395. void Xuat(float[], int, string);
01396.
01397. void Xuat(float[], int);
01398. void HoanVi(float&, float&);
01399. void SapTang(float[], int);
01400.
01401. int main()
01402. {
01403.    float b[100000];
01404.    int n;
01405.    for (int i = 1; i <= 13; i++)
01406.    {
01407.       string FileName = getInputFileName(i);
01408.       Nhap(b, n, FileName);
01409.
01410.       cout << "\n" << FileName << endl;
01411.       Xuat(b, n);
01412.
01413.       SapTang(b, n);
01414.       cout << "\nMang sau khi sap tang:";
01415.       Xuat(b, n);
01416.
```

```
01417.        FileName = getOutputFileName(i);
01418.        Xuat(b, n, FileName);
01419.     }
01420.     return 1;
01421.  }
01422.
01423.  string getInputFileName(int n)
01424.  {
01425.     string fileName = string("floatdata");
01426.     if (n < 10)
01427.        fileName += "0" + to_string(n) + ".inp";
01428.     else
01429.        fileName += to_string(n) + ".inp";
01430.     return fileName;
01431.  }
01432.
01433.  string getOutputFileName(int n)
01434.  {
01435.     string fileName = string("floatdata");
01436.     if (n < 10)
01437.        fileName += "0" + to_string(n) + ".out";
01438.     else
01439.        fileName += to_string(n) + ".out";
01440.     return fileName;
01441.  }
01442.
01443.  void Nhap(float a[], int& n, string filename)
01444.  {
01445.     ifstream fi(filename);
01446.     fi >> n;
01447.     for (int i = 0; i < n; i++)
01448.        fi >> a[i];
01449.  }
01450.
01451.  void Xuat(float a[], int n, string filename)
01452.  {
01453.     ofstream fo(filename);
01454.     fo << n << endl;
01455.     for (int i = 0; i < n; i++)
01456.        fo<<setw(8)<<setprecision(5)<<a[i];
01457.  }
01458.
01459.  void Xuat(float a[], int n)
```

```
01460. {
01461.    for (int i = 0; i < n; i++)
01462.       cout<<setw(8)<<setprecision(5)<<a[i];
01463. }
01464.
01465. void HoanVi(float& a, float& b)
01466. {
01467.    float temp = a;
01468.    a = b;
01469.    b = temp;
01470. }
01471.
01472. void SapTang(float a[], int n)
01473. {
01474.    for (int i = 0; i <= n - 2; i++)
01475.       for (int j = i + 1; j <= n - 1; j++)
01476.       if (a[i] > a[j])
01477.          HoanVi(a[i], a[j]);
01478. }
```

01.15 KỸ THUẬT XÓA

Bài cơ sở 032. Định nghĩa hàm xóa phần tử tại vị trí vt trong mảng một chiều các số nguyên.

- Ví dụ: hãy xóa phần tử tại vị trí vt = 6.

19	29	7	761	2230	232	227	9889	87	53

- Mảng sau khi xóa:

19	29	7	761	2230	232	9889	87	53

- Khai báo hàm.

```
01479. void XoaViTri(int [],int &,int);
```

- Định nghĩa hàm.

```
01480. void XoaViTri(int a[],int &n,int vt)
01481. {
01482.     for(int i=vt;i<=n-2;i++)
01483.         a[i] = a[i+1];
01484.     n--;
01485. }
```

Bài cơ sở 033. Định nghĩa hàm xóa các số đối xứng trong mảng một chiều các số nguyên.

- Ví dụ:

19	29	7	761	2230	232	227	9889	87	53

– Các giá trị đối xứng:

19	29	7	761	2230	232	227	9889	87	53

– Mảng sau khi xóa các giá trị đối xứng:

19	29	761	2230	227	87	53

– Khai báo hàm.

```
01486.  void XoaDoiXung(int [],int &);
```

– Định nghĩa hàm.

```
01487.  void XoaDoiXung(int a[],int &n)
01488.  {
01489.      for(int i=n-1;i>=0;i--)
01490.          if(ktDoiXung(a[i]))
01491.              XoaViTri(a,n,i);
01492.  }
```

Chương trình 015. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều nguyên.
b. Xuất mảng.
c. Xóa các giá trị đối xứng có trong mảng.

– Chương trình

```
01493.  #include <iostream>
01494.  #include <iomanip>
01495.  #include <cstdlib>
01496.  #include <ctime>
01497.  using namespace std;
01498.
01499.  bool ktDoiXung(int);
01500.  void XoaViTri(int[],int&,int);
01501.  void XoaDoiXung(int[],int&);
01502.  void Nhap(int[],int&);
01503.  void Xuat(int[],int);
01504.
01505.  int main()
01506.  {
01507.     int b[100];
01508.     int k;
01509.
01510.     Nhap(b, k);
01511.     cout << "\nMang ban dau:";
01512.     Xuat(b, k);
01513.
```

```
01514.     XoaDoiXung(b, k);
01515.     cout << “\nMang sau khi xoa so doi xung:";
01516.     Xuat(b, k);
01517.     return 0;
01518. }
01519.
01520. void Nhap(int a[], int &n)
01521. {
01522.     cout << “Nhap n: ";
01523.     cin >> n;
01524.     srand(time(NULL));
01525.     for(int i = 0;i < n; i++)
01526.        a[i] = rand()%(200+1) - 100;
01527. }
01528.
01529. void Xuat(int a[], int n)
01530. {
01531.     for (int i = 0; i < n; i++)
01532.        cout<<setw(8)<<a[i];
01533. }
01534.
01535. bool ktDoiXung(int k)
01536. {
01537.     int dn = 0;
01538.     for(int t=k=abs(k);t!=0;t=t/10)
01539.        dn = dn*10 + t%10;
01540.     return (dn==k);
01541. }
01542.
01543. void XoaViTri(int a[], int &n, int vt)
01544. {
01545.     for(int i=vt;i<=n-2;i++)
01546.        a[i] = a[i+1];
01547.     n--;
01548. }
01549.
01550. void XoaDoiXung(int a[], int &n)
01551. {
01552.     for(int i=n-1;i>=0;i--)
01553.        if(ktDoiXung(a[i]))
01554.           XoaViTri(a,n,i);
01555. }
```

Chương trình 016. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ các file có tên intdata01.inp; intdata02.inp; intdata03.inp; intdata04.inp; intdata05.inp; intdata06.inp; intdata07.inp; intdata08.inp; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
b. Xuất mảng.
c. Xóa các giá trị đối xứng có trong mảng.
d. Xuất mảng sau khi xóa ra file intdata01.out; intdata02.out; intdata03.out; intdata04.out; intdata05.out; intdata06.out; intdata07.out; intdata08.out; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

– Chương trình.

```
01556. #include <iostream>
01557. #include <iomanip>
01558. #include <fstream>
01559. #include <string>
01560. using namespace std;
01561.
01562. string getInputFileName(int);
01563. string getOutputFileName(int);
01564. void Nhap(int[], int&, string);
01565. void Xuat(int[], int, string);
01566.
01567. void Xuat(int[], int);
01568. bool ktDoiXung(int);
01569. void XoaViTri(int[], int&, int);
01570. void XoaDoiXung(int[], int&);
01571.
01572. int main()
01573. {
01574.    int b[100000];
01575.    int n;
01576.    for (int i = 1; i <= 13; i++)
01577.    {
01578.       string FileName = getInputFileName(i);
01579.       Nhap(b, n, FileName);
01580.
01581.       cout << "\n" << FileName << endl;
01582.       Xuat(b, n);
01583.
01584.       XoaDoiXung(b, n);
01585.       cout << "\nMang sau khi xoa so doi xung:";
```

```
01586.       Xuat(b, n);
01587.
01588.       FileName = getOutputFileName(i);
01589.       Xuat(b, n, FileName);
01590.    }
01591.    return 1;
01592. }
01593.
01594. string getInputFileName(int n)
01595. {
01596.    string fileName = string("intdata");
01597.    if (n < 10)
01598.       fileName += "0" + to_string(n) + ".inp";
01599.    else
01600.       fileName += to_string(n) + ".inp";
01601.    return fileName;
01602. }
01603.
01604. string getOutputFileName(int n)
01605. {
01606.    string fileName = string("intdata");
01607.    if (n < 10)
01608.       fileName += "0" + to_string(n) + ".out";
01609.    else
01610.       fileName += to_string(n) + ".out";
01611.    return fileName;
01612. }
01613.
01614. void Nhap(int a[], int& n, string filename)
01615. {
01616.    ifstream fi(filename);
01617.    fi >> n;
01618.    for (int i = 0; i < n; i++)
01619.       fi >> a[i];
01620. }
01621.
01622. void Xuat(int a[], int n, string filename)
01623. {
01624.    ofstream fo(filename);
01625.    fo << n << endl;
01626.    for (int i = 0; i < n; i++)
01627.       fo << setw(8) << a[i];
01628. }
```

```
01629.
01630.  void Xuat(int a[], int n)
01631.  {
01632.     for (int i = 0; i < n; i++)
01633.        cout << setw(8) << a[i];
01634.  }
01635.
01636.  bool ktDoiXung(int k)
01637.  {
01638.     int dn = 0;
01639.     for (int t = k = abs(k); t != 0; t = t / 10)
01640.        dn = dn * 10 + t % 10;
01641.     return (dn == k);
01642.  }
01643.
01644.  void XoaViTri(int a[], int& n, int vt)
01645.  {
01646.     for (int i = vt; i <= n - 2; i++)
01647.        a[i] = a[i + 1];
01648.     n--;
01649.  }
01650.
01651.  void XoaDoiXung(int a[], int& n)
01652.  {
01653.     for (int i = n - 1; i >= 0; i--)
01654.        if (ktDoiXung(a[i]))
01655.           XoaViTri(a, n, i);
01656.  }
```

01.16 KỸ THUẬT THÊM

Bài cơ sở 034. Định nghĩa hàm thêm giá trị x tại vị trí vt trong mảng một chiều các số nguyên.

– Ví dụ: hãy thêm giá trị $x = 45$ tại vị trí $vt = 6$.

0	1	2	3	4	5	6	7	8
19	29	7	761	2230	232	227	87	53

– Mảng sau khi thêm giá trị $x = 45$ tại vị trí $vt = 6$.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	2230	232	45	227	87	53

– Khai báo hàm.

```
01657.  void ThemViTri(int [],int &,int,int);
```

– Định nghĩa hàm.

```
01658.  void ThemViTri(int a[],int &n,int x,int vt)
01659.  {
01660.     for(int i=n-1;i>=vt;i--)
01661.        a[i+1] = a[i];
01662.     a[vt] = x;
01663.     n++;
01664.  }
```

Bài cơ sở 035. Viết hàm thêm các giá trị 0 vào sau các giá trị lẻ trong mảng.

- Ví dụ:

19	20	72	761	87	53

- Các giá trị lẻ trong mảng

19	20	72	761	87	53

- Mảng sau khi thêm các giá trị 0.

19	0	20	72	761	0	87	0	53	0

- Khai báo hàm.

```
01665.  void ThemKhong(int [],int &);
```

- Định nghĩa hàm.

```
01666.  void ThemKhong(int a[],int &n)
01667.  {
01668.     for(int i=0;i<=n-1;i++)
01669.        if(ktLe(a[i]))
01670.           ThemViTri(a,n,i+1,0);
01671.  }
```

Chương trình 017. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều nguyên.
b. Xuất mảng.
c. Thêm các giá trị 0 vào sau các giá trị lẻ có trong mảng.

- Chương trình

```
01672.  #include <iostream>
01673.  #include <iomanip>
01674.  #include <cstdlib>
01675.  #include <ctime>
01676.  using namespace std;
01677.
01678.  void Nhap(int[], int&);
01679.  void Xuat(int[], int);
01680.  void ThemViTri(int[], int&, int, int);
```

```
01681. void ThemKhong(int[], int&);
01682.
01683. int main()
01684. {
01685.    int b[100];
01686.    int k;
01687.
01688.    Nhap(b, k);
01689.    cout << "\nMang ban dau:";
01690.    Xuat(b, k);
01691.
01692.    ThemKhong(b, k);
01693.    cout << "\nMang sau khi bien doi:";
01694.    Xuat(b, k);
01695.    return 0;
01696. }
01697.
01698. void Nhap(int a[], int &n)
01699. {
01700.    cout << "Nhap n: ";
01701.    cin >> n;
01702.    srand(time(NULL));
01703.    for (int i = 0; i < n; i++)
01704.       a[i] = rand()%(200+1) - 100;
01705. }
01706.
01707. void Xuat(int a[], int n)
01708. {
01709.    for (int i = 0; i < n; i++)
01710.       cout<<setw(8)<<a[i];
01711. }
01712.
01713. void ThemViTri(int a[],int &n,int vt,int x)
01714. {
01715.    for(int i=n-1;i>=vt;i--)
01716.       a[i+1] = a[i];
01717.    a[vt] = x;
01718.    n++;
01719. }
01720.
01721. void ThemKhong(int a[], int&n)
01722. {
01723.    for(int i=0;i<=n-1;i++)
```

```
01724.       if(a[i]%2!=0)
01725.          ThemViTri(a,n,i+1,0);
01726. }
```

Chương trình 018. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ các file có tên intdata01.inp; intdata02.inp; intdata03.inp; intdata04.inp; intdata05.inp; intdata06.inp; intdata07.inp; intdata08.inp; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
b. Xuất mảng.
c. Thêm các giá trị 0 vào sau các giá trị lẻ có trong mảng.
d. Xuất mảng sau khi xóa ra file intdata01.out; intdata02.out; intdata03.out; intdata04.out; intdata05.out; intdata06.out; intdata07.out; intdata08.out; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

– Chương trình.

```
01727. #include <iostream>
01728. #include <iomanip>
01729. #include <fstream>
01730. #include <string>
01731. using namespace std;
01732.
01733. string getInputFileName(int);
01734. string getOutputFileName(int);
01735. void Nhap(int[], int&, string);
01736. void Xuat(int[], int, string);
01737.
01738. void Xuat(int[], int);
01739. void ThemViTri(int[], int&, int, int);
01740. void ThemKhong(int[], int&);
01741.
01742. int main()
01743. {
01744.    int b[200000];
01745.    int n;
01746.    for (int i = 1; i <= 13; i++)
01747.    {
01748.       string FileName = getInputFileName(i);
01749.       Nhap(b, n, FileName);
01750.
01751.       cout << "\n" << FileName << endl;
01752.       Xuat(b, n);
```

```
01753.
01754.        ThemKhong(b, n);
01755.
01756.        cout << "\nMang sau khi bien doi:";
01757.        Xuat(b, n);
01758.
01759.        FileName = getOutputFileName(i);
01760.        Xuat(b, n, FileName);
01761.     }
01762.     return 1;
01763.  }
01764.
01765.  string getInputFileName(int n)
01766.  {
01767.     string fileName = string("intdata");
01768.     if (n < 10)
01769.        fileName += "0" + to_string(n) + ".inp";
01770.     else
01771.        fileName += to_string(n) + ".inp";
01772.     return fileName;
01773.  }
01774.
01775.  string getOutputFileName(int n)
01776.  {
01777.     string fileName = string("intdata");
01778.     if (n < 10)
01779.        fileName += "0" + to_string(n) + ".out";
01780.     else
01781.        fileName += to_string(n) + ".out";
01782.     return fileName;
01783.  }
01784.
01785.  void Nhap(int a[], int& n, string filename)
01786.  {
01787.     ifstream fi(filename);
01788.     fi >> n;
01789.     for (int i = 0; i < n; i++)
01790.        fi >> a[i];
01791.  }
01792.
01793.  void Xuat(int a[], int n, string filename)
01794.  {
01795.     ofstream fo(filename);
```

```
01796.    fo << n << endl;
01797.    for (int i = 0; i < n; i++)
01798.       fo << setw(8) << a[i];
01799. }
01800.
01801. void Xuat(int a[], int n)
01802. {
01803.    for (int i = 0; i < n; i++)
01804.       cout << setw(8) << a[i];
01805. }
01806.
01807. bool ktDoiXung(int k)
01808. {
01809.    int dn = 0;
01810.    for (int t = k = abs(k); t != 0; t = t / 10)
01811.       dn = dn * 10 + t % 10;
01812.    return (dn == k);
01813. }
01814.
01815. void ThemViTri(int a[], int& n, int vt, int x)
01816. {
01817.    for (int i = n - 1; i >= vt; i--)
01818.       a[i + 1] = a[i];
01819.    a[vt] = x;
01820.    n++;
01821. }
01822.
01823. void ThemKhong(int a[], int& n)
01824. {
01825.    for (int i = 0; i <= n - 1; i++)
01826.       if (a[i] % 2 != 0)
01827.          ThemViTri(a, n, i + 1, 0);
01828. }
```

01.17 KỸ THUẬT XỬ LÝ MẢNG CON

Bài cơ sở 036. Định nghĩa hàm xuất mảng con có độ dài l bắt đầu tại vị trí vt trong mảng một chiều các số nguyên.

– Ví dụ: cho mảng ban đầu có 10 phần tử, xuất mảng con có độ dài là 4 bắt đầu tại vị trí vt = 3.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	2230	232	227	9889	87	53

– Mảng con có độ dài là 4 bắt đầu tại vị trí vt = 3.

0	1	2	3	4	5	6	7	8	9
19	29	7	**761**	**2230**	**232**	**227**	9889	87	53

– Kết quả: 761, 2230, 232, 227.

– Khai báo hàm.

```
void XuatCon(int [],int,int,int);
```

– Định nghĩa hàm.

```
void XuatCon(int a[],int n,int vt,int l)
{
    for(int i=0;i<=l-1;i++)
        cout << setw(8) << a[vt+i];
}
```

Bài cơ sở 037. Định nghĩa hàm xuất tất cả các mảng con có độ dài *l* trong mảng một chiều các số nguyên.

– Ví dụ: cho mảng ban đầu có 10 phần tử, xuất tất cả mảng con có độ dài là 4.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	2230	232	227	9889	87	53

– Các mảng con có độ dài là 4.

0	1	2	3	4	5	6	7	8	9
19	**29**	**7**	**761**	2230	232	227	9889	87	53

19	**29**	**7**	**761**	**2230**	232	227	9889	87	53

19	29	**7**	**761**	**2230**	**232**	227	9889	87	53

19	29	7	**761**	**2230**	**232**	**227**	9889	87	53

19	29	7	761	**2230**	**232**	**227**	**9889**	87	53

19	29	7	761	2230	**232**	**227**	**9889**	**87**	53

19	29	7	761	2230	232	**227**	**9889**	**87**	**53**

– Kết quả:
  + 19 29 7 761,
  + 29 7 761 2230,
  + 7 761 2230 232,
  + 761 2230 232 227,
  + 2230 232 227 9889,
  + 232 227 9889 87,

+ 227 9889 87 53.

– Khai báo hàm.

```
01835.  void XuatCon(int [],int,int);
```

– Định nghĩa hàm.

```
01836.  void XuatCon(int a[],int n,int l)
01837.  {
01838.     for(int vt=0;vt<=n-l;vt++)
01839.     {
01840.        cout<<endl;
01841.        XuatCon(a,n,vt,l);
01842.     }
01843.  }
```

Bài cơ sở 038. Định nghĩa hàm xuất tất cả các mảng con trong mảng một chiều các số nguyên.

– Ví dụ: Cho mảng ban đầu có 5 phần tử.

0	1	2	3	4
89	29	07	67	38

– Các mảng con có độ dài là 01.

0	1	2	3	4
89	29	07	67	38
89	29	07	67	38
89	29	07	67	38
89	29	07	67	38
89	29	07	67	38

– Các mảng con có độ dài là 02.

0	1	2	3	4
89	29	07	67	38
89	29	07	67	38
89	29	07	67	38
89	29	07	67	38

– Các mảng con có độ dài là 03.

0	1	2	3	4
89	29	07	67	38
89	29	07	67	38
89	29	07	67	38

– Các mảng con có độ dài là 04.

0	1	2	3	4
89	29	07	67	38
89	29	07	67	38

– Các mảng con có độ dài là 05.

0	1	2	3	4
89	29	07	67	38

– Kết quả: các mảng con trong mảng ban đầu.

+ Các mảng con độ dài là 1: 89, 29, 07, 67, 38.
+ Các mảng con độ dài là 2: 89 29, 29 07, 07 67, 67 38.
+ Các mảng con độ dài là 3: 89 29 07, 29 07 67, 07 67 38.
+ Các mảng con độ dài là 4: 89 29 07 67, 29 07 67 38.
+ Các mảng con độ dài là 5: 89 29 07 67 38.

– Khai báo hàm.

```
01844. void XuatCon(int [],int);
```

– Định nghĩa hàm.

```
01845. void XuatCon(int a[],int n)
01846. {
01847.    for(int l=1;l<=n;l++)
01848.       XuatCon(a,n,l);
01849. }
```

Chương trình 019. Viết chương trình thực hiện các yêu cầu sau:
a. Tạo ngẫu nhiên mảng một chiều nguyên.
b. Xuất mảng.
c. Xuất tất cả các mảng con có trong mảng.

– Chương trình

```
01850. #include <iostream>
01851. #include <iomanip>
01852. #include <cstdlib>
01853. #include <ctime>
01854. using namespace std;
01855.
01856. void Nhap(int[],int&);
01857. void Xuat(int[],int);
01858.
01859. void XuatCon(int[],int,int,int);
01860. void XuatCon(int[],int,int);
01861. void XuatCon(int[],int);
01862.
01863. int main()
01864. {
01865.    int b[100];
01866.    int k;
01867.
01868.    Nhap(b, k);
01869.    cout << "\nMang ban dau: \n";
01870.    Xuat(b, k);
01871.
```

```
01872.     cout << “\nTat ca mang Con: \n";
01873.     XuatCon(b,k);
01874.     cout<<endl<<"Ket Thuc.\n";
01875.     return 0;
01876.  }
01877.
01878.  void Nhap(int a[],int &n)
01879.  {
01880.     cout << “Nhap n: ";
01881.     cin >> n;
01882.     srand(time(NULL));
01883.     for (int i = 0; i < n; i++)
01884.        a[i] = rand()%(200+1) - 100;
01885.  }
01886.
01887.  void Xuat(int a[],int n)
01888.  {
01889.     for (int i = 0; i < n; i++)
01890.        cout<<setw(8)<<a[i];
01891.  }
01892.
01893.  void XuatCon(int a[],int n,int vt,int l)
01894.  {
01895.     for(int i=0;i<=l-1;i++)
01896.        cout<<setw(8)<<a[vt+i];
01897.  }
01898.
01899.  void XuatCon(int a[],int n,int l)
01900.  {
01901.     for(int vt=0;vt<=n-l;vt++)
01902.     {
01903.        cout<<endl;
01904.        XuatCon(a,n,vt,l);
01905.     }
01906.  }
01907.
01908.  void XuatCon(int a[],int n)
01909.  {
01910.     for(int l=1;l<=n;l++)
01911.        XuatCon(a,n,l);
01912.  }
```

Chương trình 020. Viết chương trình thực hiện các yêu cầu sau:
a. Nhập mảng một chiều từ file có tên intdata01.inp;
b. Xuất mảng ra màn hình.
c. Xuất tất cả các mảng con có trong mảng.

– Chương trình.

```
01913. #include <iostream>
01914. #include <iomanip>
01915. #include <fstream>
01916. #include <string>
01917. using namespace std;
01918.
01919. string getInputFileName(int);
01920. void Nhap(int[], int&, string);
01921.
01922. void Xuat(int[], int);
01923. void XuatCon(int[], int, int, int);
01924. void XuatCon(int[], int, int);
01925. void XuatCon(int[], int);
01926.
01927. int main()
01928. {
01929.    int b[100000];
01930.    int n;
01931.
01932.    string FileName = getInputFileName(1);
01933.    Nhap(b, n, FileName);
01934.
01935.    cout << "\n" << FileName << endl;
01936.    Xuat(b, n);
01937.
01938.    cout << "\nTat ca mang Con: \n";
01939.    XuatCon(b, n);
01940.    cout << "Ket Thuc.\n";
01941.
01942.    return 1;
01943. }
01944.
01945. string getInputFileName(int n)
01946. {
01947.    string fileName = string("intdata");
01948.    if (n < 10)
01949.       fileName += "0" + to_string(n) + ".inp";
```

```
01950.    else
01951.       fileName += to_string(n) + ".inp";
01952.    return fileName;
01953. }
01954.
01955. void Nhap(int a[], int& n, string filename)
01956. {
01957.    ifstream fi(filename);
01958.    fi >> n;
01959.    for (int i = 0; i < n; i++)
01960.       fi >> a[i];
01961. }
01962.
01963. void Xuat(int a[], int n)
01964. {
01965.    for (int i = 0; i < n; i++)
01966.       cout << setw(8) << a[i];
01967. }
01968.
01969. void XuatCon(int a[], int n, int vt, int l)
01970. {
01971.    for (int i = 0; i <= l - 1; i++)
01972.       cout << setw(8) << a[vt + i];
01973. }
01974.
01975. void XuatCon(int a[], int n, int l)
01976. {
01977.    for (int vt = 0; vt <= n - l; vt++)
01978.    {
01979.       XuatCon(a, n, vt, l);
01980.       cout << endl;
01981.    }
01982. }
01983.
01984. void XuatCon(int a[], int n)
01985. {
01986.    for (int l = 1; l <= n; l++)
01987.       XuatCon(a, n, l);
01988. }
```

01.18 BÀI TẬP MẢNG MỘT CHIỀU

01.18.01 Kỹ thuật nhập xuất mảng

Bài 001.Viết chương trình thực hiện các yêu cầu sau:
+ Tạo ngẫu nhiên mảng một chiều các số nguyên với yêu cầu mỗi phần tử trong mảng là số nguyên nằm trong đoạn $[-100, 100]$.
+ Xuất mảng ra màn hình.

– Khai báo hàm.

```
01989.  void Nhap(int [],int &);
01990.  void Xuat(int [],int);
```

– Định nghĩa hàm tạo ngẫu nhiên mảng một chiều các số nguyên với yêu cầu mỗi phần tử trong mảng là số nguyên nằm trong đoạn $[-100, 100]$.

```
01991.  void Nhap(int a[],int &n)
01992.  {
01993.     cout << "Nhap n: ";
01994.     cin >> n;
01995.     srand(time(NULL));
01996.     for (int i=0; i<=n-1; i++)
01997.        a[i] = rand()%(200+1) - 100;
01998.  }
```

– Định nghĩa hàm xuất.

```
01999.  void Xuat(int a[], int n)
02000.  {
02001.     cout << n << endl;
02002.     for (int i=0; i<=n-1; i++)
02003.        cout << setw(10) << a[i];
02004.  }
```

Bài 002.Viết chương trình thực hiện các yêu cầu sau:
+ Tạo ngẫu nhiên mảng một chiều các số thực với yêu cầu mỗi phần tử trong mảng là số thực nằm trong đoạn $[-100, 100]$.
+ Xuất mảng ra màn hình.

– Khai báo hàm.

```
02005.  void Nhap(float [],int &);
02006.  void Xuat(float [],int);
```

– Định nghĩa hàm tạo ngẫu nhiên mảng một chiều các số thực với yêu cầu mỗi phần tử trong mảng là số thực nằm trong đoạn $[-100, 100]$.

```
02007.  void Nhap(float a[],int &n)
```

```
02008. {
02009.    cout << “Nhap n: ";
02010.    cin >> n;
02011.    srand(time(NULL));
02012.    for (int i=0; i<=n-1; i++)
02013.       a[i] = -100.0 +
02014.          (rand()/(RAND_MAX/(100.0-(-100.0))));
02015. }
```

– Định nghĩa hàm xuất.

```
02016. void Xuat(float a[], int n)
02017. {
02018.    cout << n << endl;
02019.    for (int i=0; i<=n-1; i++)
02020.       cout<<setw(10)<<setprecision(5)<<a[i];
02021. }
```

Bài 003.Viết chương trình thực hiện các yêu cầu sau:
+ Nhập mảng một chiều các số nguyên từ tập tin.
+ Xuất mảng ra màn hình.

– Khai báo hàm.

```
02022. void Nhap(string,int [],int &);
02023. void Xuat(int [],int);
```

– Định nghĩa hàm nhập.

```
02024. void Nhap(string filename, int a[], int &n)
02025. {
02026.    ifstream fi(filename);
02027.    fi >> n;
02028.    for (int i=0; i<=n-1; i++)
02029.       fi >> a[i];
02030. }
```

– Định nghĩa hàm xuất.

```
02031. void Xuat(int a[], int n)
02032. {
02033.    cout << n << endl;
02034.    for (int i=0; i<=n-1; i++)
02035.       cout << setw(10) << a[i];
02036. }
```

Bài 004.Viết chương trình thực hiện các yêu cầu sau:
+ Nhập mảng một chiều các số thực từ tập tin.
+ Xuất mảng ra màn hình.

– Khai báo hàm.

```
02037.  void Nhap(float [],int &);
02038.  void Xuat(float [],int);
```

– Định nghĩa hàm nhập.

```
02039.  void Nhap(string filename, float a[], int &n)
02040.  {
02041.     ifstream fi(filename);
02042.     fi >> n;
02043.     for (int i=0; i<=n-1; i++)
02044.        fi >> a[i];
02045.  }
```

– Định nghĩa hàm xuất.

```
02046.  void Xuat(float a[], int n)
02047.  {
02048.     cout << n << endl;
02049.     for (int i=0; i<=n-1; i++)
02050.        cout<<setw(10)<<setprecision(5)<<a[i];
02051.  }
```

01.18.02 Kỹ thuật Liệt Kê

Bài 005.(Hạt nhân) Viết hàm liệt kê các giá trị chẵn trong mảng một chiều các số nguyên.

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Các giá trị chẵn trong mảng.

19	29	62	76	99	23	12	98	88	11

– Kết quả: 62, 76, 12, 98, 88.

– Khai báo hàm.

```
02052.  void LietKe(int [],int);
```

– Định nghĩa hàm.

```
02053.  void LietKe(int a[], int n)
02054.  {
02055.     for (int i=0; i<=n-1; i++)
02056.        if (a[i] % 2 == 0)
02057.           cout << setw(10) << a[i];
02058.  }
```

Bài 006.Hãy liệt kê các số âm trong mảng một chiều các số thực.

– Ví dụ:

19	29	-62	76	-99	-23	12	-98	88	-11

– Các số âm trong mảng.

19	29	-62	76	-99	-23	12	-98	88	-11

– Kết quả: -62, -99, -23, -98, -11.

– Khai báo hàm.

```
02059.  void LietKe(float [],int);
```

– Định nghĩa hàm.

```
02060.  void LietKe(float a[], int n)
02061.  {
02062.     for (int i=0; i<=n-1; i++)
02063.        if (a[i] < 0)
02064.           cout<<setw(8)<<setprecision(5)<<a[i];
02065.  }
```

Bài 007.Hãy liệt kê các số dương trong mảng một chiều các số thực.

– Ví dụ:

19	29	-62	76	-99	-23	12	-98	88	-11

– Các số dương trong mảng.

19	29	-62	76	-99	-23	12	-98	88	-11

– Kết quả: 19, 29, 76, 12, 88.

– Khai báo hàm.

```
02066.  void LietKe(float [],int);
```

– Định nghĩa hàm.

```
02067.  void LietKe(float a[], int n)
02068.  {
02069.     for (int i=0; i<=n-1; i++)
02070.        if (a[i] > 0)
02071.           cout<<setw(8)<<setprecision(5)<<a[i];
02072.  }
```

Bài 008.Hãy liệt kê các giá trị trong mảng một chiều các số nguyên có chữ số đầu tiên là chữ số lẻ.

– Ví dụ:

191	29	622	763	244	647	367	234	8712	134

– Các giá trị trong mảng có chữ số đầu tiên là chữ số lẻ.

191	29	622	763	244	647	367	234	8712	134

– Kết quả: 191, 763, 367, 134.

– Khai báo hàm.

```
02073.  int ChuSoDau(int);
02074.  void LietKe(int [],int);
```

– Định nghĩa hàm tìm chữ số đầu tiên.

```
02075. int ChuSoDau(int n)
02076. {
02077.    int dt = abs(n);
02078.    while (dt >= 10)
02079.       dt /= 10;
02080.    return dt;
02081. }
```

– Định nghĩa hàm liệt kê.

```
02082. void LietKe(int a[], int n)
02083. {
02084.    for (int i=0; i<=n-1; i++)
02085.       if (ChuSoDau(a[i])%2!=0)
02086.          cout << a[i];
02087. }
```

Bài 009.Hãy liệt kê các giá trị trong mảng các số nguyên có chữ số đầu tiên là chữ số chẵn.

– Ví dụ:

191	29	622	763	244	647	367	234	8712	134

– Các giá trị trong mảng có chữ số đầu tiên là chữ số chẵn.

191	29	622	763	244	647	367	234	8712	134

– Kết quả: 29, 622, 244, 647, 234, 8712.

– Khai báo hàm.

```
02088. int ChuSoDau(int);
02089. void LietKe(int [],int);
```

– Định nghĩa hàm tìm chữ số đầu tiên.

```
02090. int ChuSoDau(int n)
02091. {
02092.    int dt = abs(n);
02093.    while (dt >= 10)
02094.       dt /= 10;
02095.    return dt;
02096. }
```

– Định nghĩa hàm liệt kê.

```
02097. void LietKe(int a[], int n)
02098. {
02099.    for (int i=0; i<=n-1; i++)
02100.       if (ChuSoDau(a[i]) % 2 == 0)
02101.          cout << a[i];
02102. }
```

Bài 010.Hãy liệt kê các giá trị có toàn chữ số lẻ trong mảng một chiều các số nguyên.

- Ví dụ:

191	29	622	753	244	647	397	7139	8712	134

- Các giá trị có toàn chữ số lẻ.

191	29	622	753	244	647	397	7139	8712	134

- Kết quả: 191, 753, 397, 7139.
- Khai báo hàm.

```
02103.  bool ktToanLe(int);
02104.  void LietKe(int[],int);
```

- Định nghĩa hàm kiểm tra số nguyên dương toàn lẻ.

```
02105.  bool ktToanLe(int n)
02106.  {
02107.     int flag = true;
02108.     int t = abs(n);
02109.     while (t != 0)
02110.     {
02111.        int dv = t%10;
02112.        if (dv % 2 == 0)
02113.           flag = false;
02114.        t /= 10;
02115.     }
02116.     return flag;
02117.  }
```

- Định nghĩa hàm liệt kê.

```
02118.  void LietKe(int a[], int n)
02119.  {
02120.     for (int i=0; i<=n-1; i++)
02121.        if (ktToanLe(a[i]))
02122.           cout << a[i];
02123.  }
```

Bài 011.Cho mảng một chiều các số nguyên. Hãy viết hàm liệt kê các giá trị trong mảng có dạng 3^m. Nếu mảng không tồn tại giá trị dạng 3^m thì hàm sẽ trả về giá trị 0.

- Ví dụ:

81	29	62	09	99	23	27	1	88	243

- Các giá trị trong mảng có dạng 3^m.

81	29	62	09	99	23	27	1	88	243

- Kết quả: 81, 09, 27, 1, 243.
- Khai báo hàm.

```
02124.  bool ktDang3m(int);
02125.  void LietKe(int [],int);
```

– Định nghĩa hàm.

```
02126.  bool ktDang3m(int n)
02127.  {
02128.     if(n < 1)
02129.        return false;
02130.     bool flag = true;
02131.     int t = n;
02132.     while (t > 1)
02133.     {
02134.        int du = t % 3;
02135.        if (du != 0)
02136.           flag = false;
02137.        t /= 3;
02138.     }
02139.     return flag;
02140.  }
```

– Định nghĩa hàm liệt kê.

```
02141.  void LietKe(int a[], int n)
02142.  {
02143.     for (int i=0; i<=n-1; i++)
02144.        if (ktDang3m(a[i]))
02145.           cout << a[i];
02146.  }
```

Bài 012.(Hạt nhân) Viết hàm liệt kê các vị trí mà giá trị tại đó là giá trị âm trong mảng một chiều các số thực (133).

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	-62	76	-99	-23	12	-98	88	-11

– Các vị trí mà giá trị tại đó là giá trị âm.

0	1	2	3	4	5	6	7	8	9
19	29	-62	76	-99	-23	12	-98	88	-11

– Kết quả: 2, 4, 5, 7, 9.

– Khai báo hàm.

```
02147.  void LietKe(float [],int);
```

– Định nghĩa hàm.

```
02148.  void LietKe(float a[], int n)
02149.  {
02150.     for (int i=0; i<=n-1; i++)
```

```
02151.        if (a[i] < 0)
02152.           cout << i ;
02153.  }
```

Bài 013.Hãy liệt kê các vị trí mà giá trị tại đó là số nguyên tố trong mảng một chiều các số nguyên.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	-62	76	-99	23	12	-98	88	11

- Các vị trí mà giá trị tại đó là số nguyên tố.

0	1	2	3	4	5	6	7	8	9
19	29	-62	76	-99	23	12	-98	88	11

- Kết quả: 0, 1, 5, 9.
- Khai báo hàm.

```
02154.  bool ktNguyenTo(int);
02155.  void LietKe(int [],int);
```

- Định nghĩa hàm kiểm tra nguyên tố.

```
02156.  bool ktNguyenTo(int k)
02157.  {
02158.     int dem = 0;
02159.     for(int i=1; i<=k; i++)
02160.        if(k%i==0)
02161.           dem++;
02162.     if(dem==2)
02163.        return true;
02164.     return false;
02165.  }
```

- Định nghĩa hàm liệt kê.

```
02166.  void LietKe(int a[], int n)
02167.  {
02168.     for (int i=0; i<=n-1; i++)
02169.        if (ktNguyenTo(a[i]))
02170.           cout << i ;
02171.  }
```

Bài 014.Hãy liệt kê các vị trí mà giá trị tại vị trí đó là số chính phương trong mảng một chiều các số nguyên.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	81	-62	76	-99	25	12	-98	16	11

- Các vị trí mà giá trị tại vị trí đó là số chính phương.

0	1	2	3	4	5	6	7	8	9
19	81	-62	76	-99	25	12	-98	16	11

– Kết quả: 1, 5, 8.

– Khai báo hàm.

```
02172. bool ktChinhPhuong(int);
02173. void LietKe(int [],int);
```

– Định nghĩa hàm.

```
02174. bool ktChinhPhuong(int n)
02175. {
02176.
02177.    for (int i = 0; (i * i) <= n; i++)
02178.       if (i * i == n)
02179.          return true;
02180.    return false;
02181. }
```

– Định nghĩa hàm liệt kê.

```
02182. void LietKe(int a[], int n)
02183. {
02184.    for (int i=0; i<=n-1; i++)
02185.       if (ktChinhPhuong(a[i]))
02186.          cout << i ;
02187. }
```

Bài 015.(Hạt nhân) Hãy liệt kê các giá trị trong mảng một chiều các số thực thuộc đoạn $[x, y]$ cho trước.

– Ví dụ: cho mảng sau và $[x, y] \equiv [30, 50]$

19	29	62	46	99	23	12	35	88	31

– Các giá trị trong mảng thuộc đoạn $[30, 50]$.

19	29	62	46	99	23	12	35	88	31

– Kết quả: 46, 35, 31.

– Khai báo hàm.

```
02188. void LietKe(float [],int, int, int);
```

– Định nghĩa hàm.

```
02189. void LietKe(float a[],int n,int x,int y)
02190. {
02191.    for (int i=0; i<=n-1; i++)
02192.       if (a[i] >= x && a[i] <= y)
02193.          cout << a[i] ;
02194. }
```

Bài 016.Hãy liệt kê các giá trị chẵn trong mảng một chiều các số nguyên thuộc đoạn $[x, y]$ cho trước. Trong đó x, y là số nguyên.

- Ví dụ: cho mảng sau và $[x, y] \equiv [30, 50]$.

19	29	62	46	99	23	12	35	88	36

- Các giá trị trong mảng thuộc đoạn $[30, 50]$.

19	29	62	46	99	23	12	35	88	36

- Các giá trị chẵn trong mảng thuộc đoạn $[30, 50]$.

19	29	62	46	99	23	12	35	88	36

- Kết quả: 46, 36.
- Khai báo hàm.

```
02195. void LietKe(int [],int,int,int);
```

- Định nghĩa hàm.

```
02196. void LietKe(int a[], int n, int x, int y)
02197. {
02198.    for (int i=0; i<=n-1; i++)
02199.       if (a[i] >= x && a[i] <= y )
02200.          if(a[i] % 2 == 0)
02201.             cout << a[i];
02202. }
```

Bài 017.Hãy xuất mảng theo yêu cầu các phần tử chẵn nằm trên một hàng. Các phần tử lẻ nằm ở hàng tiếp theo.

- Ví dụ:

19	29	62	76	99	23	12	98	88	11

- Các giá trị chẵn lẻ trong mảng.

19	29	62	76	99	23	12	98	88	11

- Kết quả:
  + 62, 76, 12, 98, 88.
  + 19, 29, 99, 23, 11.
- Khai báo hàm.

```
02203. void LietKe(int [],int);
```

- Định nghĩa hàm.

```
02204. void LietKe(int a[], int n)
02205. {
02206.    for (int i=0; i<=n-1; i++)
02207.       if (a[i] % 2 == 0)
02208.          cout << a[i] ;
02209.    cout << endl;
02210.    for (int i=0; i<=n-1; i++)
02211.       if (a[i] % 2 != 0)
```

```
02212.          cout << a[i] ;
02213.  }
```

Bài 018.(Hạt nhân) Hãy liệt kê các vị trí mà giá trị tại đó là giá trị lớn nhất trong mảng một chiều các số thực.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	98	76	81	23	12	98	88	11

– Các vị trí mà giá trị tại đó là giá trị lớn nhất.

0	1	2	3	4	5	6	7	8	9
19	29	98	76	81	23	12	98	88	11

– Kết quả: 2, 7.

– Khai báo hàm.

```
02214.  float LonNhat(float [], int);
02215.  void LietKe(float [],int);
```

– Định nghĩa hàm tìm giá trị lớn nhất.

```
02216.  float LonNhat(float a[], int n)
02217.  {
02218.     float lc = a[0];
02219.     for (int i=0; i<=n-1; i++)
02220.        if (a[i] > lc)
02221.           lc = a[i];
02222.     return lc;
02223.  }
```

– Định nghĩa hàm liệt kê.

```
02224.  void LietKe(float a[], int n)
02225.  {
02226.     float lc = LonNhat(a,n);
02227.     for (int i=0; i<=n-1; i++)
02228.        if (a[i] == lc)
02229.           cout << i ;
02230.  }
```

Bài 019.Hãy liệt kê các vị trí mà giá trị tại đó là giá trị nhỏ nhất trong mảng một chiều các số thực.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	11	76	81	23	12	98	88	11

– Các vị trí mà giá trị tại đó là giá trị nhỏ nhất.

0	1	2	3	4	5	6	7	8	9
19	29	11	76	81	23	12	98	88	11

– Kết quả: 2, 9.
– Khai báo hàm.

```
02231. float NhoNhat(float [], int );
02232. void LietKe(float [], int);
```

– Định nghĩa hàm tìm giá trị nhỏ nhất.

```
02233. float NhoNhat(float a[], int n)
02234. {
02235.    float lc = a[0];
02236.    for (int i=0; i<=n-1; i++)
02237.       if (a[i] < lc)
02238.          lc = a[i];
02239.    return lc;
02240. }
```

– Định nghĩa hàm liệt kê.

```
02241. void LietKe(float a[], int n)
02242. {
02243.    float lc = NhoNhat(a,n);
02244.    for (int i=0; i<=n-1; i++)
02245.       if (a[i] == lc)
02246.          cout << i ;
02247. }
```

Bài 020.(Hạt nhân) Hãy liệt kê các giá trị cực tiểu trong mảng một chiều các số thực. Một phần tử được gọi là cực tiểu khi nhỏ hơn các phần tử lân cận.

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Các giá trị cực tiểu trong mảng.

19	29	62	76	99	23	**12**	98	88	**11**

– Kết quả: 19, 12, 11.
– Lân cận của các phần tử trong mảng.

0	1	2	3	4	5	6	7	8	9

+ Phần tử đầu tiên có một lân cận bên phải.
+ Phần tử cuối cùng có một lân cận bên trái.
+ Các phần tử có chỉ số từ $1 \rightarrow (n-2)$ có hai lân cận.

– Khai báo hàm.

```
02248. void LietKe(float [],int);
```

– Định nghĩa hàm.

```
02249. void LietKe(float a[], int n)
02250. {
```

```
02251.     if (n == 1)
02252.     {
02253.        return ;
02254.     }
02255.     if (a[0] < a[1])
02256.        cout << a[0] ;
02257.     for (int i = 1; i <= n - 2; i++)
02258.        if (a[i] < a[i - 1] && a[i] < a[i + 1])
02259.           cout << a[i] ;
02260.     if (a[n - 1] < a[n - 2])
02261.        cout << a[n - 1] ;
02262.  }
```

Bài 021.(Hạt nhân) Hãy liệt kê các giá trị trong mảng mà thỏa điều kiện lớn hơn trị tuyệt đối của giá trị đứng liền sau nó trong mảng một chiều số thực.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	98	88	11

- Các phần tử trong mảng có phần tử đứng liền sau nó.

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	98	88	11

- Các giá trị trong mảng lớn hơn trị tuyệt đối của giá trị đứng liền sau nó.

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	98	88	11

- Kết quả: 99, 23, 98, 88.
- Khai báo hàm.

```
02263.  void LietKe(float [],int);
```

- Định nghĩa hàm.

```
02264.  void LietKe(float a[], int n)
02265.  {
02266.     for (int i = 0; i <= n - 2; i++)
02267.        if (a[i] > abs(a[i + 1]))
02268.           cout << a[i] ;
02269.
02270.  }
```

Bài 022.(Hạt nhân) Hãy liệt kê các giá trị trong mảng mà thỏa điều kiện nhỏ hơn trị tuyệt đối của giá trị đứng liền sau nó và lớn hơn giá trị đứng liền trước nó.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	58	88	11

– Các phần tử trong mảng có phần tử đứng liền sau và phần tử đứng liền trước.

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	58	88	11

– Các giá trị trong mảng thỏa điều kiện nhỏ hơn trị tuyệt đối của giá trị đứng liền sau nó và lớn hơn giá trị đứng liền trước nó.

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	58	88	11

– Kết quả: 29, 62, 76, 58.

– Khai báo hàm.

```
02271.  void LietKe(int [],int);
```

– Định nghĩa hàm.

```
02272.  void LietKe(int a[], int n)
02273.  {
02274.     for (int i = 1; i <= n - 2; i++)
02275.        if (a[i]<abs(a[i + 1]) && a[i] > a[i-1])
02276.           cout << a[i] ;
02277.  }
```

Bài 023.Cho mảng một chiều các số nguyên. Hãy viết hàm liệt kê các giá trị chẵn có ít nhất một lân cận cũng là giá trị chẵn.

– Lân cận của các phần tử trong mảng.

0	1	2	3	4	5	6	7	8	9

+ Phần tử đầu tiên có một lân cận bên phải.
+ Phần tử cuối cùng có một lân cận bên trái.
+ Các phần tử có chỉ số từ $1 \rightarrow (n-2)$ có hai lân cận.

– Ví dụ:

12	29	62	76	99	80	23	12	98	88	11

– Các giá trị chẵn trong mảng.

12	29	62	76	99	80	23	12	98	88	11

– Các giá trị chẵn có ít nhất một lân cận cũng là giá trị chẵn.

12	29	62	76	99	80	23	12	98	88	11

– Khai báo hàm.

```
02278.  void LietKe(int [],int);
```

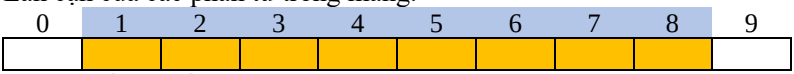
– Định nghĩa hàm.

```
02279.  void LietKe(int a[], int n)
02280.  {
02281.     if (n <= 1)
```

```
02282.     {
02283.        return;
02284.     }
02285.     if (a[0]%2 == 0 && a[1]%2 == 0)
02286.        cout << a[0] ;
02287.     for (int i = 1; i <= n - 2; i++)
02288.        if (a[i] % 2 == 0)
02289.           if (a[i-1]% 2==0 || a[i+1]% 2==0)
02290.              cout << a[i] ;
02291.     if (a[n - 1]%2 == 0 && a[n-2]%2 ==0)
02292.        cout << a[n - 1] ;
02293.  }
```

Bài 024.Cho mảng một chiều các số thực. Hãy viết hàm liệt kê tất cả các giá trị trong mảng có ít nhất một lân cận trái dấu với nó.

- Lân cận của các phần tử trong mảng.



  + Phần tử đầu tiên có một lân cận bên phải.
  + Phần tử cuối cùng có một lân cận bên trái.
  + Các phần tử có chỉ số từ $1 \to (n-2)$ có hai lân cận.
- Ví dụ:

-19	29	62	76	99	-23	12	98	88	11

- Các giá trị trong mảng có ít nhất một lân cận trái dấu.

-19	29	62	76	99	-23	12	98	88	11

- Kết quả: $-19, 29, 99, -23, 12$.
- Khai báo hàm.

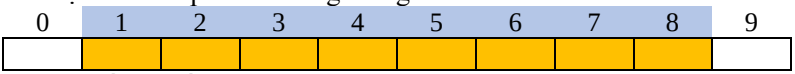
```
02294.  void LietKe(float [],int);
```

- Định nghĩa hàm.

```
02295.  void LietKe(float a[], int n)
02296.  {
02297.     if (n == 1)
02298.        return ;
02299.     if (a[0] * a[1] < 0)
02300.        cout << a[0] ;
02301.     for (int i = 1; i <= n - 2; i++)
02302.        if (a[i] * a[i + 1] < 0
02303.            || a[i] * a[i-1] < 0)
02304.           cout << a[i] ;
02305.     if (a[n - 1] * a[n - 2] < 0)
02306.        cout << a[n - 1] ;
02307.  }
```

Bài 025.Hãy liệt kê các giá trị cực đại trong mảng một chiều các số thực. Một phần tử được gọi là cực đại khi lớn hơn các phần tử lân cận.

- Lân cận của các phần tử trong mảng.



  + Phần tử đầu tiên có một lân cận bên phải.
  + Phần tử cuối cùng có một lân cận bên trái.
  + Các phần tử có chỉ số từ $1 \rightarrow (n-2)$ có hai lân cận.
- Ví dụ:

19	29	62	76	99	23	12	18	-88	111

- Các giá trị cực đại trong mảng.

19	29	62	76	99	23	12	18	-88	111

- Kết quả: 99, 18, 111.
- Khai báo hàm.

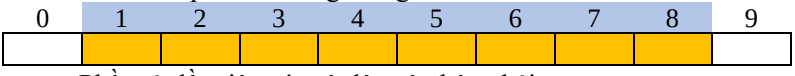
```
02308. void LietKe(float [],int);
```

- Định nghĩa hàm.

```
02309. void LietKe(float a[], int n)
02310. {
02311.    if (n == 1)
02312.       return ;
02313.    if (a[0] > a[1])
02314.       cout << a[0] ;
02315.    for (int i = 1; i <= n - 2; i++)
02316.       if (a[i] > a[i - 1] && a[i] > a[i + 1])
02317.          cout << a[i] ;
02318.    if (a[n - 1] > a[n - 2])
02319.       cout << a[n - 1] ;
02320. }
```

Bài 026.Hãy liệt kê các giá trị cực trị trong mảng một chiều các số thực. Một phần tử được gọi là cực trị khi nó là cực đại hoặc cực tiểu.

- Lân cận của các phần tử trong mảng.



  + Phần tử đầu tiên có một lân cận bên phải.
  + Phần tử cuối cùng có một lân cận bên trái.
  + Các phần tử có chỉ số từ $1 \rightarrow (n-2)$ có hai lân cận.
- Ví dụ:

19	29	62	76	99	23	12	18	-88	111

– Các giá trị cực đại trong mảng.

19	29	62	76	99	23	12	18	-88	111

– Các giá trị cực tiểu trong mảng.

19	29	62	76	99	23	12	18	-88	111

– Các giá trị cực trị trong mảng.

19	29	62	76	99	23	12	18	-88	111

– Kết quả: 19, 99, 12, 18, -88, 111.

– Khai báo hàm.

```
02321. void LietKe(float [],int);
```

– Định nghĩa hàm.

```
02322. void LietKe(float a[], int n)
02323. {
02324.    if (n == 1)
02325.       return ;
02326.    if (a[0]<a[1] || a[0]>a[1])
02327.       cout << a[0] ;
02328.    for (int i = 1; i <= n - 2; i++)
02329.       if ((a[i]<a[i-1] && a[i]<a[i+1]) ||
02330.           (a[i]>a[i-1] && a[i]>a[i+1]))
02331.          cout << a[i] ;
02332.    if (a[n-1]<a[n-2] || a[n-1] > a[n-2])
02333.       cout << a[n - 1] ;
02334. }
```

Bài 027.(Hạt nhân) Hãy liệt kê các vị trí trong mảng một chiều các số thực mà giá trị tại vị trí đó bằng giá trị âm đầu tiên trong mảng.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	-29	62	-76	99	23	-29	98	88	-29

– Âm đầu tiên là.

19	-29	62	-76	99	23	-29	98	88	-29

– Các vị trí giá trị tại vị trí đó bằng giá trị âm đầu tiên.

0	1	2	3	4	5	6	7	8	9
19	-29	62	-76	99	23	-29	98	88	-29

– Kết quả: 1, 6, 9.

– Khai báo hàm.

```
02335. float AmDau(float [], int );
02336. void LietKe(float [],int);
```

– Định nghĩa hàm tìm giá trị âm đầu tiên.

```
02337. float AmDau(float a[], int n)
02338. {
```

```
02339.     for (int i=0; i<=n-1; i++)
02340.        if (a[i] < 0)
02341.           return a[i];
02342.     return 0;
02343.  }
```

– Định nghĩa hàm liệt kê.

```
02344.  void LietKe(float a[], int n)
02345.  {
02346.     float ad = AmDau(a,n);
02347.     if (ad == 0)
02348.     {
02349.        cout << "\nKhong co gia tri am";
02350.        return;
02351.     }
02352.     for (int i=0; i<=n-1; i++)
02353.        if (a[i] == ad)
02354.           cout << i ;
02355.  }
```

Bài 028.(Hạt nhân) Hãy liệt kê các vị trí mà giá trị tại các vị trí đó bằng giá trị dương nhỏ nhất trong mảng một chiều các số thực.

– Ví dụ:

19	-29	11	76	99	11	-12	-98	88	11

– Dương nhỏ nhất.

19	-29	11	76	99	11	-12	-98	88	11

– Các vị trí mà giá trị tại các vị trí đó bằng giá trị dương nhỏ nhất.

0	1	2	3	4	5	6	7	8	9
19	-29	11	76	99	11	-12	-98	88	11

– Kết quả: 2, 5, 9.

– Khai báo hàm.

```
02356.  float DuongDau(float [], int );
02357.  float DuongNhoNhat(float [], int );
02358.  void LietKe(float [],int);
```

– Định nghĩa hàm tìm giá trị dương đầu tiên.

```
02359.  float DuongDau(float a[], int n)
02360.  {
02361.     for (int i=0; i<=n-1; i++)
02362.        if (a[i] > 0)
02363.           return a[i];
02364.     return -1;
02365.  }
```

– Định nghĩa hàm tìm giá trị dương nhỏ nhất.

```
02366.  float DuongNhoNhat(float a[], int n)
02367.  {
02368.     float lc = DuongDau(a, n);
02369.     if(lc==-1)
02370.        return -1;
02371.     for (int i=0; i<=n-1; i++)
02372.        if (a[i] > 0 && a[i] < lc)
02373.           lc = a[i];
02374.     return lc;
02375.  }
```

– Định nghĩa hàm liệt kê.

```
02376.  void LietKe(float a[], int n)
02377.  {
02378.     float dd = DuongNhoNhat(a,n);
02379.     if (dd == -1)
02380.     {
02381.        cout << "\nKhong co gia tri duong";
02382.        return;
02383.     }
02384.     for (int i=0; i<=n-1; i++)
02385.        if (a[i] == dd)
02386.           cout << i ;
02387.  }
```

Bài 029.Hãy liệt kê các vị trí chẵn lớn nhất trong mảng một chiều các số nguyên.

– Ví dụ:

19	-29	11	76	99	11	-12	-98	76	11

– Các giá trị chẵn

19	-29	11	76	99	11	-12	-98	76	11

– Chẵn lớn nhất.

19	-29	11	76	99	11	-12	-98	76	11

– Các vị trí mà giá trị tại các vị trí đó bằng giá trị chẵn lớn nhất.

0	1	2	3	4	5	6	7	8	9
19	-29	11	76	99	11	-12	-98	76	11

– Kết quả: 3, 8.

– Khai báo hàm.

```
02388.  int ChanDau(int [], int );
02389.  int ChanLonNhat(int [], int );
02390.  void LietKe(int [],int);
```

– Định nghĩa hàm tìm giá trị chẵn đầu.

```
02391.  int ChanDau(int a[], int n)
02392.  {
02393.     for (int i=0; i<=n-1; i++)
02394.        if (a[i] % 2 == 0)
02395.           return a[i];
02396.     return -1;
02397.  }
```

– Định nghĩa hàm tìm giá trị chẵn lớn nhất.

```
02398.  int ChanLonNhat(int a[], int n)
02399.  {
02400.     int lc = ChanDau(a, n);
02401.     if (lc == -1)
02402.        return -1;
02403.     for (int i=0; i<=n-1; i++)
02404.        if (a[i] % 2 == 0 && a[i] > lc)
02405.           lc = a[i];
02406.     return lc;
02407.  }
```

– Định nghĩa hàm liệt kê.

```
02408.  void LietKe(int a[], int n)
02409.  {
02410.     int dd = ChanLonNhat(a, n);
02411.     if (dd == -1)
02412.     {
02413.        cout << "\nKhong co Chan";
02414.        return;
02415.     }
02416.     for (int i=0; i<=n-1; i++)
02417.        if (a[i] == dd)
02418.           cout << i ;
02419.  }
```

Bài 030.(Hạt nhân) Cho mảng số thực có nhiều hơn hai giá trị và các giá trị trong mảng khác nhau từng đôi một. Hãy viết hàm liệt kê tất cả các cặp giá trị (x, y) với x, y là hai giá trị nằm trong mảng.

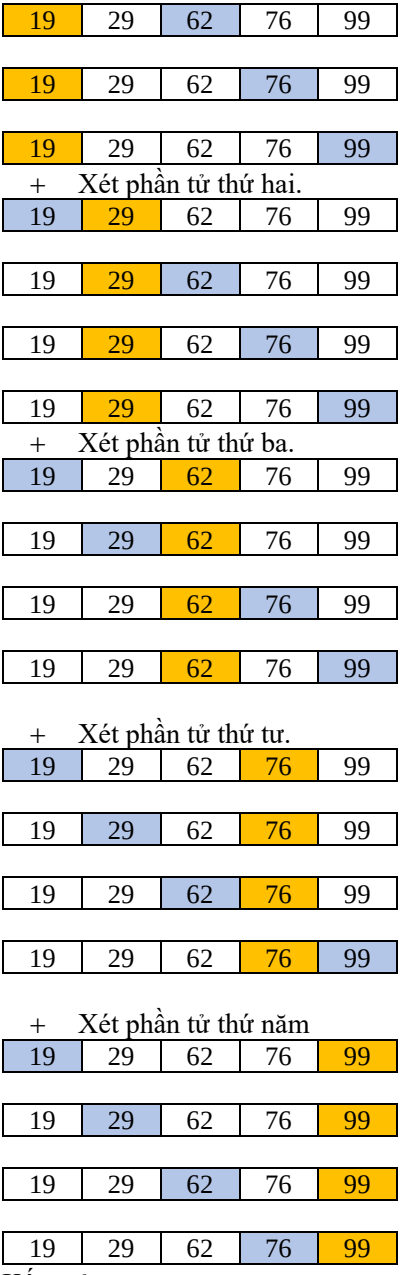
– Ví dụ:

19	29	62	76	99

– Các cặp giá trị (x, y) với x, y là hai giá trị nằm trong mảng.

+ Xét phần tử thứ nhất.

19	29	62	76	99



– Kết quả:
  + $(19, 29), (19, 62), (19, 76), (19, 99)$.
  + $(29,19), (29, 62), (29, 76), (29, 99)$.
  + $(62, 19), (62,29), (62, 76), (62, 99)$.

+ $(76,19),(76,29),(76,62),(76,99)$.
+ $(99,19),(99,29),(99,62),(99,76)$.

– Khai báo hàm.

```
02420. void LietKe(float [],int);
```

– Định nghĩa hàm.

```
02421. void LietKe(float a[], int n)
02422. {
02423.    for (int i = 0; i<=n-1; i++)
02424.       for (int j = 0; j<=n-1; j++)
02425.          if (i != j)
02426.          {
02427.             cout << "(" << a[i] << "," ;
02428.             cout << a[j] << ")";
02429.          }
02430. }
```

Bài 031.Cho mảng số thực có nhiều hơn hai giá trị và các giá trị trong mảng khác nhau từng đôi một. Hãy viết hàm liệt kê tất cả các cặp giá trị (x,y) trong mảng thỏa điều kiện $x \le y$.

– Ví dụ:

19	9	6	7	99

– Các cặp giá trị có trong mảng.

+ $(19,9),(19,6),(19,7),(19,99)$.
+ $(9,19),(9,6),(9,7),(9,99)$.
+ $(6,19),(6,9),(6,7),(6,99)$.
+ $(7,19),(7,9),(7,6),(7,99)$.
+ $(99,19),(99,9),(99,6),(99,7)$.

– Các cặp giá trị (x,y) thỏa điều kiện $x \le y$.

+ $(19,99)$.
+ $(9,19),(9,99)$.
+ $(6,19),(6,9),(6,7),(6,99)$.
+ $(7,19),(7,9),(7,99)$.

– Khai báo hàm.

```
02431. void LietKe(float [],int);
```

– Định nghĩa hàm.

```
02432. void LietKe(float a[], int n)
02433. {
02434.    for (int i = 0; i <= n-1; i++)
02435.       for (int j = 0; j <= n - 1; j++)
02436.          if (i != j && a[i] <= a[j])
02437.             cout << "("<<a[i]<<","<<a[j]<<")";
```

```
02438. }
```

Bài 032. (*) Cho mảng số thực có nhiều hơn ba giá trị và các giá trị trong mảng khác nhau từng đôi một. Hãy liệt kê tất cả các bộ ba giá trị (x, y, z) thỏa điều kiện $x = y + z$ với x, y, z là ba giá trị khác nhau trong mảng (195).

– Ví dụ:

13	9	6	7	99

- Phần tử đầu tiên.
  - (13,9,6), (13,9,7), (13,9,99).
  - (13,6,9), (13,6,7), (13,6,99).
  - (13,7,9), (13,7,6), (13,7,99).
  - (13,99,9), (13,99,6), (13,99,7).
- Phần tử thứ hai.
  - (9,13,6), (9,13,7), (9,13,99).
  - (9,6,13), (9,6,7), (9,6,99).
  - (9,7,13), (9,7,6), (9,7,99).
  - (9,99,13), (9,99,6), (9,99,7)
- Phần tử thứ ba.
  - (6,13,6), (6,13,7), (6,13,99).
  - (6,9,13), (6,9,7), (6,9,99).
  - (6,7,13), (6,7,9), (6,7,99).
  - (6,99,13), (6,99,9), (6,99,7)
- Phần tử thứ tư.
  - (7,13,9), (7,13,6), (7,13,99).
  - (7,9,13), (7,9,6), (7,9,99).
  - (7,6,13), (7,6,9), (7,6,99).
  - (7,99,13), (7,99,9), (7,99,6)
- Phần tử thứ năm.
  - (99,13,9),(99,13,6), (99,13,7).
  - (99,9,13), (99,9,6), (99,9,7).
  - (99,6,13), (99,6,9), (99,6,7).
  - (99,7,13), (99,7,9), (99,7,6)
- Các bộ ba giá trị (x, y, z) thỏa điều kiện $x = y + z$ là (13,6,7), (13,7,6).
- Khai báo hàm.

```
02439.  void LietKe(float [],int);
```

- Định nghĩa hàm.

```
02440.  void LietKe(float a[], int n)
02441.  {
02442.     for (int i=0; i<=n-1; i++)
02443.        for (int j=0; j<=n-1; j++)
02444.           for (int k=0; k<=k-1; k++)
02445.              if(i!=j && j!=k && i!=k
02446.                 a[i] == a[j] + a[k])
02447.              {
02448.                 cout << “("<<a[i]<<","<<a[j];
02449.                 cout << “,"<<a[z]<<")"<<endl;
02450.              }
```

```
02451. }
```

01.18.03 Kỹ thuật Tính toán

Bài 033.(Hạt nhân) Tính tổng các phần tử trong mảng các số thực.

- Ví dụ:

19	29	62	76	99

- Kết quả: 285.
- Khai báo hàm.

```
02452. float Tong(float [],int);
```

- Định nghĩa hàm.

```
02453. float Tong(float a[], int n)
02454. {
02455.    float s = 0;
02456.    for (int i=0; i<=n-1; i++)
02457.       s += a[i];
02458.    return s;
02459. }
```

Bài 034.(Hạt nhân) Tính tổng các giá trị dương trong mảng các số thực.

- Ví dụ:

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

- Các số dương trong mảng

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

- Kết quả: 101.7
- Khai báo hàm.

```
02460. float TongDuong(float [],int);
```

- Định nghĩa hàm.

```
02461. float TongDuong(float a[], int n)
02462. {
02463.    float s = 0;
02464.    for (int i=0; i<=n-1; i++)
02465.      if(a[i] > 0)
02466.         s += a[i];
02467.    return s;
02468. }
```

Bài 035.Tính tổng các giá trị có chữ số hàng chục là chữ số 5 có trong mảng các số nguyên.

- Ví dụ:

191	-41	644	151	99	253	-12	-58	28	11

– Các giá trị có chữ số hàng chục là chữ số 5.

191	-41	644	151	99	253	-12	-58	28	11

– Kết quả: 346.

– Khai báo hàm.

```
02469.  int HangChuc(int);
02470.  int TongGiaTri(int [],int);
```

– Định nghĩa hàm.

```
02471.  int HangChuc(int n)
02472.  {
02473.     int t = abs(n);
02474.     return (t / 10) % 10;
02475.  }
```

– Định nghĩa hàm tính tổng.

```
02476.  int TongGiaTri(int a[], int n)
02477.  {
02478.     int s = 0;
02479.     for (int i=0; i<=n-1; i++)
02480.     {
02481.        if (HangChuc(a[i]) == 5)
02482.           s += a[i];
02483.     }
02484.     return s;
02485.  }
```

Bài 036. Tính tổng các giá trị chính phương trong mảng một chiều các số nguyên.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	81	-62	76	-99	25	12	-98	16	11

– Các số chính phương.

0	1	2	3	4	5	6	7	8	9
19	81	-62	76	-99	25	12	-98	16	11

– Kết quả: 122.

– Khai báo hàm.

```
02486.  bool ktChinhPhuong(int);
02487.  int TongChinhPhuong(int [],int);
```

– Định nghĩa hàm.

```
02488.  bool ktChinhPhuong(int n)
02489.  {
02490.     for (int i = 1; (i * i) <= n; i++)
```

```
02491.       if (i * i == n)
02492.          return true;
02493.    return false;
02494. }
```

– Định nghĩa hàm tính tổng.

```
02495. int TongChinhPhuong (int a[], int n)
02496. {
02497.    int s = 0;
02498.    for (int i=0; i<=n-1; i++)
02499.       if (ktChinhPhuong(a[i]))
02500.          s += a[i];
02501.    return s;
02502. }
```

Bài 037.Tính tổng các giá trị đối xứng trong mảng các số nguyên.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	181	-62	76	-9	25	12	1221	16	11

– Các số đối xứng.

0	1	2	3	4	5	6	7	8	9
19	181	-62	76	-9	25	12	1221	16	11

– Kết quả: 1404.

– Khai báo hàm.

```
02503. bool ktDoiXung(int);
02504. int TongDoiXung(int [],int);
```

– Định nghĩa hàm.

```
02505. bool ktDoiXung(int n)
02506. {
02507.    int t = abs(n);
02508.    int dn = 0;
02509.    while (t != 0)
02510.    {
02511.       dn = dn * 10 + t % 10;
02512.       t /= 10;
02513.    }
02514.    if (dn == abs(n))
02515.       return true;
02516.    return false;
02517. }
```

– Định nghĩa hàm tính tổng đối xứng.

```
02518. int TongDoiXung (int a[], int n)
```

```
02519. {
02520.    int s = 0;
02521.    for (int i=0; i<=n-1; i++)
02522.       if (ktDoiXung(a[i]))
02523.          s += a[i];
02524.    return s;
02525. }
```

Bài 038.Tính tổng các giá trị có chữ số đầu tiên là chữ số lẻ trong mảng một chiều các số nguyên.

- Ví dụ:

19	29	62	76	99	23	12	98	88	11

- Các giá trị có chữ số đầu tiên là chữ số lẻ:

19	29	62	76	99	23	12	98	88	11

- Kết quả: 315.
- Khai báo hàm.

```
02526. int ChuSoDau(int);
02527. int TongGiaTri(int [],int);
```

- Định nghĩa hàm.

```
02528. int ChuSoDau(int n)
02529. {
02530.    int t = abs(n);
02531.    while (t >= 10)
02532.       t /= 10;
02533.    return t;
02534. }
```

- Định nghĩa hàm tính tổng.

```
02535. int Tong(int a[], int n)
02536. {
02537.    int s = 0;
02538.    for (int i=0; i<=n-1; i++)
02539.    {
02540.       if (ChuSoDau(a[i]) & 1)
02541.          s += a[i];
02542.    }
02543.    return s;
02544. }
```

Bài 039.Tính tổng các giá trị có chữ số đầu tiên là chữ số chẵn có trong mảng các số nguyên.

- Ví dụ:

191	29	622	763	244	647	367	234	8712	134

– Các giá trị trong mảng có chữ số đầu tiên là chữ số chẵn.

191	29	622	763	244	647	367	234	8712	134

– Kết quả: $29 + 622 + 244 + 647 + 234 + 8712 = 10{,}488$.

– Khai báo hàm.

```
02545.  int ChuSoDau(int);
02546.  int TongGiaTri(int [],int);
```

– Định nghĩa hàm.

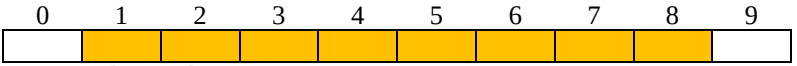
```
02547.  int ChuSoDau(int n)
02548.  {
02549.     int t = abs(n);
02550.     while (t >= 10)
02551.        t /= 10;
02552.     return t;
02553.  }
```

– Định nghĩa hàm tính tổng.

```
02554.  int TongGiaTri (int a[], int n)
02555.  {
02556.     int s = 0;
02557.     for (int i=0; i<=n-1; i++)
02558.        if (ChuSoDau(a[i]) % 2 == 0)
02559.           s += a[i];
02560.     return s;
02561.  }
```

Bài 040.(Hạt nhân) Tính tổng các giá trị nhỏ hơn các giá trị xung quanh trong mảng một chiều các số thực.

– Lân cận của các phần tử trong mảng.



+ Phần tử đầu tiên có một lân cận bên phải.
+ Phần tử cuối cùng có một lân cận bên trái.
+ Các phần tử có chỉ số từ $1 \rightarrow (n-2)$ có hai lân cận.

– Ví dụ:

19	29	62	76	99	23	12	18	-88	111

– Các giá trị cực tiểu trong mảng.

19	29	62	76	99	23	12	18	-88	111

– Kết quả: $19 + 12 + (-88) = -57$.

– Khai báo hàm.

```
02562.  float TongCucTieu(float [],int);
```

– Định nghĩa hàm.

```
02563.  float TongCucTieu (float a[], int n)
02564.  {
02565.     float s = 0;
02566.     if (n == 1)
02567.        s += a[0];
02568.     for (int i = 1; i <= n - 2; i++)
02569.        if (a[i] < a[i + 1] && a[i] < a[i - 1])
02570.           s += a[i];
02571.     if (a[n - 1] < a[n - 2])
02572.        s += a[n - 1];
02573.     return s;
02574.  }
```

Bài 041.Tính tổng các giá trị lớn hơn giá trị đứng liền trước nó trong mảng một chiều các số thực.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	58	88	11

– Các phần tử trong mảng có phần tử đứng liền trước.

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	58	88	11

– Các giá trị trong mảng thỏa điều kiện lớn hơn giá trị đứng liền trước nó.

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	58	88	11

– Kết quả: 412.

– Khai báo hàm.

```
02575.  float TongGiaTri(float [],int);
```

– Định nghĩa hàm.

```
02576.  float TongGiaTri(float a[],int n)
02577.  {
02578.      float s = 0;
02579.      for(int i=1;i<n;i++)
02580.          if(a[i]>a[i-1])
02581.              s = s + a[i];
02582.      return s;
02583.  }
```

Bài 042.Tính tổng các giá trị lớn hơn trị tuyệt đối của giá trị đứng liền sau nó trong mảng một chiều các số thực.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9

19	29	62	76	99	23	12	58	88	11

– Các phần tử trong mảng có phần tử đứng liền sau.

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	58	88	11

– Các giá trị trong mảng thỏa điều kiện lớn hơn trị tuyệt đối của giá trị đứng liền sau.

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	58	88	11

– Kết quả: 210.

– Khai báo hàm.

```
02584. float TongGiaTri(float [],int);
```

– Định nghĩa hàm.

```
02585. float TongGiaTri(float a[],int n)
02586. {
02587.     float s = 0;
02588.     for(int i=0;i<=n-2;i++)
02589.         if(a[i]>abs(a[i+1]))
02590.             s = s + a[i];
02591.     return s;
02592. }
```

Bài 043.Tính tổng các giá trị lớn hơn các giá trị xung quanh trong mảng một chiều các số thực.

– Lân cận của các phần tử trong mảng.

0	1	2	3	4	5	6	7	8	9

+ Phần tử đầu tiên có một lân cận bên phải.
+ Phần tử cuối cùng có một lân cận bên trái.
+ Các phần tử có chỉ số từ $1 \rightarrow (n-2)$ có hai lân cận.

– Ví dụ:

19	29	62	76	99	23	12	18	-88	111

– Các giá trị cực đại trong mảng.

19	29	62	76	99	23	12	18	-88	111

– Kết quả: $99 + 18 + 111 = 228$.

– Khai báo hàm.

```
02593. float TongCucDai(float [],int);
```

– Định nghĩa hàm.

```
02594. float TongCucDai(float a[],int n)
02595. {
02596.     if(n<=1)
02597.         return 0;
```

```
02598.      float s = 0;
02599.      if(a[0]>a[1])
02600.          s = s + a[0];
02601.      for(int i=1;i<=n-2;i++)
02602.          if(a[i]>a[i-1] && a[i]>a[i+1])
02603.              s = s + a[i];
02604.      if(a[n-1]>a[n-2])
02605.          s = s + a[n-1];
02606.      return s;
02607.  }
```

Bài 044. Tính tổng các phần tử "cực trị" trong mảng. Một phần tử được gọi là cực trị khi nó lớn hơn hoặc nhỏ hơn các phần tử xung quanh nó.

- Lân cận của các phần tử trong mảng.

  + Phần tử đầu tiên có một lân cận bên phải.
  + Phần tử cuối cùng có một lân cận bên trái.
  + Các phần tử có chỉ số từ $1 \rightarrow (n-2)$ có hai lân cận.
- Ví dụ:

19	29	62	76	99	23	12	18	-88	111

- Các giá trị cực đại trong mảng.

19	29	62	76	**99**	23	12	**18**	-88	**111**

- Các giá trị cực tiểu trong mảng.

19	29	62	76	99	23	**12**	18	**-88**	111

- Các giá trị cực trị trong mảng.

19	29	62	76	**99**	23	**12**	**18**	**-88**	**111**

- Kết quả: $19 + 99 + 12 + 18 + (-88) + 111 = 171$.
- Khai báo hàm.

```
02608.  int TongCucDai(int [],int);
02609.  int TongCucTieu(int [],int);
02610.  int TongCucTri(int [],int);
```

- Định nghĩa hàm tính tổng cực đại.

```
02611.  int TongCucDai(int a[],int n)
02612.  {
02613.      if(n<=1)
02614.          return 0;
02615.      int s = 0;
02616.      if(a[0]>a[1])
02617.          s = s + a[0];
02618.      for(int i=1;i<=n-2;i++)
```

```
02619.          if(a[i]>a[i-1] && a[i]>a[i+1])
02620.              s = s + a[i];
02621.      if(a[n-1]>a[n-2])
02622.          s = s + a[n-1];
02623.      return s;
02624.  }
```

– Định nghĩa hàm tính tổng cực tiểu.

```
02625.  int TongCucTieu(int a[],int n)
02626.  {
02627.      if(n<=1)
02628.          return 0;
02629.      int s = 0;
02630.      if(a[0]<a[1])
02631.          s = s + a[0];
02632.      for(int i=1;i<=n-2;i++)
02633.          if(a[i]<a[i-1] && a[i]<a[i+1])
02634.              s = s + a[i];
02635.      if(a[n-1]<a[n-2])
02636.          s = s + a[n-1];
02637.      return s;
02638.  }
```

– Định nghĩa hàm tính tổng cực trị.

```
02639.  int TongCucTri(int a[],int n)
02640.  {
02641.      int s1 = TongCucDai(a,n);
02642.      int s2 = TongCucTieu(a,n);
02643.      return (s1+s2);
02644.  }
```

Bài 045.(Hạt nhân) Tính trung bình cộng các số dương trong mảng một chiều các số thực.

– Khái niệm: trung bình cộng của các số dương trong mảng bằng tổng các số dương chia cho số lượng số dương trong mảng.

– Ví dụ:

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

– Các số dương trong mảng

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

– Kết quả: $\frac{101.7}{5} = 20.34$

– Khai báo hàm.

```
02645.  float TongDuong(float [],int);
02646.  int DemDuong(float [],int);
```

```
02647.  float TrungBinhCong(float [],int);
```

– Định nghĩa hàm tính tổng giá trị dương.

```
02648.  float TongDuong(float a[],int n)
02649.  {
02650.      float s = 0;
02651.      for (int i = 0; i < n; i++)
02652.          if(a[i]>0)
02653.              s = s + a[i];
02654.      return s;
02655.  }
```

– Định nghĩa hàm đếm dương giá trị dương.

```
02656.  int DemDuong(float a[],int n)
02657.  {
02658.      int dem = 0;
02659.      for (int i = 0; i < n; i++)
02660.          if(a[i]>0)
02661.              dem++;
02662.      return dem;
02663.  }
```

– Định nghĩa hàm tính trung bình cộng các giá trị dương.

```
02664.  float TrungBinhCong (float a[],int n)
02665.  {
02666.      float s = TongDuong(a,n);
02667.      int dem = DemDuong(a,n);
02668.      if(dem==0)
02669.          return 0;
02670.      return s/dem;
02671.  }
```

Bài 046.Tính trung bình cộng các số nguyên tố trong mảng một chiều các số nguyên.

– Ví dụ:

19	29	62	76	223	23	227	98	88	53

– Các số nguyên tố có trong mảng

19	29	62	76	223	23	227	98	88	53

– Kết quả: $\frac{19+29+223+23+227+53}{6} = \frac{574}{6} = 95.6.$

– Khai báo hàm.

```
02672.  bool ktNguyenTo(int);
02673.  int DemNguyenTo(int [],int);
02674.  int TongNguyenTo(int [],int);
02675.  float TrungBinhCong(int [],int);
```

– Định nghĩa hàm kiểm tra nguyên tố.

```
02676.  bool ktNguyenTo(int k)
02677.  {
02678.     int dem = 0;
02679.     for(int i=1; i<=k; i++)
02680.        if(k%i==0)
02681.           dem++;
02682.     if(dem==2)
02683.        return true;
02684.     return false;
02685.  }
```

– Định nghĩa hàm đếm số lượng giá trị nguyên tố.

```
02686.  int DemNguyenTo(int a[],int n)
02687.  {
02688.     int dem = 0;
02689.     for (int i = 0; i < n; i++)
02690.         if(ktNguyenTo(a[i]))
02691.             dem++;
02692.     return dem;
02693.  }
```

– Định nghĩa hàm tính tổng các giá trị nguyên tố.

```
02694.  int TongNguyenTo(int a[],int n)
02695.  {
02696.     int s = 0;
02697.     for (int i = 0; i < n; i++)
02698.         if(ktNguyenTo(a[i]))
02699.             s = s + a[i];
02700.     return s;
02701.  }
```

– Định nghĩa hàm tính trung bình cộng giá trị nguyên tố.

```
02702.  float TrungBinhCong (int a[],int n)
02703.  {
02704.      int s = TongNguyenTo(a,n);
02705.      int dem = DemNguyenTo(a,n);
02706.      if(dem==0)
02707.          return 0;
02708.      return (float)s/dem;
02709.  }
```

Bài 047.Tính trung bình cộng các giá trị lớn hơn giá trị x trong mảng một chiều các số thực.

– Ví dụ: cho mảng sau và $x = 10.6$

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

– Các giá trị lớn hơn 10.6

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

– Kết quả: 44.05

– Khai báo hàm.

```
02710. float TongGiaTri(float [],int, float);
02711. int DemGiaTri(float [],int, float);
02712. float TrungBinhCong(float [],int, float);
```

– Định nghĩa hàm tính tổng các giá trị lớn hơn x.

```
02713. float TongGiaTri(float a[],int n, float x)
02714. {
02715.    float s = 0;
02716.    for (int i = 0; i < n; i++)
02717.        if(a[i]>x)
02718.            s = s + a[i];
02719.    return s;
02720. }
```

– Định nghĩa hàm đếm số lượng các giá trị lớn hơn x.

```
02721. int DemGiaTri(float a[],int n, float x)
02722. {
02723.    int dem = 0;
02724.    for (int i = 0; i < n; i++)
02725.        if(a[i]>x)
02726.            dem++;
02727.    return dem;
02728. }
```

– Định nghĩa hàm tính tổng trung bình cộng các giá trị lớn hơn x.

```
02729. float TrungBinhCong(float a[],int n, float x)
02730. {
02731.     float s= TongGiaTri(a,n,x);
02732.     int dem = DemgiaTri(a,n,x);
02733.     if(dem==0)
02734.         return 0;
02735.     return s/dem;
02736. }
```

Bài 048.Tính trung bình nhân các giá trị dương có trong mảng một chiều các số thực. Biết rằng trung bình nhân các số dương trong mảng bằng căn bậc số lượng số dương của tích các số dương trong mảng.

– Ví dụ:

1.9	2.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

– Các số dương trong mảng

1.9	2.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

– Kết quả: 5.368

– Khai báo hàm.

```
02737. float TichDuong(float [], int);
02738. int DemDuong(float [], int);
02739. float TrungBinhNhan(float [],int);
```

– Định nghĩa hàm tính tích các giá trị dương.

```
02740. float TichDuong(float a[], int n)
02741. {
02742.    float T = 01;
02743.    for (int i=0; i<n; i++)
02744.       if (a[i] > 0)
02745.          T = T * a[i];
02746.    return T;
02747. }
```

– Định nghĩa hàm đếm số lượng giá trị dương.

```
02748. int DemDuong(float a[], int n)
02749. {
02750.    int dem = 0;
02751.    for (int i=0; i<n; i++)
02752.       if (a[i] > 0)
02753.          dem++;
02754.    return dem;
02755. }
```

– Định nghĩa hàm tính trung bình nhân các giá trị dương.

```
02756. float TrungBinhNhan(float a[], int n)
02757. {
02758.    float T = TichDuong(a, n);
02759.    int dem = DemDuong(a, n);
02760.    if (dem == 0)
02761.       return 0;
02762.    return pow(T, (float)1 / dem);
02763. }
```

Bài 049. (*) Cho mảng số thực có nhiều hơn hai giá trị và các giá trị trong mảng khác nhau từng đôi một. Tính khoảng cách trung bình giữa các cặp giá trị trong mảng.

– Ví dụ:

1.9	2.9	-0.6	7.6

- Các cặp giá trị có trong mảng
  + Xét phần tử đầu tiên
    - $(1.9; 2.9)$ khoảng cách 1.
    - $(1.9; -0.6)$ khoảng cách 2.5.
    - $(1.9;\ 7.6)$ khoảng cách 5.7.
  + Xét phần tử thứ hai
    - $(2.9; 1.9)$ khoảng cách 1.
    - $(2.9; -0.6)$ khoảng cách 3.5.
    - $(2.9;\ 7.6)$ khoảng cách 4.7.
  + Xét phần tử thứ ba
    - $(-0.6; 1.9)$ khoảng cách 2.5.
    - $(-0.6; 2.9)$ khoảng cách 3.5.
    - $(-0.6;\ 7.6)$ khoảng cách 7.0.
  + Xét phần tử thứ tư
    - $(7.6; 1.9)$ khoảng cách 5.7.
    - $(7.6;\ 2.9)$ khoảng cách 4.7.
    - $(7.6; -0.6)$ khoảng cách 8.2.
- Số lượng các cặp giá trị trong mảng: 12.
- Tổng khoảng cách: 50.
- Trung bình khoảng cách là: 4.1667.
- Khai báo hàm.

```
02764. float TrungBinh(float [],int);
```

- Định nghĩa hàm.

```
02765. float TrungBinh(float a[], int n)
02766. {
02767.    float s = 0;
02768.    int dem = 0;
02769.    for (int i=0; i<n; i++)
02770.       for (int j=0; j<n; j++)
02771.          if (i != j)
02772.          {
02773.             s = s + abs(a[i] - a[j]);
02774.             dem++;
02775.          }
02776.    if (dem == 0)
02777.       return 0;
02778.    return s / dem;
02779. }
```

01.18.04 Kỹ thuật Đếm

Bài 050.(Hạt nhân) Đếm số lượng giá trị chẵn có trong mảng một chiều các số nguyên.

- Ví dụ:

19	29	62	76	99	23	12	98	88	11

- Các giá trị chẵn trong mảng.

19	29	62	76	99	23	12	98	88	11

- Kết quả: 5.
- Khai báo hàm.

```
02780. int  DemChan (int [],int);
```

- Định nghĩa hàm đếm số lượng giá trị chẵn.

```
02781. int DemChan (int a[], int n)
02782. {
02783.    int  dem = 0;
02784.    for (int i=0; i<n; i++)
02785.       if (a[i] % 2 == 0)
02786.          dem++;
02787.    return dem;
02788. }
```

Bài 051.Đếm số lượng giá trị dương chia hết cho 7 trong mảng một chiều các số nguyên.

- Ví dụ:

19	343	62	76	49	23	63	98	88	11

- Các giá trị dương chia hết cho 7.

19	343	62	76	49	23	63	98	88	11

- Kết quả: 3.
- Khai báo hàm.

```
02789. int  DemGiaTri(int [],int);
```

- Định nghĩa hàm.

```
02790. int DemGiaTri(int a[], int n)
02791. {
02792.    int dem = 0;
02793.    for (int i=0; i<n; i++)
02794.       if (a[i] > 0 && a[i] % 7 == 0)
02795.          dem++;
02796.    return dem;
02797. }
```

Bài 052.Hãy đếm số lượng giá trị có chữ số tận cùng bằng 5 trong mảng các số nguyên.

- Ví dụ:

19	345	62	76	95	23	12	5	88	11

- Các giá trị có chữ số tận cùng bằng 5.

19	345	62	76	95	23	12	5	88	11

- Kết quả: 3.
- Khai báo hàm.

```
02798.  int  DemGiaTri(int [],int);
```

- Định nghĩa hàm.

```
02799.  int DemGiaTri(int a[], int n)
02800.  {
02801.     int dem = 0;
02802.     for (int i=0; i<n; i++)
02803.        if (abs(a[i]) % 10 == 5)
02804.           dem++;
02805.     return dem;
02806.  }
```

Bài 053.(Hạt nhân) Đếm số lần xuất hiện của giá trị x trong mảng một chiều các số thực.

- Ví dụ: cho mảng sau và $x = 10.6$

1.9	2.9	-0.6	10.6	-9.9	-23	10.6	-0.9	8.8	-1.1

- Các giá trị bằng 10.6

1.9	2.9	-0.6	10.6	-9.9	-23	10.6	-0.9	8.8	-1.1

- Kết quả: 2.
- Khai báo hàm.

```
02807.  int  TanSuat(float [],int, float);
```

- Định nghĩa hàm.

```
02808.  int TanSuat(float a[], int n, float x)
02809.  {
02810.     int dem = 0;
02811.     for (int i=0; i<n; i++)
02812.        if (a[i] == x)
02813.           dem++;
02814.     return dem;
02815.  }
```

Bài 054.Đếm số lượng giá trị đối xứng trong mảng các số nguyên.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	181	-62	76	-9	25	12	1221	16	11

– Các số đối xứng.

0	1	2	3	4	5	6	7	8	9
19	181	-62	76	-9	25	12	1221	16	11

– Kết quả: 4.

– Khai báo hàm.

```
02816. bool ktDoiXung(int);
02817. int  DemDoiXung(int [],int);
```

– Định nghĩa hàm.

```
02818. bool ktDoiXung(int n)
02819. {
02820.    int t = n;
02821.    int dn = 0;
02822.    int dv;
02823.    while (t != 0)
02824.    {
02825.       dv = t % 10;
02826.       dn = dn * 10 + dv;
02827.       t /= 10;
02828.    }
02829.    if (dn == n)
02830.       return true;
02831.    return false;
02832. }
```

– Định nghĩa hàm đếm số lượng giá trị đối xứng.

```
02833. int DemDoiXung(int a[], int n)
02834. {
02835.    int dem = 0;
02836.    for (int i=0; i<n; i++)
02837.       if (ktDoiXung(a[i]))
02838.          dem++;
02839.    return dem;
02840. }
```

Bài 055.Hãy đếm số lượng "số nguyên tố" có trong mảng một chiều các số nguyên.

– Ví dụ:

19	29	62	76	223	23	227	98	88	53

– Các số nguyên tố có trong mảng

19	29	62	76	223	23	227	98	88	53

– Kết quả: 6.
– Khai báo hàm.

```
02841. bool ktNguyenTo(int);
02842. int  DemNguyenTo(int [],int);
```

– Định nghĩa hàm kiểm tra nguyên tố.

```
02843. bool ktNguyenTo(int k)
02844. {
02845.    int dem = 0;
02846.    for(int i=1; i<=k; i++)
02847.       if(k%i==0)
02848.          dem++;
02849.    if(dem==2)
02850.       return true;
02851.    return false;
02852. }
```

– Định nghĩa hàm đếm số lượng số nguyên tố.

```
02853. int DemNguyenTo(int a[], int n)
02854. {
02855.    int dem = 0;
02856.    for (int i=0; i<n; i++)
02857.       if (ktNguyenTo(a[i]))
02858.          dem++;
02859.    return dem;
02860. }
```

Bài 056.Hãy đếm số lượng "số hoàn thiện" có trong mảng một chiều các số nguyên.

– Ví dụ:

0	6	62	76	28	496	6	98	88	0

– Các số hoàn thiện có trong mảng.

0	6	62	76	28	496	6	98	88	0

– Kết quả: 6.
– Khai báo hàm.

```
02861. bool ktHoanThien(int);
02862. int  DemHoanThien(int [],int);
```

– Định nghĩa hàm kiểm tra một số nguyên có phải có số hoàn thiện không?

```
02863. bool ktHoanThien(int k)
02864. {
02865.     int s = 0;
02866.     for(int i=1; i<k; i++)
```

```
02867.          if(k%i==0)
02868.              s = s + i;
02869.      if(s==k)
02870.          return true;
02871.      return false;
02872.  }
```

– Định nghĩa hàm đếm số lượng số hoàn thiện.

```
02873.  int DemHoanThien (int a[], int n)
02874.  {
02875.     int dem = 0;
02876.     for (int i=0; i<n; i++)
02877.        if (ktHoanThien (a[i]))
02878.           dem++;
02879.     return dem;
02880.  }
```

Bài 057.Hãy cho biết sự tương quan giữa số lượng số chẵn và số lượng số lẻ trong mảng các số nguyên.

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Các giá trị chẵn trong mảng.

19	29	62	76	99	23	12	98	88	11

– Kết quả: số lượng số chẵn bằng số lượng số lẻ.

– Khai báo hàm.

```
02881.  int DemChan(int [], int);
02882.  int DemLe(int [], int);
02883.  int  TuongQuan(int [],int);
```

– Hàm TuongQuan trả về 1 trong 3 giá trị −1, 0 và 1.

+ Giá trị −1 có nghĩa số lượng số chẵn nhiều hơn số lẻ.
+ Giá trị 0 có nghĩa số lượng số lẻ bằng số lượng số chẵn.
+ Giá trị +1 có nghĩa số lẻ nhiều hơn số chẵn.

– Định nghĩa hàm đếm số lượng giá trị chẵn.

```
02884.  int DemChan(int a[], int n)
02885.  {
02886.     int dem = 0;
02887.     for (int i=0; i<=n-1; i++)
02888.        if (a[i] % 2 == 0)
02889.           dem++;
02890.     return dem;
02891.  }
```

– Định nghĩa hàm đếm số lượng giá trị lẻ.

```
02892.  int DemLe (int a[], int n)
02893.  {
02894.     int dem = 0;
02895.     for (int i=0; i<=n-1; i++)
02896.        if (a[i] % 2 != 0)
02897.           dem++;
02898.     return dem;
02899.  }
```

– Định nghĩa hàm tương quan chẵn lẻ.

```
02900.  int TuongQuanChanLe (int a[], int n)
02901.  {
02902.     int x = DemChan(a, n);
02903.     int y = DemLe(a, n);
02904.     if (x > y)
02905.        return -1;
02906.     if (x < y)
02907.        return 1;
02908.     return 0;
02909.  }
```

Bài 058.(Hạt nhân) Hãy đếm số lượng các giá trị lớn nhất có trong mảng một chiều các số thực.

– Ví dụ:

1.9	2.9	-0.6	10.6	-9.9	-23	10.6	-0.9	8.8	-1.1

– Giá trị lớn nhất là 10.6

– Đếm lớn nhất.

1.9	2.9	-0.6	10.6	-9.9	-23	10.6	-0.9	8.8	-1.1

– Kết quả: 2.

– Khai báo hàm.

```
02910.  float LonNhat (float [], int);
02911.  int  DemLonNhat (float [],int);
```

– Định nghĩa hàm tìm giá trị lớn nhất.

```
02912.  float LonNhat (float a[], int n)
02913.  {
02914.     float lc = a[0];
02915.     for (int i=0; i<=n-1; i++)
02916.        if (a[i] > lc)
02917.           lc = a[i];
02918.     return lc;
02919.  }
```

– Định nghĩa hàm đếm số lượng giá trị lớn nhất.

```
02920.  int DemLonNhat (float a[], int n)
02921.  {
02922.     float ln = LonNhat(a, n);
02923.     int dem = 0;
02924.     for (int i=0; i<n; i++)
02925.        if (a[i] == ln)
02926.           dem++;
02927.     return dem;
02928.  }
```

Bài 059.(Hạt nhân) Hãy xác định số lượng các phần tử kề nhau mà cả hai đều chẵn.

- Ví dụ:

90	29	62	76	99	34	23	12	98	88	11

- Các giá trị chẵn trong mảng.

90	29	62	76	99	34	23	12	98	88	11

- Kết quả: 5.
- Khai báo hàm.

```
02929.  int  DemGiaTri (int[],int);
```

- Định nghĩa hàm.

```
02930.  int DemGiaTri (int a[], int n)
02931.  {
02932.     if (n <= 1)
02933.        return 0;
02934.     int dem = 0;
02935.     if (a[0]%2==0 && a[1]%2==0)
02936.        dem++;
02937.     for (int i = 1; i <= n - 2; i++)
02938.        if(a[i]%2==0)
02939.           if(a[i - 1]%2==0 || a[i + 1]%2==0)
02940.              dem++;
02941.     if (a[n - 1]%2==0 && a[n - 2]%2==0)
02942.        dem++;
02943.     return dem;
02944.  }
```

Bài 060.Hãy xác định số lượng các phần tử kề nhau mà cả hai số trái dấu nhau trong mảng một chiều các số thực.

- Ví dụ:

1.9	2.9	-0.6	10.6	-9.9	-23	-1.6	-0.9	8.8	-1.1

- Các giá trị kề nhau và trái dấu nhau.

1.9	2.9	-0.6	10.6	-9.9	-23	-1.6	-0.9	8.8	-1.1

– Kết quả: 7.
– Khai báo hàm.

```
02945. int  DemGiaTri(float [],int);
```

– Định nghĩa hàm.

```
02946. int DemGiaTri(float a[], int n)
02947. {
02948.    if (n <= 1)
02949.       return 0;
02950.    int dem = 0;
02951.    if (a[0] * a[1] < 0)
02952.       dem++;
02953.    for (int i = 1; i <= n - 2; i++)
02954.       if((a[i]*a[i-1]<0) || (a[i]*a[i+1]<0))
02955.          dem++;
02956.    if (a[n - 1] * a[n - 2] < 0)
02957.       dem++;
02958.    return dem;
02959. }
```

Bài 061.Hãy xác định số lượng các phần tử kề nhau mà số đứng sau cùng dấu số đứng trước và có giá trị tuyệt đối lớn hơn.

– Ví dụ:

1.9	2.9	-0.6	10.6	-9.9	-23	-1.6	-0.9	8.8	-1.1

– Các phần tử trong mảng có số đứng trước.

1.9	2.9	-0.6	10.6	-9.9	-23	-1.6	-0.9	8.8	-1.1

– Các phần tử kề nhau mà số đứng sau cùng dấu số đứng trước và có giá trị tuyệt đối lớn hơn.

1.9	2.9	-0.6	10.6	-9.9	-23	-1.6	-0.9	8.8	-1.1

– Kết quả: 2.
– Khai báo hàm.

```
02960. int  DemGiaTri(float [],int);
```

– Định nghĩa hàm.

```
02961. int DemGiaTri(float a[], int n)
02962. {
02963.    if (n <= 1)
02964.       return 0;
02965.    int dem = 0;
02966.    for (int i = 1; i <= n - 1; i++)
02967.       if ((a[i] * a[i-1] > 0) &&
02968.          abs(a[i]) > abs(a[i - 1])))
```

```
02969.          dem++;
02970.     return dem;
02971. }
```

Bài 062.Hãy đếm số lượng phần tử cực trị trong mảng một chiều các số thực.

- Lân cận của các phần tử trong mảng.

  + Phần tử đầu tiên có một lân cận bên phải.
  + Phần tử cuối cùng có một lân cận bên trái.
  + Các phần tử có chỉ số từ $1 \rightarrow (n-2)$ có hai lân cận.
- Ví dụ:

19	29	62	76	99	23	12	18	-88	111

- Các giá trị cực đại trong mảng.

19	29	62	76	99	23	12	18	-88	111

- Các giá trị cực tiểu trong mảng.

19	29	62	76	99	23	12	18	-88	111

- Các giá trị cực trị trong mảng.

19	29	62	76	99	23	12	18	-88	111

- Kết quả: 6.
- Khai báo hàm.

```
02972. int DemCucDai(float [], int);
02973. int DemCucTieu(float [], int);
02974. int  DemCucTri(float [],int);
```

- Định nghĩa hàm đếm số lượng giá trị cực đại.

```
02975. int DemCucDai(float a[], int n)
02976. {
02977.    if (n <= 1)
02978.       return 0;
02979.    int dem = 0;
02980.    if (a[0] > a[1])
02981.       dem++;
02982.    for (int i = 1; i <= n - 2; i++)
02983.       if (a[i] > a[i - 1] && a[i] > a[i + 1])
02984.          dem++;
02985.    if (a[n - 1] > a[n - 2])
02986.       dem++;
02987.    return dem;
02988. }
```

- Định nghĩa hàm đếm số lượng giá trị cực tiểu.

```
02989.  int DemCucTieu(float a[], int n)
02990.  {
02991.     if (n <= 1)
02992.        return 0;
02993.     int dem = 0;
02994.     if (a[0] < a[1])
02995.        dem++;
02996.     for (int i = 1; i <= n - 2; i++)
02997.        if (a[i] < a[i - 1] && a[i] < a[i + 1])
02998.           dem++;
02999.     if (a[n - 1] < a[n - 2])
03000.        dem++;
03001.     return dem;
03002.  }
```

– Định nghĩa hàm đếm số lượng giá trị cực trị.

```
03003.  int DemCucTri(float a[], int n)
03004.  {
03005.     int dem1 = DemCucDai(a, n);
03006.     int dem2 = DemCucTieu(a, n);
03007.     return (dem1 + dem2);
03008.  }
```

Bài 063.(Hạt nhân) Hãy liệt kê các giá trị xuất hiện trong mảng một chiều các số nguyên đúng một lần.

– Ví dụ:

19	29	62	19	99	29	12	19	88	11

– Số lượng giá trị phân biệt.

19	29	62	19	99	29	12	19	88	11

– Kết quả: 62, 99, 12, 88, 11.

– Khai báo hàm.

```
03009.  int TanSuat(int [], int, int);
03010.  void  LietKe(int [],int);
```

– Định nghĩa hàm đếm tần suất xuất hiện của giá trị x.

```
03011.  int TanSuat(int a[], int n, int x)
03012.  {
03013.     int dem = 0;
03014.     for (int i=0; i<=n-1; i++)
03015.        if (a[i] == x)
03016.           dem++;
03017.     return dem;
03018.  }
```

– Định nghĩa hàm liệt kê.

```
03019.  void LietKe(int a[], int n)
03020.  {
03021.     for (int i=0; i<n; i++)
03022.     {
03023.        int kq = TanSuat(a, n, a[i]);
03024.        if (kq == 1)
03025.           cout << setw(8) << a[i];
03026.     }
03027.  }
```

Bài 064.(Hạt nhân) Hãy đếm số lượng các giá trị phân biệt có trong mảng một chiều các số nguyên.

– Ví dụ:

19	29	62	19	99	29	12	19	88	11

– Số lượng giá trị phân biệt.

19	29	62	19	99	29	12	19	88	11

– Kết quả: 7.

– Khai báo hàm.

```
03028.  int  DemPhanBiet(int [],int);
```

– Định nghĩa hàm.

```
03029.  int DemPhanBiet(int a[], int n)
03030.  {
03031.     int dem = 0;
03032.     for (int i=0; i<n; i++)
03033.     {
03034.        bool flag = true;
03035.        for(int j = 0; j < i; j++)
03036.           if(a[j]==a[i])
03037.              flag  = false;
03038.        if(flag==true)
03039.           dem++;
03040.     }
03041.     return dem;
03042.  }
```

Bài 065.(*) Hãy đếm số lượng giá trị phân biệt có trong mảng một chiều các số thực.

– Ví dụ:

19	29	62	19	99	29	12	19	88	11

– Số lượng giá trị phân biệt.

19	29	62	19	99	29	12	19	88	11

– Kết quả: 7.

– Khai báo hàm.

```
03043. int  DemPhanBiet(float [],int);
```

– Định nghĩa hàm.

```
03044. int DemPhanBiet(float a[], int n)
03045. {
03046.    int dem = 0;
03047.    for (int i=0; i<n; i++)
03048.    {
03049.       bool flag = true;
03050.       for(int j = 0; j < i; j++)
03051.          if(a[j]==a[i])
03052.             flag  = false;
03053.       if(flag==true)
03054.          dem++;
03055.    }
03056.    return dem;
03057. }
```

Bài 066.(*) Hãy đếm số lượng số nguyên tố phân biệt trong mảng các số nguyên.

– Ví dụ:

19	29	62	76	223	23	227	98	88	19

– Các số nguyên tố có trong mảng

19	29	62	76	223	23	227	98	88	19

– Kết quả: 5.

– Khai báo hàm.

```
03058. bool ktNguyenTo(int);
03059. int  DemGiaTri(int [],int);
```

– Định nghĩa hàm kiểm tra nguyên tố.

```
03060. bool ktNguyenTo(int k)
03061. {
03062.    int dem = 0;
03063.    for(int i=1; i<=k; i++)
03064.       if(k%i==0)
03065.          dem++;
03066.    if(dem==2)
03067.       return true;
03068.    return false;
03069. }
```

– Định nghĩa hàm đếm số lượng số nguyên tố phân biệt

```
03070.  int DemGiaTri (int a[], int n)
03071.  {
03072.     int dem = 0;
03073.     for (int i=0; i<n; i++)
03074.     {
03075.        int flag = 1;
03076.        for (int j = 0; j <= i - 1; j++)
03077.           if (a[j] == a[i])
03078.              flag = 0;
03079.        if(ktNguyenTo(a[i]) && flag == 1)
03080.           dem++;
03081.     }
03082.     return dem;
03083.  }
```

Bài 067. (*) Cho hai mảng số nguyên a, b. Đếm số lượng giá trị chỉ xuất hiện một trong hai mảng số nguyên (234).

– Ví dụ:

+ Mảng a.

19	29	19	62	62	99

+ Mảng b.

14	29	45	76	45	14	14	29

+ Các giá trị chỉ xuất hiện trong mảng a.

19	29	19	62	62	99

+ Các giá trị chỉ xuất hiện trong mảng b.

14	29	45	76	45	14	14	29

– Kết quả: 6.

– Khai báo hàm.

```
03084.  int TanSuat(int [], int, int);
03085.  int  DemGiaTri(int [],int,int [],int);
```

– Định nghĩa hàm.

```
03086.  int TanSuat(int a[], int n, int x)
03087.  {
03088.     int dem = 0;
03089.     for (int i=0; i<=n-1; i++)
03090.        if (a[i] == x)
03091.           dem++;
03092.     return dem;
03093.  }
```

– Định nghĩa hàm đếm số lượng giá trị chỉ xuất hiện một trong hai mảng.

```
03094.  int DemGiaTri(int a[],int n,
03095.                int b[],int m)
03096.  {
03097.     int dem = 0;
03098.     for (int i=0; i<n; i++)
03099.     {
03100.        int flag = 1;
03101.        for (int j = 0; j <= i - 1; j++)
03102.           if (a[j] == a[i])
03103.              flag = 0;
03104.        if (TanSuat(b, m, a[i])==0 && flag == 1)
03105.           dem++;
03106.     }
03107.     for (i = 0; i < m; i++)
03108.     {
03109.        int flag = 1;
03110.        for (int j = 0; j <= i - 1; j++)
03111.           if (b[j] == b[i])
03112.              flag = 0;
03113.        if (TanSuat(a, n, b[i])==0 && flag == 1)
03114.           dem++;
03115.     }
03116.     return dem;
03117.  }
```

Bài 068. (*) Hãy liệt kê tần suất xuất hiện của các giá trị trong mảng các số nguyên. Lưu ý: mỗi giá trị liệt kê một lần (230, 233).

– Ví dụ:

19	29	19	76	99	29	19	98	88	11

– Các giá trị có trong mảng.

19	29	19	76	99	29	19	98	88	11

– Kết quả.

19	29	76	99	98	88	11
3	2	1	1	1	1	1

– Khai báo hàm.

```
03118.  int TanSuat(int [], int, int);
03119.  void  LietKe(int [],int);
```

– Định nghĩa hàm.

```
03120.  int TanSuat(int a[], int n, int x)
03121.  {
```

```
03122.     int dem = 0;
03123.     for (int i=0; i<=n-1; i++)
03124.        if (a[i] == x)
03125.           dem++;
03126.     return dem;
03127.  }
```

– Định nghĩa hàm liệt kê tần suất xuất hiện của các giá trị trong mảng.

```
03128.  void LietKe(int a[], int n)
03129.  {
03130.     for (int i=0; i<n; i++)
03131.     {
03132.        int flag = 1;
03133.        for (int j = 0; j <= i - 1; j++)
03134.           if (a[j] == a[i])
03135.              flag = 0;
03136.        if (flag == 1)
03137.        {
03138.           int dem = TanSuat(a, n, a[i]);
03139.           cout << "\nGia tri :" << a[i];
03140.           cout << "(" << dem << ")";
03141.        }
03142.     }
03143.  }
```

Bài 069. (*) Hãy liệt kê các giá trị xuất hiện quá một lần trong mảng các số nguyên. Lưu ý: mỗi giá trị liệt kê một lần.

– Ví dụ:

19	29	19	76	99	29	19	98	88	11

– Các giá trị có trong mảng.

19	29	19	76	99	29	19	98	88	11

– Kết quả: 19, 29.

– Khai báo hàm.

```
03144.  int TanSuat(int [], int, int);
03145.  void  LietKe(int [],int);
```

– Định nghĩa hàm.

```
03146.  int TanSuat(int a[], int n, int x)
03147.  {
03148.     int dem = 0;
03149.     for (int i=0; i<=n-1; i++)
03150.        if (a[i] == x)
```

```
03151.          dem++;
03152.     return dem;
03153. }
```

– Định nghĩa hàm liệt kê tần suất của các giá trị trong mảng.

```
03154. void LietKe(int a[], int n)
03155. {
03156.    for (int i=0; i<n; i++)
03157.    {
03158.       int flag = 1;
03159.       for (int j = 0; j <= i - 1; j++)
03160.          if (a[j] == a[i])
03161.             flag = 0;
03162.       int kq = TanSuat(a, n, a[i]);
03163.       if (flag == 1 && kq > 1)
03164.          cout << setw(8) << a[i];
03165.    }
03166. }
```

Bài 070. (*) Cho hai mảng các số nguyên *a* và *b*. Liệt kê các giá trị chỉ xuất hiện một trong hai mảng.

– Ví dụ:

+ Mảng *a*:

19	29	19	62	62	99

+ Mảng *b*.

14	29	45	76	45	14	14	29

+ Các giá trị chỉ xuất hiện trong mảng *a*.

19	29	19	62	62	99

+ Các giá trị chỉ xuất hiện trong mảng *b*.

14	29	45	76	45	14	14	29

– Kết quả: 19, 62, 99, 14, 45, 76.

– Khai báo hàm.

```
03167. int TanSuat(int [], int, int);
03168. void LietKe(int [],int,int [],int);
```

– Định nghĩa hàm.

```
03169. int TanSuat(int a[], int n, int x)
03170. {
03171.    int dem = 0;
03172.    for (int i=0; i<=n-1; i++)
03173.       if (a[i] == x)
03174.          dem++;
03175.    return dem;
```

```
03176. }
```

– Định nghĩa hàm liệt kê.

```
03177. void LietKe(int a[],int n,int b[],int m)
03178. {
03179.
03180.    for (int i=0; i<n; i++)
03181.    {
03182.       int flag = 1;
03183.       for (int j = 0; j <= i - 1; j++)
03184.          if (a[j] == a[i])
03185.             flag = 0;
03186.       if (TanSuat(b, m, a[i]) == 0 && flag)
03187.          cout << a[i];
03188.    }
03189.    for (int i = 0; i < m; i++)
03190.    {
03191.       int flag = 1;
03192.       for (int j = 0; j <= i - 1; j++)
03193.          if (b[j] == b[i])
03194.             flag = 0;
03195.       if (TanSuat(a, n, b[i]) == 0 && flag)
03196.          cout << b[i];
03197.    }
03198. }
```

Bài 071. (*) Cho hai mảng các số nguyên *a* và *b*. Hãy cho biết số lần xuất hiện của mảng *a* trong mảng *b*.

– Ví dụ:

+ Mảng *b*:

0	1	2	3	4	5	6	7	8	9
19	29	89	88	29	89	88	29	34	23

+ Mảng *a*:

29	89	88	29

– Các mảng con có độ dài là 4.

0	1	2	3	4	5	6	7	8	9
19	29	89	88	29	89	88	29	34	23

19	29	89	88	29	89	88	29	34	23

19	29	89	88	29	89	88	29	34	23

19	29	89	88	29	89	88	29	34	23

19	29	89	88	29	89	88	29	34	23

19	29	89	88	29	89	88	29	34	23

19	29	89	88	29	89	88	29	34	23

– Kết quả: 2.

– Khai báo hàm.

```
03199.  int  DemXuatHien(int [],int,int [],int);
```

– Định nghĩa hàm.

```
03200.  int DemXuatHien(int a[],int n,int b[],int m)
03201.  {
03202.     if (n > m)
03203.        return 0;
03204.     int dem = 0;
03205.     for (int vt = 0; vt <= m - n; vt++)
03206.     {
03207.        int flag = 1;
03208.        for (int i=0; i<n; i++)
03209.           if (a[i] != b[vt + i])
03210.              flag = 0;
03211.        if (flag == 1)
03212.           dem++;
03213.     }
03214.     return dem;
03215.  }
```

01.18.05 Kỹ thuật Tìm kiếm – Kỹ thuật Đặt lính canh

Bài 072.(Hạt nhân) Viết hàm tìm "giá trị lớn nhất" trong mảng số thực.

– Ví dụ:

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

– Kết quả: 76.

– Khai báo hàm.

```
03216.  float LonNhat(float [],int);
```

– Định nghĩa hàm tìm giá trị lớn nhất.

```
03217.  float LonNhat(float a[], int n)
03218.  {
03219.     float lc = a[0];
03220.     for (int i=0; i<n; i++)
03221.        if (a[i] > lc)
```

```
03222.          lc = a[i];
03223.    return lc;
03224. }
```

Bài 073.Tìm "giá trị nhỏ nhất" trong mảng một chiều số thực.

– Ví dụ:

1.9	2.9	-0.6	-34	-9.9	-23	12.1	-0.9	8.8	-1.1

– Kết quả: -34.

```
03225. float NhoNhat(float [],int);
```

– Định nghĩa hàm tìm giá trị nhỏ nhất.

```
03226. float NhoNhat(float a[], int n)
03227. {
03228.    float lc = a[0];
03229.    for (int i=0; i<n; i++)
03230.       if (a[i] < lc)
03231.          lc = a[i];
03232.    return lc;
03233. }
```

Bài 074.(Hạt nhân) Tìm "một vị trí mà giá trị tại vị trí đó là giá trị nhỏ nhất" trong mảng một chiều các số thực.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
1.9	2.9	-0.6	-34	-9.9	-23	12.1	-0.9	8.8	-1.1

– Kết quả: 3.

– Khai báo hàm.

```
03234. int TimViTri(float [],int);
```

– Định nghĩa hàm tìm vị trí giá trị nhỏ nhất.

```
03235. int TimViTri (float a[], int n)
03236. {
03237.    int lc = 0;
03238.    for (int i=0; i<n; i++)
03239.       if (a[i] < a[lc])
03240.          lc = i;
03241.    return lc;
03242. }
```

Bài 075.(Hạt nhân) Hãy tìm giá trị trong mảng số thực "xa giá trị x nhất".

– Ví dụ: cho giá trị $x = 80$.

19	29	62	76	99	23	12	98	88	11

- Khoảng cách từ $x = 80$ tới các giá trị trong mảng.
  + Phần tử 19. Khoảng cách tới $x = 80$ là $|80 - 19| = 61$.
  + Phần tử 29. Khoảng cách tới $x = 80$ là $|80 - 29| = 51$.
  + Phần tử 62. Khoảng cách tới $x = 80$ là $|80 - 62| = 18$.
  + Phần tử 76. Khoảng cách tới $x = 80$ là $|80 - 76| = 4$.
  + Phần tử 99. Khoảng cách tới $x = 80$ là $|80 - 99| = 19$.
  + Phần tử 23. Khoảng cách tới $x = 80$ là $|80 - 23| = 57$.
  + Phần tử 12. Khoảng cách tới $x = 80$ là $|80 - 12| = 68$.
  + Phần tử 98. Khoảng cách tới $x = 80$ là $|80 - 98| = 18$.
  + Phần tử 88. Khoảng cách tới $x = 80$ là $|80 - 88| = 8$.
  + Phần tử 11. Khoảng cách tới $x = 80$ là $|80 - 11| = 69$.
- Kết quả: 11.
- Khai báo hàm.

```
03243.  float XaNhat(float [],int, float);
```

- Định nghĩa hàm tìm giá trị trong mảng số thực "xa giá trị x nhất".

```
03244.  float XaNhat(float a[], int n, float x)
03245.  {
03246.     float lc = a[0];
03247.     for (int i=0; i<n; i++)
03248.        if (abs(a[i] - x) > abs(lc - x))
03249.           lc = a[i];
03250.     return lc;
03251.  }
```

Bài 076.Hãy tìm một vị trí trong mảng một chiều các số thực mà giá trị tại vị trí đó là giá trị "gần giá trị x nhất".

- Ví dụ: cho giá trị $x = 15.8$

0	1	2	3	4	5	6	7	8	9
1.9	2.9	-0.6	34	-9.9	-23	12.1	-0.9	8.8	-1.1

- Khoảng cách từ $x = 15.8$ tới các giá trị trong mảng.
  + Phần tử 1.9. Khoảng cách tới 15.8 là $|15.8 - 1.9| = 13.9$.
  + Phần tử 2.9. Khoảng cách tới 15.8 là $|15.8 - 2.9| = 12.9$.
  + Phần tử -0.6. Khoảng cách tới 15.8 là $|15.8 + 0.6| = 16.4$.
  + Phần tử 34. Khoảng cách tới 15.8 là $|15.8 - 34| = 18.2$.
  + Phần tử -9.9. Khoảng cách tới 15.8 là $|15.8 + 9.9| = 25.7$.
  + Phần tử -23. Khoảng cách tới 15.8 là $|15.8 + 23| = 38.8$.
  + Phần tử 12.1. Khoảng cách tới 15.8 là $|15.8 - 12.1| = 3.7$.

+ Phần tử -0.9. Khoảng cách tới 15.8 là $|15.8 + 0.9| = 16.7$.
+ Phần tử 8.8. Khoảng cách tới 15.8 là $|15.8 - 8.8| = 7$.
+ Phần tử -1.1. Khoảng cách tới 15.8 là $|15.8 + 1.1| = 16.9$.

– Kết quả: 6.
– Khai báo hàm.

```
03252.  int TimViTri(float [],int, float);
```

– Định nghĩa hàm tìm một vị trí trong mảng một chiều các số thực mà giá trị tại vị trí đó là giá trị "gần giá trị x nhất".

```
03253.  int TimViTri (float a[], int n, float x)
03254.  {
03255.     int lc = 0;
03256.     for (int i=0; i<n; i++)
03257.        if (abs(a[i] - x) < abs(a[lc] - x))
03258.           lc = i;
03259.     return lc;
03260.  }
```

Bài 077.(Hạt nhân) Cho mảng một chiều các số thực hãy tìm đoạn $[x, y]$ sao cho đoạn này chứa tất cả các giá trị trong mảng.

– Ví dụ:

1.9	2.9	-0.6	34	-9.9	-23	12.1	-0.9	8.8	-1.1

– Giá trị lớn nhất trong mảng.

1.9	2.9	-0.6	34	-9.9	-23	12.1	-0.9	8.8	-1.1

– Giá trị nhỏ nhất trong mảng.

1.9	2.9	-0.6	34	-9.9	-23	12.1	-0.9	8.8	-1.1

– Kết quả: $[-23,34]$.
– Khai báo hàm.

```
03261.  float LonNhat(float [], int);
03262.  float NhoNhat(float [], int);
03263.  void TimDoan(float [],int, float &, float &);
```

– Định nghĩa hàm tìm giá trị lớn nhất.

```
03264.  float LonNhat(float a[], int n)
03265.  {
03266.     float lc = a[0];
03267.     for (int i=0; i<n; i++)
03268.        if (a[i] > lc)
03269.           lc = a[i];
03270.     return lc;
03271.  }
```

– Định nghĩa hàm tìm giá trị nhỏ nhất.

```
03272. float NhoNhat(float a[], int n)
03273. {
03274.    float lc = a[0];
03275.    for (int i=0; i<n; i++)
03276.       if (a[i] < lc)
03277.          lc = a[i];
03278.    return lc;
03279. }
```

– Định nghĩa hàm tìm đoạn.

```
03280. void TimDoan(  float a[],float n,
03281.                float &x, float &y)
03282. {
03283.    x = NhoNhat(a, n);
03284.    y = LonNhat(a, n);
03285. }
```

Bài 078.Cho mảng một chiều các số thực hãy tìm giá trị x sao cho đoạn $[-x,x]$ chứa tất cả các giá trị trong mảng.

– Ví dụ 01:

+ Cho mảng.

1.9	2.9	-0.6	34	-9.9	-23	12.1	-0.9	8.8	-1.1

+ Giá trị lớn nhất, nhỏ nhất trong mảng.

1.9	2.9	-0.6	34	-9.9	-23	12.1	-0.9	8.8	-1.1

+ Kết quả: $[-34,34]$.

– Ví dụ 02:

+ Cho mảng.

1.9	2.9	-0.6	34	-9.9	-23	12.1	-0.9	8.8	-1.1

+ Giá trị lớn nhất, nhỏ nhất trong mảng.

1.9	2.9	-0.6	34	-9.9	-53	12.1	-0.9	8.8	-1.1

+ Kết quả: $[-53,53]$.

– Khai báo hàm.

```
03286. float TimX(float [],int);
```

– Định nghĩa hàm tìm giá trị x sao cho đoạn $[-x,x]$ chứa tất cả các giá trị trong mảng.

```
03287. float TimX(float a[], int n)
03288. {
03289.    float lc = abs(a[0]);
03290.    for (int i=0; i<n; i++)
03291.       if (abs(a[i]) > lc)
03292.          lc = abs(a[i]);
```

```
03293.     return lc;
03294.  }
```

Bài 079.(Hạt nhân) Tìm "giá trị dương đầu tiên" trong mảng một chiều các số thực. Nếu mảng không có giá trị dương thì hàm trả về giá trị không dương là 0.

- Ví dụ:

1.9	2.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

- Các số dương trong mảng

1.9	2.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

- Kết quả: 1.9
- Khai báo hàm.

```
03295.  float DuongDau(float [],int);
```

- Định nghĩa hàm tìm "giá trị dương đầu tiên" trong mảng một chiều các số thực.

```
03296.  float DuongDau(float a[], int n)
03297.  {
03298.     for (int i=0; i<n; i++)
03299.        if (a[i] > 0)
03300.           return a[i];
03301.     return -1;
03302.  }
```

Bài 080.Tìm giá trị âm đầu tiên trong mảng một chiều các số thực. Nếu mảng không có giá trị âm thì thì trả về giá trị không âm là giá trị 0.

- Ví dụ:

19	29	-62	76	-99	-23	12	-98	88	-11

- Các số âm trong mảng.

19	29	-62	76	-99	-23	12	-98	88	-11

- Kết quả: -62.
- Khai báo hàm.

```
03303.  float AmDau(float [],int);
```

- Định nghĩa hàm tìm giá trị âm đầu tiên trong mảng một chiều các số thực.

```
03304.  float AmDau(float a[], int n)
03305.  {
03306.     for (int i=0; i<n; i++)
03307.        if (a[i] < 0)
03308.           return a[i];
03309.     return 0;
03310.  }
```

Bài 081.Cho mảng một chiều các số thực hãy tìm giá trị đầu tiên lớn hơn giá trị 2003. Nếu mảng không có giá trị thỏa điều kiện trên thì hàm trả về giá trị là 0.

- Ví dụ:

1.9	2.9	-0.6	7.6	2005	-23	12.1	5712	8.8	-1.1

- Các giá trị lớn hơn giá trị 2003.

1.9	2.9	-0.6	7.6	2005	-23	12.1	5712	8.8	-1.1

- Kết quả: 2005.
- Khai báo hàm.

```
03311. float DauTien(float [],int);
```

- Định nghĩa hàm.

```
03312. float DauTien(float a[], int n)
03313. {
03314.    for (int i=0; i<n; i++)
03315.       if (a[i] > 2003)
03316.       {
03317.          float lc = a[i];
03318.          return lc;
03319.       }
03320.    return 0;
03321. }
```

Bài 082.Viết hàm tìm "số chẵn đầu tiên" trong mảng các số nguyên. Nếu mảng không có giá trị chẵn thì hàm sẽ trả về giá trị không chẵn là -1.

- Ví dụ:

90	29	62	76	99	34	23	12	98	88	11

- Các giá trị chẵn trong mảng.

90	29	62	76	99	34	23	12	98	88	11

- Kết quả: 90.
- Khai báo hàm.

```
03322. int DauTien(int [],int);
```

- Định nghĩa hàm tìm "số chẵn đầu tiên" trong mảng các số nguyên.

```
03323. int ChanDau(int a[], int n)
03324. {
03325.    for (int i=0; i<n; i++)
03326.       if (a[i] % 2 == 0)
03327.          return a[i];
03328.    return -1;
```

```
03329.  }
```

Bài 083.Cho mảng một chiều các số nguyên hãy tìm giá trị đầu tiên trong mảng nằm trong khoảng (x, y) cho trước. Nếu mảng không có giá trị thỏa điều kiện trên thì hàm trả về giá trị là x.

- Ví dụ: cho mảng và $(x, y) = (10,20)$

19	29	62	76	99	23	12	98	88	11

- Các giá trị thuộc khoảng $(10,20)$.

19	29	62	76	99	23	12	98	88	11

- Kết quả: 19.
- Khai báo hàm.

```
03330.  int DauTien(int [],int,int,int);
```

- Định nghĩa hàm.

```
03331.  int DauTien(int a[], int n, int x, int y)
03332.  {
03333.     for (int i=0; i<n; i++)
03334.        if (a[i] > x && a[i] < y)
03335.           return a[i];
03336.     return x;
03337.  }
```

Bài 084.Cho mảng một chiều các số thực. Hãy viết hàm tìm một vị trí trong mảng thỏa hai điều kiện: có hai giá trị lân cận và giá trị tại vị trí đó bằng tích hai giá trị lân cận. Nếu mảng không tồn tại giá trị như vậy thì hàm trả về giá trị -1.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	8	72	9	18	2	0	0	88	11

- Các phần tử có hai giá trị lân cận.

19	8	72	9	18	2	0	0	88	11

- Các giá trị bằng tích hai giá trị lân cận.

19	8	72	9	18	2	0	0	88	11

- Kết quả: 2.
- Khai báo hàm.

```
03338.  int TimViTri(float [],int);
```

- Định nghĩa hàm.

```
03339.  int TimViTri(float a[], int n)
03340.  {
03341.     for (int i = 1; i <= n - 2; i++)
03342.        if (a[i] == (a[i - 1] * a[i + 1]))
03343.           return i;
```

```
03344.     return -1;
03345.  }
```

Bài 085.(Hạt nhân) Tìm "số chẵn cuối cùng" trong mảng một chiều các số nguyên. Nếu mảng không có giá trị chẵn thì hàm trả về giá trị không chẵn là −1.

- Ví dụ:

90	29	62	76	99	34	23	12	98	88	11

- Các giá trị chẵn trong mảng.

90	29	62	76	99	34	23	12	98	88	11

- Kết quả: 88.
- Khai báo hàm.

```
03346.  int ChanCuoi(int [],int);
```

- Định nghĩa hàm.

```
03347.  int ChanCuoi(int a[], int n)
03348.  {
03349.     for (int i = n - 1; i >= 0; i--)
03350.        if (a[i] % 2 == 0)
03351.           return a[i];
03352.     return -1;
03353.  }
```

Bài 086.Tìm "số dương cuối cùng" trong mảng số thực. Nếu mảng không có giá trị dương thì trả về giá trị không dương là 0.

- Ví dụ:

1.9	2.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

- Các số dương trong mảng

1.9	2.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

- Kết quả: 8.8
- Khai báo hàm.

```
03354.  float DuongCuoi(float [],int);
```

- Định nghĩa hàm.

```
03355.  float DuongCuoi(float a[], int n)
03356.  {
03357.     for (int i = n - 1; i >= 0; i--)
03358.        if (a[i] > 0)
03359.           return a[i];
03360.     return 0;
03361.  }
```

Bài 087.Cho mảng một chiều các số thực hãy viết hàm tìm giá trị âm cuối cùng lớn hơn giá trị −1. Nếu mảng không có giá trị thỏa điều kiện trên thì hàm trả về giá trị không âm là 0.

- Ví dụ:

19	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

- Các số âm trong mảng.

19	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

- Các số âm trong mảng lớn hơn -1.

19	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

- Kết quả: -0.98.
- Khai báo hàm.

```
03362.  float CuoiCung(float [],int);
```

- Định nghĩa hàm.

```
03363.  float CuoiCung(float a[], int n)
03364.  {
03365.     for (int i = n - 1; i >= 0; i--)
03366.        if (a[i] < 0 && a[i]>-1)
03367.           return a[i];
03368.     return 0;
03369.  }
```

Bài 088.Tìm số chính phương đầu tiên trong mảng một chiều các số nguyên. Nếu mảng không có số chính phương thì hàm trả về giá trị là −1.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	81	-62	76	-99	25	12	-98	16	11

- Các số chính phương.

0	1	2	3	4	5	6	7	8	9
19	81	-62	76	-99	25	12	-98	16	11

- Kết quả: 81.
- Khai báo hàm.

```
03370.  bool ktChinhPhuong(int);
03371.  int ChinhPhuongDau(int [],int);
```

- Định nghĩa hàm.

```
03372.  bool ktChinhPhuong(int n)
03373.  {
03374.     for (int i = 1; (i * i) <= n; i++)
03375.        if (i * i == n)
03376.           return true;
03377.     return false;
```

```
03378. }
```

– Định nghĩa hàm tìm giá trị chính phương đầu.

```
03379. int ChinhPhuongDau(int a[], int n)
03380. {
03381.    for (int i=0; i<n; i++)
03382.       if (ktChinhPhuong(a[i]))
03383.          return a[i];
03384.    return -1;
03385. }
```

Bài 089. Tìm "số nguyên tố đầu tiên" trong mảng một chiều các số nguyên. Nếu mảng không có số nguyên tố thì trả về giá trị là 0.

– Ví dụ:

19	29	62	76	223	23	227	98	88	19

– Các số nguyên tố có trong mảng

19	29	62	76	223	23	227	98	88	19

– Kết quả: 19.

– Khai báo hàm.

```
03386. bool ktNguyenTo(int);
03387. int NguyenToDau(int [],int);
```

– Định nghĩa hàm kiểm tra nguyên tố.

```
03388. bool ktNguyenTo(int k)
03389. {
03390.    int dem = 0;
03391.    for(int i=1; i<=k; i++)
03392.       if(k%i==0)
03393.          dem++;
03394.    if(dem==2)
03395.       return true;
03396.    return false;
03397. }
```

– Định nghĩa hàm tìm số nguyên tố đầu.

```
03398. int NguyenToDau(int a[], int n)
03399. {
03400.    for (int i=0; i<n; i++)
03401.       if (ktNguyenTo(a[i]))
03402.          return a[i];
03403.    return 0;
03404. }
```

Bài 090.Tìm "số hoàn thiện đầu tiên" trong mảng một chiều các số nguyên. Nếu mảng không có số hoàn thiện thì trả về giá trị là −1.

- Ví dụ:

97	6	62	76	28	496	6	98	88	18

- Các số hoàn thiện có trong mảng.

97	6	62	76	28	496	6	98	88	18

- Kết quả: 6.
- Khai báo hàm.

```
03405.  bool ktHoanThien(int);
03406.  int HoanThienDau(int [],int);
```

- Định nghĩa hàm.

```
03407.  bool ktHoanThien(int k)
03408.  {
03409.      int s = 0;
03410.      for(int i=1; i<k; i++)
03411.          if(k%i==0)
03412.              s = s + i;
03413.      if(s==k)
03414.          return true;
03415.      return false;
03416.  }
```

- Định nghĩa hàm tìm số hoàn thiện đầu.

```
03417.  int HoanThienDau(int a[], int n)
03418.  {
03419.     for (int i=0; i<n; i++)
03420.        if (ktHoanThien(a[i]))
03421.           return a[i];
03422.     return -1;
03423.  }
```

Bài 091.Cho mảng một chiều các số nguyên hãy viết hàm tìm giá trị đối xứng đầu tiên trong mảng. Nếu mảng không có số đối xứng hàm trả về giá trị không đối xứng là 10.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	181	-62	76	-9	25	12	1221	16	11

- Các số đối xứng.

0	1	2	3	4	5	6	7	8	9
19	181	-62	76	-9	25	12	1221	16	11

- Kết quả: 181.
- Khai báo hàm.

```
03424.  bool ktDoiXung(int);
03425.  int DoiXungDau(int [],int);
```

– Định nghĩa hàm.

```
03426.  bool ktDoiXung(int n)
03427.  {
03428.     int t = n;
03429.     int dn = 0;
03430.     while (t != 0)
03431.     {
03432.        dn = dn * 10 + t % 10;
03433.        t /= 10;
03434.     }
03435.     if (dn == n)
03436.        return true;
03437.     return false;
03438.  }
```

– Định nghĩa hàm tìm số đối xứng đầu.

```
03439.  int DoiXungDau(int a[], int n)
03440.  {
03441.     for (int i = 0; i <= n-1; i++)
03442.        if (ktDoiXung(a[i]))
03443.           return a[i];
03444.     return 10;
03445.  }
```

Bài 092.Hãy tìm giá trị đầu tiên trong mảng một chiều các số nguyên có chữ số đầu tiên là chữ số lẻ. Nếu trong mảng không tồn tại giá trị như vậy hàm sẽ trả về giá trị 0.

– Ví dụ:

191	29	622	763	244	647	367	234	8712	134

– Các giá trị trong mảng có chữ số đầu tiên là chữ số chẵn.

191	29	622	763	244	647	367	234	8712	134

– Kết quả: 191.

– Khai báo hàm.

```
03446.  int ChuSoDau(int);
03447.  int TimGiaTri(int [],int);
```

– Định nghĩa hàm.

```
03448.  int ChuSoDau(int n)
03449.  {
03450.     int t = abs(n);
03451.     while (t >= 10)
```

```
03452.        t /= 10;
03453.    return t;
03454. }
```

– Định nghĩa hàm tìm giá trị đầu tiên trong mảng một chiều các số nguyên có chữ số đầu tiên là chữ số lẻ.

```
03455. int TimGiaTri (int a[], int n)
03456. {
03457.    for (int i = 0; i <= n-1; i++)
03458.       if (ChuSoDau(a[i]) & 1)
03459.          return a[i];
03460.    return 0;
03461. }
```

Bài 093.Cho mảng một chiều các số nguyên. Hãy viết hàm tìm giá trị đầu tiên trong mảng có dạng 2^m. Nếu mảng không tồn tại giá trị dạng 2^m thì hàm sẽ trả về giá trị 0.

– Ví dụ:

64	29	62	16	99	23	128	1	88	08

– Các giá trị trong mảng có dạng 2^m.

64	29	62	16	99	23	128	1	88	08

– Kết quả: 217.

– Khai báo hàm.

```
03462. bool ktDang2m(int);
03463. int TimGiaTri(int [],int);
```

– Định nghĩa hàm.

```
03464. bool ktDang2m(int n)
03465. {
03466.    if(n < 1)
03467.       return false;
03468.    bool flag = true;
03469.    int t = abs(n);
03470.    while (t > 1)
03471.    {
03472.       int du = t % 2;
03473.       if (du != 0)
03474.          flag = false;
03475.       t /= 2;
03476.    }
03477.    return flag;
03478. }
```

– Định nghĩa hàm tìm giá trị đầu tiên trong mảng có dạng 2^m.

```
03479.  int TimGiaTri (int a[], int n)
03480.  {
03481.     for (int i=0; i<n; i++)
03482.        if (ktDang2m(a[i]))
03483.           return a[i];
03484.     return 0;
03485.  }
```

Bài 094. Tìm "số nguyên tố cuối cùng" trong mảng một chiều các số nguyên. Nếu mảng không có số nguyên tố thì trả về giá trị không nguyên tố là 0.

- Ví dụ:

19	29	62	76	223	23	227	98	88	269

- Các số nguyên tố có trong mảng

19	29	62	76	223	23	227	98	88	269

- Kết quả: 269.
- Khai báo hàm.

```
03486.  bool ktNguyenTo(int);
03487.  int NguyenToCuoi(int [],int);
```

- Định nghĩa hàm kiểm tra nguyên tố.

```
03488.  bool ktNguyenTo(int k)
03489.  {
03490.     int dem = 0;
03491.     for(int i=1; i<=k; i++)
03492.        if(k%i==0)
03493.           dem++;
03494.     if(dem==2)
03495.        return true;
03496.     return false;
03497.  }
```

- Định nghĩa hàm tìm "số nguyên tố cuối cùng" trong mảng một chiều các số nguyên.

```
03498.  int NguyenToCuoi(int a[], int n)
03499.  {
03500.     for (int i = n-1; i >= 0; i--)
03501.        if (ktNguyenTo(a[i]))
03502.           return a[i];
03503.     return 0;
03504.  }
```

Bài 095.Tìm "số hoàn thiện cuối cùng" trong mảng một chiều các số nguyên. Nếu mảng không có số hoàn thiện thì hàm sẽ trả về giá trị không hoàn thiện là −1.

- Ví dụ:

97	6	62	76	28	496	8128	98	88	18

- Các số hoàn thiện có trong mảng.

97	6	62	76	28	496	8128	98	88	18

- Kết quả: 8128.
- Khai báo hàm.

```
03505.  bool ktHoanThien(int);
03506.  int HoanThienCuoi(int [],int);
```

- Định nghĩa hàm.

```
03507.  bool ktHoanThien(int k)
03508.  {
03509.      int s = 0;
03510.      for(int i=1; i<k; i++)
03511.          if(k%i==0)
03512.              s = s + i;
03513.      if(s==k)
03514.          return true;
03515.      return false;
03516.  }
```

- Định nghĩa hàm tìm "số hoàn thiện cuối cùng" trong mảng một chiều các số nguyên.

```
03517.  int HoanThienCuoi(int a[], int n)
03518.  {
03519.     for (int i = n-1; i >= 0; i--)
03520.        if (ktHoanThien(a[i]) == 1)
03521.           return a[i];
03522.     return -1;
03523.  }
```

Bài 096.(Hạt nhân) Tìm "vị trí của giá trị chẵn đầu tiên" trong mảng một chiều các số nguyên. Nếu mảng không có giá trị chẵn thì hàm sẽ trả về giá trị là −1.

- Ví dụ:

91	29	62	76	99	34	23	12	98	88	11

- Các giá trị chẵn trong mảng.

91	29	62	76	99	34	23	12	98	88	11

- Kết quả: 2.
- Khai báo hàm.

```
03524.  int ViTriDau(int [],int);
```

– Định nghĩa hàm tìm "vị trí của giá trị chẵn đầu tiên" trong mảng một chiều các số nguyên.

```
03525.  int ViTriDau (int a[], int n)
03526.  {
03527.     for (int i=0; i<n; i++)
03528.        if (a[i] % 2 == 0)
03529.           return i;
03530.     return -1;
03531.  }
```

Bài 097.(Hạt nhân) Tìm "vị trí số hoàn thiện cuối cùng" trong mảng một chiều các số nguyên. Nếu mảng không có số hoàn thiện thì trả về giá trị -1.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
97	6	62	76	28	496	8128	98	88	18

– Các số hoàn thiện có trong mảng.

0	1	2	3	4	5	6	7	8	9
97	6	62	76	28	496	8128	98	88	18

– Kết quả: 6.

– Khai báo hàm.

```
03532.  bool ktHoanThien(int);
03533.  int ViTriCuoi(int [],int);
```

– Định nghĩa hàm.

```
03534.  bool ktHoanThien(int k)
03535.  {
03536.      int s = 0;
03537.      for(int i=1; i<k; i++)
03538.          if(k%i==0)
03539.              s = s + i;
03540.      if(s==k)
03541.          return true;
03542.      return false;
03543.  }
```

– Định nghĩa hàm tìm "vị trí số hoàn thiện cuối cùng" trong mảng một chiều các số nguyên.

```
03544.  int ViTriCuoi(int a[], int n)
03545.  {
03546.     for (int i = n - 1; i >= 0; i--)
03547.        if (ktHoanThien(a[i]))
```

```
03548.          return i;
03549.     return -1;
03550. }
```

Bài 098.(Hạt nhân) Hãy tìm "giá trị dương nhỏ nhất" trong mảng các số thực. Nếu mảng không có giá trị dương thì trả về giá trị không dương là 0.

- Ví dụ:

1.9	2.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

- Các số dương trong mảng

1.9	2.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

- Kết quả: 1.9
- Khai báo hàm.

```
03551. float DuongDau(float [], int);
03552. float DuongNhoNhat(float [],int);
```

- Định nghĩa hàm tìm giá trị dương đầu tiên trong mảng.

```
03553. float DuongDau(float a[], int n)
03554. {
03555.    for (int i=0; i<n; i++)
03556.       if (a[i] > 0)
03557.          return a[i];
03558.    return 0;
03559. }
```

- Định nghĩa hàm tìm giá trị dương nhỏ nhất trong mảng.

```
03560. float DuongNhoNhat(float a[], int n)
03561. {
03562.    float lc = DuongDau(a, n);
03563.    if (lc == 0)
03564.       return 0;
03565.    for (int i=0; i<n; i++)
03566.       if (a[i] > 0 && a[i] < lc)
03567.          lc = a[i];
03568.    return lc;
03569. }
```

Bài 099.(Hạt nhân) Hãy tìm "vị trí giá trị dương nhỏ nhất" trong mảng một chiều các số thực. Nếu mảng không có giá trị dương thì hàm trả về một giá trị ngoài đoạn $[0, n-1]$ là -1 nhằm mô tả không có vị trí nào thỏa điều kiện.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9

15.9	12.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

– Các số dương trong mảng

0	1	2	3	4	5	6	7	8	9
15.9	12.9	-0.6	7.6	-9.9	-23	12.1	-0.9	8.8	-1.1

– Kết quả: 3.

– Khai báo hàm.

```
03570.  int ViTriDau(float [], int);
03571.  int TimViTri(float [],int);
```

– Định nghĩa hàm tìm vị trí dương đầu tiên trong mảng.

```
03572.  int ViTriDau(float a[], int n)
03573.  {
03574.     for (int i=0; i<n; i++)
03575.        if (a[i] > 0)
03576.           return i;
03577.     return -1;
03578.  }
```

– Định nghĩa hàm tìm vị trí dương nhỏ nhất trong mảng.

```
03579.  int TimViTri (float a[], int n)
03580.  {
03581.     int lc = ViTriDau(a, n);
03582.     if (lc == -1)
03583.        return -1;
03584.     for (int i=0; i<n; i++)
03585.        if (a[i] > 0 && a[i] < a[lc])
03586.           lc = i;
03587.     return lc;
03588.  }
```

Bài 100.Hãy tìm "giá trị âm lớn nhất" trong mảng các số thực. Nếu mảng không có giá trị âm thì trả về giá trị 0.

– Ví dụ:

1.04	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

– Các số âm trong mảng.

19	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

– Các số âm trong mảng lớn hơn -1.

19	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

– Kết quả: -0.6.

– Khai báo hàm.

```
03589.  float AmDau(float [], int);
03590.  float AmLonNhat(float [],int);
```

– Định nghĩa hàm tìm giá trị âm đầu tiên trong mảng.

```
03591.  float AmDau(float a[], int n)
03592.  {
03593.     for (int i=0; i<n; i++)
03594.        if (a[i] < 0)
03595.           return a[i];
03596.     return 0;
03597.  }
```

– Định nghĩa hàm tìm giá trị âm lớn nhất trong mảng.

```
03598.  float AmLonNhat(float a[], int n)
03599.  {
03600.     float lc = AmDau(a, n);
03601.     if (lc == 0)
03602.        return 0;
03603.     for (int i=0; i<n; i++)
03604.        if (a[i]<0 && a[i]>lc)
03605.           lc = a[i];
03606.     return lc;
03607.  }
```

Bài 101.Hãy tìm "số nguyên tố lớn nhất" trong mảng một chiều các số nguyên. Nếu mảng không có số nguyên tố thì trả về giá trị 0.

– Ví dụ:

19	29	62	76	223	23	227	98	88	19

– Các số nguyên tố có trong mảng

19	29	62	76	223	23	227	98	88	19

– Kết quả: 227.

– Khai báo hàm.

```
03608.  bool ktNguyenTo(int);
03609.  int NguyenToDau(int [], int);
03610.  int TimGiaTri(int [],int);
```

– Định nghĩa hàm kiểm tra nguyên tố.

```
03611.  bool ktNguyenTo(int k)
03612.  {
03613.     int dem = 0;
03614.     for(int i=1; i<=k; i++)
03615.        if(k%i==0)
03616.           dem++;
03617.     if(dem==2)
03618.        return true;
03619.     return false;
03620.  }
```

– Định nghĩa hàm tìm giá trị nguyên tố đầu tiên trong mảng.

```
03621.  int NguyenToDau(int a[], int n)
03622.  {
03623.     for (int i=0; i<n; i++)
03624.        if (ktNguyenTo(a[i]))
03625.           return a[i];
03626.     return 0;
03627.  }
```

– Định nghĩa hàm tìm giá trị nguyên tố lớn nhất trong mảng.

```
03628.  int TimGiaTri (int a[], int n)
03629.  {
03630.     int lc = NguyenToDau(a, n);
03631.     if (lc == 0)
03632.        return 0;
03633.     for (int i=0; i<n; i++)
03634.        if (ktNguyenTo(a[i]) && a[i] > lc)
03635.           lc = a[i];
03636.     return lc;
03637.  }
```

Bài 102.Hãy tìm "hoàn thiện nhỏ nhất" trong mảng một chiều các số nguyên. Nếu mảng không có số hoàn thiện thì hàm trả về giá trị không hoàn thiện là −1.

– Ví dụ:

97	6	62	76	28	496	8128	98	88	18

– Các số hoàn thiện có trong mảng.

97	6	62	76	28	496	8128	98	88	18

– Kết quả: 6.

– Khai báo hàm.

```
03638.  bool ktHoanThien(int);
03639.  int HoanThienDau(int [], int);
03640.  int TimGiaTri(int [],int);
```

– Định nghĩa hàm.

```
03641.  bool ktHoanThien(int k)
03642.  {
03643.      int s = 0;
03644.      for(int i=1; i<k; i++)
03645.          if(k%i==0)
03646.              s = s + i;
03647.      if(s==k)
03648.          return true;
```

```
03649.      return false;
03650.  }
```

– Định nghĩa hàm tìm giá trị hoàn thiện đầu tiên trong mảng.

```
03651.  int HoanThienDau(int a[], int n)
03652.  {
03653.     for (int i=0; i<n; i++)
03654.        if (ktHoanThien(a[i]) == 1)
03655.           return a[i];
03656.     return -1;
03657.  }
```

– Định nghĩa hàm tìm giá trị hoàn thiện nhỏ nhất trong mảng.

```
03658.  int TimGiaTri (int a[], int n)
03659.  {
03660.     int lc = HoanThienDau(a, n);
03661.     if (lc == -1)
03662.        return -1;
03663.     for (int i=0; i<n; i++)
03664.        if (ktHoanThien(a[i]) == 1 && a[i] < lc)
03665.           lc = a[i];
03666.     return lc;
03667.  }
```

Bài 103.Hãy tìm "giá trị chẵn nhỏ nhất" trong mảng một chiều các số nguyên. Nếu mảng không có giá trị chẵn thì hàm trả về giá trị không chẵn là −1.

– Ví dụ:

91	29	62	76	99	34	23	12	98	88	11

– Các giá trị chẵn trong mảng.

91	29	62	76	99	34	23	12	98	88	11

– Kết quả: 12.

– Khai báo hàm.

```
03668.  int ChanDau(int [], int);
03669.  int ChanNhoNhat(int [],int);
```

– Định nghĩa hàm tìm giá trị chẵn đầu tiên trong mảng.

```
03670.  int ChanDau(int a[], int n)
03671.  {
03672.     for (int i=0; i<n; i++)
03673.        if (a[i] % 2 == 0)
03674.           return a[i];
03675.     return -1;
03676.  }
```

– Định nghĩa hàm tìm giá trị chẵn nhỏ nhất trong mảng.

```
03677.  int ChanNhoNhat(int a[], int n)
03678.  {
03679.     int lc = ChanDau(a, n);
03680.     if (lc == -1)
03681.        return -1;
03682.     for (int i=0; i<n; i++)
03683.        if (a[i] % 2 == 0 && a[i] < lc)
03684.           lc = a[i];
03685.     return lc;
03686.  }
```

Bài 104.Hãy tìm "vị trí giá trị âm lớn nhất" trong mảng các số thực. Nếu mảng không có giá trị âm thì hàm trả về −1.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

– Các số âm trong mảng.

0	1	2	3	4	5	6	7	8	9
19	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

– Âm lớn nhất: -0.6

– Kết quả: 2.

– Khai báo hàm.

```
03687.  int ViTriAmDau(float [], int);
03688.  int TimViTri(float [],int);
```

– Định nghĩa hàm tìm vị trí âm đầu.

```
03689.  int ViTriAmDau(float a[], int n)
03690.  {
03691.     for (int i=0; i<n; i++)
03692.        if (a[i] < 0)
03693.           return i;
03694.     return -1;
03695.  }
```

– Định nghĩa hàm tìm vị trí âm lớn nhất.

```
03696.  int TimViTri (float a[], int n)
03697.  {
03698.     int lc = ViTriAmDau(a, n);
03699.     if (lc == -1)
03700.        return -1;
03701.     for (int i=0; i<n; i++)
03702.        if (a[i]<0 && a[i]>a[lc])
```

```
03703.          lc = i;
03704.    return lc;
03705. }
```

Bài 105.Hãy tìm giá trị thỏa điều kiện toàn chữ số lẻ và là giá trị lớn nhất thỏa điều kiện ấy trong mảng một chiều các số nguyên (nếu mảng không có giá trị thỏa điều kiện trên thì hàm trả về giá trị 0).

- Ví dụ:

191	29	622	753	244	647	397	7139	8712	134

- Các giá trị có toàn chữ số lẻ.

191	29	622	753	244	647	397	7139	8712	134

- Kết quả: 7139.
- Khai báo hàm.

```
03706. bool ktToanLe(int);
03707. int ToanLeDauTien(int [], int);
03708. int TimGiaTri(int [],int);
```

- Định nghĩa hàm.

```
03709. bool ktToanLe(int n)
03710. {
03711.
03712.    int t = n;
03713.    while (t != 0)
03714.    {
03715.       if (t % 2 == 0)
03716.          return false;
03717.       t /= 10;
03718.    }
03719.    return true;
03720. }
```

- Định nghĩa hàm giá trị toàn lẻ đầu tiên.

```
03721. int ToanLeDauTien(int a[], int n)
03722. {
03723.    for (int i=0; i<n; i++)
03724.       if (ktToanLe(a[i]) != 0)
03725.          return a[i];
03726.    return 0;
03727. }
```

- Định nghĩa hàm giá trị toàn lẻ lớn nhất.

```
03728. int TimGiaTri (int a[], int n)
03729. {
03730.    int lc = ToanLeDauTien(a, n);
```

```
03731.     if (lc == 0)
03732.        return 0;
03733.     for (int i=0; i<n; i++)
03734.        if (ktToanLe(a[i]) == 1 && a[i] > lc)
03735.           lc = a[i];
03736.     return lc;
03737.  }
```

Bài 106.Cho mảng một chiều các số nguyên. Hãy viết hàm tìm giá trị lớn nhất trong mảng có dạng 5^m. Nếu mảng không tồn tại giá trị dạng 5^m thì hàm sẽ trả về giá trị 0.

- Ví dụ:

125	29	62	5	99	23	1	625	88	25

- Các giá trị trong mảng có dạng 5^m.

125	29	62	5	99	23	1	625	88	25

- Kết quả: 781.
- Khai báo hàm.

```
03738.  bool ktDang5m(int);
03739.  int DauTien(int [], int);
03740.  int TimGiaTri(int [],int);
```

- Định nghĩa hàm.

```
03741.  bool ktDang5m(int n)
03742.  {
03743.     if(n < 1)
03744.        return false;
03745.     bool flag = true;
03746.     int t = abs(n);
03747.     while (t > 1)
03748.     {
03749.        int du = t % 5;
03750.        if (du != 0)
03751.           flag = false;
03752.        t /= 5;
03753.     }
03754.     return flag;
03755.  }
```

- Định nghĩa hàm giá trị dạng 5^m đầu tiên.

```
03756.  int DauTien(int a[], int n)
03757.  {
03758.     for (int i=0; i<n; i++)
03759.        if (ktDang5m(a[i]) != 0)
```

```
03760.          return a[i];
03761.    return 0;
03762. }
```

– Định nghĩa hàm giá trị dạng 5^m lớn nhất.

```
03763. int TimGiaTri (int a[], int n)
03764. {
03765.    int lc = DauTien(a, n);
03766.    if (lc == 0)
03767.       return 0;
03768.    for (int i=0; i<n; i++)
03769.       if (ktDang5m(a[i]) == 1 && a[i] > lc)
03770.          lc = a[i];
03771.    return lc;
03772. }
```

Bài 107.(Hạt nhân) Hãy tìm một giá trị có số lần xuất hiện nhiều nhất trong mảng các số nguyên.

– Ví dụ:

19	29	19	76	99	29	19	98	88	11

– Các giá trị có trong mảng.

19	29	19	76	99	29	19	98	88	11

– Tần suất xuất hiện của các giá trị trong mảng.

19	29	76	99	98	88	11
3	2	1	1	1	1	1

– Kết quả: 19.

– Khai báo hàm.

```
03773. int TanSuat(int [], int, int);
03774. int TimGiaTri(int [],int);
```

– Định nghĩa hàm.

```
03775. int TanSuat(int a[], int n, int x)
03776. {
03777.    int dem = 0;
03778.    for (int i=0; i<=n-1; i++)
03779.       if (a[i] == x)
03780.          dem++;
03781.    return dem;
03782. }
```

– Định nghĩa hàm tìm giá trị xuất hiện nhiều nhất trong mảng.

```
03783. int TimGiaTri (int a[], int n)
03784. {
03785.    int lc = a[0];
```

```
03786.     for (int i=0; i<=n-1; i++)
03787.        if(TanSuat(a,n,a[i])>TanSuat(a,n,lc))
03788.           lc = a[i];
03789.     return lc;
03790.  }
```

Bài 108.(*) Cho mảng một chiều các số nguyên dương. Hãy viết hàm tìm ước chung lớn nhất của tất cả các phần tử trong mảng.

- Ví dụ:

16	40	96	56

- Kết quả: 8.
- Khai báo hàm.

```
03791.  int ucln(int, int);
03792.  int TimUCLN(int [],int);
```

- Định nghĩa hàm tìm ước chung lớn nhất của hai số nguyên.

```
03793.  int ucln(int a, int b)
03794.  {
03795.     a = abs(a);
03796.     b = abs(b);
03797.     while(a*b!=0)
03798.     {
03799.        if(a>b)
03800.           a=a-b;
03801.        else
03802.           b=b-a;
03803.     }
03804.     return (a+b);
03805.  }
```

- Định nghĩa hàm tìm ước chung lớn nhất của mảng một chiều các số nguyên.

```
03806.  int TimUCLN (int a[], int n)
03807.  {
03808.     int lc = a[0];
03809.     for (int i=0; i<n; i++)
03810.        lc = ucln(lc, a[i]);
03811.     return lc;
03812.  }
```

Bài 109. (*) Cho mảng một chiều các số nguyên dương. Hãy viết hàm tìm bội chung nhỏ nhất của tất cả các phần tử trong mảng (172).

- Ví dụ:

16	40	9	5

– Kết quả: 720.

– Khai báo hàm.

```
03813. int ucln(int, int);
03814. int bcnn(int, int);
03815. int TimBCNN(int [],int);
```

– Định nghĩa hàm tìm ước chung lớn nhất của hai số nguyên.

```
03816. int ucln(int a, int b)
03817. {
03818.    a = abs(a);
03819.    b = abs(b);
03820.    while(a*b!=0)
03821.    {
03822.       if(a>b)
03823.          a=a-b;
03824.       else
03825.          b=b-a;
03826.    }
03827.    return (a+b);
03828. }
```

– Định nghĩa hàm tìm bội chung nhỏ nhất của hai số nguyên.

```
03829. int bcnn(int a, int b)
03830. {
03831.    return abs(a*b)/ucln(a,b);
03832. }
```

– Định nghĩa hàm tìm bội chung nhỏ nhất của mảng một chiều các số nguyên.

```
03833. int TimBCNN (int a[], int n)
03834. {
03835.    int lc = a[0];
03836.    for (int i=0; i<n; i++)
03837.       lc = bcnn(lc, a[i]);
03838.    return lc;
03839. }
```

Bài 110. (*) Cho mảng một chiều các số nguyên. Hãy viết hàm tìm số nguyên tố nhỏ nhất lớn hơn mọi giá trị có trong mảng.

– Ví dụ.

97	29	62	76	223	23	5	98	88	19

– Giá trị lớn nhất trong mảng 223.

97	29	62	76	223	23	5	98	88	19

– Kết quả: 227.

– Khai báo hàm.

```
03840. bool ktNguyenTo(int);
03841. int LonNhat(int [],int);
03842. int TimGiaTri(int [],int);
```

– Định nghĩa hàm kiểm tra nguyên tố.

```
03843. bool ktNguyenTo(int k)
03844. {
03845.    int dem = 0;
03846.    for(int i=1; i<=k; i++)
03847.       if(k%i==0)
03848.          dem++;
03849.    if(dem==2)
03850.       return true;
03851.    return false;
03852. }
```

– Định nghĩa hàm tìm giá trị lớn nhất.

```
03853. int LonNhat(int a[], int n)
03854. {
03855.    int lc = a[0];
03856.    for (int i=0; i<n; i++)
03857.       if (a[i] > lc)
03858.          lc = a[i];
03859.    return lc;
03860. }
```

– Định nghĩa hàm tìm giá trị.

```
03861. int TimGiaTri (int a[], int n)
03862. {
03863.    int lc = LonNhat(a,n)+1;
03864.    while(ktNguyenTo(lc)==0)
03865.       lc++;
03866.    return lc;
03867. }
```

Bài 111. (*) Cho mảng một chiều các số nguyên. Hãy viết hàm tìm số chẵn lớn nhất nhỏ hơn mọi giá trị lẻ có trong mảng nếu mảng không có giá trị lẻ thì hàm trả về giá trị không chẵn là -1.

– Ví dụ:

91	29	62	76	99	34	23	12	98	88	111

– Các giá trị lẻ trong mảng.

91	29	62	76	99	34	23	12	98	88	111

– Giá trị lẻ nhỏ nhất trong mảng.

91	29	62	76	99	34	23	12	98	88	111

– Kết quả: 22.

– Khai báo hàm.

```
03868. int LeDauTien(int [], int);
03869. int LeNhoNhat(int [], int);
03870. int TimGiaTri(int [],int);
```

– Định nghĩa hàm tìm giá trị lẻ đầu tiên.

```
03871. int LeDauTien(int a[], int n)
03872. {
03873.    for (int i=0; i<n; i++)
03874.       if (a[i] % 2 != 0)
03875.          return a[i];
03876.    return 0;
03877. }
```

– Định nghĩa hàm tìm giá trị lẻ nhỏ nhất.

```
03878. int LeNhoNhat (int a[], int n)
03879. {
03880.    int lc = LeDauTien(a, n);
03881.    if (lc == 0)
03882.       return 0;
03883.    for (int i=0; i<n; i++)
03884.       if (a[i] % 2 != 0 && a[i] < lc)
03885.          lc = a[i];
03886.    return lc;
03887. }
```

– Định nghĩa hàm tìm giá trị.

```
03888. int TimGiaTri(int a[],int n)
03889. {
03890.    int lc = LeNhoNhat(a,n);
03891.    if(lc==0)
03892.       return -1;
03893.    return lc-1;
03894. }
```

Bài 112. (*) Hãy liệt kê các giá trị có số lần xuất hiện nhiều nhất trong mảng các số nguyên.

– Ví dụ:

19	29	19	76	99	29	19	98	88	11

– Các giá trị có trong mảng.

19	29	19	76	99	29	99	98	88	11

– Tần suất xuất hiện của các giá trị trong mảng.

Giá trị	19	29	76	99	98	88	11
Tần suất	2	2	1	2	1	1	1

– Kết quả: 19, 29, 99.

– Khai báo hàm.

```
03895. int TanSuat(int [], int, int);
03896. int TimGiaTri(int [],int);
03897. void LietKe(int [],int);
```

– Định nghĩa hàm tần suất

```
03898. int TanSuat(int a[], int n, int x)
03899. {
03900.    int dem = 0;
03901.    for (int i=0; i<=n-1; i++)
03902.       if (a[i] == x)
03903.          dem++;
03904.    return dem;
03905. }
```

– Định nghĩa hàm.

```
03906. int TimGiaTri(int a[],int n)
03907. {
03908.    int lc = a[0];
03909.    for (int i=0; i<=n-1; i++)
03910.       if(TanSuat(a,n,a[i])>TanSuat(a,n,lc))
03911.          lc = a[i];
03912.    return lc;
03913. }
```

– Định nghĩa hàm.

```
03914. void LietKe(int a[],int n)
03915. {
03916.    int lc = TimGiaTri(a,n);
03917.    int ts = TanSuat(a,n,lc);
03918.    for (int i=0; i<=n-1; i++)
03919.    {
03920.       int flag = 1;
03921.       for (int j = 0; j <= i - 1; j++)
03922.          if (a[j]==a[i])
03923.             flag = 0;
03924.       if(TanSuat(a,n,a[i])==ts && flag)
03925.          cout << a[i];
03926.    }
03927. }
```

Bài 113. (*) Cho mảng một chiều các số nguyên. Hãy viết hàm tìm chữ số xuất hiện nhiều nhất trong mảng.

- Ví dụ:

19	29	62	706	99	23	12	98	88	11

- Tần suất xuất hiện của các chữ số trong mảng.

Chữ số	0	1	2	3	4	5	6	7	8	9
Tần suất	1	4	4	1	0	0	2	1	3	5

- Kết quả: 9.
- Khai báo hàm.

```
03928.  int ViTriLonNhat(int [], int);
03929.  int TimChuSo(int [],int);
```

- Định nghĩa hàm tìm vị trí lớn nhất.

```
03930.  int ViTriLonNhat(int a[], int n)
03931.  {
03932.     int lc = 0;
03933.     for (int i=0; i<=n-1; i++)
03934.        if (a[i] > a[lc])
03935.           lc = i;
03936.     return lc;
03937.  }
```

- Định nghĩa hàm tìm chữ số xuất hiện nhiều nhất.

```
03938.  int TimChuSo (int a[], int n)
03939.  {
03940.     int dem[10] = { 0,0,0,0,0,0,0,0,0,0 };
03941.     for (int i=0; i<n; i++)
03942.     {
03943.        if (a[i] == 0)
03944.           dem[0]++;
03945.        int t = abs(a[i]);
03946.        while (t != 0)
03947.        {
03948.           int dv = t % 10;
03949.           dem[dv]++;
03950.           t = t / 10;
03951.        }
03952.     }
03953.     return ViTriLonNhat(dem, 10);
03954.  }
```

Bài 114. (*) Cho mảng một chiều các số nguyên. Hãy viết hàm tìm một chữ số xuất hiện ít nhất trong mảng.

– Ví dụ:

19	29	62	706	99	23	12	98	88	11

– Tần suất xuất hiện của các chữ số trong mảng.

Chữ số	0	1	2	3	4	5	6	7	8	9
Tần suất	1	4	4	1	0	0	2	1	3	5

– Kết quả: 0.

– Khai báo hàm.

```
03955.  int ViTriDuongDau(int [], int);
03956.  int ViTriDuongNhoNhat(int [], int);
03957.  int TimChuSo(int [],int);
```

– Định nghĩa hàm tìm vị trí dương đầu.

```
03958.  int ViTriDuongDau (int a[], int n)
03959.  {
03960.     for (int i=0; i<=n-1; i++)
03961.        if(a[i] > 0)
03962.           return i;
03963.     return -1;
03964.  }
```

– Định nghĩa hàm tìm vị trí dương nhỏ nhất.

```
03965.  int ViTriDuongNhoNhat(int a[], int n)
03966.  {
03967.     int lc = ViTriDuongDau(a,n);
03968.     if(lc==-1)
03969.        return -1;
03970.     for (int i=0; i<=n-1; i++)
03971.        if (a[i]>0 && a[i] < a[lc])
03972.           lc = i;
03973.     return lc;
03974.  }
```

– Định nghĩa hàm tìm chữ số xuất hiện ít nhất.

```
03975.  int TimChuSo(int a[], int n)
03976.  {
03977.     int dem[10] = { 0,0,0,0,0,0,0,0,0,0 };
03978.     for (int i=0; i<n; i++)
03979.     {
03980.        if (a[i] == 0)
03981.           dem[0]++;
03982.        int t = abs(a[i]);
03983.        while (t != 0)
03984.        {
03985.           int dv = t % 10;
```

```
03986.           dem[dv]++;
03987.           t = t / 10;
03988.        }
03989.     }
03990.     return ViTriDuongNhoNhat(dem, 10);
03991. }
```

Bài 115. (*) Cho mảng số thực có nhiều hơn hai giá trị và các giá trị trong mảng khác nhau từng đôi một. Hãy viết hàm tìm hai giá trị gần nhau nhất trong mảng.

- Ví dụ:

19	9	6	7	20

- Các cặp giá trị có trong mảng.
  + $(19, 9), (19, 6), (19, 7), (19, 20)$.
  + $(9,19), (9, 6), (9, 7), (9, 20)$.
  + $(6, 19), (6,9), (6, 7), (6, 20)$.
  + $(7, 19), (7,9), (7, 6), (7, 20)$.
  + $(20, 19), (20,9), (20, 6), (20, 7)$.
- Kết quả: $(19, 20)$.
- Khai báo hàm.

```
03992. void GanNhat(float [],int, float &, float &);
```

- Định nghĩa hàm hàm tìm hai giá trị gần nhau nhất.

```
03993. void GanNhat(  float a[],int n,
03994.                float &x, float &y)
03995. {
03996.    x = a[0];
03997.    y = a[1];
03998.    for (int i = 0; i <= n - 2; i++)
03999.       for (int j = i + 1; j <= n - 1; j++)
04000.          if (abs(a[i] - a[j]) < abs(x - y))
04001.          {
04002.             x = a[i];
04003.             y = a[j];
04004.          }
04005. }
```

Bài 116. (*) Cho mảng số nguyên có nhiều hơn hai giá trị. Hãy liệt kê tất cả các cặp giá trị gần nhau nhất trong mảng.

- Ví dụ:

19	9	6	7	20

- Các cặp giá trị có trong mảng.

+ $(19,9),(19,6),(19,7),(19,20)$.
+ $(9,19),(9,6),(9,7),(9,20)$.
+ $(6,19),(6,9),(6,7),(6,20)$.
+ $(7,19),(7,9),(7,6),(7,20)$.
+ $(20,19),(20,9),(20,6),(20,7)$.

– Kết quả: $(19,20),(6,7),(7,6),(20,19)$.

– Khai báo hàm.

```
04006.  void GanNhat(int [],int,int &,int &);
04007.  void LietKe(int [],int);
```

– Định nghĩa hàm hàm tìm hai giá trị gần nhau nhất.

```
04008.  void GanNhat(int a[],int n, int &x, int &y)
04009.  {
04010.     x = a[0];
04011.     y = a[1];
04012.     for (int i = 0; i <= n - 2; i++)
04013.        for (int j = i + 1; j <= n - 1; j++)
04014.           if (abs(a[i] - a[j]) < abs(x - y))
04015.           {
04016.              x = a[i];
04017.              y = a[j];
04018.           }
04019.  }
```

– Định nghĩa hàm liệt kê.

```
04020.  void LietKe (int a[], int n)
04021.  {
04022.     int x,y;
04023.     GanNhat(a, n, x, y);
04024.     for (int i=0; i<=n-1; i++)
04025.        for (int j = 0; j <= n - 1; j++)
04026.           if(i!=j && abs(a[i]-a[j])==abs(x-y))
04027.              cout << "("<<a[i]<< ","<<a[j]<<")";
04028.  }
```

01.18.06 Kỹ thuật Đặt cờ hiệu

Bài 117.(Hạt nhân) Hãy kiểm tra mảng số nguyên có tồn tại giá trị không hay không? Nếu mảng không tồn tại giá trị không trả về giá trị 0, ngược lại trả về $+1$.

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Kết quả: 0.

– Khai báo hàm.

```
04029.  int ktKhong(int [],int);
```

– Định nghĩa hàm.

```
04030.  int ktKhong (int a[], int n)
04031.  {
04032.     int flag = 0;
04033.     for (int i=0; i<=n-1; i++)
04034.        if (a[i] == 0)
04035.           flag = 1;
04036.     return flag;
04037.  }
```

Bài 118.Hãy kiểm tra mảng số nguyên có tồn tại hai giá trị không liên tiếp hay không?

– Ví dụ:

19	0	62	76	99	0	0	98	0	11

– Kết quả: 1.

– Khai báo hàm.

```
04038.  int ktTonTai(int [],int);
```

– Định nghĩa hàm.

```
04039.  int ktTonTai (int a[], int n)
04040.  {
04041.     int flag = 0;
04042.     for (int i = 0; i <= n - 2; i++)
04043.        if (a[i] == 0 && a[i + 1] == 0)
04044.           flag = 1;
04045.     return flag;
04046.  }
```

Bài 119.Hãy kiểm tra mảng số nguyên có tồn tại giá trị chẵn hay không? Nếu không tồn tại giá trị chẵn trả về giá trị 0, ngược lại trả về +1.

– Ví dụ:

91	29	62	76	99	34	23	12	98	88	11

– Các giá trị chẵn trong mảng.

91	29	62	76	99	34	23	12	98	88	11

– Kết quả: 1.

– Khai báo hàm.

```
04047.  int ktTonTai (int [],int);
```

– Định nghĩa hàm.

```
04048.  int ktTonTai (int a[], int n)
```

```
04049. {
04050.    int flag = 0;
04051.    for (int i=0; i<n; i++)
04052.       if (a[i] % 2 == 0)
04053.          flag = 1;
04054.    return flag;
04055. }
```

Bài 120.Hãy kiểm tra mảng số nguyên có tồn tại số nguyên tố hay không? Nếu có trả về +1, nếu không trả về 0.

- Ví dụ:

97	29	62	76	223	23	5	98	88	19

- Các số nguyên tố có trong mảng

97	29	62	76	223	23	5	98	88	19

- Kết quả: 1.
- Khai báo hàm.

```
04056. bool ktNguyenTo (int);
04057. int ktTonTai (int [],int);
```

- Định nghĩa hàm kiểm tra nguyên tố.

```
04058. bool ktNguyenTo(int k)
04059. {
04060.    int dem = 0;
04061.    for (int i=1; i<=k; i++)
04062.       if(k%i==0)
04063.          dem++;
04064.    if(dem==2)
04065.       return true;
04066.    return false;
04067. }
```

- Định nghĩa hàm kiểm tra mảng có tồn tại số nguyên tố hay ko.

```
04068. int ktTonTai(int a[], int n)
04069. {
04070.    int flag = 0;
04071.    for (int i=0; i<n; i++)
04072.       if (ktNguyenTo(a[i]) == true)
04073.          flag = 1;
04074.    return flag;
04075. }
```

Bài 121.(Hạt nhân) Hãy kiểm tra mảng có thỏa mãn tính chất sau không: "Mảng không có tồn tại số hoàn thiện lớn hơn 256". Nếu thỏa trả về +1, nếu không trả về 0.

- Ví dụ:

97	6	62	76	28	496	8128	98	88	18

- Các số hoàn thiện có trong mảng.

97	6	62	76	28	496	8128	98	88	18

- Kết quả: 0 (có tồn tại).
- Khai báo hàm.

```
04076.  bool ktHoanThien(int);
04077.  int ktTinhChat(int [],int);
```

- Định nghĩa hàm.

```
04078.  bool ktHoanThien(int k)
04079.  {
04080.      int s = 0;
04081.      for(int i=1; i<k; i++)
04082.          if(k%i==0)
04083.              s = s + i;
04084.      if(s==k)
04085.          return true;
04086.      return false;
04087.  }
```

- Định nghĩa hàm kiểm tra mảng có thỏa mãn tính chất.

```
04088.  int ktTinhChat(int a[], int n)
04089.  {
04090.     int dem = 0;
04091.     for (int i=0; i<n; i++)
04092.        if (ktHoanThien(a[i]) && a[i] > 256)
04093.           dem++;
04094.     if (dem == 0)
04095.        return 1;
04096.     return 0;
04097.  }
```

Bài 122.(Hạt nhân) Hãy cho biết mảng các số nguyên có toàn giá trị chẵn hay không? Nếu mảng có tồn tại giá trị lẻ trả về giá trị 0, ngược lại trả về +1.

- Ví dụ:

91	29	52	86	99	824	23	2	82	18	11

- Các giá trị chẵn trong mảng.

91	29	52	86	99	824	23	2	82	18	11

- Kết quả: 0.
- Khai báo hàm.

```
04098.  int ktToanChan(int [],int);
```

- Định nghĩa hàm.

```
04099.  int ktToanChan(int a[], int n)
04100.  {
04101.     int dem = 0;
04102.     for (int i=0; i<n; i++)
04103.        if (a[i] % 2 == 0)
04104.           dem++;
04105.     if (dem == n)
04106.        return 1;
04107.     return 0;
04108.  }
```

Bài 123.Hãy kiểm tra mảng một chiều các số nguyên có đối xứng hay không?

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
11	16	1221	12	25	25	12	1221	16	11

- Kết quả: 1.
- Khai báo hàm.

```
04109.  int ktDoiXung(int [],int);
```

- Định nghĩa hàm.

```
04110.  int ktDoiXung(int a[], int n)
04111.  {
04112.     int flag = 1;
04113.     int d = 0;
04114.     int c = n - 1;
04115.     while (d < c)
04116.     {
04117.        if (a[d] != a[c])
04118.           flag = 0;
04119.        d++;
04120.        c--;
04121.     }
04122.     return flag;
04123.  }
```

Bài 124.Ta định nghĩa một mảng có tính chẵn lẻ, khi tổng của hai phần tử liên tiếp trong mảng luôn luôn là số lẻ. Hãy viết hàm kiểm tra mảng a có tính chẵn lẻ hay không?

- Ví dụ 01:

91	29	62	76	99	34	23	12	98	88	11

  + Các giá trị chẵn trong mảng.

91	29	62	76	99	34	23	12	98	88	11

  + Kết quả: 0.
- Ví dụ 02:

91	20	57	76	99	34	23	12	9	88	11

  + Các giá trị chẵn trong mảng.

91	20	57	76	99	34	23	12	9	88	11

  + Kết quả: 1.
- Khai báo hàm.

```
04124. int ktChanLe(int [],int);
```

- Định nghĩa hàm.

```
04125. int ktChanLe(int a[], int n)
04126. {
04127.    int flag = 1;
04128.    for (int i = 0; i <= n - 2; i++)
04129.       if ((a[i] + a[i + 1]) % 2 == 0)
04130.          flag = 0;
04131.    return flag;
04132. }
```

Bài 125.Hãy kiểm tra mảng có tăng dần hay không?

- Ví dụ:

19	29	62	76	99	123	182	200	211	321

- Kết quả: 1.
- Các phần tử trong mảng có giá trị đứng đằng sau nó.

19	29	62	76	99	123	182	200	211	321

- Mảng được gọi là tăng dần khi các bất đẳng thức sau được thỏa.
  $19 \le 29 \le 62 \le 76 \le 99 \le 123 \le 182 \le 200 \le 211 \le 321$
- Tổng quát: mảng tăng dần khi mọi phần tử trong mảng đều nhỏ hơn bằng phần tử đứng sau $a_0 \le a_1 \le a_2 \le \cdots \le a_{n-2} \le a_{n-1}$.
- Khai báo hàm.

```
04133. int ktTang(float [],int);
```

- Định nghĩa hàm.

```
04134. int ktTang(float a[], int n)
04135. {
```

```
04136.     int flag = 1;
04137.     for (int i = 0; i <= n - 2; i++)
04138.        if (a[i] > a[i + 1])
04139.           flag = 0;
04140.     return flag;
04141.  }
```

Bài 126.Hãy kiểm tra mảng có giảm dần hay không?

- Ví dụ:

19	29	62	76	99	23	12	98	88	11

- Kết quả: 0.
- Các phần tử trong mảng có giá trị đứng đằng sau nó.

19	29	62	76	99	123	182	200	211	321

- Tổng quát: mảng giảm dần khi mọi phần tử trong mảng đều lớn hơn bằng phần tử đứng sau $a_0 \geq a_1 \geq a_2 \geq \cdots \geq a_{n-2} \geq a_{n-1}$.
- Khai báo hàm.

```
04142.  int ktGiam(float [],int);
```

- Định nghĩa hàm.

```
04143.  int ktGiam(float a[], int n)
04144.  {
04145.     int flag = 1;
04146.     for (int i = 0; i <= n - 2; i++)
04147.        if (a[i] < a[i + 1])
04148.           flag = 0;
04149.     return flag;
04150.  }
```

Bài 127.Hãy cho biết các phần tử trong mảng có lập thành cấp số cộng không?

- Ví dụ 01:
  + Cho mảng sau:

19	29	62	76	99	23	12	98	88	11

  + Kết quả: 0.
- Ví dụ 02:
  + Cho mảng sau:

5	17	29	41	53	65	77	89	101	113

  + Kết quả: 1.
- Khai báo hàm.

```
04151.  int ktCSC(float [],int);
```

- Định nghĩa hàm.

```
04152.  int ktCSC(float a[], int n)
```

```
04153. {
04154.    if (n <= 1)
04155.       return 0;
04156.    int flag = 1;
04157.    for (int i = 0; i <= n - 2; i++)
04158.       if ((a[i] - a[i + 1]) != (a[0] - a[1]))
04159.          flag = 0;
04160.    return flag;
04161. }
```

Bài 128.Hãy cho biết các phần tử trong mảng có bằng nhau không?

- Ví dụ:

19	29	62	76	99	23	12	98	88	11

- Kết quả: 0.
- Khai báo hàm.

```
04162. int ktBang(float [],int);
```

- Định nghĩa hàm.

```
04163. int ktBang(float a[], int n)
04164. {
04165.    int flag = 1;
04166.    for (int i=0; i<n; i++)
04167.       if (a[i] != a[0])
04168.          flag = 0;
04169.    return flag;
04170. }
```

Bài 129.Người ta định nghĩa một mảng được gọi là "dạng sóng" khi phần tử có trị số *i* lớn hơn hoặc nhỏ hơn hai phần tử xung quanh nó. Hãy viết hàm kiểm tra trong *a* sóng hay không?

- Ví dụ 01:
  + Cho mảng.

19	29	6	76	9	23	12	98	88	94

  + Kết quả: 1.
- Ví dụ 02:
  + Cho mảng.

19	29	62	76	99	23	12	98	88	11

  + Kết quả: 0.
- Khai báo hàm.

```
04171. int ktSong(int [],int);
```

- Định nghĩa hàm.

```
04172. int ktSong(int a[], int n)
```

```
04173. {
04174.    if (n <= 1)
04175.       return 0;
04176.    if (n == 2)
04177.       if (a[0] == a[1])
04178.          return 0;
04179.       else
04180.          return 1;
04181.    int flag = 1;
04182.    for (int i = 1; i <= n - 2; i++)
04183.       if ((a[i]-a[i-1])*(a[i]-a[i + 1]) <= 0)
04184.          flag = 0;
04185.    return flag;
04186. }
```

Bài 130.Hãy cho biết tất cả các phần tử trong mảng *a* có nằm trong mảng *b* hay không?

- Ví dụ 01:
  - + Mảng a.

19	29	88	39	59

  - + Mảng b.

19	29	62	76	39	23	12

  - + Kết quả: 0.
- Ví dụ 01:
  - + Mảng a.

19	29	88	39	59

  - + Mảng b.

19	29	53	76	39	88	59

  - + Kết quả: 1.
- Khai báo hàm.

```
04187. int TanSuat(int [], int, int);
04188. int ktThuoc(int [],int,int [],int);
```

- Định nghĩa hàm.

```
04189. int TanSuat(int a[], int n, int x)
04190. {
04191.    int dem = 0;
04192.    for (int i=0; i<n; i++)
04193.       if (a[i] == x)
04194.          dem++;
04195.    return dem;
04196. }
```

– Định nghĩa hàm kiểm tra tất cả các phần tử trong mảng a có nằm trong mảng b hay không.

```
04197.  int ktThuoc(int a[],int n,int b[],int m)
04198.  {
04199.     int flag = 1;
04200.     for (int i=0; i<n; i++)
04201.        if (TanSuat(b, m, a[i]) == 0)
04202.           flag = 0;
04203.     return flag;
04204.  }
```

Bài 131.Hãy đếm số lượng giá trị trong mảng các số nguyên thỏa tính chất: "lớn hơn tất cả các giá trị đứng đằng trước nó".

– Ví dụ:

19	29	62	7	98	23	12	99	88	11

– Các phần tử có giá trị đứng đằng trước nó.

19	29	62	7	98	23	12	99	88	11

– Các phần tử lớn hơn tất cả các giá trị đứng đằng trước nó.

19	29	62	7	98	23	12	99	88	11

– Kết quả: 4.

– Khai báo hàm.

```
04205.  int DemGiaTri(int [],int);
```

– Định nghĩa hàm.

```
04206.  int DemGiaTri(int a[], int n)
04207.  {
04208.     int dem = 0;
04209.     for (int i = 1; i < n; i++)
04210.     {
04211.        int flag = 1;
04212.        for (int j = 0; j <= i - 1; j++)
04213.           if (a[j] >= a[i])
04214.              flag = 0;
04215.        if (flag == 1)
04216.           dem++;
04217.     }
04218.     return dem;
04219.  }
```

01.18.07 Kỹ thuật Xây dựng mảng

Bài 132.(Hạt nhân) Cho mảng một chiều các số nguyên *a*. Hãy tạo mảng *b* từ mảng *a*, sao cho mảng *b* chỉ chứa các giá trị lẻ (307).

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Các giá trị lẻ trong mảng.

19	29	62	76	99	23	12	98	88	11

– Kết quả.

19	29	99	23	11

– Khai báo hàm.

```
04220. void XayDung(int [],int,int [],int &);
```

– Định nghĩa hàm.

```
04221. void XayDung(int a[], int n, int b[], int &k)
04222. {
04223.    k = 0;
04224.    for (int i=0; i<n; i++)
04225.       if (a[i] % 2 != 0)
04226.       {
04227.          b[k] = a[i];
04228.          k++;
04229.       }
04230. }
```

Bài 133.Cho mảng một chiều các số thực *a*. Hãy tạo mảng *b* từ mảng *a*, sao cho mảng *b* chỉ chứa các giá trị âm.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

– Các số âm trong mảng.

0	1	2	3	4	5	6	7	8	9
19	29	-0.6	76	-9.9	-23	12	-0.98	88	-1.1

– Kết quả:

0	1	2	3	4
-0.6	-9.9	-23	-0.98	-1.1

– Khai báo hàm.

```
04231. void XayDung(float [],int, float [],int &);
```

– Định nghĩa hàm.

```
04232. void XayDung(  float a[],int n,
04233.                float b[],int &k)
04234. {
```

```
04235.     k = 0;
04236.     for (int i=0; i<n; i++)
04237.        if (a[i] < 0)
04238.           b[k++] = a[i];
04239.  }
```

Bài 134.Cho mảng một chiều các số nguyên a. Hãy tạo mảng b từ mảng a, sao cho mảng b chỉ chứa các số nguyên tố trong mảng a.

- Ví dụ:

97	29	62	76	223	23	5	98	88	19

- Các số nguyên tố có trong mảng.

97	29	62	76	223	23	5	98	88	19

- Kết quả.

97	29	223	23	5	19

- Khai báo hàm.

```
04240.  bool ktNguyenTo(int);
04241.  void XayDung(int [],int,int [],int &);
```

- Định nghĩa hàm kiểm tra nguyên tố.

```
04242.  bool ktNguyenTo(int k)
04243.  {
04244.     int dem = 0;
04245.     for(int i=1; i<=k; i++)
04246.        if(k%i==0)
04247.           dem++;
04248.     if(dem==2)
04249.        return true;
04250.     return false;
04251.  }
```

- Định nghĩa hàm xây dựng mảng.

```
04252.  void XayDung(int a[], int n, int b[], int &k)
04253.  {
04254.     k = 0;
04255.     for (int i=0; i<n; i++)
04256.        if (ktNguyenTo(a[i]))
04257.        {
04258.           b[k] = a[i];
04259.           k++;
04260.        }
04261.  }
```

Bài 135.(Hạt nhân) Cho mảng một chiều các số thực a. Hãy tạo mảng b từ mảng a, với $b[i]$ = tổng các phần tử lân cận với $a[i]$ trong mảng a.

- Ví dụ:

19	29	62	76	99	23	12	98	88	11

- Kết quả.

29	81	105	161	99	111	121	100	109	88

- Lân cận của các phần tử trong mảng.

0	1	2	3	4	5	6	7	8	9

  + Phần tử đầu tiên có một lân cận bên phải.
  + Phần tử cuối cùng có một lân cận bên trái.
  + Các phần tử có chỉ số từ $1 \rightarrow (n-2)$ có hai lân cận.

- Khai báo hàm.

```
04262. void XayDung(float [],int, float [],int&);
```

- Định nghĩa hàm.

```
04263. void XayDung(float a[],int n,float b[],int&k)
04264. {
04265.    if (n == 1)
04266.    {
04267.       k = 1;
04268.       b[0] = 0;
04269.       return;
04270.    }
04271.    k = n;
04272.    b[0] = a[1];
04273.    for (int i = 1; i <= n - 2; i++)
04274.       b[i] = a[i - 1] + a[i + 1];
04275.    b[k - 1] = a[n - 2];
04276. }
```

Bài 136.Hãy tạo mảng b từ mảng a các số nguyên bằng cách thêm các giá trị 0, 1 để mảng có tính "tính chẵn lẻ"s.

- Ví dụ:

19	29	62	76	88	99

- Các giá trị chẵn trong mảng.

19	29	62	76	88	99

- Kết quả.

19	0	29	62	1	76	1	88	99

- Khai báo hàm.

```
04277. void XayDung(int [],int,int [],int&);
```

– Định nghĩa hàm.

```
04278. void XayDung(int a[], int n, int b[], int &k)
04279. {
04280.    k = 0;
04281.    b[k++] = a[0];
04282.    for (int i = 1; i <= n - 1; i++)
04283.    {
04284.       if(a[i]%2==0 && b[k-1]%2==0)
04285.          b[k++] = 1;
04286.       else
04287.          if(a[i]%2!=0 && b[k-1]%2!=0)
04288.             b[k++] = 0;
04289.       b[k++] = a[i];
04290.    }
04291. }
```

01.18.08 Kỹ thuật Sắp xếp

Bài 137.(Hạt nhân) Hãy sắp xếp các giá trị trong mảng các số thực tăng dần.

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Kết quả.

11	12	19	23	29	62	76	88	98	99

– Khai báo hàm.

```
04292. void HoanVi(float &, float &);
04293. void SapTang(float [],int);
```

– Định nghĩa hàm hoán vị.

```
04294. void HoanVi(float &x, float &y)
04295. {
04296.    float temp = x;
04297.    x = y;
04298.    y = temp;
04299. }
```

– Định nghĩa hàm hàm sắp tăng.

```
04300. void SapTang(float a[], int n)
04301. {
04302.    for (int i = 0; i <= n - 2; i++)
04303.       for (int j = i + 1; j <= n - 1; j++)
04304.          if (a[i] > a[j])
04305.             HoanVi(a[i],a[j]);
```

```
04306. }
```

Bài 138.(Hạt nhân) Hãy sắp xếp mảng các số nguyên giảm dần.

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Kết quả.

99	98	88	76	62	29	23	19	12	11

– Khai báo hàm.

```
04307. void HoanVi(int &, int &);
04308. void SapGiam(int [],int);
```

– Định nghĩa hàm hoán vị.

```
04309. void HoanVi(int &x, int &y)
04310. {
04311.    int temp = x;
04312.    x = y;
04313.    y = temp;
04314. }
```

– Định nghĩa hàm sắp giảm.

```
04315. void SapGiam(int a[], int n)
04316. {
04317.    for (int i = 0; i <= n - 2; i++)
04318.       for (int j = i + 1; j <= n - 1; j++)
04319.          if (a[i] < a[j])
04320.             HoanVi(a[i],a[j]);
04321. }
```

Bài 139.(Hạt nhân) Hãy sắp xếp các giá trị tại các vị trí lẻ trong mảng các số nguyên tăng dần các giá trị khác giữ nguyên giá trị và vị trí.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	98	88	11

– Các vị trí lẻ trong mảng.

19	29	62	76	99	23	12	98	88	11

– Kết quả.

19	11	62	23	99	29	12	76	88	98

– Khai báo hàm.

```
04322. void ViTriLeTang(int [],int);
```

– Định nghĩa hàm sắp các giá trị tại các vị trí lẻ tăng dần.

```
04323. void ViTriLeTang(int a[], int n)
04324. {
04325.    for (int i = 0; i <= n - 2; i++)
```

```
04326.        for (int j = i + 1; j <= n - 1; j++)
04327.           if (i%2!=0 && j%2!=0 && a[i] > a[j])
04328.              HoanVi(a[i],a[j]);
04329.  }
```

Bài 140.(Hạt nhân) Hãy sắp xếp các số nguyên tố trong mảng các số nguyên tăng dần các giá trị khác giữ nguyên giá trị và vị trí.

– Ví dụ:

19	29	62	7	99	23	12	98	88	11

– Các số nguyên tố trong mảng.

19	29	62	7	99	23	12	98	88	11

– Kết quả.

7	11	62	19	99	23	12	98	88	29

– Khai báo hàm.

```
04330.  bool ktNguyenTo(int);
04331.  void NguyenToTang(int [],int);
```

– Định nghĩa hàm kiểm tra nguyên tố.

```
04332.  bool ktNguyenTo(int k)
04333.  {
04334.     int dem = 0;
04335.     for(int i=1; i<=k; i++)
04336.        if(k%i==0)
04337.           dem++;
04338.     if(dem==2)
04339.        return true;
04340.     return false;
04341.  }
```

– Định nghĩa hàm sắp các số nguyên tố tăng dần.

```
04342.  void NguyenToTang(int a[], int n)
04343.  {
04344.     for (int i = 0; i <= n - 2; i++)
04345.        for (int j = i + 1; j <= n - 1; j++)
04346.           if(ktNguyenTo(a[i]) &&
04347.               ktNguyenTo(a[j])&& a[i] > a[j])
04348.              HoanVi(a[i],a[j]);
04349.  }
```

Bài 141.Hãy sắp xếp các số hoàn thiện trong mảng các số nguyên giảm dần các giá trị khác giữ nguyên giá trị và vị trí.

– Ví dụ:

97	8128	62	76	28	6	496	98	88	18

– Các số hoàn thiện có trong mảng.

97	8128	62	76	28	6	496	98	88	18

– Kết quả:

97	8128	62	76	496	28	6	98	88	18

– Khai báo hàm.

```
04350.  bool ktHoanThien(int);
04351.  void HoanThienGiam(int [],int);
```

– Định nghĩa hàm kiểm tra hoàn thiện.

```
04352.  bool ktHoanThien(int k)
04353.  {
04354.      int s = 0;
04355.      for(int i=1; i<k; i++)
04356.          if(k%i==0)
04357.              s = s + i;
04358.      if(s==k)
04359.          return true;
04360.      return false;
04361.  }
```

– Định nghĩa hàm sắp các số hoàn thiện giảm dần.

```
04362.  void HoanThienGiam(int a[], int n)
04363.  {
04364.     for (int i = 0; i <= n - 2; i++)
04365.        for (int j = i + 1; j <= n - 1; j++)
04366.           if (ktHoanThien(a[i]) &&
04367.              ktHoanThien(a[j]) && a[i] < a[j])
04368.              HoanVi(a[i],a[j]);
04369.  }
```

Bài 142.Hãy sắp xếp các số dương trong mảng các số thực tăng dần các số âm giữ nguyên vị trí của chúng trong mảng.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.98	8.8	-1.1

– Các số dương trong mảng.

0	1	2	3	4	5	6	7	8	9
1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.98	8.8	-1.1

– Kết quả:

0	1	2	3	4	5	6	7	8	9
1.9	2.9	-0.6	8.8	-9.9	-23	12.1	-0.98	76	-1.1

– Khai báo hàm.

```
04370.  void DuongTang(float [],int);
```

– Định nghĩa hàm sắp các số dương tăng dần.

```
04371. void DuongTang(float a[], int n)
04372. {
04373.    for (int i=0; i<=n-2; i++)
04374.       for (int j=i+1; j<=n-1; j++)
04375.          if (a[i]>0 && a[j]>0 && a[i] > a[j])
04376.             HoanVi(a[i],a[j]);
04377. }
```

Bài 143.(*) Cho hai mảng a, b. Hãy cho biết mảng b có phải là hoán vị của mảng a hay không?

– Ví dụ 01:

+ Mảng a:

19	29	62	76	99	23	12	98	88	11

+ Mảng b.

12	62	23	19	99	11	29	88	76	98

+ Kết quả: $+1$.

– Ví dụ 02:

+ Mảng a:

19	29	62	76	89	23	12	98	88	11

+ Mảng b.

12	62	23	19	99	11	29	88	76	98

+ Kết quả: 0.

– Khai báo hàm.

```
04378. void SapTang(int [], int);
04379. int ktHoanVi(int [],int,int [],int);
```

– Định nghĩa hàm sắp mảng một chiều các số thực tăng dần.

```
04380. void SapTang(int a[], int n)
04381. {
04382.    for (int i=0; i<=n-2; i++)
04383.       for (int j=i+1; j<=n-1; j++)
04384.          if (a[i] > a[j])
04385.             HoanVi(a[i],a[j]);
04386. }
```

– Định nghĩa hàm kiểm tra hai mảng có là hoán vị của nhau hay không?

```
04387. int ktHoanVi(int a[],int n,int b[],int m)
04388. {
04389.    if (m != n)
04390.       return 0;
04391.    SapTang(a, n);
```

```
04392.     SapTang(b, m);
04393.
04394.     int flag = 1;
04395.     for (int i=0; i<n; i++)
04396.        if (a[i] != b[i])
04397.           flag = 0;
04398.     return flag;
04399.  }
```

Bài 144.(Hạt nhân) Hãy sắp xếp các số chẵn trong mảng các số nguyên tăng dần, các số lẻ cũng tăng dần. Vị trí tương đối giữa các số chẵn và số lẻ không đổi.

– Ví dụ:

91	29	62	76	99	34	23	12	98	88	11

– Các giá trị chẵn lẻ trong mảng.

91	29	62	76	99	34	23	12	98	88	11

– Kết quả:

11	23	12	34	29	62	91	76	88	98	99

– Khai báo hàm.

```
04400.  void ChanTang(int [], int);
04401.  void LeTang(int [], int);
04402.  void ChanTangLeTang(int [],int);
```

– Định nghĩa hàm sắp các giá trị chẵn tăng dần.

```
04403.  void ChanTang(int a[], int n)
04404.  {
04405.     for (int i = 0; i <= n - 2; i++)
04406.        for (int j = i + 1; j <= n - 1; j++)
04407.           if (!a[i]%2 && !a[j]%2 && a[i]> a[j])
04408.              HoanVi(a[i],a[j]);
04409.  }
```

– Định nghĩa hàm sắp các giá trị lẻ tăng dần.

```
04410.  void LeTang(int a[], int n)
04411.  {
04412.     for (int i = 0; i <= n - 2; i++)
04413.        for (int j = i + 1; j <= n - 1; j++)
04414.           if (a[i]%2 && a[j]%2 && a[i] > a[j])
04415.              HoanVi(a[i],a[j]);
04416.  }
```

– Định nghĩa hàm sắp các giá trị chẵn tăng dần, các giá trị lẻ cũng tăng dần.

```
04417.  void ChanTangLeTang(int a[], int n)
```

```
04418. {
04419.     ChanTang(a, n);
04420.     LeTang(a, n);
04421. }
```

Bài 145.Hãy sắp xếp các số dương trong mảng các số thực tăng dần, các số âm giảm dần. Vị trí tương đối giữa các số âm và số dương không đổi.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

- Các giá trị âm dương trong mảng.

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

- Kết quả:

1.9	2.9	-0.6	8.8	-0.9	-1.1	12.1	-9.9	76	-23

- Cách làm
  + Các số dương trong mảng.

0	1	2	3	4	5	6	7	8	9
1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

  + Sắp các số dương tăng

1.9	2.9	-0.6	8.8	-9.9	-23	12.1	-0.9	76	-1.1

  + Các số âm trong mảng.

1.9	2.9	-0.6	8.8	-9.9	-23	12.1	-0.9	76	-1.1

  + Sắp các số âm giảm dần

1.9	2.9	-0.6	8.8	-0.9	-1.1	12.1	-9.9	76	-23

- Khai báo hàm.

```
04422. void DuongTang(float [], int);
04423. void AmGiam(float [], int);
04424. void DuongTangAmGiam(float [],int);
```

- Định nghĩa hàm sắp các giá trị dương tăng dần.

```
04425. void DuongTang(float a[], int n)
04426. {
04427.    for (int i = 0; i <= n - 2; i++)
04428.       for (int j = i + 1; j <= n - 1; j++)
04429.          if (a[i]>0 && a[j]>0 && a[i]>a[j])
04430.             HoanVi(a[i],a[j]);
04431. }
```

- Định nghĩa hàm sắp các giá trị âm giảm dần.

```
04432. void AmGiam(float a[], int n)
04433. {
04434.    for (int i = 0; i <= n - 2; i++)
```

```
04435.       for (int j = i + 1; j <= n - 1; j++)
04436.          if (a[i]<0 && a[j]<0 && a[i] < a[j])
04437.             HoanVi(a[i],a[j]);
04438. }
```

– Định nghĩa hàm sắp các giá trị dương tăng dần, các giá trị âm giảm dần.

```
04439. void DuongTangAmGiam(float a[], int n)
04440. {
04441.    DuongTang(a, n);
04442.    AmGiam(a, n);
04443. }
```

Bài 146. (*) Hãy trộn hai mảng tăng dần lại thành 1 mảng được sắp thứ tự tăng dần.

– Ví dụ:

+ Mảng a.

19	29	62	76	99

+ Mảng b.

11	29	76	88

+ Kết quả.

11	19	29	29	62	76	76	88	99

– Khai báo hàm.

```
04444. void Tron(int [],int,int [],int,int [],int&);
```

– Định nghĩa hàm.

```
04445. void Tron(int a[], int n, int b[], int m,
04446.             int c[], int &p)
04447. {
04448.    int i = 0;
04449.    int j = 0;
04450.    p = 0;
04451.    while (!(i >= n && j >= m))
04452.    {
04453.       if ((i<n && j<m && a[i]<b[j]) || (j>=m))
04454.          c[p++] = a[i++];
04455.       else
04456.          c[p++] = b[j++];
04457.    }
04458. }
```

Bài 147. (*) Cho hai mảng tăng dần. Hãy trộn hai mảng lại thành một mảng được sắp thứ tự giảm dần.

– Ví dụ:
  + Mảng a.

19	29	62	76	99

  + Mảng b.

11	29	76	88

  + Kết quả.

99	88	76	76	62	29	29	19	11

– Khai báo hàm.

```
04459. void Tron(int [],int,int [],int,int [],int&);
```

– Định nghĩa hàm.

```
04460. void Tron(int a[],int n,int b[],int m,
04461.              int c[],int &p)
04462. {
04463.    int i = 0;
04464.    int j = 0;
04465.    p = 0;
04466.    while (!(i >= n && j >= m))
04467.    {
04468.       if ((i<n && j<m && a[i]>b[j]) || (j>=m))
04469.          c[p++] = a[i++];
04470.       else
04471.          c[p++] = b[j++];
04472.    }
04473. }
```

01.18.09 Kỹ thuật Xử lý trên mảng

Bài 148.(Hạt nhân) Hãy biến đổi mảng các số thực bằng cách thay các giá trị lớn nhất bằng giá trị nhỏ nhất và ngược lại.

– Ví dụ:

19	99	11	76	99	11	12	98	88	11

– Giá trị lớn nhất, giá trị nhỏ nhất.

19	99	11	76	99	11	12	98	88	11

– Kết quả.

19	11	99	76	11	99	12	98	88	99

– Khai báo hàm.

```
04474. float LonNhat(float [], int);
04475. float NhoNhat(float [], int);
04476. void BienDoi(float [], int);
```

– Định nghĩa hàm tìm giá trị lớn nhất

```
04477. float LonNhat(float a[], int n)
```

```
04478. {
04479.    float lc = a[0];
04480.    for (int i=0; i<n; i++)
04481.       if (a[i] > lc)
04482.          lc = a[i];
04483.    return lc;
04484. }
```

– Định nghĩa hàm tìm giá trị nhỏ nhất

```
04485. float NhoNhat(float a[], int n)
04486. {
04487.    float lc = a[0];
04488.    for (int i=0; i<n; i++)
04489.       if (a[i] < lc)
04490.          lc = a[i];
04491.    return lc;
04492. }
```

– Định nghĩa hàm biến đổi mảng các số thực

```
04493. void BienDoi(float a[], int n)
04494. {
04495.    float ln = LonNhat(a, n);
04496.    float nn = NhoNhat(a, n);
04497.    for (int i=0; i<n; i++)
04498.    {
04499.       if (a[i] == ln)
04500.          a[i] = nn;
04501.       else
04502.          if (a[i] == nn)
04503.             a[i] = ln;
04504.    }
04505. }
```

– Cách làm của người thông minh.

```
04506. void BienDoi(float a[], int n)
04507. {
04508.    float ln = LonNhat(a, n);
04509.    float nn = NhoNhat(a, n);
04510.    for (int i=0; i<n; i++)
04511.       if (a[i] == ln || a[i] == nn)
04512.             a[i] = ln + nn – a[i];
04513. }
```

Bài 149.(Hạt nhân) Hãy nguyên hóa mảng bằng cách thay tất cả các phần tử trong mảng bằng số nguyên gần nó nhất.

– Ví dụ:

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

– Các giá trị âm và dương trong mảng.

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

– Kết quả:

2	3	-1	76	-10	-23	12	-1	9	-1

– Khai báo hàm.

```
04514.  void NguyenHoa(float [],int);
```

– Định nghĩa hàm nguyên hóa mảng

```
04515.  void NguyenHoa(float a[], int n)
04516.  {
04517.     for (int i=0; i<n; i++)
04518.     {
04519.        if (a[i] > 0)
04520.           a[i] = int(a[i] + 0.5);
04521.        else
04522.           a[i] = int(a[i] - 0.5);
04523.     }
04524.  }
```

Bài 150.(Hạt nhân) Hãy đưa số một về đầu mảng.

– Ví dụ:

19	1	62	1	99	23	1	98	1	1

– Các giá trị một trong mảng.

19	1	62	1	99	23	1	98	1	1

– Kết quả.

1	1	1	1	1	23	62	98	19	99

– Khai báo hàm.

```
04525.  void MotVeDau(float [],int);
```

– Định nghĩa hàm đưa số một về đầu mảng

```
04526.  void MotVeDau(float a[], int n)
04527.  {
04528.     int vt = 0;
04529.     for (int i=0; i<n; i++)
04530.        if (a[i] == 1)
04531.        {
04532.           float temp = a[i];
04533.           a[i] = a[vt];
04534.           a[vt] = temp;
04535.           vt++;
04536.        }
```

```
04537. }
```

Bài 151.Hãy đưa các số chia hết cho 3 về đầu mảng.

- Ví dụ:

19	29	66	76	99	23	12	98	88	33

- Các giá trị chia hết cho 3.

19	29	**66**	76	**99**	23	**12**	98	88	**33**

- Kết quả.

66	**99**	**12**	**33**	29	23	19	98	88	76

- Khai báo hàm.

```
04538. void DuaVeDau(int [],int);
```

- Định nghĩa hàm đưa các số chia hết cho 3 về đầu mảng

```
04539. void DuaVeDau(int a[], int n)
04540. {
04541.    int vt = 0;
04542.    for (int i=0; i<n; i++)
04543.       if (a[i] % 3 == 0)
04544.       {
04545.          int temp = a[i];
04546.          a[i] = a[vt];
04547.          a[vt] = temp;
04548.          vt++;
04549.       }
04550. }
```

Bài 152.(Hạt nhân) Hãy đưa các số nguyên tố về cuối mảng.

- Ví dụ:

19	29	62	76	99	23	12	98	88	11

- Các số nguyên tố trong mảng.

19	**29**	62	76	99	**23**	12	98	88	**11**

- Kết quả.

12	98	62	76	99	88	**19**	**29**	**23**	**11**

- Khai báo hàm.

```
04551. bool ktNguyenTo(int);
04552. void DuaVeCuoi(int [],int);
```

- Định nghĩa hàm kiểm tra nguyên tố.

```
04553. bool ktNguyenTo(int k)
04554. {
04555.    int dem = 0;
04556.    for(int i=1; i<=k; i++)
04557.       if(k%i==0)
```

```
04558.            dem++;
04559.      if(dem==2)
04560.         return true;
04561.      return false;
04562.  }
```

– Định nghĩa hàm đưa các số nguyên tố về cuối mảng

```
04563.  void DuaVeCuoi(int a[], int n)
04564.  {
04565.      int vt = n - 1;
04566.      for (int i = n - 1; i >= 0; i--)
04567.         if (ktNguyenTo(a[i]))
04568.         {
04569.            int temp = a[i];
04570.            a[i] = a[vt];
04571.            a[vt] = temp;
04572.            vt--;
04573.         }
04574.  }
```

Bài 153.Hãy đưa các số chẵn trong mảng về đầu mảng, số lẻ về cuối mảng và các phần tử 0 nằm ở giữa.

– Ví dụ:

91	0	62	76	99	34	23	12	98	88	0

– Các giá trị lẻ và chẵn khác không trong mảng.

91	0	62	76	99	34	23	12	98	88	0

– Kết quả:

62	76	34	12	98	88	0	0	91	99	23

– Khai báo hàm.

```
04575.  void ChanVeDau(int [], int);
04576.  void LeVeCuoi(int [], int);
04577.  void ChanDauLeCuoi(int [],int);
```

– Định nghĩa hàm đưa các số chẵn khác không về đầu mảng

```
04578.  void ChanVeDau(int a[], int n)
04579.  {
04580.      int vt = 0;
04581.      for (int i=0; i<n; i++)
04582.         if (a[i] % 2 == 0 && a[i] != 0)
04583.         {
04584.            int temp = a[i];
04585.            a[i] = a[vt];
04586.            a[vt] = temp;
```

```
04587.          vt++;
04588.       }
04589. }
```

– Định nghĩa hàm đưa các số lẻ về cuối mảng

```
04590. void LeVeCuoi(int a[], int n)
04591. {
04592.    int vt = n - 1;
04593.    for (int i = n - 1; i >= 0; i--)
04594.       if (a[i] % 2 != 0)
04595.       {
04596.          int temp = a[i];
04597.          a[i] = a[vt];
04598.          a[vt] = temp;
04599.          vt--;
04600.       }
04601. }
```

– Định nghĩa hàm

```
04602. void ChanDauLeCuoi(int a[], int n)
04603. {
04604.    ChanVeDau(a, n);
04605.    LeVeCuoi(a, n);
04606. }
```

Bài 154.(Hạt nhân) Hãy đảo ngược mảng ban đầu (283).

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Kết quả.

11	88	98	12	23	99	76	62	29	19

– Khai báo hàm.

```
04607. void DaoMang(float [],int);
```

– Định nghĩa hàm đảo ngược mảng ban đầu.

```
04608. void DaoMang(float a[], int n)
04609. {
04610.    int d = 0;
04611.    int c = n - 1;
04612.    while (d < c)
04613.    {
04614.       float temp = a[d];
04615.       a[d] = a[c];
04616.       a[c] = temp;
04617.       d++;
```

```
04618.        c--;
04619.     }
04620.  }
```

Bài 155.Hãy đảo ngược thứ tự các giá trị chẵn có trong mảng các số nguyên, các giá trị lẻ vẫn giữ nguyên giá trị và vị trí trong mảng.

- Ví dụ:

91	29	62	76	99	34	23	12	98	88	11

- Các giá trị chẵn trong mảng.

91	29	62	76	99	34	23	12	98	88	11

- Kết quả.

91	29	88	98	99	12	23	34	76	62	11

- Khai báo hàm.

```
04621.  void DaoMang(int [], int);
04622.  void DaoChan(int [],int);
```

- Định nghĩa hàm đảo ngược mảng ban đầu

```
04623.  void DaoMang(int a[], int n)
04624.  {
04625.     int d = 0;
04626.     int c = n - 1;
04627.     while (d < c)
04628.     {
04629.        int temp = a[d];
04630.        a[d] = a[c];
04631.        a[c] = temp;
04632.        d++;
04633.        c--;
04634.     }
04635.  }
```

- Định nghĩa hàm đảo ngược thứ tự các giá trị chẵn có trong mảng các số nguyên.

```
04636.  void DaoChan(int a[], int n)
04637.  {
04638.     int b[100];
04639.     int k;
04640.     k = 0;
04641.     for (int i=0; i<n; i++)
04642.        if (a[i] % 2 == 0)
04643.           b[k++] = a[i];
04644.     DaoMang(b, k);
04645.     k = 0;
```

```
04646.     for (int i=0; i<n; i++)
04647.        if (a[i] % 2 == 0)
04648.           a[i] = b[k++];
04649.  }
```

Bài 156.Hãy đảo ngược thứ tự các số dương có trong mảng một chiều các số thực, các giá trị âm vẫn giữ nguyên giá trị và vị trí trong mảng.

- Ví dụ:

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

- Các số dương trong mảng.

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

- Đảo các số dương trong mảng.

8.8	12.1	-0.6	76	-9.9	-23	2.9	-0.9	1.9	-1.1

- Kết quả.

8.8	12.1	-0.6	76	-9.9	-23	2.9	-0.9	1.9	-1.1

- Khai báo hàm.

```
04650.  void DaoMang(float [], int);
04651.  void DaoDuong(float [],int);
```

- Định nghĩa hàm đảo mảng.

```
04652.  void DaoMang(float a[], int n)
04653.  {
04654.     int d = 0;
04655.     int c = n - 1;
04656.     while (d < c)
04657.     {
04658.        float temp = a[d];
04659.        a[d] = a[c];
04660.        a[c] = temp;
04661.        d++;
04662.        c--;
04663.     }
04664.  }
```

- Định nghĩa hàm đảo các giá trị dương trong mảng.

```
04665.  void DaoDuong(float a[], int n)
04666.  {
04667.     float b[100];
04668.     int k;
04669.     k = 0;
04670.     for (int i=0; i<n; i++)
04671.        if (a[i] > 0)
```

```
04672.           b[k++] = a[i];
04673.     DaoMang(b, k);
04674.     k = 0;
04675.     for (int i=0; i<n; i++)
04676.        if (a[i] > 0)
04677.           a[i] = b[k++];
04678.  }
```

Bài 157.Hãy đảo ngược thứ tự các số chẵn và các số lẻ trong mảng mà vẫn giữ nguyên vị trí tương đối của chúng.

– Ví dụ:

91	29	62	76	99	34	23	12	98	88	11

– Các giá trị chẵn trong mảng.

91	29	62	76	99	34	23	12	98	88	11

– Kết quả.

11	23	88	98	99	12	29	34	76	62	91

– Khai báo hàm.

```
04679.  void DaoMang(int [], int);
04680.  void DaoChan(int [], int);
04681.  void DaoLe(int [], int);
04682.  void DaoChanLe(int [],int);
```

– Định nghĩa hàm đảo mảng.

```
04683.  void DaoMang(int a[], int n)
04684.  {
04685.     int d = 0;
04686.     int c = n - 1;
04687.     while (d < c)
04688.     {
04689.        int temp = a[d];
04690.        a[d] = a[c];
04691.        a[c] = temp;
04692.        d++;
04693.        c--;
04694.     }
04695.  }
```

– Định nghĩa hàm đảo các giá trị chẵn trong mảng.

```
04696.  void DaoChan(int a[], int n)
04697.  {
04698.     int b[100];
04699.     int k;
04700.     k = 0;
```

```
04701.     for (int i=0; i<n; i++)
04702.        if (a[i] % 2 == 0)
04703.           b[k++] = a[i];
04704.     DaoMang(b, k);
04705.     k = 0;
04706.     for (int i=0; i<n; i++)
04707.        if (a[i] % 2 == 0)
04708.           a[i] = b[k++];
04709.  }
```

– Định nghĩa hàm đảo các giá trị lẻ trong mảng.

```
04710.  void DaoLe(int a[], int n)
04711.  {
04712.     int b[100];
04713.     int k;
04714.     k = 0;
04715.     for (int i=0; i<n; i++)
04716.        if (a[i] % 2 != 0)
04717.           b[k++] = a[i];
04718.     DaoMang(b, k);
04719.     k = 0;
04720.     for (int i=0; i<n; i++)
04721.        if (a[i] % 2 == 0)
04722.           a[i] = b[k++];
04723.  }
```

– Định nghĩa hàm đảo ngược thứ tự các số chẵn và các số lẻ trong mảng mà vẫn giữ nguyên vị trí tương đối của chúng.

```
04724.  void DaoChanLe(int a[],int n)
04725.  {
04726.     DaoChan(a,n);
04727.     DaoLe(a,n);
04728.  }
```

Bài 158.Hãy "dịch trái xoay vòng" các phần tử trong mảng.

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Kết quả.

29	62	76	99	23	12	98	88	11	19

– Khai báo hàm.

```
04729.  void DichTrai(float [],int);
```

– Định nghĩa hàm.

```
04730.  void DichTrai(float a[], int n)
```

```
04731. {
04732.    float temp = a[0];
04733.    for (int i = 0; i <= n - 2; i++)
04734.       a[i] = a[i + 1];
04735.    a[n - 1] = temp;
04736. }
```

Bài 159.Hãy "dịch phải xoay vòng" các phần tử trong mảng.

– Ví dụ:

19	29	62	76	99	23	12	98	88	11

– Kết quả.

11	19	29	62	76	99	23	12	98	88

– Khai báo hàm.

```
04737. void DichPhai(float [],int);
```

– Định nghĩa hàm.

```
04738. void DichPhai(float a[], int n)
04739. {
04740.    float temp = a[n - 1];
04741.    for (int i = n - 1; i >= 1; i--)
04742.       a[i] = a[i - 1];
04743.    a[0] = temp;
04744. }
```

01.18.10 Kỹ thuật Xóa

Bài 160.(Hạt nhân) Xóa các phần tử có chỉ số k trong mảng một chiều các số thực.

– Ví dụ: hãy xóa phần tử tại vị trí $k = 6$.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	223	232	227	989	87	53

– Mảng sau khi xóa:

19	29	7	761	223	232	989	87	53

– Ý tưởng.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	223	232	227	989	87	53

– Khai báo hàm.

```
04745. void XoaViTri(float [], int&, int);
```

– Định nghĩa hàm xóa phần tử tại vị trí k trong mảng.

```
04746. void XoaViTri(float a[], int &n, int k)
04747. {
04748.    for (int i = k; i <= n - 2; i++)
```

```
04749.        a[i] = a[i + 1];
04750.     n--;
04751.  }
```

Bài 161.(Hạt nhân) Hãy xóa tất cả số âm trong mảng các số thực.

- Ví dụ:

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

- Các số âm trong mảng

1.9	2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

- Kết quả.

1.9	2.9	76	12.1	8.8

- Khai báo hàm.

```
04752.  void XoaViTri(float [], int &, int);
04753.  void XoaAm(float [], int &);
```

- Định nghĩa hàm xóa phần tử tại vị trí *k* trong mảng.

```
04754.  void XoaViTri(float a[], int &n, int k)
04755.  {
04756.     for (int i = k; i <= n - 2; i++)
04757.        a[i] = a[i + 1];
04758.     n--;
04759.  }
```

- Định nghĩa hàm xóa các giá trị âm trong mảng.

```
04760.  void XoaAm(float a[], int &n)
04761.  {
04762.     for (int i = n - 1; i >= 0; i--)
04763.        if (a[i] < 0)
04764.           XoaViTri(a, n, i);
04765.  }
```

- Giải thích: tại sao khi xóa các giá trị âm ta không thể duyệt mảng từ đầu đến cuối.
  + Cho mảng.

0	1	2	3	4	5	6	7	8	9
-1.9	-2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

  + Xét phần tử có chỉ số 0.

0	1	2	3	4	5	6	7	8	9
-1.9	-2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

-2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

  + Xét phần tử có chỉ số 1.

0	1	2	3	4	5	6	7	8
-2.9	-0.6	76	-9.9	-23	12.1	-0.9	8.8	-1.1

-2.9	76	-9.9	-23	12.1	-0.9	8.8	-1.1

+ Dễ thấy ta sẽ xóa sót phần tử -2.9

Bài 162.Hãy xóa tất cả số chẵn trong mảng các số nguyên.

– Ví dụ:

91	29	62	76	99	34	23	12	98	88	11

– Các giá trị chẵn trong mảng.

91	29	62	76	99	34	23	12	98	88	11

– Kết quả.

91	29	99	23	11

– Khai báo hàm.

```
04766.  void XoaViTri(int [], int &, int);
04767.  void XoaChan(int [],int &);
```

– Định nghĩa hàm xóa phần tử tại vị trí *k* trong mảng.

```
04768.  void XoaViTri(int a[], int &n, int k)
04769.  {
04770.     for (int i = k; i <= n - 2; i++)
04771.        a[i] = a[i + 1];
04772.     n--;
04773.  }
```

– Định nghĩa hàm xóa các giá trị chẵn.

```
04774.  void XoaChan(int a[], int &n)
04775.  {
04776.     for (int i = n - 1; i >= 0; i--)
04777.        if (a[i]%2==0)
04778.           XoaViTri(a, n, i);
04779.  }
```

Bài 163.Hãy xóa tất cả "số chính phương" trong mảng một chiều các số nguyên.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	81	-62	76	-99	25	12	-98	16	11

– Các số chính phương.

0	1	2	3	4	5	6	7	8	9
19	81	-62	76	-99	25	12	-98	16	11

– Kết quả:

0	1	2	3	4	5	6
19	-62	76	-99	12	-98	11

– Khai báo hàm.

```
04780.  void XoaViTri(int [], int& ,int);
04781.  bool ktChinhPhuong(int);
04782.  void XoaChinhPhuong(int [],int&);
```

– Định nghĩa hàm xóa phần tử tại vị trí *k* trong mảng.

```
04783.  void XoaViTri(int a[], int &n, int k)
04784.  {
04785.     for (int i = k; i <= n - 2; i++)
04786.        a[i] = a[i + 1];
04787.     n--;
04788.  }
```

– Định nghĩa hàm kiểm tra số nguyên có phải là số chính phương hay không.

```
04789.  bool ktChinhPhuong(int n)
04790.  {
04791.     for (int i = 1; (i * i) <= n; i++)
04792.        if (i * i == n)
04793.           return true;
04794.     return false;
04795.  }
```

– Định nghĩa hàm xóa các số chính phương trong mảng.

```
04796.  void XoaChinhPhuong(int a[], int &n)
04797.  {
04798.     for (int i = n - 1; i >= 0; i--)
04799.        if (ktChinhPhuong(a[i]))
04800.           XoaViTri(a, n, i);
04801.  }
```

Bài 164.Hãy xóa tất cả các "số nguyên tố" trong mảng số nguyên.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
20	11	-62	76	-99	131	12	-98	101	10

– Các số chính phương.

0	1	2	3	4	5	6	7	8	9
20	11	-62	76	-99	131	12	-98	101	10

– Kết quả:

0	1	2	3	4	5	6
20	-62	76	-99	12	-98	10

– Khai báo hàm.

```
04802.  void XoaViTri(int [], int&, int);
04803.  bool ktNguyenTo(int);
04804.  void XoaNguyenTo(int [],int&);
```

– Định nghĩa hàm kiểm tra nguyên tố.

```
04805. bool ktNguyenTo(int k)
04806. {
04807.    int dem = 0;
04808.    for(int i=1; i<=k; i++)
04809.       if(k%i==0)
04810.          dem++;
04811.    if(dem==2)
04812.       return true;
04813.    return false;
04814. }
```

– Định nghĩa hàm xóa phần tử tại vị trí *k* trong mảng.

```
04815. void XoaViTri(int a[], int &n, int k)
04816. {
04817.    for (int i = k; i <= n - 2; i++)
04818.       a[i] = a[i + 1];
04819.    n--;
04820. }
```

– Định nghĩa hàm xóa các số nguyên tố trong mảng.

```
04821. void XoaNguyenTo(int a[], int &n)
04822. {
04823.    for (int i = n - 1; i >= 0; i--)
04824.       if (ktNguyenTo(a[i]))
04825.          XoaViTri(a, n, i);
04826. }
```

Bài 165.Hãy xóa tất cả số lớn nhất trong mảng các số thực.

– Ví dụ:

19	29	62	76	99	23	12	99	88	11

– Giá trị lớn nhất trong mảng.

19	29	62	76	99	23	12	99	88	11

– Kết quả.

19	29	62	76	23	12	88	11

– Khai báo hàm.

```
04827. void XoaViTri(float [], int& , int);
04828. float LonNhat(float [], int);
04829. void XoaLonNhat(float [],int&);
```

– Định nghĩa hàm xóa phần tử tại vị trí *k* trong mảng..

```
04830. void XoaViTri(float a[], int &n, int k)
04831. {
04832.    for (int i = k; i <= n - 2; i++)
```

```
04833.        a[i] = a[i + 1];
04834.     n--;
04835.  }
```

– Định nghĩa hàm tìm giá trị lớn nhất.

```
04836.  float LonNhat(float a[], int n)
04837.  {
04838.     int lc = a[0];
04839.     for (int i=0; i<n; i++)
04840.        if (a[i] > lc)
04841.           lc = a[i];
04842.     return lc;
04843.  }
```

– Định nghĩa hàm xóa các giá trị lớn nhất trong mảng.

```
04844.  void XoaLonNhat(float a[], int &n)
04845.  {
04846.     float ln = LonNhat(a, n);
04847.     for (int i = n - 1; i >= 0; i--)
04848.        if (a[i] == ln)
04849.           XoaViTri(a, n, i);
04850.  }
```

Bài 166.Hãy xóa tất cả các phần tử có giá trị trùng với x.

– Ví dụ: cho mảng sau và $x = 19$.

19	29	62	76	19	23	12	98	19	11

– Các phần tử có giá trị bằng 19.

19	29	62	76	19	23	12	98	19	11

– Kết quả.

29	62	76	23	12	98	11

– Khai báo hàm.

```
04851.  void XoaViTri(float [], int&, int);
04852.  void XoaTrungX(float [], int&, float);
```

– Định nghĩa hàm xóa phần tử tại vị trí k trong mảng.

```
04853.  void XoaViTri(float a[], int &n, int k)
04854.  {
04855.     for (int i = k; i <= n - 2; i++)
04856.        a[i] = a[i + 1];
04857.     n--;
04858.  }
```

– Định nghĩa hàm xóa các phần tử trùng với x.

```
04859.  void XoaTrungX(float a[], int &n, float x)
04860.  {
```

```
04861.     for (int i = n - 1; i >= 0; i--)
04862.        if (a[i] == x)
04863.           XoaViTri(a, n, i);
04864.  }
```

Bài 167.(*) Hãy xóa tất cả các phần tử trùng nhau trong mảng và chỉ giữ lại duy nhất một phần tử.

- Ví dụ:

19	29	19	76	88	29	12	19	11	19

- Các giá trị có trong mảng.

19	29	19	76	88	29	12	19	11	19

- Kết quả.

19	29	76	88	12	11

- Khai báo hàm.

```
04865.  void XoaViTri(float [], int&, int);
04866.  void XoaTrung(float [],int&);
```

- Định nghĩa hàm xóa phần tử tại vị trí *k* trong mảng.

```
04867.  void XoaViTri(float a[], int &n, int k)
04868.  {
04869.     for (int i = k; i <= n - 2; i++)
04870.        a[i] = a[i + 1];
04871.     n--;
04872.  }
```

- Định nghĩa hàm xóa các phần tử trùng nhau trong mảng.

```
04873.  void XoaTrung(float a[], int &n)
04874.  {
04875.     for (int i = n - 1; i >= 0; i--)
04876.     {
04877.        int flag = 0;
04878.        for (int j = 0; j <= i - 1; j++)
04879.           if (a[j] == a[i])
04880.              flag = 1;
04881.        if (flag == 1)
04882.           XoaViTri(a, n, i);
04883.     }
04884.  }
```

Bài 168.(*) Hãy xóa tất cả các phần tử có tần suất xuất hiện trong mảng nhiều hơn một lần.

- Ví dụ:

19	29	19	76	88	29	12	19	11	19

– Các giá trị có trong mảng.

– Kết quả.

– Khai báo hàm.

```
04885. void XoaViTri(float [], int&, int);
04886. void XoaTrung(float [], int&, int);
04887. int TanSuat(float [], int , float);
04888. void XoaPhanTu(float [], int&);
```

– Định nghĩa hàm xóa phần tử tại vị trí *k* trong mảng.

```
04889. void XoaViTri(float a[], int &n, int k)
04890. {
04891.    for (int i = k; i <= n - 2; i++)
04892.       a[i] = a[i + 1];
04893.    n--;
04894. }
```

– Định nghĩa hàm xóa các phần tử trong mảng trùng với giá trị *x*.

```
04895. void XoaTrung(float a[], int &n, float x)
04896. {
04897.    for (int i = n - 1; i >= 0; i--)
04898.       if (a[i] == x)
04899.          XoaViTri(a, n, i);
04900. }
```

– Định nghĩa hàm đếm tần suất xuất hiện của giá trị *x* trong mảng.

```
04901. int TanSuat(float a[], int n, float x)
04902. {
04903.    int dem = 0;
04904.    for (int i=0; i<=n-1; i++)
04905.       if (a[i] == x)
04906.          dem++;
04907.    return dem;
04908. }
```

– Định nghĩa hàm xóa tất cả các phần tử có tần suất xuất hiện trong mảng nhiều hơn một lần.

```
04909. void XoaPhanTu (float a[], int &n)
04910. {
04911.    for (int i = n - 1; i >= 0; i--)
04912.       if (TanSuat(a,n,a[i]) > 1)
04913.          XoaTrung(a, n, a[i]);
04914. }
```

01.18.11 Kỹ thuật Thêm

Bài 169.(Hạt nhân) Hãy thêm một phần tử có giá trị x vào mảng tại vị trí k.

- Ví dụ: hãy thêm giá trị $x = 45$ tại vị trí $k = 6$.

0	1	2	3	4	5	6	7	8
19	29	7	761	2230	232	227	87	53

- Mảng sau khi thêm giá trị $x = 45$ tại vị trí $vt = 6$.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	2230	232	45	227	87	53

- Khai báo hàm.

```
04915. void ThemViTri(float [], int&, float, int);
```

- Định nghĩa hàm thêm giá trị x vào mảng tại vị trí k.

```
04916. void ThemViTri(   float a[],int &n,
04917.                   float x,int k)
04918. {
04919.    for (int i = n; i >= k + 1; i--)
04920.       a[i] = a[i - 1];
04921.    a[k] = x;
04922.    n++;
04923. }
```

Bài 170.(Hạt nhân) Hãy thêm một giá trị x vào trong mảng tăng mà vẫn giữ nguyên tính đơn điệu tăng của mảng.

- Ví dụ: cho mảng sau và $x = 23$.

0	1	2	3	4	5	6	7	8
11	12	19	29	62	76	88	98	99

- .

11	12	19	23	29	62	76	88	98	99

- Khai báo hàm.

```
04924. void ThemBaoToan(float [], int&, float);
```

- Định nghĩa hàm.

```
04925. void ThemBaoToan(float a[], int &n, float x)
04926. {
04927.    int i = n - 1;
04928.    while (i >= 0 && a[i] > x)
04929.    {
04930.       a[i + 1] = a[i];
04931.       i--;
04932.    }
04933.    a[i + 1] = x;
```

```
04934.     n++;
04935.  }
```

Bài 171.(Hạt nhân) Hãy viết hàm nhập mảng một chiều các số thực sao cho khi mảng nhập xong các giá trị trong mảng được sắp giảm dần và không sắp xếp các giá trị trong mảng sau khi nhập.

– Khai báo hàm.

```
04936.  void NhapGiam(float [],int&);
```

– Nhắc lại định nghĩa hàm mảng một chiều các số thực từ bàn phím.

```
04937.  void Nhap (float a[], int &n)
04938.  {
04939.     cout << "Nhap n : ";
04940.     cin >> n;
04941.     for (int i=0; i<n; i++)
04942.     {
04943.        cout << "Nhap a[” << i << “]: ";
04944.        cin >> a[i];
04945.     }
04946.  }
```

– Ý tưởng thô.

```
04947.  void Nhap (float a[], int &n)
04948.  {
04949.     cout << "Nhap n : ";
04950.     cin >> n;
04951.     for (int i=0; i<n; i++)
04952.     {
04953.        int x;
04954.        cout << "Nhap a[” << i << “]: ";
04955.        cin >> x;
04956.        Chèn x vào trong mảng có i phần tử đã
04957.        được sắp giảm để được mảng có i+1 phần
04958.        tử sắp giảm.
04959.     }
04960.  }
```

– Định nghĩa hàm (cách 01).

```
04961.  void Nhap (float a[], int &n)
04962.  {
04963.     cout << "Nhap n : ";
04964.     cin >> n;
04965.     for (int i=0; i<n; i++)
```

```
04966.     {
04967.        float x;
04968.        cout << "Nhap a[” << i << “]: ";
04969.        cin >> x;
04970.        ThemBaoToan(a,i,x);
04971.     }
04972.  }
```

– Định nghĩa hàm (cách 02).

```
04973.  void NhapGiam(float a[], int &n)
04974.  {
04975.     cout << "Nhap n : ";
04976.     cin >> n;
04977.     for (int i=0; i<n; i++)
04978.     {
04979.        float x;
04980.        cout << "Nhap a[” << i << “]: ";
04981.        cin >> x;
04982.        for (int j=i-1; j>=0 && a[j]<x; j--)
04983.           a[j + 1] = a[j];
04984.        a[j + 1] = x;
04985.     }
04986.  }
```

Bài 172.Hãy viết hàm nhập mảng một chiều các số thực sao cho khi mảng nhập xong các giá trị trong mảng được sắp tăng dần và không sắp xếp các giá trị trong mảng sau khi nhập.

– Khai báo hàm.

```
04987.  void NhapTang(float [],int&);
```

– Định nghĩa hàm.

```
04988.  void NhapTang(float a[], int &n)
04989.  {
04990.     cout << "Nhap n : ";
04991.     cin >> n;
04992.     for (int i=0; i<n; i++)
04993.     {
04994.        cout << "Nhap a[" << i << "]: ";
04995.        cin >> a[i];
04996.        int x = a[i];
04997.        int j;
04998.        for (j = i - 1; j >= 0 && a[j] > x; j--)
04999.           a[j + 1] = a[j];
05000.        a[j + 1] = x;
```

```
05001.     }
05002.  }
```

01.18.12 Kỹ thuật Xử lý Mảng con

Bài 173.(Hạt nhân) Định nghĩa hàm xuất mảng con có độ dài l bắt đầu tại vị trí vt trong mảng một chiều các số nguyên.

- Ví dụ: cho mảng ban đầu có 10 phần tử, xuất mảng con có độ dài là 4 bắt đầu tại vị trí $vt = 3$.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	257	232	227	988	87	53

- Mảng con có độ dài là 4 bắt đầu tại vị trí $vt = 3$.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	257	232	227	988	87	53

- Kết quả: $761, 257, 232, 227$.
- Các phần tử trong mảng con có độ dài l được đánh chỉ số nội bộ từ 0 tới $l - 1$.

0	1	2	3
761	257	232	227

- Khai báo hàm.

```
05003.  void XuatCon(int [],int,int,int);
```

- Định nghĩa hàm.

```
05004.  void XuatCon(int a[],int n,int vt,int l)
05005.  {
05006.     for(int i=0;i<=l-1;i++)
05007.        cout << setw(8) << a[vt+i];
05008.  }
```

Bài 174.(Hạt nhân) Định nghĩa hàm xuất tất cả các mảng con có độ dài l trong mảng một chiều các số nguyên.

- Ví dụ: cho mảng ban đầu có 10 phần tử, xuất tất cả mảng con có độ dài là 4.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	2230	232	227	9889	87	53

- Các mảng con có độ dài là 4.

0	1	2	3	4	5	6	7	8	9
19	29	7	761	2230	232	227	9889	87	53

19	29	7	761	2230	232	227	9889	87	53

19	29	7	761	2230	232	227	9889	87	53

19	29	7	761	2230	232	227	9889	87	53

19	29	7	761	2230	232	227	9889	87	53

19	29	7	761	2230	232	227	9889	87	53

19	29	7	761	2230	232	227	9889	87	53

- Kết quả:
  + 19 29 7 761,
  + 29 7 761 2230,
  + 7 761 2230 232,
  + 761 2230 232 227,
  + 2230 232 227 9889,
  + 232 227 9889 87,
  + 227 9889 87 53.
- Khai báo hàm.

```
void XuatCon(int [],int,int);
```

- Định nghĩa hàm.

```
void XuatCon(int a[],int n,int l)
{
    for(int vt=0;vt<=n-l;vt++)
    {
        cout<<endl;
        XuatCon(a,n,vt,l);
    }
}
```

- Nhận xét:
  + 0 là *vị trí bắt đầu* của mảng con đầu tiên trong mảng a có n phần tử.
  + $n - l$ là *vị trí bắt đầu* của mảng con cuối cùng trong mảng a có n phần tử.

Bài 175.(Hạt nhân) Định nghĩa hàm xuất tất cả các mảng con trong mảng một chiều các số nguyên.

- Ví dụ: Cho mảng ban đầu có 5 phần tử.

0	1	2	3	4
89	29	07	67	38

- Các mảng con có độ dài là 01.

0	1	2	3	4
89	29	07	67	38

89	29	07	67	38

89	29	07	67	38

89	29	07	67	38

89	29	07	67	38

– Các mảng con có độ dài là 02.

0	1	2	3	4
89	29	07	67	38

89	29	07	67	38

89	29	07	67	38

89	29	07	67	38

– Các mảng con có độ dài là 03.

0	1	2	3	4
89	29	07	67	38

89	29	07	67	38

89	29	07	67	38

– Các mảng con có độ dài là 04.

0	1	2	3	4
89	29	07	67	38

89	29	07	67	38

– Các mảng con có độ dài là 05.

0	1	2	3	4
89	29	07	67	38

– Kết quả: các mảng con trong mảng ban đầu.
  + Các mảng con độ dài là 1: 89, 29, 07, 67, 38.
  + Các mảng con độ dài là 2: 89 29, 29 07, 07 67, 67 38.
  + Các mảng con độ dài là 3: 89 29 07, 29 07 67, 07 67 38.
  + Các mảng con độ dài là 4: 89 29 07 67, 29 07 67 38.
  + Các mảng con độ dài là 5: 89 29 07 67 38.

– Khai báo hàm.

```
05018.  void XuatCon(int [],int);
```

– Định nghĩa hàm.

```
05019.  void XuatCon(int a[],int n)
```

```
05020. {
05021.    for(int l=1;l<=n;l++)
05022.       XuatCon(a,n,l);
05023. }
```

– Định nghĩa hàm (tích hợp ba hàm thành một hàm).

```
05024. void XuatCon(int a[],int n)
05025. {
05026.    for(int l=1;l<=n;l++)
05027.    {
05028.       for(int vt=0;vt<=n-l;vt++)
05029.       {
05030.          for(int i=0;i<l;i++)
05031.             cout << setw(8) << a[vt+i];
05032.          cout << endl;
05033.       }
05034.    }
05035. }
```

Bài 176. Liệt kê tất cả các mảng con có độ dài lớn hơn 2 trong mảng một chiều các số nguyên.

– Ví dụ: cho mảng ban đầu có 5 phần tử.

0	1	2	3	4
89	29	07	67	38

– Các mảng con có độ dài là 03.

0	1	2	3	4
89	29	07	67	38

89	29	07	67	38

89	29	07	67	38

– Các mảng con có độ dài là 04.

0	1	2	3	4
89	29	07	67	38

89	29	07	67	38

– Các mảng con có độ dài là 05.

0	1	2	3	4
89	29	07	67	38

– Kết quả: các mảng con có độ dài lớn hơn 2.

+ Các mảng con độ dài là 3: 89 29 07, 29 07 67, 07 67 38.
+ Các mảng con độ dài là 4: 89 29 07 67, 29 07 67 38.
+ Các mảng con độ dài là 5: 89 29 07 67 38.

– Khai báo hàm.

```
05036.  void LietKe(int [],int);
```

– Định nghĩa hàm.

```
05037.  void LietKe(int a[], int n)
05038.  {
05039.     for (int l = 3; l <= n; l++)
05040.     {
05041.        for (int vt = 0; vt <= n - l; vt++)
05042.        {
05043.           for (int i = 0; i < l; i++)
05044.              cout << setw(8) << a[vt + i];
05045.           cout << "\n";
05046.        }
05047.     }
05048.  }
```

Bài 177.(Hạt nhân) Liệt kê các dãy con tăng trong mảng một chiều các số nguyên.

– Ví dụ: Cho mảng ban đầu có 5 phần tử.

0	1	2	3	4
89	29	07	67	38

– Các mảng con có độ dài là 01.

0	1	2	3	4
89	29	07	67	38
89	**29**	07	67	38
89	29	**07**	67	38
89	29	07	**67**	38
89	29	07	67	**38**

– Các mảng con có độ dài là 02.

0	1	2	3	4
89	**29**	07	67	38
89	**29**	**07**	67	38
89	29	**07**	**67**	38
89	29	07	**67**	**38**

– Các mảng con có độ dài là 03.

0	1	2	3	4
89	29	07	67	38

89	29	07	67	38

89	29	07	67	38

– Các mảng con có độ dài là 04.

0	1	2	3	4
89	29	07	67	38

89	29	07	67	38

– Các mảng con có độ dài là 05.

0	1	2	3	4
89	29	07	67	38

– Kết quả: các mảng con tăng trong mảng ban đầu. 89, 29, 07, 67, 38, 07 67.

– Khai báo hàm.

```
05049.  int ktCon(int [], int, int, int);
05050.  void LietKe(int [],int);
```

– Định nghĩa hàm kiểm tra mảng con có độ dài *l* bắt đầu tại vị trí vt có tăng không.

```
05051.  int ktCon(int a[], int n, int vt, int l)
05052.  {
05053.     int flag = 1;
05054.     for (int i = 0; i <= l - 2; i++)
05055.        if (a[vt + i] > a[vt + i + 1])
05056.           flag = 0;
05057.     return flag;
05058.  }
```

– Định nghĩa hàm liệt kê các mảng con tăng.

```
05059.  void LietKe(int a[], int n)
05060.  {
05061.     for (int l = 1; l <= n; l++)
05062.     {
05063.        for (int vt = 0; vt <= n - l; vt++)
05064.        {
05065.           if (ktCon(a, n, vt, l) == 1)
05066.           {
05067.              for (int i = 0; i < l; i++)
05068.                 cout << setw(8) << a[vt + i];
05069.              cout << "\n";
05070.           }
```

```
05071.         }
05072.     }
05073. }
```

Bài 178.Xuất và tính tổng từng mảng con tăng trong mảng một chiều các số nguyên.

– Ví dụ: cho mảng ban đầu có 5 phần tử.

0	1	2	3	4
89	29	07	67	38

– Các mảng con có độ dài là 01.

0	1	2	3	4
89	29	07	67	38

89	**29**	07	67	38

89	29	**07**	67	38

89	29	07	**67**	38

89	29	07	67	**38**

– Các mảng con có độ dài là 02.

0	1	2	3	4
89	**29**	07	67	38

89	**29**	**07**	67	38

89	29	**07**	**67**	38

89	29	07	**67**	**38**

– Các mảng con có độ dài là 03.

0	1	2	3	4
89	**29**	**07**	67	38

89	**29**	**07**	**67**	38

89	29	**07**	**67**	**38**

– Các mảng con có độ dài là 04.

0	1	2	3	4
89	**29**	**07**	**67**	38

89	**29**	**07**	**67**	**38**

– Các mảng con có độ dài là 05.

0	1	2	3	4
89	29	07	67	38

- Kết quả: các mảng con tăng trong mảng ban đầu.
  - 89 có tổng 89.
  - 29 có tổng 29.
  - 07 có tổng 07.
  - 67 có tổng 67.
  - 38 có tổng 38.
  - 07 67 có tổng 74.
- Khai báo hàm.

```
05074.  int ktCon(int [], int, int, int);
05075.  int TongCon(int [], int, int, int);
05076.  void LietKe(int [],int);
```

- Định nghĩa hàm kiểm tra mảng con có độ dài *l* bắt đầu tại vị trí vt có tăng không.

```
05077.  int ktCon(int a[], int n, int vt, int l)
05078.  {
05079.     int flag = 1;
05080.     for (int i = 0; i <= l - 2; i++)
05081.        if (a[vt + i] > a[vt + i + 1])
05082.           flag = 0;
05083.     return flag;
05084.  }
```

- Định nghĩa hàm tính tổng các giá trị trong mảng con có độ dài *l* bắt đầu tại vị trí vt.

```
05085.  int TongCon(int a[],int n,int vt,int l)
05086.  {
05087.     int s = 0;
05088.     for (int i = 0; i < l; i++)
05089.        s = s + a[vt + i];
05090.     return s;
05091.  }
```

- Định nghĩa hàm xuất và tính tổng từng mảng con tăng.

```
05092.  void LietKe(int a[], int n)
05093.  {
05094.     for (int l = 1; l <= n; l++)
05095.     {
05096.        for (int vt = 0; vt <= n - l; vt++)
05097.        {
05098.           if (ktCon(a, n, vt, l) == 1)
05099.           {
```

```
05100.              cout << TongCon(a, n, vt, l);
05101.              for (int i = 0; i < l; i++)
05102.                 cout << setw(8) << a[vt + i];
05103.              cout << "\n";
05104.           }
05105.        }
05106.     }
05107.  }
```

Bài 179. Liệt kê các dãy con toàn dương có độ dài lớn hơn 1 trong mảng một chiều số nguyên.

- Ví dụ:

0	1	2	3	4	5	6	7	8	9
15.9	12.9	-0.6	7.6	9.9	23	12.1	-0.9	8.8	-1.1

- Các số dương trong mảng

0	1	2	3	4	5	6	7	8	9
15.9	12.9	-0.6	7.6	9.9	23	12.1	-0.9	8.8	-1.1

- Các dãy con toàn dương có độ dài lớn 1 là:
  - + Dãy con toàn dương có độ dài là 2.
    - 15.9 12.9
    - 7.6 9.9
    - 9.9 23
    - 23 12.1
  - + Dãy con toàn dương có độ dài là 3.
    - 7.6 9.9 23
    - 9.9 23 12.1
  - + Dãy con toàn dương có độ dài là 4.
    - 7.6 9.9 23 12.1
- Khai báo hàm.

```
05108.  int ktCon(int [], int, int, int);
05109.  void LietKe(int [],int);
```

- Định nghĩa hàm kiểm tra mảng con có độ dài *l* bắt đầu tại vị trí vt có toàn dương không.

```
05110.  int ktCon(int a[], int n, int vt, int l)
05111.  {
05112.     int flag = 1;
05113.     for (int i = 0; i <= l - 1; i++)
05114.        if (a[vt + i] <= 0)
05115.           flag = 0;
05116.     return flag;
05117.  }
```

– Định nghĩa hàm liệt kê các dãy con toàn dương có độ dài lớn hơn 1.

```
05118.  void LietKe(int a[], int n)
05119.  {
05120.     for (int l = 2; l <= n; l++)
05121.     {
05122.        for (int vt = 0; vt <= n - l; vt++)
05123.        {
05124.           if (ktCon(a, n, vt, l) == 1)
05125.           {
05126.              for (int i = 0; i < l; i++)
05127.                 cout << setw(8) << a[vt + i];
05128.              cout << "\n";
05129.           }
05130.        }
05131.     }
05132.  }
```

Bài 180.Đếm số lượng mảng con tăng có độ dài lớn hơn 1 trong mảng một chiều các số nguyên.

– Ví dụ: cho mảng ban đầu có 5 phần tử.

0	1	2	3	4
89	29	07	67	88

– Các mảng con có độ dài là 01.

0	1	2	3	4
89	29	07	67	88

0	1	2	3	4
89	**29**	07	67	88

0	1	2	3	4
89	29	**07**	67	88

0	1	2	3	4
89	29	07	**67**	88

0	1	2	3	4
89	29	07	67	**88**

– Các mảng con có độ dài là 02.

0	1	2	3	4
89	**29**	07	67	88

0	1	2	3	4
89	**29**	**07**	67	88

0	1	2	3	4
89	29	**07**	**67**	88

89	29	07	67	88

– Các mảng con có độ dài là 03.

0	1	2	3	4
89	29	07	67	88

89	29	07	67	88

89	29	07	67	88

– Các mảng con có độ dài là 04.

0	1	2	3	4
89	29	07	67	88

89	29	07	67	88

– Các mảng con có độ dài là 05.

0	1	2	3	4
89	29	07	67	88

– Các mảng con tăng trong mảng ban đầu. 89, 29, 07, 67, 38, 07 67, 67 88, 07 67 88.

– Kết quả: 3.

– Khai báo hàm.

```
05133.  int ktCon(int [], int, int, int);
05134.  int DemConTang(int [],int);
```

– Định nghĩa hàm kiểm tra mảng con có độ dài *l* bắt đầu tại vị trí vt có tăng không.

```
05135.  int ktCon(int a[], int n, int vt, int l)
05136.  {
05137.     int flag = 1;
05138.     for (int i = 0; i <= l - 2; i++)
05139.        if (a[vt + i] > a[vt + i + 1])
05140.           flag = 0;
05141.     return flag;
05142.  }
```

– Định nghĩa hàm đếm số lượng mảng con tăng có độ dài lớn hơn 1.

```
05143.  int DemConTang(int a[], int n)
05144.  {
05145.     int dem = 0;
05146.     for (int l = 2; l <= n; l++)
05147.     {
05148.        for (int vt = 0; vt <= n - l; vt++)
05149.        {
05150.           if (ktCon(a, n, vt, l) == 1)
```

```
05151.                dem++;
05152.            }
05153.        }
05154.        return dem;
05155.    }
```

Bài 181.Đếm số lượng mảng con giảm trong mảng một chiều các số nguyên.

– Ví dụ: cho mảng ban đầu có 5 phần tử.

0	1	2	3	4
89	29	07	67	88

– Các mảng con có độ dài là 01.

0	1	2	3	4
89	29	07	67	88

89	**29**	07	67	88

89	29	**07**	67	88

89	29	07	**67**	88

89	29	07	67	**88**

– Các mảng con có độ dài là 02.

0	1	2	3	4
89	**29**	07	67	88

89	**29**	**07**	67	88

89	29	**07**	**67**	88

89	29	07	**67**	**88**

– Các mảng con có độ dài là 03.

0	1	2	3	4
89	**29**	**07**	67	88

89	**29**	**07**	**67**	88

89	29	**07**	**67**	**88**

– Các mảng con có độ dài là 04.

0	1	2	3	4
89	**29**	**07**	**67**	88

89	29	07	67	88

– Các mảng con có độ dài là 05.

0	1	2	3	4
89	29	07	67	88

– Các mảng con giảm trong mảng ban đầu. 89, 29, 07, 67, 38, 89 29, 29 07, 89 29 07.

– Kết quả: 8.

– Khai báo hàm.

```
05156.  int ktCon(int [], int, int, int);
05157.  int DemConGiam(int [],int);
```

– Định nghĩa hàm kiểm tra mảng con có độ dài *l* bắt đầu tại vị trí vt có giảm không.

```
05158.  int ktCon(int a[], int n, int vt, int l)
05159.  {
05160.     int flag = 1;
05161.     for (int i = 0; i <= l - 2; i++)
05162.        if (a[vt + i] < a[vt + i + 1])
05163.           flag = 0;
05164.     return flag;
05165.  }
```

– Định nghĩa hàm đếm số lượng mảng con giảm.

```
05166.  int DemConGiam(int a[], int n)
05167.  {
05168.     int dem = 0;
05169.     for (int l = 1; l <= n; l++)
05170.     {
05171.        for (int vt = 0; vt <= n - l; vt++)
05172.        {
05173.           if (ktCon(a, n, vt, l) == 1)
05174.              dem++;
05175.        }
05176.     }
05177.     return dem;
05178.  }
```

Bài 182. Hãy liệt kê các mảng con tăng có chứa giá trị lớn nhất trong mảng các số nguyên.

– Ví dụ: cho mảng ban đầu có 5 phần tử.

0	1	2	3	4
66	29	62	88	10

– Các mảng con có độ dài là 01.

0	1	2	3	4

66	29	62	88	10

66	29	62	88	10

66	29	62	88	10

66	29	62	88	10

66	29	62	88	10

– Các mảng con có độ dài là 02.

0	1	2	3	4
66	29	62	88	10

66	29	62	88	10

66	29	62	88	10

66	29	62	88	10

– Các mảng con có độ dài là 03.

0	1	2	3	4
66	29	62	88	10

66	29	62	88	10

66	29	62	88	10

– Các mảng con có độ dài là 04.

0	1	2	3	4
66	29	62	88	10

66	29	62	88	10

– Các mảng con có độ dài là 05.

0	1	2	3	4
66	29	62	88	10

– Các mảng con tăng trong mảng ban đầu. 66, 29, 62, 88, 10, 29 62, 62 88, 29 62 88.

– Kết quả: 88, 62 88, 29 62 88.

– Khai báo hàm.

```
05179.  void LietKe(int [],int);
05180.  int LonNhat(int [], int);
05181.  int ktCon(int [], int, int, int);
```

– Định nghĩa hàm kiểm tra mảng con có độ dài l bắt đầu tại vị trí vt có tăng không.

```
05182.  int ktCon(int a[], int n, int vt, int l)
05183.  {
05184.     int flag = 1;
05185.     for (int i = 0; i <= l - 2; i++)
05186.        if (a[vt + i] > a[vt + i + 1])
05187.           flag = 0;
05188.     return flag;
05189.  }
```

– Định nghĩa hàm tìm giá trị lớn nhất.

```
05190.  int LonNhat(int a[], int n)
05191.  {
05192.     int lc = a[0];
05193.     for (int i=0; i<n; i++)
05194.        if (a[i] > lc)
05195.           lc = a[i];
05196.     return lc;
05197.  }
```

– Định nghĩa hàm liệt kê các mảng con tăng có chứa giá trị lớn nhất.

```
05198.  void LietKe(int a[], int n)
05199.  {
05200.     int ln = LonNhat(a,n);
05201.     for (int l = 1; l <= n; l++)
05202.     {
05203.        for (int vt = 0; vt <= n - l; vt++)
05204.        {
05205.           if(ktCon(a,n,vt,l) && a[vt+l-1]==ln)
05206.           {
05207.              for (int i = 0; i <= l - 1; i++)
05208.                 cout <<setw(3)<< a[vt + i];
05209.              cout << endl;
05210.           }
05211.        }
05212.     }
05213.  }
```

Bài 183.(*) Cho hai mảng số nguyên *a* và *b*. Hãy cho biết mảng *a* có phải là mảng con trong mảng *b* hay không?

– Ví dụ:

+ Mảng a.

76	99	23	12

+ Mảng b.

19	29	62	76	99	23	12	98	88	11

+ Kết quả: 1.

– Ý tưởng.

0	1	2	3	4	5	6	7	8	9
19	29	62	76	99	23	12	98	88	11

19	29	62	76	99	23	12	98	88	11

19	29	62	76	99	23	12	98	88	11

19	29	62	76	99	23	12	98	88	11

19	29	62	76	99	23	12	98	88	11

19	29	62	76	99	23	12	98	88	11

19	29	62	76	99	23	12	98	88	11

– Giả sử:
  + Mảng a có n phần tử.
  + Mảng b có m phần tử.
  + 0 là *vị trí bắt đầu* của mảng con đầu tiên có độ dài là n trong mảng b có m phần tử.
  + $m - n$ là *vị trí bắt đầu* của mảng con cuối cùng có độ dài là n trong mảng b có m phần tử.

– Khai báo hàm.

```
int ktCon(int [],int,int [],int);
```

– Định nghĩa hàm.

```
int ktCon(int a[], int n, int b[], int m)
{
    if (n > m)
        return 0;
    int flag = 0;
    for (int vt = 0; vt <= m - n; vt++)
    {
        int co = 1;
        for (int i=0; i<n; i++)
            if (a[i] != b[vt + i])
                co = 0;
        if (co == 1)
            flag = 1;
    }
    return flag;
```

```
}
```

Bài 184.(*) Cho hai mảng số nguyên *a* và *b*. Hãy đếm số lần xuất hiện của mảng *a* trong mảng *b*.

– Ví dụ:

+ Mảng *a*.

76	99	23	12

+ Mảng *b*.

19	29	62	76	99	23	12	98	88	11

+ Kết quả: 1.

– Ý tưởng.

0	1	2	3	4	5	6	7	8	9
19	**29**	**62**	**76**	99	23	12	98	88	11

19	**29**	**62**	**76**	**99**	23	12	98	88	11

19	29	**62**	**76**	**99**	**23**	12	98	88	11

19	29	62	**76**	**99**	**23**	**12**	98	88	11

19	29	62	76	**99**	**23**	**12**	**98**	88	11

19	29	62	76	99	**23**	**12**	**98**	**88**	11

19	29	62	76	99	23	**12**	**98**	**88**	**11**

– Khai báo hàm.

```
int DemCon(int [],int,int [],int);
```

– Định nghĩa hàm.

```
int DemCon(int a[], int n,int b[], int m)
{
    if (n > m)
        return 0;
    int dem = 0;
    for (int vt = 0; vt <= m - n; vt++)
    {
        int flag = 1;
        for (int i=0; i<n; i++)
            if (a[i] != b[vt + i])
                flag = 0;
        if (flag == 1)
            dem++;
    }
```

```
05246.     return dem;
05247.  }
```

Bài 185.(*) Tìm dãy con toàn dương dài nhất trong mảng các số thực.

- Ví dụ:

15.9	12.9	-0.6	7.6	9.9	23	12.1	-0.9	8.8	-1.1

- Các số dương trong mảng

15.9	12.9	-0.6	7.6	9.9	23	12.1	-0.9	8.8	-1.1

- Kết quả: 7.6 9.9 23 12.1
- Khai báo hàm.

```
05248.  int ktCon(float [], int, int, int);
05249.  void DuongDaiNhat(float [],int,int &,int&);
```

- Định nghĩa hàm kiểm tra mảng con có độ dài l bắt đầu tại vị trí vt có toàn dương không.

```
05250.  int ktCon(float a[], int n, int vt, int l)
05251.  {
05252.     int flag = 1;
05253.     for (int i = 0; i <= l - 1; i++)
05254.        if (a[vt + i] <= 0)
05255.           flag = 0;
05256.     return flag;
05257.  }
```

- Định nghĩa hàm tìm dãy con toàn dương dài nhất.

```
05258.  void DuongDaiNhat(float a[], int n,
05259.                    int &vtd, int &vtc)
05260.  {
05261.     for (int l = n; l >= 1; l--)
05262.        for (int vt = 0; vt <= n - l; vt++)
05263.           if (ktCon(a, n, vt, l) == 1)
05264.           {
05265.              vtd = vt;
05266.              vtc = vt + l - 1;
05267.              return;
05268.           }
05269.     vtd = vtc = -1;
05270.  }
```

Bài 186. (*) Cho mảng một chiều các số nguyên và một số nguyên M. Hãy tìm một mảng con sao cho tổng các phần tử trong mảng bằng M.

- Ví dụ: cho mảng sau và $M = 100$

0	1	2	3	4	5	6	7	8	9
19	29	62	8	30	23	12	98	88	11

– Kết quả: $vt = 2, l = 3$.

0	1	2	3	4	5	6	7	8	9
19	29	62	8	30	23	12	98	88	11

– Khai báo hàm.

```
05271. int TongCon(int [],int,int,int);
05272. void TimCon(int [],int,int,int&,int&);
```

– Định nghĩa hàm tính tổng các phần tử trong mảng con có độ dài l bắt đầu tại vị trí vt.

```
05273. int TongCon(int a[],int n,int vt,int l)
05274. {
05275.    int s = 0;
05276.    for (int i = 0; i <= l - 1; i++)
05277.       s = s + a[vt + i];
05278.    return s;
05279. }
```

– Định nghĩa hàm tìm một mảng con sao cho tổng các phần tử trong mảng bằng M.

```
05280. void TimCon(int a[], int n,int M,
05281.              int &vtd, int &vtc)
05282. {
05283.    for (int l = 1; l <= n; l++)
05284.       for (int vt = 0; vt <= n - l; vt++)
05285.          if (TongCon(a, n, vt, l) == M)
05286.          {
05287.             vtd = vt;
05288.             vtc = vt + l - 1;
05289.             return;
05290.          }
05291.    vtd = vtc = -1;
05292. }
```

Bài 187. (*) Tìm dãy con toàn dương có tổng lớn nhất trong mảng một chiều các số thực.

– Ví dụ:

151.9	12.3	-0.6	7.6	9.9	23	12.1	-0.9	8.8	-1.1

– Các số dương trong mảng

151.9	12.3	-0.6	7.6	9.9	23	12.1	-0.9	8.8	-1.1

– Kết quả: 151.9 12.3

– Khai báo hàm.

```
05293.  int ViTriDuongDau(float [], int);
05294.  float TongCon(float [], int, int, int);
05295.  int ktCon(float a[], int, int, int);
05296.  void DuongLonNhat(float [],int,int &,int&);
```

– Định nghĩa hàm tính tổng các phần tử trong mảng con có độ dài *l* bắt đầu tại vị trí vt.

```
05297.  float TongCon(float a[],int n,int vt,int l)
05298.  {
05299.     int s = 0;
05300.     for (int i = 0; i < l; i++)
05301.        s = s + a[vt + i];
05302.     return s;
05303.  }
```

– Định nghĩa hàm kiểm tra mảng con có độ dài *l* bắt đầu tại vị trí vt có toàn dương không.

```
05304.  int ktCon(float a[], int n, int vt, int l)
05305.  {
05306.     int flag = 1;
05307.     for (int i = 0; i <= l - 1; i++)
05308.        if (a[vt + i] <= 0)
05309.           flag = 0;
05310.     return flag;
05311.  }
```

– Định nghĩa hàm tìm vị trí dương đầu tiên trong mảng.

```
05312.  int ViTriDuongDau(float a[], int n)
05313.  {
05314.     for (int i=0; i<n; i++)
05315.        if (a[i] > 0)
05316.           return i;
05317.     return -1;
05318.  }
```

– Định nghĩa hàm tìm dãy con toàn dương có tổng lớn nhất.

```
05319.  void DuongLonNhat(float a[], int n,
05320.                     int &vtd, int &vtc)
05321.  {
05322.     int vt = ViTriDuongDau(a,n);
05323.     if(vt==-1)
05324.     {
05325.        vtd = vtc = -1;
05326.        return;
05327.     }
```

```
05328.     int smax = a[vt];
05329.     vtd = vtc = vt;
05330.     for (int l = 1; l <= n; l++)
05331.        for (int vt = 0; vt <= n - l; vt++)
05332.           if (ktCon(a, n, vt, l)==1 &&
05333.              TongCon(a, n, vt, l) > smax)
05334.           {
05335.              vtd = vt;
05336.              vtc = vt + l - 1;
05337.              smax = TongCon(a, n, vt, l);
05338.           }
05339.  }
```

Bài 188.(Hạt nhân) Tìm mảng con có tổng lớn nhất trong mảng một chiều các số thực.

– Ví dụ:

0	1	2	3	4	5	6	7	8	9
19	29	-62	23	7	36	-99	17	8	-11

– Kết quả: vtd $= 3, vtc = 5$.

19	29	-62	23	7	36	-99	17	8	-11

– Khai báo hàm.

```
05340.  void ConLonNhat(float [],int,int &,int &);
05341.  float TongCon(float [], int, int, int);
```

– Định nghĩa hàm tính tổng các phần tử trong mảng con có độ dài l bắt đầu tại vị trí vt trong mảng a có n phần tử.

```
05342.  float TongCon(float a[],int n,int vt,int l)
05343.  {
05344.     float s = 0;
05345.     for (int i = 0; i < l; i++)
05346.        s = s + a[vt + i];
05347.     return s;
05348.  }
```

– Định nghĩa hàm tìm mảng con có tổng lớn nhất.

```
05349.  void ConLonNhat(float a[],int n,
05350.                   int &vtd,int &vtc)
05351.  {
05352.     vtd = vtc = 0;
05353.     float sln = a[0];
05354.     for (int l = 1; l <= n; l++)
05355.        for (int vt = 0; vt <= n - l; vt++)
05356.           if (TongCon(a, n, vt, l) > sln)
05357.           {
```

```
05358.              vtd = vt;
05359.              vtc = vt + l - 1;
05360.              sln = TongCon(a, n, vt, l);
05361.           }
05362.  }
```

CHƯƠNG 02. THUẬT TOÁN INTERCHANGE SORT

02.01 BÀI TOÁN DẪN NHẬP

Bài cơ sở 039. Viết hàm liệt kê tất cả các cặp giá trị trong mảng một chiều các số nguyên. Lưu ý: cặp (1,2) và cặp (2,1) là giống nhau.

- Ví dụ:

12	43	01	34	22

- Các cặp giá trị trong mảng là:
  + (12,43), (12,01), (12,34), (12,22).
  + (43,01), (43,34), (43,22) .
  + (01,34), (01,22) .
  + (34,22).
- Khai báo hàm.

```
05363.  void LietKe(int [],int);
```

- Hàm cài đặt.

```
05364.  void LietKe(int a[],int n)
05365.  {
05366.     for(int i=0; i<=n-2; i++)
05367.        for(int j=i+1; j<=n-1; j++)
05368.           cout <<a[i]<<a[j]<<endl;
05369.  }
```

02.02 NGHỊCH THẾ

- Khái niệm: Một cặp giá trị (a_i, a_j) được gọi là nghịch thế khi a_i và a_j không thỏa điều kiện sắp thứ tự.

- Ví dụ 1: Cho mảng một chiều các số thực a có n phần tử: a_0, a_1, a_2, …, a_{n-2}, a_{n-1}. Hãy sắp mảng theo thứ tự tăng dần. Khi đó cặp giá trị (a_i, a_j) với $(i \le j)$ được gọi là nghịch thế khi $a_i \ge a_j$.
- Ví dụ 3: Hãy liệt kê các cặp giá trị nghịch thế trong mảng sau, biết rằng yêu cầu là sắp xếp mảng tăng dần.

14	29	-1	10	05	23

- Các cặp giá trị trong mảng.
  + (14, 29), (14, -1), (14, 10), (14, 05), (14, 23).
  + (29, -1), (29, 10), (29, 05), (29, 23).
  + (-1, 10), (-1, 05), (-1, 23).
  + (10, 05), (10, 23).
  + (05, 23).
- Các cặp giá trị nghịch thế trong mảng: (14, -1), (14,10), (14,5), (29,-1), (29,10), (29,5), (29,23), (10,5).

02.03 TƯ TƯỞNG THUẬT TOÁN INTERCHANGE SORT

- Thuật toán Interchange Sort sẽ duyệt qua tất cả các cặp giá trị trong mảng và hoán vị hai giá trị trong một cặp nếu cặp giá trị đó là nghịch thế.

02.04 HÀM CÀI ĐẶT THUẬT TOÁN INTERCHANGE SORT

Bài cơ sở 040. Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Interchange sort.

- Khai báo hàm.

```
05370. void InterchangeSort(int [], int);
```

- Định nghĩa hàm (phiên bản 01).

```
05371. void InterchangeSort(int a[], int n)
05372. {
05373.    for(int i=0; i<=n-2; i++)
05374.       for(int j=i+1; j<=n-1; j++)
05375.          if(a[i] > a[j])
05376.             HoanVi(a[i],a[j]);
05377. }
```

02.05 MÔ TẢ TRỰC QUAN THUẬT TOÁN INTERCHANGE SORT

Bài cơ sở 041. Hãy mô tả trực quan sắp xếp mảng 24, 45, 23, 13, 43, -1 tăng dần với phiên bản 01.

– Mảng ban đầu:

24	45	23	13	43	-1

– Phiên bản cài đặt 01.

```
05378.  void InterchangeSort(int a[], int n)
05379.  {
05380.     for(int i=0; i<=n-2; i++)
05381.        for(int j=i+1; j<=n-1; j++)
05382.           if(a[i] > a[j])
05383.              swap(a[i],a[j]);
05384.  }
```

– Thứ tự các bước khi sắp tăng dần mảng trên bằng thuật toán Interchange Sort.

+ Bước 1:

24	45	23	13	43	-1

24	45	23	13	43	-1

23	45	24	13	43	-1

13	45	24	23	43	-1

13	45	24	23	43	-1

-1	45	24	23	43	13

+ Bước 2:

-1	45	24	23	43	13

-1	24	45	23	43	13

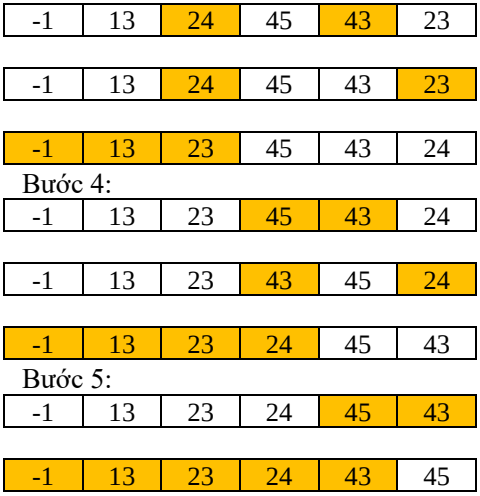
-1	23	45	24	43	13

-1	23	45	24	43	13

-1	13	45	24	43	23

+ Bước 3:

-1	13	45	24	43	23



02.06 CÁC PHIÊN BẢN CÀI ĐẶT KHÁC CỦA THUẬT TOÁN INTERCHANGE SORT

Bài cơ sở 042. Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Interchange sort.

– Khai báo hàm.

```
05385. void InterchangeSort(int [], int);
```

– Phiên bản cài đặt 02.

```
05386. void InterchangeSort(int a[], int n)
05387. {
05388.    for(int i=0; i<=n-2; i++)
05389.       for(int j=n-1; j>=i+1; j--)
05390.             swap(a[i],a[j]);
05391. }
```

– Phiên bản cài đặt 03.

```
05392. void InterchangeSort(int a[], int n)
05393. {
05394.    for(int i=n-1; i>=1; i--)
05395.       for(int j=0; j<=i-1; j++)
05396.          if(a[i] < a[j])
05397.             swap(a[i],a[j]);
05398. }
```

– Phiên bản cài đặt 04.

```
05399. void InterchangeSort(int a[], int n)
05400. {
```

```
05401.     for(int i=n-1; i>=1; i--)
05402.        for(int j=i-1; j>=0; j--)
05403.           if(a[i] < a[j])
05404.              swap(a[i],a[j]);
05405.  }
```

– Phiên bản cài đặt 05.

```
05406.  void InterchangeSort(int a[], int n)
05407.  {
05408.     for(int i=0; i<=n-2; i++)
05409.        for(int j=n-1; j>=i+1; j--)
05410.           if(a[i] > a[j])
05411.              swap(a[i],a[j]);
05412.  }
```

02.07 PHIÊN BẢN CÀI ĐẶT THUẬT TOÁN INTERCHANGE SORT BẰNG ĐỆ QUI

Bài cơ sở 043. Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Interchange sort.

– Phiên bản cài đặt 06 – Sử dụng kỹ thuật Đệ qui.

```
05413.  void InterchangeSort(int a[],int n)
05414.  {
05415.     if(n<=1)
05416.        return;
05417.     for(int i=0; i<=n-2; i++)
05418.        if(a[i]>a[n-1])
05419.           swap(a[i],a[n-1]);
05420.     InterchangeSort(a,n-1);
05421.  }
```

02.08 PHÂN TÍCH ĐỘ PHỨC TẠP CỦA THUẬT TOÁN INTERCHANGE SORT

02.08.01 Phân tích phiên bản không đệ qui

– Nhìn lại hàm InterchangeSort.

```
05422.  void InterchangeSort(int a[], int n)
05423.  {
05424.     for(int i=0; i<=n-2; i++)
05425.        for(int j=i+1; j<=n-1; j++)
05426.           if(a[i] > a[j])
```

```
05427.              swap(a[i],a[j]);
05428.  }
```

– Phân tích độ phức tạp của thuật toán.

```
05429.  Biến i chạy từ 0 đến n-2.
05430.  Khi i = 0
05431.     Biến j chạy từ 1 đến n-1.
05432.     Có n-1 lần chạy
05433.  Khi i = 1
05434.     Biến j chạy từ 2 đến n-1.
05435.     Có n-2 lần chạy
05436.  Khi i = 2
05437.     Biến j chạy từ 3 đến n-1.
05438.     Có n-3 lần chạy
05439.  …
05440.  Khi i = k
05441.     Biến j chạy từ k+1 đến n-1.
05442.     Có (n-1-k) lần chạy
05443.  …
05444.  Khi i = n-2
05445.     Biến j chạy từ n-1 đến n-1.
05446.     Có 1 lần chạy
05447.
05448.  Tổng số lần chạy là:
05449.  = (n-1)+(n-2)+⋯+(n-1-k)+⋯+3+2+1
05450.  = (n−1)n / 2
05451.  = n²−n / 2
05452.  = 1/2 n² − 1/2 n
05453.  ≈ 1/2 n².
05454.  ≈ n².
```

– Vậy độ phức tạp của thuật toán Interchange sort là: $O(n^2)$.

02.08.02 Phân tích phiên bản đệ qui

– Nhìn lại hàm InterchangeSort phiên bản đệ qui.

```
05455.  void InterchangeSort(int a[],int n)
05456.  {
05457.     if(n<=1)
05458.        return;
05459.     for(int i=0; i<=n-2; i++)
05460.        if(a[i]>a[n-1])
```

```
05461.          swap(a[i],a[n-1]);
05462.
05463.    InterchangeSort(a,n-1);
05464. }
```

– Phân tích độ phức tạp của thuật toán.

02.09 ÁP DỤNG THUẬT TOÁN INTERCHANGE SORT

Bài cơ sở 044. Định nghĩa hàm sắp mảng một chiều các phân số tăng dần bằng thuật toán Interchange Sort.

– Khai báo kiểu dữ liệu biểu diễn phân số.

```
05465. struct phanso
05466. {
05467.    int tu;
05468.    int mau;
05469. };
05470. typedef struct phanso PHANSO;
```

– Định nghĩa hàm SoSanh.

```
05471. int SoSanh(PHANSO x, PHANSO y)
05472. {
05473.    int a = (int)x.tu/x.mau;
05474.    int b = (int)y.tu/y.mau;
05475.    if(a>b)
05476.          return 1;
05477.    if(a<b)
05478.          return -1;
05479.    return 0;
05480. }
```

– Hàm cài đặt hàm InterchangeSort.

```
05481. void InterchangeSort(int a[], int n)
05482. {
05483.    for(int i=0; i<=n-2; i++)
05484.       for(int j=i+1; j<=n-1; j++)
05485.          if(SoSanh(a[i],a[j]==-1)
05486.          {
05487.             PHANSO temp = a[i];
05488.             a[i] = a[j];
05489.             a[j] = temp;
05490.          }
05491. }
```

02.10 BÀI TẬP VỀ THUẬT TOÁN INTERCHANGE SORT

Bài 189. Định nghĩa hàm đếm số lượng phép hoán vị khi cài đặt thuật toán Interchange sort với các phiên bản cài đặt 1, 2, 3, 4, 5, 6.

Bài 190. Hãy phân tích độ phức tạp của thuật toán Interchange sort theo các phiên bản cài đặt 02, 03, 04, 05.

Bài 191. Hãy mô tả trực quan sắp xếp mảng 24, 45, 23, 13, 43, -1 tăng dần với phiên bản cài đặt 02, 03, 04, 05, 06.

Bài 192. Hãy định nghĩa hàm sắp xếp mảng một chiều các phân số tăng dần bằng thuật toán interchange sort.

Bài 193. Hãy định nghĩa hàm sắp xếp (*bằng thuật toán interchange sort*) mảng một chiều các số phức tăng dần theo phần thực.

Bài 194. Hãy định nghĩa hàm sắp xếp (*bằng thuật toán interchange sort*) mảng một chiều tọa độ các điểm trong mặt phẳng Oxy giảm dần theo khoảng cách đến gốc tọa độ.