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

CHUYÊN ĐỀ ĐỆ QUY

(Bản thảo ngày 16/05/2023)

Nguyễn Tần Trần Minh Khang
Võ Duy Nguyên

THÀNH PHỐ HỒ CHÍ MINH 2022

MỤC LỤC

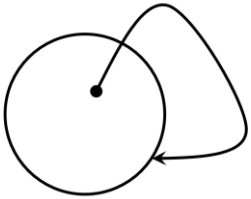
CHƯƠNG 01. ĐỆ QUY TUYẾN TÍNH

01.01 KHÁI NIỆM ĐỆ QUY TUYẾN TÍNH

- Khái niệm: Một hàm được gọi là đệ quy tuyến tính (linear recursion, single recursion, linear recursive) nếu trong thân của hàm đó có duy nhất một lời gọi hàm gọi lại chính nó một cách tường minh.

01.02 HÌNH ẢNH ĐỆ QUY TUYẾN TÍNH

- Hình vẽ minh họa



- Trong hình vẽ minh họa trên ta có thể hiểu như sau:
  + Hàm là vòng tròn.
  + Lời gọi hàm được minh họa bởi vòng cung có mũi tên.
  + Lời gọi hàm bắt đầu tại một điểm trong vòng tròn và kết thúc với mũi tên tại biên vòng tròn.

01.03 CẤU TRÚC HÀM ĐỆ QUY TUYẾN TÍNH

```
00001.  KDL TenHam(<ThamSo>)
00002.  {
00003.      if <điều kiện dừng>
00004.      {
00005.          ...
00006.          return <Giá Trị Trả Về>;
00007.      }
00008.      ...
00009.      ...TenHam(<ThamSo>;
00010.      ...
00011.  }
```

01.04 KỸ THUẬT TÍNH TOÁN ĐỆ QUY

01.04.01 Kỹ thuật Đệ quy Tính tổng Tính tích

01.04.01.1 Bài cơ sở Kỹ thuật Đệ quy Tính tổng Tính tích

> Bài cơ sở 001. Viết hàm đệ quy tính tổng $S(n) = 1 + 2 + 3 + \cdots + n$.

- Ta có:
  - $S(n) = 1 + 2 + 3 + \cdots + (n-1) + n$.
  - $S(n-1) = 1 + 2 + 3 + \cdots + (n-1)$.
- Suy ra: $S(n) = S(n-1) + n$.
- Điều kiện dừng:
  - Phương án 01: $S(0) = 0$.
  - Phương án 02: $S(1) = 1$.
- Khai báo hàm.

```
00012. int Tong(int);
```

- Định nghĩa hàm theo phương án 01.

```
00013. int Tong(int n)
00014. {
00015.     if(n==0)
00016.        return 0;
00017.     int s = Tong(n-1);
00018.     return (s+n);
00019. }
```

- Định nghĩa hàm theo phương án 02.

```
00020. int Tong(int n)
00021. {
00022.     if(n==1)
00023.        return 1;
00024.     int s = Tong(n-1);
00025.     return (s+n);
00026. }
```

- Định nghĩa hàm theo phương án 03.

```
00027. int Tong(int n)
00028. {
00029.     if(n==1)
00030.        return 1;
00031.     return (Tong(n-1)+n);
00032. }
```

Chương trình 001. Viết chương trình bằng kỹ thuật đệ quy tính $S(n) = 1 + 2 + 3 + \cdots + (n-1) + n$.

– Chương trình

```
00033. #include <iostream>
00034. using namespace std;
00035.
00036. int Tong(int);
00037.
00038. int main()
00039. {
00040.    int n;
00041.    cout << "Nhap n: ";
00042.    cin >> n;
00043.
00044.    int kq = Tong(n);
00045.    cout << "Ket qua: " << kq;
00046.    return 1;
00047. }
00048.
00049. int Tong(int n)
00050. {
00051.    if (n == 1)
00052.       return 1;
00053.    return Tong(n - 1) + n;
00054. }
```

Bài cơ sở 002. Viết hàm đệ quy tính $T(n) = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$.

– Ta có:
  + $T(n) = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$.
  + $T(n-1) = 1 \times 2 \times 3 \times \cdots \times (n-1)$.
– Suy ra: $T(n) = T(n-1) \times n$.
– Điều kiện dừng:
  + Phương án 01: $T(0) = 1$.
  + Phương án 02: $T(1) = 1$.
– Khai báo hàm.

```
00055. int GiaiThua(int);
```

– Định nghĩa hàm theo phương án 01.

```
00056. int GiaiThua(int n)
00057. {
```

```
00058.     if(n==0)
00059.        return 1;
00060.     int T = GiaiThua(n-1);
00061.     return(T*n);
00062.  }
```

– Định nghĩa hàm theo phương án 02.

```
00063.  int GiaiThua(int n)
00064.  {
00065.     if(n==1)
00066.        return 1;
00067.     int T = GiaiThua(n-1);
00068.     return(T*n);
00069.  }
```

– Định nghĩa hàm theo phương án 03.

```
00070.  int GiaiThua(int n)
00071.  {
00072.     if(n==1)
00073.        return 1;
00074.     return(GiaiThua(n-1)*n);
00075.  }
```

Chương trình 002. Viết chương trình bằng kỹ thuật đệ quy tính $T(n) = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$.

– Chương trình

```
00076.  #include <iostream>
00077.  using namespace std;
00078.
00079.  int GiaiThua(int);
00080.
00081.  int main()
00082.  {
00083.     int n;
00084.     cout << "Nhap n: ";
00085.     cin >> n;
00086.
00087.     int kq = GiaiThua(n);
00088.     cout << "Ket qua: " << kq;
00089.     return 1;
00090.  }
00091.
00092.  int GiaiThua(int n)
```

```
00093.  {
00094.     if (n == 1)
00095.        return 1;
00096.     return GiaiThua(n - 1) * n;
00097.  }
```

Bài cơ sở 003. Viết hàm đệ quy tính $T(x,n) = x^n$.

- Ta có:
  - $T(x,n) = x^n$.
  - $T(x,n-1) = x^{n-1}$.
- Suy ra: $T(x,n) = T(x,n-1) \times x$.
- Điều kiện dừng: $T(x,0) = 1$.
- Khai báo hàm.

```
00098.  float LuyThua(float,int);
```

- Định nghĩa hàm đệ quy theo phương án 01.

```
00099.  float LuyThua(float x,int n)
00100.  {
00101.     if(n==0)
00102.        return 1;
00103.     float T = LuyThua(x,n-1);
00104.     return(T*x);
00105.  }
```

- Định nghĩa hàm đệ quy theo phương án 02.

```
00106.  float LuyThua(float x,int n)
00107.  {
00108.     if(n==1)
00109.        return x;
00110.     float T = LuyThua(x,n-1);
00111.     return(T*x);
00112.  }
```

- Định nghĩa hàm đệ quy theo phương án 03.

```
00113.  float LuyThua(float x,int n)
00114.  {
00115.     if(n==0)
00116.        return 1;
00117.     return (LuyThua(x,n-1)*x);
00118.  }
```

–

Chương trình 003. Viết chương trình bằng kỹ thuật đệ quy tính $T(x,n) = x^n$.

– Chương trình

```
00119. #include <iostream>
00120. #include <iomanip>
00121. using namespace std;
00122.
00123. float LuyThua(float, int);
00124.
00125. int main()
00126. {
00127.    int n;
00128.    cout << "Nhap n: ";
00129.    cin >> n;
00130.
00131.    float x;
00132.    cout << "Nhap x: ";
00133.    cin >> x;
00134.
00135.    float kq = LuyThua(x, n);
00136.    cout << setw(8) << setprecision(5);
00137.    cout << "Ket qua: " << kq;
00138.    return 1;
00139. }
00140.
00141. float LuyThua(float x, int n)
00142. {
00143.    if (n == 0)
00144.       return 1;
00145.    return LuyThua(x, n - 1) * x;
00146. }
```

01.04.01.2 Bài tập Kỹ thuật Đệ quy Tính tổng Tính tích

Bài 001.Viết hàm đệ quy tính $S(n) = 1^2 + 2^2 + 3^2 + \cdots + (n-1)^2 + n^2$.

– Ta có:
  + $S(n) = 1^2 + 2^2 + 3^2 + \cdots + (n-1)^2 + n^2$.
  + $S(n-1) = 1^2 + 2^2 + 3^2 + \cdots + (n-1)^2$.
– Suy ra: $S(n) = S(n-1) + n^2$.
– Điều kiện dừng: $S(0) = 0$.

– Dữ liệu test

```
00147. Dữ liệu vào: n=10
00148. Dữ liệu ra: 285
00149.
00150. Dữ liệu vào:n=143
00151. Dữ liệu ra:984,984
00152.
00153. Dữ liệu vào:n=10000
00154. Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00155. int Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00156. int Tong(int n)
00157. {
00158.    if(n==0)
00159.       return 0;
00160.    int s = Tong(n-1);
00161.    return (s + n*n);
00162. }
```

Bài 002.Viết hàm đệ quy tính $S(n) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{(n-1)} + \frac{1}{n}$.

– Ta có:

+ $S(n) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{(n-1)} + \frac{1}{n}$.
+ $S(n-1) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{(n-1)}$.

– Suy ra: $S(n) = S(n-1) + \frac{1}{n}$.

– Điều kiện dừng: $S(0) = 0$.

– Dữ liệu test

```
00163. Dữ liệu vào: n=10
00164. Dữ liệu ra: 2.92897
00165.
00166. Dữ liệu vào: n=143
00167. Dữ liệu ra: 5.54355
00168.
00169. Dữ liệu vào: n=10000
00170. Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00171. float Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00172. float Tong(int n)
```

```
00173. {
00174.    if(n==0)
00175.       return 0;
00176.    float s = Tong(n-1);
00177.    return (s + (float)1/n);
00178. }
```

Bài 003.Viết hàm đệ quy tính $S(n) = \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2(n-1)} + \frac{1}{2n}$.

- Ta có:
  - $S(n) = \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2(n-1)} + \frac{1}{2n}$.
  - $S(n-1) = \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2(n-1)}$.
- Suy ra: $S(n) = S(n-1) + \frac{1}{2n}$.
- Điều kiện dừng: $S(0) = 0$.
- Dữ liệu test

```
00179. Dữ liệu vào: n=10
00180. Dữ liệu ra: 1.46448
00181.
00182. Dữ liệu vào: n=143
00183. Dữ liệu ra: 2.77178
00184.
00185. Dữ liệu vào: n=10000
00186. Dữ liệu ra: Tràn stack
```

- Khai báo hàm.

```
00187. float Tong(int);
```

- Định nghĩa hàm đệ quy.

```
00188. float Tong(int n)
00189. {
00190.    if(n==0)
00191.       return 0;
00192.    float s = Tong(n-1);
00193.    return (s + (float)1/(2*n));
00194. }
```

- Định nghĩa hàm đệ quy.

```
00195. float Tong(int n)
00196. {
00197.    if(n==0)
00198.       return 0;
00199.    float s = Tong(n-1);
00200.    return (s + (float)1/2/n);
```

```
00201. }
```

Bài 004.Viết hàm đệ quy tính $S(n) = 1 + \frac{1}{3} + \frac{1}{5} + \cdots + \frac{1}{2(n-1)+1} + \frac{1}{2n+1}$.

- Ta có:
  - $S(n) = 1 + \frac{1}{3} + \frac{1}{5} + \cdots + \frac{1}{2(n-1)+1} + \frac{1}{2n+1}$.
  - $S(n-1) = 1 + \frac{1}{3} + \frac{1}{5} + \cdots + \frac{1}{2(n-1)+1}$.
- Suy ra: $S(n) = S(n-1) + \frac{1}{2n+1}$.
- Điều kiện dừng: $S(0) = 1$.
- Dữ liệu test

```
00202. Dữ liệu vào: n=10
00203. Dữ liệu ra: 2.18087
00204.
00205. Dữ liệu vào: n=143
00206. Dữ liệu ra: 3.46666
00207.
00208. Dữ liệu vào:n=10000
00209. Dữ liệu ra: Tràn stack
```

- Khai báo hàm.

```
00210. float Tong(int);
```

- Định nghĩa hàm đệ quy.

```
00211. float Tong(int n)
00212. {
00213.     if(n==0)
00214.        return 1;
00215.     float s = Tong(n-1);
00216.     return (s + (float)1/(2*n+1));
00217. }
```

Bài 005.Viết hàm đệ quy tính $S(n) = \frac{1}{1\times2} + \frac{1}{2\times3} + \cdots + \frac{1}{(n-1)\times n} + \frac{1}{n\times(n+1)}$.

- Ta có:
  - $S(n) = \frac{1}{1\times2} + \frac{1}{2\times3} + \cdots + \frac{1}{(n-1)\times n} + \frac{1}{n\times(n+1)}$.
  - $S(n-1) = \frac{1}{1\times2} + \frac{1}{2\times3} + \cdots + \frac{1}{(n-1)\times n}$.
- Suy ra: $S(n) = S(n-1) + \frac{1}{n\times(n+1)}$.
- Điều kiện dừng: $S(0) = 0$.

– Dữ liệu test

```
00218. Dữ liệu vào: n=10
00219. Dữ liệu ra: 0.90909
00220.
00221. Dữ liệu vào: n=143
00222. Dữ liệu ra: 0.99305
00223.
00224. Dữ liệu vào: n = 10000
00225. Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00226. float Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00227. float Tong(int n)
00228. {
00229.    if(n==0)
00230.       return 0;
00231.    float s = Tong(n-1);
00232.    return (s + (float)1/n/(n+1));
00233. }
```

Bài 006.Viết hàm đệ quy tính $S(n) = \frac{1}{2} + \frac{2}{3} + \cdots + \frac{(n-1)}{n} + \frac{n}{(n+1)}$.

– Ta có:

+ $S(n) = \frac{1}{2} + \frac{2}{3} + \cdots + \frac{(n-1)}{n} + \frac{n}{(n+1)}$.
+ $S(n-1) = \frac{1}{2} + \frac{2}{3} + \cdots + \frac{(n-1)}{n}$.

– Suy ra: $S(n) = S(n-1) + \frac{n}{(n+1)}$.

– Điều kiện dừng: $S(0) = 0$.

– Dữ liệu test

```
00234. Dữ liệu vào: n=10
00235. Dữ liệu ra: 7.98012
00236.
00237. Dữ liệu vào: n=143
00238. Dữ liệu ra: 138.45000
00239.
00240. Dữ liệu vào: n = 10000
00241. Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00242. float Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00243. float Tong(int n)
```

```
00244.  {
00245.      if(n==0)
00246.         return 0;
00247.      float s = Tong(n-1);
00248.      return (s + (float)n/(n+1));
00249.  }
```

Bài 007.Viết hàm đệ quy tính $S(n) = \frac{1}{2} + \frac{3}{4} + \cdots + \frac{2(n-1)+1}{2(n-1)+2} + \frac{2n+1}{2n+2}$.

- Ta có:
  - $S(n) = \frac{1}{2} + \frac{3}{4} + \cdots + \frac{2(n-1)+1}{2(n-1)+2} + \frac{2n+1}{2n+2}$.
  - $S(n-1) = \frac{1}{2} + \frac{3}{4} + \cdots + \frac{2(n-1)+1}{2(n-1)+2}$.
- Suy ra: $S(n) = S(n-1) + \frac{2n+1}{2n+2}$.
- Điều kiện dừng: $S(0) = 0.5$
- Dữ liệu test

```
00250.  Dữ liệu vào: n=10
00251.  Dữ liệu ra: 9.49006
00252.
00253.  Dữ liệu vào: n=143
00254.  Dữ liệu ra: 141.22500
00255.
00256.  Dữ liệu vào: n = 10000
00257.  Dữ liệu ra: Tràn stack
```

- Khai báo hàm.

```
00258.  float Tong(int);
```

- Định nghĩa hàm đệ quy.

```
00259.  float Tong(int n)
00260.  {
00261.      if(n==0)
00262.         return 0.5;
00263.      float s = Tong(n-1);
00264.      return (s + (float)(2*n+1)/(2*n+2));
00265.  }
```

Bài 008.Viết hàm đệ quy tính $T(n) = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$.

- Ta có:
  - $T(n) = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$.
  - $T(n-1) = 1 \times 2 \times 3 \times \cdots \times (n-1)$.
- Suy ra: $T(n) = T(n-1) \times n$.
- Điều kiện dừng: $T(0) = 1$.

– Dữ liệu test

```
00266. Dữ liệu vào: n=10
00267. Dữ liệu ra: 3,628,800
00268.
00269. Dữ liệu vào: n=53
00270. Dữ liệu ra: Tràn số
00271.
00272. Dữ liệu vào: n = 10000
00273. Dữ liệu ra: Tràn số, Tràn stack.
```

– Khai báo hàm.

```
00274. int GiaiThua(int);
```

– Định nghĩa hàm đệ quy.

```
00275. int GiaiThua(int n)
00276. {
00277.    if(n==0)
00278.       return 1;
00279.    int T = GiaiThua(n-1);
00280.    return(T*n);
00281. }
```

Bài 009.Viết hàm đệ quy tính $T(x,n) = x^n$.

– Ta có:
  + $T(x,n) = x^n$.
  + $T(x,n-1) = x^{n-1}$.
– Suy ra: $T(x,n) = T(x,n-1) \times x$.
– Điều kiện dừng: $T(x,0) = 1$.
– Dữ liệu test

```
00282. Dữ liệu vào: x=2, n=10
00283. Dữ liệu ra: 1,024
00284.
00285. Dữ liệu vào: x= 3.6, n=11
00286. Dữ liệu ra: 1,316,217.03842
00287.
00288. Dữ liệu vào: x=1, n = 10,000
00289. Dữ liệu ra: Tràn stack.
```

– Định nghĩa hàm đệ quy.

```
00290. long double LuyThua(double,int);
```

– Định nghĩa hàm đệ quy.

```
00291. long double LuyThua(double x,int n)
00292. {
00293.    if(n==0)
```

```
00294.        return 1;
00295.     float T = LuyThua(x,n-1);
00296.     return (T*x);
00297.  }
```

Bài 010.Viết hàm đệ quy tính $S(n) = 1^3 + 2^3 + 3^3 + \cdots + n^3$.

- Ta có:
  - $S(n) = 1^3 + 2^3 + 3^3 + \cdots + (n-1)^3 + n^3$.
  - $S(n-1) = 1^3 + 2^3 + 3^3 + \cdots + (n-1)^3$.
- Suy ra: $S(n) = S(n-1) + n^3$.
- Điều kiện dừng: $S(0) = 0$.
- Dữ liệu test

```
00298.  Dữ liệu vào: n=10
00299.  Dữ liệu ra: 3,025
00300.
00301.  Dữ liệu vào: n=143
00302.  Dữ liệu ra: 106,007,616
00303.
00304.  Dữ liệu vào: n = 10000
00305.  Dữ liệu ra: Tràn stack
```

- Khai báo hàm.

```
00306.  int Tong(int);
```

- Định nghĩa hàm đệ quy.

```
00307.  int Tong(int n)
00308.  {
00309.     if(n==0)
00310.        return 0;
00311.     int s = Tong(n-1);
00312.     return (s + n*n*n);
00313.  }
```

Bài 011.Viết hàm đệ quy tính $S(n) = 1^4 + 2^4 + 3^4 + \cdots + n^4$.

- Ta có:
  - $S(n) = 1^4 + 2^4 + 3^4 + \cdots + (n-1)^4 + n^4$.
  - $S(n-1) = 1^4 + 2^4 + 3^4 + \cdots + (n-1)^4$.
- Suy ra: $S(n) = S(n-1) + n^4$.
- Điều kiện dừng: $S(0) = 0$.
- Dữ liệu test

```
00314.  Dữ liệu vào: n=10
00315.  Dữ liệu ra: 25,333
00316.
```

```
00317.  Dữ liệu vào: n=143
00318.  Dữ liệu ra: tràn số.
00319.
00320.  Dữ liệu vào: n = 10000
00321.  Dữ liệu ra: Tràn số, Tràn stack
```

– Khai báo hàm.

```
00322.  int Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00323.  int Tong(int n)
00324.  {
00325.      if(n==0)
00326.          return 0;
00327.      int s = Tong(n-1);
00328.      return (s + n*n*n*n);
00329.  }
```

Bài 012.Viết hàm đệ quy tính $T(n)=\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)\cdots\left(1+\frac{1}{n^2}\right)$.

– Ta có:

+ $T(n)=\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)\cdots\left(1+\frac{1}{(n-1)^2}\right)\left(1+\frac{1}{n^2}\right)$.
+ $T(n-1)=\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)\cdots\left(1+\frac{1}{(n-1)^2}\right)$.

– Suy ra: $T(n)=T(n-1)\left(1+\frac{1}{n^2}\right)$.

– Điều kiện dừng: $T(0)=1$.

– Dữ liệu test

```
00330.  Dữ liệu vào: n=10
00331.  Dữ liệu ra: 3.34285
00332.
00333.  Dữ liệu vào: n=143
00334.  Dữ liệu ra: 3.65055
00335.
00336.  Dữ liệu vào: n = 10000
00337.  Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00338.  float Tinh(int);
```

– Định nghĩa hàm đệ quy.

```
00339.  float Tinh(int n)
00340.  {
00341.      if(n==0)
00342.          return 1;
00343.      float t = Tinh(n-1);
```

```
00344.     return (t * (1 + (float)1/(n*n));
00345. }
```

Bài 013.Viết hàm đệ quy tính $S(n) = 1.2 + 2.3 + 3.4 + \cdots + n(n+1)$.

- Ta có:
  - $S(n) = 1.2 + 2.3 + 3.4 + \cdots + (n-1)n + n(n+1)$.
  - $S(n-1) = 1.2 + 2.3 + 3.4 + \cdots + (n-1)n$.
- Suy ra: $S(n) = S(n-1) + n(n+1)$.
- Điều kiện dừng: $S(0) = 0$.
- Dữ liệu test

```
00346. Dữ liệu vào: n=10
00347. Dữ liệu ra: 440
00348.
00349. Dữ liệu vào: n=143
00350. Dữ liệu ra: 995280
00351.
00352. Dữ liệu vào: n = 10000
00353. Dữ liệu ra: Tràn stack
```

- Khai báo hàm.

```
00354. int Tong(int);
```

- Định nghĩa hàm đệ quy.

```
00355. int Tong(int n)
00356. {
00357.     if(n==0)
00358.        return 0;
00359.     int s = Tong(n-1);
00360.     return (s + n*(n+1));
00361. }
```

Bài 014.Viết hàm đệ quy tính $S(n) = 1.2.3 + 2.3.4 + 3.4.5 + \cdots + n(n+1)(n+2)$.

- Ta có:
  - $S(n) = 1.2.3 + 2.3.4 + 3.4.5 + \cdots + n(n+1)(n+2)$.
  - $S(n-1) = 1.2.3 + 2.3.4 + 3.4.5 + \cdots + (n-1)n(n+1)$.
- Suy ra: $S(n) = S(n-1) + n(n+1)(n+2)$.
- Điều kiện dừng: $S(0) = 0$.
- Dữ liệu test

```
00362. Dữ liệu vào: n=10
00363. Dữ liệu ra: 4290
```

```
00364.
00365. Dữ liệu vào: n=143
00366. Dữ liệu ra: 108,983,160
00367.
00368. Dữ liệu vào: n = 10000
00369. Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00370. int Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00371. int Tong(int n)
00372. {
00373.     if(n==0)
00374.        return 0;
00375.     int s = Tong(n-1);
00376.     return (s + n*(n+1)*(n+2));
00377. }
```

Bài 015. Viết hàm đệ quy tính $S(n) = \frac{1}{1.2} + \frac{1}{2.3} + \frac{1}{3.4} + \cdots + \frac{1}{(n-1).n} + \frac{1}{n.(n+1)}$.

– Ta có:

+ $S(n) = \frac{1}{1.2} + \frac{1}{2.3} + \frac{1}{3.4} + \cdots + \frac{1}{(n-1).n} + \frac{1}{n.(n+1)}$.
+ $S(n-1) = \frac{1}{1.2} + \frac{1}{2.3} + \frac{1}{3.4} + \cdots + \frac{1}{(n-1).n}$.

– Suy ra: $S(n) = S(n-1) + \frac{1}{n.(n+1)}$.

– Điều kiện dừng: $S(0) = 0$.

– Dữ liệu test

```
00378. Dữ liệu vào: n=10
00379. Dữ liệu ra: 0.90909
00380.
00381. Dữ liệu vào: n=143
00382. Dữ liệu ra: 0.99305
00383.
00384. Dữ liệu vào: n = 10000
00385. Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00386. float Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00387. float Tong(int n)
00388. {
```

```
00389.     if(n==0)
00390.        return 0;
00391.     float s = Tong(n-1);
00392.     return (s + (float)1/n/(n+1));
00393. }
```

Bài 016.Viết hàm đệ quy tính $S(n) = \frac{1}{1.2.3} + \frac{1}{2.3.4} + \frac{1}{3.4.5} + \cdots + \frac{1}{(n-1).n.(n+1)} + \frac{1}{n.(n+1).(n+2)}$.

- Ta có:
  - $S(n) = \frac{1}{1.2.3} + \frac{1}{2.3.4} + \cdots + \frac{1}{(n-1).n.(n+1)} + \frac{1}{n.(n+1).(n+2)}$.
  - $S(n-1) = \frac{1}{1.2.3} + \frac{1}{2.3.4} + \cdots + \frac{1}{(n-1).n.(n+1)}$.
- Suy ra: $S(n) = S(n-1) + \frac{1}{n.(n+1).(n+2)}$.
- Điều kiện dừng: $S(0) = 0$.
- Dữ liệu test

```
00394. Dữ liệu vào: n=10
00395. Dữ liệu ra: 0.24621
00396.
00397. Dữ liệu vào: n=143
00398. Dữ liệu ra: 0.24997
00399.
00400. Dữ liệu vào: n = 10000
00401. Dữ liệu ra: Tràn stack
```

- Khai báo hàm.

```
00402. float Tong(int);
```

- Định nghĩa hàm đệ quy.

```
00403. float Tong(int n)
00404. {
00405.     if(n==0)
00406.        return 0;
00407.     float s = Tong(n-1);
00408.     return (s + (float)1/n/(n+1)/(n+2));
00409. }
```

Bài 017.Viết hàm đệ quy tính $S(n) = \frac{1}{1.2.3.4} + \frac{1}{2.3.4.5} + \frac{1}{3.4.5.6} + \cdots + \frac{1}{(n-1).n.(n+1)(n+2)} + \frac{1}{n.(n+1).(n+2).(n+3)}$

- Ta có:

+ $S(n) = \frac{1}{1.2.3.4} + \frac{1}{2.3.4.5} + \frac{1}{3.4.5.6} + \cdots + \frac{1}{(n-1).n.(n+1)(n+2)} + \frac{1}{n.(n+1).(n+2).(n+3)}$

.

+ $S(n-1) = \frac{1}{1.2.3.4} + \frac{1}{2.3.4.5} + \cdots + \frac{1}{(n-1).n.(n+1)(n+2)}$.

– Suy ra: $S(n) = S(n-1) + \frac{1}{n.(n+1).(n+2).(n+3)}$.

– Điều kiện dừng: $S(0) = 0$.

– Dữ liệu test

```
00410. Dữ liệu vào: n=10
00411. Dữ liệu ra: 0.05536
00412.
00413. Dữ liệu vào: n=143
00414. Dữ liệu ra: 0.05555
00415.
00416. Dữ liệu vào: n = 10000
00417. Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00418. float Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00419. float Tong(int n)
00420. {
00421.     if(n==0)
00422.        return 0;
00423.     float s = Tong(n-1);
00424.     return (s + (float)1/n/(n+1)/(n+2)/(n+3));
00425. }
```

Bài 018.Viết hàm đệ quy tính $S(n) = \frac{1}{\sqrt{1}+\sqrt{2}} + \frac{1}{\sqrt{2}+\sqrt{3}} + \cdots + \frac{1}{\sqrt{n}+\sqrt{n+1}}$.

– Ta có:

+ $S(n) = \frac{1}{\sqrt{1}+\sqrt{2}} + \frac{1}{\sqrt{2}+\sqrt{3}} + \cdots + \frac{1}{\sqrt{n-1}+\sqrt{n}} + \frac{1}{\sqrt{n}+\sqrt{n+1}}$.

+ $S(n-1) = \frac{1}{\sqrt{1}+\sqrt{2}} + \frac{1}{\sqrt{2}+\sqrt{3}} + \cdots + \frac{1}{\sqrt{n-1}+\sqrt{n}}$.

– Suy ra: $S(n) = S(n-1) + \frac{1}{\sqrt{n}+\sqrt{n+1}}$.

– Điều kiện dừng: $S(0) = 0$.

– Dữ liệu test

```
00426. Dữ liệu vào: n=10
00427. Dữ liệu ra: 2.31662
00428.
00429. Dữ liệu vào: n=143
```

```
00430.  Dữ liệu ra: 11
00431.
00432.  Dữ liệu vào: n = 10000
00433.  Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00434.  float Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00435.  float Tong(int n)
00436.  {
00437.      if(n==0)
00438.         return 0;
00439.      float s = Tong(n-1);
00440.      return (s + 1/(sqrt(n)+sqrt(n+1)));
00441.  }
```

Bài 019.Viết hàm đệ quy tính $S(n) = \frac{1}{2\sqrt{1}+1\sqrt{2}} + \frac{1}{3\sqrt{2}+2\sqrt{3}} + \cdots + \frac{1}{(n+1)\sqrt{n}+n\sqrt{n+1}}$.

– Ta có:

+ $S(n) = \frac{1}{2\sqrt{1}+1\sqrt{2}} + \frac{1}{3\sqrt{2}+2\sqrt{3}} + \cdots + \frac{1}{n\sqrt{n-1}+(n-1)\sqrt{n}} + \frac{1}{(n+1)\sqrt{n}+n\sqrt{n+1}}$.
+ $S(n-1) = \frac{1}{2\sqrt{1}+1\sqrt{2}} + \frac{1}{3\sqrt{2}+2\sqrt{3}} + \cdots + \frac{1}{n\sqrt{n-1}+(n-1)\sqrt{n}}$.

– Suy ra: $S(n) = S(n-1) + \frac{1}{(n+1)\sqrt{n}+n\sqrt{n+1}}$.

– Điều kiện dừng: $S(0) = 0$.

– Dữ liệu test

```
00442.  Dữ liệu vào: n=10
00443.  Dữ liệu ra: 0.69848
00444.
00445.  Dữ liệu vào: n=143
00446.  Dữ liệu ra: 0.91666
00447.
00448.  Dữ liệu vào: n = 10000
00449.  Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00450.  float Tong(int);
```

– Định nghĩa hàm đệ quy.

```
00451.  float Tong(int n)
00452.  {
00453.      if(n==0)
```

```
00454.        return 0;
00455.    float s = Tong(n-1);
00456.    return (s+1/((n+1)*sqrt(n)+n*sqrt(n+1)));
00457. }
```

Bài 020.Viết hàm đệ quy tính $S(n) = \sqrt{1+\frac{1}{1^2}+\frac{1}{2^2}} + \sqrt{1+\frac{1}{2^2}+\frac{1}{3^2}} + \cdots + \sqrt{1+\frac{1}{n^2}+\frac{1}{(n+1)^2}}$.

- Ta có:
  - $S(n) = \sqrt{1+\frac{1}{1^2}+\frac{1}{2^2}} + \sqrt{1+\frac{1}{2^2}+\frac{1}{3^2}} + \cdots + \sqrt{1+\frac{1}{n^2}+\frac{1}{(n+1)^2}}$.
  - $S(n-1) = \sqrt{1+\frac{1}{1^2}+\frac{1}{2^2}} + \sqrt{1+\frac{1}{2^2}+\frac{1}{3^2}} + \cdots + \sqrt{1+\frac{1}{(n-1)^2}+\frac{1}{n^2}}$.
- Suy ra: $S(n) = S(n-1) + \sqrt{1+\frac{1}{n^2}+\frac{1}{(n+1)^2}}$.
- Điều kiện dừng: $S(0) = 0$.
- Dữ liệu test

```
00458. Dữ liệu vào: n=10
00459. Dữ liệu ra: 10.9091
00460.
00461. Dữ liệu vào: n=143
00462. Dữ liệu ra: 143.993
00463.
00464. Dữ liệu vào: n = 10000
00465. Dữ liệu ra: Tràn stack
```

- Khai báo hàm.

```
00466. float Tong(int);
```

- Định nghĩa hàm đệ quy.

```
00467. float Tong(int n)
00468. {
00469.    if(n==0)
00470.       return 0;
00471.    float s = Tong(n-1);
00472.    return (s + sqrt(1 + (float)1/n/n +
00473.                    (float)1/(n+1)/(n+1)));
00474. }
```

Bài 021.Viết hàm đệ quy tính $S(x,n) = x(x+1)\ldots(x+n-1)(x+n)$.

- Ta có:

+ $S(x,n) = x(x+1)\ldots(x+n-1)(x+n)$.
+ $S(x,n-1) = x(x+1)\ldots(x+n-1)$.

– Suy ra: $S(x,n) = S(x,n-1)(x+n)$.
– Điều kiện dừng: $S(x,0) = x$.
– Dữ liệu test

```
00475. Dữ liệu vào: n=10, x=3.1
00476. Dữ liệu ra: 3,677,319,291.7035
00477.
00478. Dữ liệu vào: n=143, x=3.1
00479. Dữ liệu ra: Tràn số
00480.
00481. Dữ liệu vào: n = 10000, x=3.1
00482. Dữ liệu ra: Tràn số & Tràn stack
```

– Khai báo hàm.

```
00483. float Tinh(float,int);
```

– Định nghĩa hàm đệ quy.

```
00484. float Tinh(float x,int n)
00485. {
00486.    if(n==0)
00487.       return x;
00488.    float s = Tinh(x,n-1);
00489.    return (s * (x+n);
00490. }
```

01.04.02 Kỹ thuật Đệ quy Tính toán với Số nguyên

01.04.02.1 Bài cơ sở Kỹ thuật Đệ quy Tính toán với Số nguyên

> Bài cơ sở 004. Hãy tính tổng các chữ số chẵn của số nguyên n.

– Gọi:
+ $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
+ $s(n)$ là tổng các chữ số chẵn của số nguyên n.

– Ta có:
+ $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
+ $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$

– Suy ra:
+ $s(n) = s(n/10)$ nếu a_1 là chữ số lẻ.
+ $s(n) = s(n/10) + a_1$ nếu a_1 là chữ số chẵn.

– Điều kiện dừng: $s(0) = 0$.

– Khai báo hàm.

```
00491. int TongChan(int);
```

– Định nghĩa hàm đệ quy.

```
00492. int TongChan(int n)
00493. {
00494.     n = abs(n);
00495.     if(n==0)
00496.        return 0;
00497.     int s = TongChan(n/10);
00498.     int dv = n%10;
00499.     if(dv%2==0)
00500.        s = s + dv;
00501.     return s;
00502. }
```

Chương trình 004. Viết chương trình bằng kỹ thuật đệ quy tính tổng các chữ số chẵn của số nguyên n.

– Chương trình

```
00503. #include <iostream>
00504. using namespace std;
00505.
00506. int TongChan(int);
00507.
00508. int main()
00509. {
00510.     int n;
00511.     cout << "Nhap n: ";
00512.     cin >> n;
00513.
00514.     int kq = TongChan(n);
00515.     cout << "Ket qua: " << kq;
00516.     return 1;
00517. }
00518.
00519. int TongChan(int n)
00520. {
00521.     n = abs(n);
00522.     if (n == 0)
00523.        return 0;
00524.     int dv = n % 10;
00525.     int s = TongChan(n / 10);
```

```
00526.     if (dv % 2 == 0)
00527.        s = s + dv;
00528.     return s;
00529.  }
```

01.04.02.2 Bài tập Kỹ thuật Đệ quy tính toán Số nguyên

Bài 022.Viết hàm đệ quy tính tổng các chữ số của số nguyên n.

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $s(n)$ là tổng các chữ số của số nguyên n.
- Ta có:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
- Suy ra: $s(n) = s(n/10) + a_1$
- Điều kiện dừng: $s(0) = 0$.
- Dữ liệu test

```
00530.  Dữ liệu vào: n = 0
00531.  Dữ liệu ra: 0
00532.
00533.  Dữ liệu vào: n = 143
00534.  Dữ liệu ra: 8
00535.
00536.  Dữ liệu vào: n = -987
00537.  Dữ liệu ra: 24
```

- Khai báo hàm.

```
00538.  int TongChuSo(int n)
```

- Định nghĩa hàm đệ quy.

```
00539.  int TongChuSo(int n)
00540.  {
00541.     n = abs(n);
00542.     if(n==0)
00543.        return 0;
00544.     int s = TongChuSo(n/10);
00545.     int dv = n%10;
00546.     return s + dv;
00547.  }
```

- Định nghĩa hàm đệ quy.

```
00548.  int TongChuSo(int n)
00549.  {
00550.     n = abs(n);
```

```
00551.     if(n==0)
00552.        return 0;
00553.     return TongChuSo(n/10) + n%10;
00554.  }
```

Bài 023.Viết hàm đệ quy tính tích các chữ số của số nguyên n.

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $T(n)$ là tích các chữ số của số nguyên n.
- Ta có:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
- Suy ra: $T(n) = T(n/10) * abs(a_1)$
- Điều kiện dừng: $T(n) = abs(n)$ khi khi $-9 \le n \le 9$.
- Dữ liệu test

```
00555. Dữ liệu vào: n = 0
00556. Dữ liệu ra: 0
00557.
00558. Dữ liệu vào: n = 143
00559. Dữ liệu ra: 12
00560.
00561. Dữ liệu vào: n = -987
00562. Dữ liệu ra: 504
```

- Khai báo hàm.

```
00563. int TichChuSo(int n)
```

- Định nghĩa hàm đệ quy.

```
00564. int TichChuSo(int n)
00565. {
00566.     n = abs(n);
00567.     if(n<=9)
00568.        return n;
00569.     int T = TichChuSo(n/10);
00570.     int dv = n%10;
00571.     return T * dv;
00572. }
```

Bài 024.Viết hàm đệ quy tính tổng các chữ số chẵn của số nguyên n.

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - s(n) là tổng các chữ số chẵn của số nguyên n.
- Ta có:

+ $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
+ $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$

– Suy ra:
+ $s(n) = s(n/10)$ khi a_1 là số lẻ.
+ $s(n) = s(n/10) + a_1$ khi a_1 là số chẵn.

– Điều kiện dừng:
+ $s(n) = 0$ khi $-9 \le n \le 9$ và n lẻ.
+ $s(n) = n$ khi $-9 \le n \le 9$ và n chẵn.

– Dữ liệu test

```
00573. Dữ liệu vào: n=0
00574. Dữ liệu ra: 0
00575.
00576. Dữ liệu vào: n=143
00577. Dữ liệu ra: 4
00578.
00579. Dữ liệu vào: n = -987
00580. Dữ liệu ra: 8
```

– Khai báo hàm.

```
00581. int TongSoChan(int);
```

– Định nghĩa hàm đệ quy.

```
00582. int TongSoChan(int n)
00583. {
00584.    n = abs(n);
00585.    if(n<=9)
00586.    {
00587.       if(n%2!=0)
00588.          return 0;
00589.       return n;
00590.    }
00591.    int s = TongSoChan(n/10);
00592.    int dv = n%10;
00593.    if(dv%2==0)
00594.       return s + dv;
00595.    return s;
00596. }
```

Bài 025. Viết hàm đệ quy tính tích các chữ số lẻ của số nguyên n.

– Gọi:
+ $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
+ $T(n)$ là tích các chữ số lẻ của số nguyên n.

– Ta có:

+ $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
+ $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$

– Suy ra:
+ $T(n) = T(n/10)$ khi a_1 là số chẵn.
+ $T(n) = T(n/10) * \text{abs}(a_1)$ khi a_1 là số lẻ.

– Điều kiện dừng:
+ $\text{T}(n) = \text{abs}(n)$ khi khi $-9 \le n \le 9$ và n lẻ.
+ $\text{T}(n) = 1$ khi khi $-9 \le n \le 9$ và n chẵn.

– Dữ liệu test

```
00597. Dữ liệu vào: n=0
00598. Dữ liệu ra: 0
00599.
00600. Dữ liệu vào: n=143
00601. Dữ liệu ra: 3
00602.
00603. Dữ liệu vào: n = -987
00604. Dữ liệu ra: 63
```

– Khai báo hàm.

```
00605. int TichSoLe(int n)
```

– Định nghĩa hàm đệ quy.

```
00606. int TichSoLe(int n)
00607. {
00608.    n = abs(n);
00609.    if(n<=9)
00610.    {
00611.       if(n%2!=0)
00612.          return n;
00613.       return 1;
00614.    }
00615.    int T = TichSoLe(n/10);
00616.    int dv = n%10;
00617.    if(dv%2!=0)
00618.       return T * dv;
00619.    return T;
00620. }
```

01.04.01 Kỹ thuật Đệ quy tính tổng dưới căn

01.04.01.1 Bài cơ sở Kỹ thuật Đệ quy Tính tổng Tính tích

Bài cơ sở 005. Viết hàm đệ quy tính

$$S(n) = \sqrt{2 + \sqrt{2 + \cdots + \sqrt{2 + \sqrt{2}}}}$$ có n dấu căn.

- Ta có:
  - $S(0) = 0$.
  - $S(1) = \sqrt{2}$.
  - $S(2) = \sqrt{2 + \sqrt{2}}$.
  - $S(3) = \sqrt{2 + \sqrt{2 + \sqrt{2}}}$.
  - $S(4) = \sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2}}}}$.
- Nhận xét:
  - $S(1) = \sqrt{2 + S(0)}$.
  - $S(2) = \sqrt{2 + S(1)}$.
  - $S(3) = \sqrt{2 + S(2)}$.
  - $S(4) = \sqrt{2 + S(3)}$.
- Tổng quát: $S(n) = \sqrt{2 + S(n-1)}$.
- Điều kiện dừng: $S(0) = 0$.
- Dữ liệu test

```
00621. Dữ liệu vào: n=5
00622. Dữ liệu ra: 1.99759
00623.
00624. Dữ liệu vào: n=10
00625. Dữ liệu ra: 2
00626.
00627. Dữ liệu vào: n = 10000
00628. Dữ liệu ra: Tràn Stack
```

- Khai báo hàm.

```
00629. float Tinh(int);
```

- Định nghĩa hàm đệ quy.

```
00630. float Tinh(int n)
00631. {
00632.     if(n==0)
```

```
00633.        return 0;
00634.    return sqrt(2 + Tinh(n-1));
00635. }
```

Chương trình 005. Viết chương trình bằng kỹ thuật đệ quy $S(n) = \sqrt{2 + \sqrt{2 + \cdots + \sqrt{2 + \sqrt{2}}}}$ có n dấu căn.

– Chương trình

```
00636. #include <iostream>
00637. #include <iomanip>
00638. using namespace std;
00639.
00640. float Tinh(int);
00641.
00642. int main()
00643. {
00644.    int n;
00645.    cout << "Nhap n: ";
00646.    cin >> n;
00647.
00648.    float kq = Tinh(n);
00649.    cout << setw(8) << setprecision(5);
00650.    cout << "Ket qua: " << kq;
00651.    return 1;
00652. }
00653.
00654. float Tinh(int n)
00655. {
00656.    if (n == 0)
00657.       return 0;
00658.    return sqrt(2 + Tinh(n - 1));
00659. }
```

01.04.01.2 Bài tập Kỹ thuật Đệ quy tính Tổng dưới căn

Bài 026.Viết hàm đệ quy tính $S(n) = \sqrt{n + \sqrt{(n-1) + \cdots + \sqrt{2 + \sqrt{1}}}}$ có n dấu căn.

– Ta có:

+ $S(0) = 0$.

+ $S(1) = \sqrt{1}$.
+ $S(2) = \sqrt{2 + \sqrt{1}}$.
+ $S(3) = \sqrt{3 + \sqrt{2 + \sqrt{1}}}$.
+ $S(4) = \sqrt{4 + \sqrt{3 + \sqrt{2 + \sqrt{1}}}}$.
+ $S(5) = \sqrt{5 + \sqrt{4 + \sqrt{3 + \sqrt{2 + \sqrt{1}}}}}$.

– Nhận xét:
+ $S(1) = \sqrt{1 + S(0)}$.
+ $S(2) = \sqrt{2 + S(1)}$.
+ $S(3) = \sqrt{3 + S(2)}$.
+ $S(4) = \sqrt{4 + S(3)}$.
+ $S(5) = \sqrt{5 + S(4)}$.

– Tổng quát: $S(n) = \sqrt{n + S(n-1)}$.
– Điều kiện dừng: $S(0) = 0$.
– Dữ liệu test

```
00660.  Dữ liệu vào: n=10
00661.  Dữ liệu ra: 3.67598
00662.
00663.  Dữ liệu vào: n=100
00664.  Dữ liệu ra: 10.51
00665.
00666.  Dữ liệu vào: n = 10000
00667.  Dữ liệu ra: Tràn số & Tràn Stack
```

– Khai báo hàm.

```
00668.  float Tinh(int);
```

– Định nghĩa hàm đệ quy.

```
00669.  float Tinh(int n)
00670.  {
00671.      if(n==0)
00672.          return 0;
00673.      return sqrt(n + Tinh(n-1));
00674.  }
```

Bài 027.Viết hàm đệ quy tính $S(n) = \sqrt[n]{n + \sqrt[n-1]{(n-1) + \cdots + \sqrt[3]{3 + \sqrt{2}}}}$ có $(n-1)$ dấu căn.

- Ta có:
  - $S(1) = 0$
  - $S(2) = \sqrt{2}$
  - $S(3) = \sqrt[3]{3 + \sqrt{2}}$
  - $S(4) = \sqrt[4]{4 + \sqrt[3]{3 + \sqrt{2}}}$
  - $S(5) = \sqrt[5]{5 + \sqrt[4]{4 + \sqrt[3]{3 + \sqrt{2}}}}$
- Nhận xét:
  - $S(1) = 0$.
  - $S(2) = \sqrt{2 + S(1)}$.
  - $S(3) = \sqrt[3]{3 + S(2)}$.
  - $S(4) = \sqrt[4]{4 + S(3)}$.
  - $S(5) = \sqrt[5]{5 + S(4)}$.
- Tổng quát: $S(n) = \sqrt[n]{n + S(n-1)}$.
- Điều kiện dừng: $S(1) = 0$.
- Dữ liệu test

```
00675. Dữ liệu vào: n=10
00676. Dữ liệu ra: 1.27436
00677.
00678. Dữ liệu vào: n=143
00679. Dữ liệu ra: 1.03537
00680.
00681. Dữ liệu vào: n = 10000
00682. Dữ liệu ra: Tràn Stack
```

- Khai báo hàm.

```
00683. float Tinh(int);
```

- Định nghĩa hàm đệ quy.

```
00684. float Tinh(int n)
00685. {
00686.    if(n==1)
00687.       return 0;
00688.    return pow((n + Tinh(n-1)),(float)1/n);
00689. }
```

Bài 028.Viết hàm đệ quy tính $S(n) = \sqrt[n+1]{n + \sqrt[n]{(n-1) + \cdots + \sqrt[3]{2 + \sqrt{1}}}}$ có n dấu căn.

- Ta có:
  - $S(0) = 0.$
  - $S(1) = \sqrt{1}.$
  - $S(2) = \sqrt[3]{2 + \sqrt{1}}.$
  - $S(3) = \sqrt[4]{3 + \sqrt[3]{2 + \sqrt{1}}}.$
  - $S(4) = \sqrt[5]{4 + \sqrt[4]{3 + \sqrt[3]{2 + \sqrt{1}}}}.$
  - $S(5) = \sqrt[6]{5 + \sqrt[5]{4 + \sqrt[4]{3 + \sqrt[3]{2 + \sqrt{1}}}}}.$
- Nhận xét:
  - $S(0) = 0.$
  - $S(1) = \sqrt{1 + S(0)}.$
  - $S(2) = \sqrt[3]{2 + S(1)}.$
  - $S(3) = \sqrt[4]{3 + S(2)}.$
  - $S(4) = \sqrt[5]{4 + S(3)}.$
  - $S(5) = \sqrt[6]{5 + S(4)}.$
- Tổng quát: $S(n) = \sqrt[n+1]{n + S(n-1)}.$
- Dữ liệu test

```
00690.  Dữ liệu vào: n=10
00691.  Dữ liệu ra: 1.24624
00692.
00693.  Dữ liệu vào: n=143
00694.  Dữ liệu ra: 1.03511
00695.
00696.  Dữ liệu vào: n = 10000
00697.  Dữ liệu ra: Tràn Stack
```

- Khai báo hàm.

```
00698.  float Tinh(int);
```

- Định nghĩa hàm đệ quy.

```
00699.  float Tinh(int n)
00700.  {
```

```
00701.     if(n==0)
00702.        return 0;
00703.     return pow((n+Tinh(n-1)),(float)1/(n+1));
00704.  }
```

Bài 029.Viết hàm đệ quy tính $S(n) = \frac{1}{1+\frac{1}{1+\frac{1}{\ddots+\frac{1}{1+1}}}}$có n dấu phân số.

- Ta có:
  - $S(0) = 1$.
  - $S(1) = \frac{1}{1+1}$.
  - $S(2) = \frac{1}{1+\frac{1}{1+1}}$.
  - $S(3) = \frac{1}{1+\frac{1}{1+\frac{1}{1+1}}}$.
  - $S(4) = \frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+1}}}}$.
  - $S(5) = \frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+1}}}}}$.
- Nhận xét:
  - $S(0) = 1$.
  - $S(1) = \frac{1}{1+S(0)}$.
  - $S(2) = \frac{1}{1+S(1)}$.
  - $S(3) = \frac{1}{1+S(2)}$.
  - $S(4) = \frac{1}{1+S(3)}$.
  - $S(5) = \frac{1}{1+S(4)}$.
- Tổng quát: $S(n) = \frac{1}{1+S(n-1)}$.
- Điều kiện dừng: $S(0) = 1$.
- Dữ liệu test

```
00705.  Dữ liệu vào: n=10
00706.  Dữ liệu ra: 0.61805
00707.
00708.  Dữ liệu vào: n=143
00709.  Dữ liệu ra: 0.618034
00710.
00711.  Dữ liệu vào: n = 10000
```

```
00712.  Dữ liệu ra: Tràn Stack
```

– Khai báo hàm.

```
00713.  float Tinh(int);
```

– Định nghĩa hàm đệ quy.

```
00714.  float Tinh(int n)
00715.  {
00716.      if(n==0)
00717.          return 1;
00718.      return 1/(1+Tinh(n-1));
00719.  }
```

01.04.02 Kỹ thuật Đệ quy tính tổng đan dấu

01.04.02.1 Bài cơ sở Kỹ thuật Đệ quy Tính tổng đan dấu

Bài cơ sở 006. Tính tổng sau:

– Khai báo hàm.

```
00720.  int Tinh(int);
```

– Định nghĩa hàm đệ quy.

```
00721.  int Tinh(int n)
00722.  {
00723.      if(n==1)
00724.          return 2;
00725.      return TinhAn(n-1) + 2*n + 1;
00726.  }
```

Chương trình 006. Viết chương trình

– Chương trình

```
00727.
```

01.04.02.2 Bài tập Kỹ thuật Đệ quy Tính tổng đan dấu

01.04.03 Kỹ thuật Đệ quy trên Dãy số

01.04.03.1 Bài cơ sở Kỹ thuật Đệ quy trên Dãy số

Bài cơ sở 007. Tính số hạng thứ n của dãy

$$\begin{cases} a_1 = 2 \\ a_n = a_{n-1} + 2n + 1 \ (n \geq 2) \end{cases}$$

– Dữ liệu test

```
Dữ liệu vào: n=10
Dữ liệu ra: 119

Dữ liệu vào: n=20
Dữ liệu ra: 439

Dữ liệu vào: n = 4000
Dữ liệu ra: Tràn Stack
```

– Khai báo hàm.

```
int TinhAn(int);
```

– Định nghĩa hàm đệ quy.

```
int TinhAn(int n)
{
   if(n==1)
      return 2;
   return TinhAn(n-1) + 2*n + 1;
}
```

Chương trình 007. Viết chương trình bằng kỹ thuật đệ quy tính số hạng thứ n của dãy: $\begin{cases} a_1 = 2 \\ a_n = a_{n-1} + 2n + 1 \ \ (n \geq 2) \end{cases}$

– Chương trình

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

float TinhAn(int);

int main()
{
   int n;
   cout << "Nhap n: ";
   cin >> n;

   float kq = TinhAn(n);
   cout << "Ket qua: " << kq;
   return 1;
}

float TinhAn(int n)
```

```
{
   if (n == 1)
      return 2;
   return TinhAn(n - 1) + 2 * n + 1;;
}
```

01.04.03.2 Bài tập Kỹ thuật Đệ quy trên Dãy số

Bài 030.Viết hàm đệ quy tính số hạng thứ n của dãy

$$\begin{cases} a_1 = 2 \\ a_n = a_{n-1} + 2n + 1 \ (n \geq 2) \end{cases}$$

– Dữ liệu test

```
Dữ liệu vào: n=10
Dữ liệu ra: 119

Dữ liệu vào: n=143
Dữ liệu ra: 20,734

Dữ liệu vào: n = 10000
Dữ liệu ra: Tràn Stack
```

– Khai báo hàm.

```
int TinhAn(int);
```

– Định nghĩa hàm đệ quy.

```
int TinhAn(int n)
{
   if(n==1)
      return 2;
   return TinhAn(n-1) + 2*n + 1;
}
```

Bài 031.Viết hàm đệ quy tính số hạng thứ n của dãy

$$\begin{cases} a_1 = -2 \\ a_n = 5a_{n-1} + 12 \ (n \geq 2) \end{cases}$$

– Dữ liệu test

```
Dữ liệu vào: n=10
Dữ liệu ra: 1,953,122

Dữ liệu vào: n=143
Dữ liệu ra: 509,587,430

Dữ liệu vào: n = 10000
Dữ liệu ra: Tràn Stack
```

– Khai báo hàm.

```
00789.  int TinhAn(int);
```

– Định nghĩa hàm đệ quy.

```
00790.  int TinhAn(int n)
00791.  {
00792.      if(n==1)
00793.          return -2;
00794.      return 5*TinhAn(n-1) + 12;
00795.  }
```

Bài 032.Viết hàm đệ quy tính số hạng thứ n của dãy

$$\begin{cases} a_1 = 2 \\ a_n = \frac{-9a_{n-1}-24}{5a_{n-1}+13} \quad (n \geq 2) \end{cases}$$

– Dữ liệu test

```
00796.  Dữ liệu vào: n=10
00797.  Dữ liệu ra: -1.99998
00798.
00799.  Dữ liệu vào: n=143
00800.  Dữ liệu ra: -2
00801.
00802.  Dữ liệu vào: n = 10000
00803.  Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00804.  float TinhAn(int);
```

– Định nghĩa hàm đệ quy.

```
00805.  float TinhAn(int n)
00806.  {
00807.      if(n==1)
00808.          return 2;
00809.      float at = TinhAn(n-1);
00810.      return (-9*at-24)/(5*at + 13);
00811.  }
```

Bài 033.Viết hàm đệ quy tính số hạng thứ n của dãy

$$\begin{cases} a_1 = 2 \\ a_n = \frac{a_{n-1}^2+2}{2a_{n-1}} \quad (n \geq 2) \end{cases}$$

– Dữ liệu test

```
00812.  Dữ liệu vào: n=3
00813.  Dữ liệu ra: 1.41667
```

```
00814.
00815. Dữ liệu vào: n=143
00816. Dữ liệu ra: 1.41421
00817.
00818. Dữ liệu vào: n = 10000
00819. Dữ liệu ra: Tràn stack
```

– Khai báo hàm.

```
00820. float TinhAn(int);
```

– Định nghĩa hàm đệ quy.

```
00821. float TinhAn(int n)
00822. {
00823.    if(n==1)
00824.       return 2;
00825.    float at = TinhAn(n-1);
00826.    return (at*at+2)/(2*at);
00827. }
```

Bài 034.Viết hàm đệ quy tính số hạng thứ n của dãy

$$\begin{cases} a_1 = 2 \\ a_n = 5a_{n-1} + \sqrt{24a_{n-1}^2 - 8} \quad (n \geq 2) \end{cases}$$

– Dữ liệu test

```
00828. Dữ liệu vào: n=3
00829. Dữ liệu ra: 191.80800
00830.
00831. Dữ liệu vào: n=10
00832. Dữ liệu ra: 1,786,485,174.60441
00833.
00834. Dữ liệu vào: n=143
00835. Dữ liệu ra: Tràn số
00836.
00837. Dữ liệu vào: n = 10000
00838. Dữ liệu ra: Tràn số, Tràn stack
```

– Khai báo hàm.

```
00839. float TinhAn(int);
```

– Định nghĩa hàm đệ quy.

```
00840. float TinhAn(int n)
00841. {
00842.    if(n==1)
00843.       return 2;
00844.    float at = TinhAn(n-1);
00845.    return (5*at + sqrt(24*at*at-8));
```

```
00846.  }
```

Bài 035. Viết hàm đệ quy xuất ra dãy giá trị Hailstone sequences – Collatz conjecture (dãy mưa đá) của một số nguyên dương n. Biết rằng dãy Hailstone được định nghĩa như sau:

$$\begin{cases} a_1 = n \\ a_n = \frac{a_{n-1}}{2} & \text{khi } a_{n-1} = 2k \quad (n \geq 2) \\ a_n = 3a_{n-1} + 1 & \text{khi } a_{n-1} = 2k+1 (n \geq 2) \end{cases}$$

– Dữ liệu test

```
00847.  Dữ liệu vào:   n=3
00848.  Dữ liệu ra:    3 10 5 16 8 4 2 1
00849.
00850.  Dữ liệu vào:   n=11
00851.  Dữ liệu ra:    11 34 17 52 26 13 40 20 10 5 16
00852.                 8 4 2 1
00853.
00854.  Dữ liệu vào:   n = 10000
00855.  Dữ liệu ra:    10000 5000 2500 1250 625 1876
00856.                 938 469 1408 704 352 176 88 44
00857.                 22 11 34 17 52 26 13 40 20 10 5
00858.                 16 8 4 2 1
```

– Khai báo hàm.

```
00859.  void LietKe(int);
```

– Định nghĩa hàm đệ quy.

```
00860.  void LietKe(int n)
00861.  {
00862.     if(n==1)
00863.     {
00864.        cout << setw(6) << n;
00865.        return;
00866.     }
00867.     cout << setw(6) << n;
00868.     if(n%2==0)
00869.        LietKe(n/2);
00870.     else
00871.        LietKe(3*n + 1);
00872.  }
```

01.05 KỸ THUẬT ĐẾM ĐỆ QUY

01.05.01 Bài cơ sở Kỹ thuật Đếm Đệ quy

Bài cơ sở 008. Hãy đếm số lượng chữ số của số nguyên n.

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $dem(n)$ là số lượng chữ số của số nguyên n.
- Ta có:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
- Suy ra: $dem(n) = dem(n/10) + 1$.
- Điều kiện dừng: $dem(n) = 1$ khi $-9 \le n \le 9$.
- Khai báo hàm.

```
00873.  int DemChuSo(int n)
```

- Định nghĩa hàm đệ quy.

```
00874.  int DemChuSo(int n)
00875.  {
00876.     n = abs(n);
00877.     if(n<=9)
00878.        return 1;
00879.     int dem = DemChuSo(n/10);
00880.     return dem + 1;
00881.  }
```

Chương trình 008. Viết chương trình bằng kỹ thuật đệ quy đếm số lượng chữ số của số nguyên n.

- Chương trình

```
00882.  #include <iostream>
00883.  using namespace std;
00884.
00885.  int DemChuSo(int);
00886.
00887.  int main()
00888.  {
00889.     int n;
00890.     cout << "Nhap n: ";
00891.     cin >> n;
00892.
```

```
00893.     int kq = DemChuSo(n);
00894.     cout << "Ket qua: " << kq;
00895.     return 1;
00896.  }
00897.
00898.  int DemChuSo(int n)
00899.  {
00900.     n = abs(n);
00901.     if (n <= 9)
00902.        return 1;
00903.     int dem = DemChuSo(n / 10);
00904.     return  dem + 1;
00905.  }
```

01.05.01 Bài tập Kỹ thuật Đếm Đệ quy

Bài 036.Viết hàm đệ quy đếm số lượng chữ số lẻ của số nguyên n.

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $dem(n)$ là số lượng chữ số lẻ của số nguyên n.
- Ta có:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
- Suy ra:
  - $dem(n) = dem(n/10)$ khi a_1 là số chẵn.
  - $dem(n) = dem(n/10) + 1$ khi a_1 là số lẻ.
- Điều kiện dừng:
  - $dem(n) = 1$ khi $-9 \le n \le 9$ và n lẻ.
  - $dem(n) = 0$ khi $-9 \le n \le 9$ và n chẵn.
- Dữ liệu test

```
00906.  Dữ liệu vào: n=0
00907.  Dữ liệu ra: 0
00908.
00909.  Dữ liệu vào: n=143
00910.  Dữ liệu ra: 2
00911.
00912.  Dữ liệu vào: n = -987
00913.  Dữ liệu ra: 2
```

- Khai báo hàm.

```
00914.  int DemSoLe(int);
```

– Định nghĩa hàm đệ quy.

```
00915. int DemSoLe(int n)
00916. {
00917.    n = abs(n);
00918.    if(n<=9)
00919.    {
00920.       if(n%2!=0)
00921.          return 1;
00922.       return 0;
00923.    }
00924.    int dem = DemChuSo(n/10);
00925.    int dv = n%10;
00926.    if(dv%2!=0)
00927.       return dem + 1;
00928.    return dem;
00929. }
```

Bài 037. Viết hàm đệ quy đếm số lượng chữ số chẵn của số nguyên n.

– Gọi:
  + $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  + $dem(n)$ là số lượng chữ số chẵn của số nguyên n.

– Ta có:
  + $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  + $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$

– Suy ra:
  + $dem(n) = dem(n/10)$ khi a_1 là số lẻ.
  + $dem(n) = dem(n/10) + 1$ khi a_1 là số chẵn.

– Điều kiện dừng:
  + $dem(n) = 1$ khi $-9 \le n \le 9$ và n chẵn.
  + $dem(n) = 0$ khi $-9 \le n \le 9$ và n lẻ.

– Khai báo hàm.

```
00930. int DemSoChan(int);
```

– Định nghĩa hàm đệ quy.

```
00931. int DemSoChan(int n)
00932. {
00933.    n = abs(n);
00934.    if(n<=9)
00935.    {
00936.       if(n%2==0)
00937.          return 1;
00938.       return 0;
```

```
00939.     }
00940.     int dem = DemSoChan(n/10);
00941.     int dv = n%10;
00942.     if(dv%2==0)
00943.        return dem + 1;
00944.     return dem;
00945.  }
```

01.06 KỸ THUẬT TÌM KIẾM ĐỆ QUY

01.06.01 Bài cơ sở Kỹ thuật Tìm kiếm Đệ quy

Bài cơ sở 009. Tìm chữ số lớn nhất của số nguyên n.

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $max(n)$ là chữ số lớn nhất của số nguyên n.
- Ta có:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
- Suy ra:
  - $max(n) = max(n/10)$ nếu $max(n/10) \geq a_1$.
  - $max(n) = a_1$ nếu $max(n/10) < a_1$.
- Điều kiện dừng: $max(n) = |n|$ khi $-9 \leq n \leq 9$.
- Khai báo hàm.

```
00946.  int ChuSoLonNhat(int);
```

- Định nghĩa hàm đệ quy.

```
00947.  int ChuSoLonNhat(int n)
00948.  {
00949.     n = abs(n);
00950.     if(n<=9)
00951.        return n;
00952.     int lc = ChuSoLonNhat(n/10);
00953.     int dv = n%10;
00954.     if(dv>lc)
00955.        lc = dv;
00956.     return lc;
00957.  }
```

Chương trình 009. Viết chương trình bằng kỹ thuật đệ quy tìm chữ số lớn nhất của số nguyên n.

– Chương trình

```
00958. #include <iostream>
00959. using namespace std;
00960.
00961. int ChuSoLonNhat(int);
00962.
00963. int main()
00964. {
00965.    int n;
00966.    cout << "Nhap n: ";
00967.    cin >> n;
00968.
00969.    int kq = ChuSoLonNhat(n);
00970.    cout << "Ket qua: " << kq;
00971.    return 1;
00972. }
00973.
00974. int ChuSoLonNhat(int n)
00975. {
00976.    n = abs(n);
00977.    if (n <= 9)
00978.       return n;
00979.    int lc = ChuSoLonNhat(n / 10);
00980.    int dv = n % 10;
00981.    if (dv > lc)
00982.       lc = dv;
00983.    return lc;
00984. }
```

01.06.02 Bài tập Kỹ thuật Tìm kiếm Đệ quy

Bài 038.Viết hàm đệ quy tìm chữ số nhỏ nhất của số nguyên dương n.

– Gọi:
  + $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  + $min(n)$ là chữ số nhỏ nhất của số nguyên n.
– Ta có:
  + $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  + $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
– Suy ra:
  + $min(n) = min(n/10)$ nếu $min(n/10) \leq a_1$.
  + $min(n) = a_1$ nếu $min(n/10) > a_1$.

- Điều kiện dừng: $min(n) = |n|$ khi $-9 \le n \le 9$.
- Dữ liệu test

```
00985. Dữ liệu vào: n=0
00986. Dữ liệu ra: 0
00987.
00988. Dữ liệu vào: n=143
00989. Dữ liệu ra: 1
00990.
00991. Dữ liệu vào: n = 1000000000
00992. Dữ liệu ra: Tràn số
```

- Khai báo hàm.

```
00993. int ChuSoNhoNhat(int);
```

- Định nghĩa hàm đệ quy.

```
00994. int ChuSoNhoNhat(int n)
00995. {
00996.    n = abs(n);
00997.    if(n<=9)
00998.       return n;
00999.    int lc = ChuSoNhoNhat(n/10);
01000.    int dv = n%10;
01001.    if(dv<lc)
01002.       lc = dv;
01003.    return lc;
01004. }
```

Bài 039. Viết hàm đệ quy tìm chữ số đầu tiên của số nguyên dương n.

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
- Ta có:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
- Suy ra:
  - $ChuSoDau(n) = ChuSoDau(n/10)$.
- Điều kiện dừng:
  - $ChuSoDau(n) = \text{abs}(n)$ khi $-9 \le n \le 9$.
- Dữ liệu test

```
01005. Dữ liệu vào: n=0
01006. Dữ liệu ra: 0
01007.
01008. Dữ liệu vào: n=143
01009. Dữ liệu ra: 1
```

```
01010.
01011. Dữ liệu vào: n = 10000
01012. Dữ liệu ra: 1
```

– Khai báo hàm.

```
01013. int ChuSoDau(int);
```

– Định nghĩa hàm đệ quy.

```
01014. int ChuSoDau (int n)
01015. {
01016.     n = abs(n);
01017.     if(n<=9)
01018.        return n;
01019.     return ChuSoDau(n/10);
01020. }
```

Bài 040.Viết hàm đệ quy tìm ước số lẻ lớn nhất của số nguyên dương n.

– Gọi:
  + $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$.
– Suy ra:
  + $UocLeLonNhat(n) = abs(n)$ khi n lẻ.
  + $UocLeLonNhat(n) = UocLeLonNhat(n/2)$ khi n chẵn.
– Điều kiện dừng:
  + $UocLeLonNhat(n) = n$ khi n lẻ.
– Dữ liệu test

```
01021. Dữ liệu vào: n=0
01022. Dữ liệu ra: Không xác định
01023.
01024. Dữ liệu vào: n=143
01025. Dữ liệu ra: 143
01026.
01027. Dữ liệu vào: n = 10000
01028. Dữ liệu ra: 625
```

– Khai báo hàm.

```
01029. int UocLeLonNhat(int);
```

– Định nghĩa hàm đệ quy.

```
01030. int UocLeLonNhat(int n)
01031. {
01032.     n = abs(n);
01033.     if(n%2!=0)
01034.        return n;
01035.     return UocLeLonNhat(n/2);
```

```
01036. }
```

01.07 KỸ THUẬT ĐẶT CỜ HIỆU ĐỆ QUY

01.07.01 Bài cơ sở Kỹ thuật Đặt cờ hiệu Đệ quy

Bài cơ sở 010. Kiểm tra số nguyên n có tồn tại chữ số lẻ không?

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$.
  - $TonTaiLe(n) = 1$ khi n có tồn tại chữ số lẻ.
  - $TonTaiLe(n) = 0$ khi n có ko tồn tại chữ số lẻ.
- Ta có:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
- Suy ra:
  - $TonTaiLe(n) = TonTaiLe(n/10)$ nếu a_1 chẵn.
  - $TonTaiLe(n) = 1$ nếu a_1 lẻ.
- Điều kiện dừng:
  - $TonTaiLe(n) = 1$ khi $-9 \le n \le 9$ và n lẻ.
  - $TonTaiLe(n) = 0$ khi $-9 \le n \le 9$ và n chẵn.
- Khai báo hàm.

```
01037. int TonTaiLe(int);
```

- Định nghĩa hàm đệ quy.

```
01038. int TonTaiLe(int n)
01039. {
01040.    n = abs(n);
01041.    if(n<=9)
01042.    {
01043.       if(n%2!=0)
01044.          return 1;
01045.       return 0;
01046.    }
01047.    int dv = n%10;
01048.    if(dv%2!=0)
01049.       return 1;
01050.    return TonTaiLe(n/10);
01051. }
```

Chương trình 010. Viết chương trình bằng kỹ thuật đệ quy kiểm tra số nguyên n có tồn tại chữ số lẻ không?

– Chương trình

```
01052. #include <iostream>
01053. using namespace std;
01054.
01055. int TonTaiLe(int);
01056.
01057. int main()
01058. {
01059.    int n;
01060.    cout << "Nhap n: ";
01061.    cin >> n;
01062.
01063.    int kq = TonTaiLe(n);
01064.    if (kq == 1)
01065.       cout << "Ton tai chu so le";
01066.    else
01067.       cout << "Khong ton tai";
01068.    return 1;
01069. }
01070.
01071. int TonTaiLe(int n)
01072. {
01073.    n = abs(n);
01074.    if (n <= 9)
01075.    {
01076.       if (n % 2 != 0)
01077.          return 1;
01078.       return 0;
01079.    }
01080.    int dv = n % 10;
01081.    if (dv % 2 != 0)
01082.       return 1;
01083.    return TonTaiLe(n / 10);
01084. }
```

Bài cơ sở 011. Kiểm tra số nguyên n có toàn chữ số lẻ hay không?

– Gọi:

+ $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$

+ $ToanLe(n) = 1$ khi n toàn chữ số lẻ.
+ $ToanLe(n) = 0$ khi n ko toàn chữ số lẻ.

– Ta có:
+ $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
+ $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$

– Suy ra:
+ $ToanLe(n) = ToanLe(n/10)$ toàn lẻ và a_1 lẻ.

– Điều kiện dừng:
+ $ToanLe(n) = 1$ khi $-9 \le n \le 9$ và n lẻ.
+ $ToanLe(n) = 0$ khi $-9 \le n \le 9$ và n chẵn.

– Khai báo hàm.

```
01085.  int ktToanLe(int);
```

– Định nghĩa hàm đệ quy.

```
01086.  int ktToanLe(int n)
01087.  {
01088.     n = abs(n);
01089.     if(n<=9)
01090.     {
01091.        if(n%2!=0)
01092.           return 1;
01093.        return 0;
01094.     }
01095.     int dv = n%10;
01096.     if(ktToanLe(n/10)==1 && dv%2!=0)
01097.        return 1;
01098.     return 0;
01099.  }
```

– Định nghĩa hàm đệ quy (cách khác).

```
01100.  int ktToanLe(int n)
01101.  {
01102.     n = abs(n);
01103.     if(n<=9)
01104.     {
01105.        if(n%2!=0)
01106.           return 1;
01107.        return 0;
01108.     }
01109.     int dv = n%10;
01110.     if(dv%2==0)
01111.        return 0;
01112.     return ktToanLe(n/10);
```

```
01113. }
```

Chương trình 011. Viết chương trình bằng kỹ thuật đệ quy kiểm tra số nguyên n có toàn chữ số lẻ hay không?

– Chương trình

```
01114. #include <iostream>
01115. using namespace std;
01116.
01117. int ktToanLe(int);
01118.
01119. int main()
01120. {
01121.    int n;
01122.    cout << "Nhap n: ";
01123.    cin >> n;
01124.
01125.    int kq = ktToanLe(n);
01126.    if (kq == 1)
01127.       cout << "Chu so toan le";
01128.    else
01129.       cout << "Chu so khong toan le";
01130.    return 1;
01131. }
01132.
01133. int ktToanLe(int n)
01134. {
01135.    n = abs(n);
01136.    if (n <= 9)
01137.    {
01138.       if (n % 2 != 0)
01139.          return 1;
01140.       return 0;
01141.    }
01142.    int dv = n % 10;
01143.    if (ktToanLe(n / 10) == 1 && dv % 2 != 0)
01144.       return 1;
01145.    return 0;
01146. }
```

01.07.02 Bài tập Kỹ thuật Đặt cờ hiệu Đệ quy

Bài 041.Viết hàm đệ quy kiểm tra số nguyên n có tồn tại chữ số chẵn không?

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $TonTaiChan(n) = 1$ khi n có tồn tại chữ số chẵn.
  - $TonTaiChan(n) = 0$ khi n có ko tồn tại chữ số chẵn.
- Ta có:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
- Suy ra:
  - $TonTaiChan(n) = TonTaiChan(n/10)$ nếu a_1 lẻ.
  - $TonTaiChan(n) = 1$ nếu a_1 chẵn.
- Điều kiện dừng:
  - $TonTaiChan(n) = 1$ khi $-9 \le n \le 9$ và n chẵn.
  - $TonTaiChan(n) = 0$ khi $-9 \le n \le 9$ và n lẻ.
- Dữ liệu test

```
01147. Dữ liệu vào: n=0
01148. Dữ liệu ra: 1
01149.
01150. Dữ liệu vào: n=143
01151. Dữ liệu ra: 1
01152.
01153. Dữ liệu vào: n = 987
01154. Dữ liệu ra: 1
```

- Khai báo hàm.

```
01155. int TonTaiChan(int);
```

- Định nghĩa hàm đệ quy.

```
01156. int TonTaiChan(int n)
01157. {
01158.    n = abs(n);
01159.    if(n<=9)
01160.    {
01161.       if(n%2==0)
01162.          return 1;
01163.       return 0;
01164.    }
01165.    int dv = n%10;
01166.    if(dv%2==0)
01167.       return 1;
```

```
01168.     return TonTaiChan(n/10);
01169.  }
```

Bài 042.Viết hàm đệ quy kiểm tra số nguyên dương n có toàn chữ số chẵn hay không?

- Gọi:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $ToanChan(n) = 1$ khi n toàn chữ số chẵn.
  - $ToanChan(n) = 0$ khi n ko toàn chữ số chẵn.
- Ta có:
  - $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$
  - $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$
- Suy ra:
  - $ToanChan(n) = ToanChan(n/10)$ toàn chẵn và a_1 chẵn.
- Điều kiện dừng:
  - $ToanChan(n) = 1$ khi $-9 \le n \le 9$ và n chẵn.
  - $ToanChan(n) = 0$ khi $-9 \le n \le 9$ và n lẻ.
- Dữ liệu test

```
01170.  Dữ liệu vào: n=0
01171.  Dữ liệu ra: 1
01172.
01173.  Dữ liệu vào: n=143
01174.  Dữ liệu ra: 0
01175.
01176.  Dữ liệu vào: n = 10000
01177.  Dữ liệu ra: 0
```

- Khai báo hàm.

```
01178.  int ktToanChan(int);
```

- Định nghĩa hàm đệ quy.

```
01179.  int ktToanChan(int n)
01180.  {
01181.     n = abs(n);
01182.     if(n<=9)
01183.     {
01184.        if(n%2!=0)
01185.           return 1;
01186.        return 0;
01187.     }
01188.     int dv = n%10;
01189.     if(ktToanChan(n/10)==1 && dv%2!=0)
```

```
01190.        return 1;
01191.     return 0;
01192.  }
```

– Định nghĩa hàm đệ quy.

```
01193.  int ktToanChan(int n)
01194.  {
01195.     n = abs(n);
01196.     if(n<=9)
01197.     {
01198.        if(n%2!=0)
01199.           return 1;
01200.        return 0;
01201.     }
01202.     int dv = n%10;
01203.     if(dv%2==0)
01204.        return 0;
01205.     return ktToanChan(n/10);
01206.  }
```

Bài 043.Viết hàm đệ quy kiểm tra các chữ số của số nguyên dương n có tăng dần từ trái sang phải hay không?

– Gọi:
  + $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$.
  + $ktTang(n) = 1$ khi các chữ số của n tăng dần từ trái sang phải.
  + $ktTang(n) = 0$ khi các chữ số của n ko tăng dần từ trái sang phải.
– Ta có:
  + $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$.
  + $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$.
– Suy ra:
  + $ktTang(n)\ khi\ ktTang(n/10)$ và $a_2 \le a_1$.
– Điều kiện dừng:
  + $ktTang(n) = 1$ khi $-9 \le n \le 9$.
– Dữ liệu test

```
01207.  Dữ liệu vào: n=0
01208.  Dữ liệu ra: 1
01209.
01210.  Dữ liệu vào: n=143
01211.  Dữ liệu ra: 0
01212.
```

```
01213.  Dữ liệu vào: n = 10000
01214.  Dữ liệu ra: 0
```

– Khai báo hàm.

```
01215.  int ktTang(int);
```

– Định nghĩa hàm đệ quy.

```
01216.  int ktTang(int n)
01217.  {
01218.      n = abs(n);
01219.      if(n<=9)
01220.         return 1;
01221.      int dv = n%10;
01222.      int hc = (n/10)%10;
01223.      if(ktTang(n/10) && hc <= dv)
01224.         return 1;
01225.      return 0;
01226.  }
```

Bài 044. Viết hàm đệ quy kiểm tra các chữ số của số nguyên dương n có giảm dần từ trái sang phải hay không?

– Gọi:
  + $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$.
  + $ktGiam(n) = 1$ khi các chữ số của n giảm dần từ trái sang phải.
  + $ktGiam(n) = 0$ khi các chữ số của n ko giảm dần từ trái sang phải.

– Ta có:
  + $n = \overline{a_k a_{k-1} a_{k-2} \cdots a_2 a_1}$.
  + $n/10 = \overline{a_k a_{k-1} a_{k-2} \cdots a_2}$.

– Suy ra:
  + $ktGiam(n)\ khi\ ktGiam(n/10)$ và $a_2 >= a_1$.

– Điều kiện dừng:
  + $ktGiam(n) = 1$ khi $-9 \le n \le 9$.

– Dữ liệu test

```
01227.  Dữ liệu vào: n=0
01228.  Dữ liệu ra: 1
01229.
01230.  Dữ liệu vào: n=143
01231.  Dữ liệu ra: 0
01232.
01233.  Dữ liệu vào: n = 10000
01234.  Dữ liệu ra: 1
```

– Khai báo hàm.

```
01235.  int ktGiam(int);
```

– Định nghĩa hàm đệ quy.

```
01236.  int ktGiam(int n)
01237.  {
01238.      n = abs(n);
01239.      if(n<=9)
01240.          return 1;
01241.      int dv = n%10;
01242.      int hc = (n/10)%10;
01243.      if(ktGiam(n/10) && hc >= dv)
01244.          return 1;
01245.      return 0;
01246.  }
```

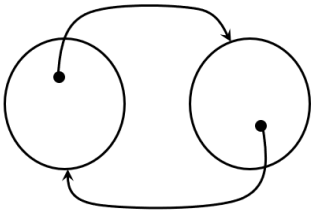
CHƯƠNG 02. ĐỆ QUY HỖ TƯƠNG

02.01 KHÁI NIỆM ĐỆ QUY HỖ TƯƠNG - MUTUAL RECURSION

- Khái niệm: Hai hàm được gọi là đệ quy hỗ tương (mutual recursion) nếu trong thân của hàm này có lời gọi hàm tới hàm kia và bên trong thân hàm kia có lời gọi hàm tới hàm này.

02.02 HÌNH ẢNH ĐỆ QUY HỖ TƯƠNG

- Hình vẽ minh họa



- Trong hình vẽ minh họa trên ta có thể hiểu như sau:
  + Hàm là vòng tròn.
  + Hai vòng tròn minh họa cho hai hàm.
  + Lời gọi hàm được minh họa bởi vòng cung có mũi tên.
  + Lời gọi hàm thứ nhất bắt đầu tại một điểm trong vòng tròn thứ nhất và kết thúc với mũi tên tại biên vòng tròn thứ hai.
  + Lời gọi hàm thứ hai bắt đầu tại một điểm trong vòng tròn thứ hai và kết thúc với mũi tên tại biên vòng tròn thứ nhất.

02.03 CẤU TRÚC HÀM ĐỆ QUY HỖ TƯƠNG

```
01247.  KDL TenHam1(<ThamSo>)
01248.  {
01249.      if <điều kiện dừng>
01250.      {
01251.          ...
01252.          return <Giá Trị Trả Về>;
01253.      }
01254.      ...TenHam2(<ThamSo>);
```

```
01255.     ...
01256.  }
01257.
01258.  KDL TenHam2(<ThamSo>)
01259.  {
01260.     if <điều kiện dừng>
01261.     {
01262.        ...
01263.        return <Giá Trị Trả Về>;
01264.     }
01265.     ...TenHam1(<ThamSo>);
01266.     ...
01267.  }
```

02.04 DÃY HOFSTADTER

Bài cơ sở 012. Định nghĩa hai hàm đệ quy hỗ tương tính số hạng thứ n của hai dãy Hofstadter Female (F) and Male (M) được định nghĩa như sau:

$$\begin{cases} F(0) = 1 \\ M(0) = 0 \\ F(n) = n - M\big(F(n-1)\big) \; với \; n > 0 \\ M(n) = n - F\big(M(n-1)\big) \; với \; n > 0 \end{cases}$$

- Dãy F: 1, 1, 2, 2, 3, 3, 4, 5, 5, 6, 6, 7, 8, 8, 9, 9, 10, 11, 11, 12, 13, 13, 14, 14, 15, 16, 16, 17, 17, 18, 19, 19, 20, 21, 21, 22, 22, 23, 24, 24, 25, 25, 26, 27, 27, 28, 29, 29, 30, 30, 31, 32, 32, 33, 34, 34, 35, 35, 36, 37, 37, 38, 38, 39, 40, 40, 41, 42, 42, 43, 43, 44, 45, 45,…
- Dãy M: 0, 0, 1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 7, 8, 9, 9, 10, 11, 11, 12, 12, 13, 14, 14, 15, 16, 16, 17, 17, 18, 19, 19, 20, 20, 21, 22, 22, 23, 24, 24, 25, 25, 26, 27, 27, 28, 29, 29, 30, 30, 31, 32, 32, 33, 33, 34, 35, 35, 36, 37, 37, 38, 38, 39, 40, 40, 41, 42, 42, 43, 43, 44, 45, 45,…
- Định nghĩa hàm đệ quy tính số hàng thứ n của dãyHofstadter Female (F) and Male (M).

```
01268.  int TinhF(int n)
01269.  {
01270.     if(n==0)
01271.        return 1;
01272.     return (n – TinhM(TinhF(n-1));
01273.  }
```

– Định nghĩa hàm đệ quy tính số hàng thứ n của dãyHofstadter Female (F) and Male (M).

```
01274. int TinhM(int n)
01275. {
01276.     if(n==0)
01277.        return 0;
01278.     return (n – TinhF(TinhM(n-1));
01279. }
```

Chương trình 012. Viết chương trình bằng kỹ thuật đệ quy hỗ tương tính số hạng thứ n của hai dãy Hofstadter Female (F) and Male (M) được định nghĩa như sau:

$$\begin{cases} F(0) = 1 \\ M(0) = 0 \\ F(n) = n - M\big(F(n-1)\big) \; với \; n > 0 \\ M(n) = n - F\big(M(n-1)\big) \; với \; n > 0 \end{cases}$$

– Chương trình

```
01280. #include <iostream>
01281. #include <iomanip>
01282. using namespace std;
01283.
01284. int TinhF(int);
01285. int TinhM(int);
01286.
01287. int main()
01288. {
01289.    int n;
01290.    cout << "Nhap n: ";
01291.    cin >> n;
01292.
01293.    int kq;
01294.    cout << "\nDay ket qua F: ";
01295.    for (int i = 0; i <= n; i++)
01296.    {
01297.       kq = TinhF(i);
01298.       cout << setw(8) << kq;
01299.    }
01300.
01301.    cout << "\nDay ket qua M: ";
01302.    for (int i = 0; i <= n; i++)
01303.    {
01304.       kq = TinhM(i);
```

```
01305.        cout << setw(8) << kq;
01306.     }
01307.     return 1;
01308.  }
01309.
01310.  int TinhF(int n)
01311.  {
01312.     if (n == 0)
01313.        return 1;
01314.     return (n - TinhM(TinhF(n - 1)));
01315.  }
01316.
01317.  int TinhM(int n)
01318.  {
01319.     if (n == 0)
01320.        return 0;
01321.     return (n - TinhF(TinhM(n - 1)));
01322.  }
```

02.05 KIỂM TRA SỐ NGUYÊN CHẴN LẺ

Bài cơ sở 013. Định nghĩa hàm kiểm tra một số nguyên dương n là một số chẵn hay số lẻ với các ràng buộc như sau:
+ Số 0 là giá trị chẵn.
+ Số 1 là giá trị lẻ.
+ Chỉ sử dụng duy nhất phép toán trừ ($operator\ -$).

– Định nghĩa hàm đệ quy kiểm tra một số nguyên n có là số chẵn không?

```
01323.  int ktChan(int n)
01324.  {
01325.     if(n==0)
01326.        return 1;
01327.     if(n==1)
01328.        return 0;
01329.     return ktLe(n-1);
01330.  }
```

– Định nghĩa hàm đệ quy kiểm tra một số nguyên n có là số lẻ không?

```
01331.  int ktLe(int n)
01332.  {
```

```
01333.     if(n==0)
01334.        return 0;
01335.     if(n==1)
01336.        return 1;
01337.     return ktChan(n-1);
01338.  }
```

Chương trình 013. Viết chương trình bằng kỹ thuật đệ quy hỗ tương kiểm tra một số nguyên dương n là một số chẵn hay số lẻ với các ràng buộc như sau:
+ Số 0 là giá trị chẵn.
+ Số 1 là giá trị lẻ.
+ Chỉ sử dụng duy nhất phép toán trừ ($operator\ -$).

– Chương trình

```
01339.  #include <iostream>
01340.  #include <iomanip>
01341.  using namespace std;
01342.
01343.  int ktLe(int);
01344.  int ktChan(int);
01345.
01346.  int main()
01347.  {
01348.     int n;
01349.     cout << "Nhap n: ";
01350.     cin >> n;
01351.
01352.     int kq = ktChan(n);
01353.     if (kq == 1)
01354.        cout << "So chan";
01355.     else
01356.        cout << "So le";
01357.     return 1;
01358.  }
01359.
01360.  int ktLe(int n)
01361.  {
01362.     if (n == 0)
01363.        return 0;
01364.     if (n == 1)
01365.        return 1;
01366.     return ktChan(n - 1);
```

```
01367.  }
01368.
01369.  int ktChan(int n)
01370.  {
01371.      if (n == 0)
01372.          return 1;
01373.      if (n == 1)
01374.          return 0;
01375.      return ktLe(n - 1);
01376.  }
```

02.06 DÃY SỐ - MINH HỌA ĐỆ QUY HỖ TƯƠNG

Bài cơ sở 014. Tính số hạng thứ n của dãy của hai dãy sau:

$$\begin{cases} x_0 = 1 \\ y_0 = 0 \\ x_n = x_{n-1} + y_{n-1} & (n \geq 2) \\ y_n = 3x_{n-1} + 2y_{n-1} & (n \geq 2) \end{cases}$$

– Tính x_5, y_5.

n	0	1	2	3	4	5	6	7	8	9
x_n	1	1	4	13	43	142	469	1.549	5.116	16.897
y_n	0	3	9	30	99	327	1.080	3.567	11.781	38.910

– Định nghĩa hàm đệ quy tính số hàng thứ n của dãy x_n.

```
01377.  int TinhXn(int n)
01378.  {
01379.      if(n==0)
01380.          return 1;
01381.      int a = TinhXn(n-1);
01382.      int b = TinhYn(n-1);
01383.      return (a + b);
01384.  }
```

– Định nghĩa hàm đệ quy tính số hàng thứ n của dãy y_n.

```
01385.  int TinhYn(int n)
01386.  {
01387.      if(n==0)
01388.          return 0;
01389.      int a = TinhXn(n-1);
01390.      int b = TinhYn(n-1);
01391.      return (3*a + 2*b);
01392.  }
```

– Một cách viết gọn hơn.

```
01393.  int TinhXn(int n)
01394.  {
01395.     if(n==0)
01396.        return 1;
01397.     return TinhXn(n-1) + TinhYn(n-1);
01398.  }
01399.
01400.  int TinhYn(int n)
01401.  {
01402.     if(n==0)
01403.        return 0;
01404.     return 3*TinhXn(n-1) + 2*TinhYn(n-1);
01405.  }
```

Chương trình 014. Viết chương trình bằng kỹ thuật đệ quy hỗ tương tính số hạng thứ n của dãy của hai dãy sau: $\begin{cases} x_0 = 1 \\ y_0 = 0 \\ x_n = x_{n-1} + y_{n-1} & (n \geq 2) \\ y_n = 3x_{n-1} + 2y_{n-1} & (n \geq 2) \end{cases}$

– Chương trình

```
01406.  #include <iostream>
01407.  #include <iomanip>
01408.  using namespace std;
01409.
01410.  int TinhXn(int);
01411.  int TinhYn(int);
01412.
01413.  int main()
01414.  {
01415.     int n;
01416.     cout << "Nhap n: ";
01417.     cin >> n;
01418.
01419.     int kq;
01420.     cout << "\nDay ket qua X: ";
01421.     for (int i = 0; i <= n; i++)
01422.     {
01423.        kq = TinhXn(i);
01424.        cout << setw(8) << kq;
01425.     }
01426.
```

```
01427.     cout << "\nDay ket qua Y: ";
01428.     for (int i = 0; i <= n; i++)
01429.     {
01430.        kq = TinhYn(i);
01431.        cout << setw(8) << kq;
01432.     }
01433.     return 1;
01434.  }
01435.
01436.  int TinhXn(int n)
01437.  {
01438.     if (n == 0)
01439.        return 1;
01440.     return TinhXn(n - 1) + TinhYn(n - 1);
01441.  }
01442.
01443.  int TinhYn(int n)
01444.  {
01445.     if (n == 0)
01446.        return 0;
01447.     return 3 * TinhXn(n - 1) + 2 * TinhYn(n - 1);
01448.  }
```

02.07 BÀI TẬP ĐỆ QUY HỖ TƯƠNG

Bài 045.Tính số hạng thứ n của hai dãy $\begin{cases} a_1 = 1 \\ b_1 = 1 \\ a_k = 3b_{k-1} + 2a_{k-1} (k \geq 2) \\ b_k = a_{k-1} + 3b_{k-1} \quad (k \geq 2) \end{cases}$

– Dữ liệu test

```
01449.  Dữ liệu vào:   n = 3
01450.  Dữ liệu ra:    22 17
01451.
01452.  Dữ liệu vào:   n = 11
01453.  Dữ liệu ra:    2,595,757 1,992,482
01454.
01455.  Dữ liệu vào:   n = 10000
01456.  Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy tính số hàng thứ n của dãy a_n.

```
01457.  int TinhAn(int n)
01458.  {
01459.     if(n==1)
```

```
01460.        return 1;
01461.     int x = TinhAn(n-1);
01462.     int y = TinhBn(n-1);
01463.     return (3*y + 2*x);
01464.  }
```

– Định nghĩa hàm đệ quy tính số hàng thứ n của dãy b_n.

```
01465.  int TinhBn(int n)
01466.  {
01467.     if(n==1)
01468.        return 1;
01469.     int x = TinhAn(n-1);
01470.     int y = TinhBn(n-1);
01471.     return (x + 3*y);
01472.  }
```

Bài 046.Tính số hạng thứ n của hai dãy $\begin{cases} a_1 = 2 \\ b_1 = 1 \\ a_k = 3b_{k-1} + 2a_{k-1} (k \geq 2) \\ b_k = a_{k-1} + 3b_{k-1} \quad (k \geq 2) \end{cases}$

– Dữ liệu test

```
01473.  Dữ liệu vào:   n = 3
01474.  Dữ liệu ra:    29 22
01475.
01476.  Dữ liệu vào:   n = 11
01477.  Dữ liệu ra:    3,381,689 2,595,757
01478.
01479.  Dữ liệu vào:   n = 10000
01480.  Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy tính số hàng thứ n của dãy a_n.

```
01481.  int TinhAn(int n)
01482.  {
01483.     if(n==1)
01484.        return 2;
01485.     int x = TinhAn(n-1);
01486.     int y = TinhBn(n-1);
01487.     return (3*y + 2*x);
01488.  }
```

– Định nghĩa hàm đệ quy tính số hàng thứ n của dãy b_n.

```
01489.  int TinhBn(int n)
01490.  {
01491.     if(n==1)
```

```
01492.        return 1;
01493.     int x = TinhAn(n-1);
01494.     int y = TinhBn(n-1);
01495.     return (x + 3*y);
01496.  }
```

Bài 047.Tính số hạng thứ n của hai dãy $\begin{cases} a_1 = 2 \\ b_1 = 1 \\ a_n = a_{n-1}^2 + 2b_{n-1}^2 & (n \geq 2) \\ b_n = 2a_{n-1}b_{n-1} & (n \geq 2) \end{cases}$

– Dữ liệu test

```
01497.  Dữ liệu vào:   n = 3
01498.  Dữ liệu ra:    68 48
01499.
01500.  Dữ liệu vào:   n = 11
01501.  Dữ liệu ra:    Tràn số
01502.
01503.  Dữ liệu vào:   n = 10000
01504.  Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy tính số hàng thứ n của dãy a_n.

```
01505.  int TinhAn(int n)
01506.  {
01507.     if(n==1)
01508.        return 2;
01509.     int x = TinhAn(n-1);
01510.     int y = TinhBn(n-1);
01511.     return (x*x + 2*y*y);
01512.  }
```

– Định nghĩa hàm đệ quy tính số hàng thứ n của dãy b_n.

```
01513.  int TinhBn(int n)
01514.  {
01515.     if(n==1)
01516.        return 1;
01517.     int x = TinhAn(n-1);
01518.     int y = TinhBn(n-1);
01519.     return 2*x*y;
01520.  }
```

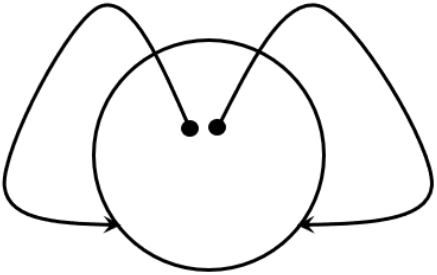
CHƯƠNG 03. ĐỆ QUY NHỊ PHÂN

03.01 KHÁI NIỆM ĐỆ QUY NHỊ PHÂN

- Khái niệm: Một hàm được gọi là đệ quy nhị phân (binary recursive, binary recursion) khi bên trong thân hàm của nó có đúng hai lời gọi hàm lại chính nó.

03.02 HÌNH ẢNH ĐỆ QUY NHỊ PHÂN

- Hình vẽ minh họa



- Trong hình vẽ minh họa trên ta có thể hiểu như sau:
  + Hàm là vòng tròn.
  + Lời gọi hàm được minh họa bởi vòng cung có mũi tên.
  + Lời gọi hàm thứ nhất bắt đầu tại một điểm trong vòng tròn và kết thúc với mũi tên tại biên vòng tròn.
  + Lời gọi hàm thứ hai bắt đầu tại một điểm khác trong vòng tròn và kết thúc với mũi tên tại biên vòng tròn.

03.03 CẤU TRÚC HÀM ĐỆ QUY NHỊ PHÂN

```
01521.  KDL TenHam(<ThamSo>)
01522.  {
01523.      if <điều kiện dừng>
01524.      {
01525.          ...
01526.          return <Giá Trị Trả Về>;
01527.      }
01528.      ...
```

```
01529.     ...TenHam(<ThamSo>);
01530.     ...
01531.     ...TenHam(<ThamSo>);
01532.     ...
01533. }
```

03.04 CẤU TRÚC DỮ LIỆU CÂY NHỊ PHÂN

Bài cơ sở 015. Hãy khai báo cấu trúc dữ liệu của cây nhị phân các số nguyên.

– Khai báo cấu trúc dữ liệu

```
01534. struct node
01535. {
01536.     int info;
01537.     struct node *pLeft;
01538.     struct node *pRight;
01539. };
01540. typedef struct node NODE;
01541. typedef NODE* TREE;
```

– Trong khai báo trên ta có khai báo kiểu cấu trúc `struct node`. Bên trong thành phần của kiểu cấu trúc này có hai thành phần `pLeft` và `pRight` là hai con trỏ có kiểu cấu trúc `struct node`.

– Cách thức khai báo kiểu dữ liệu như trên được gọi là khai báo kiểu dữ liệu đệ quy.

03.05 DÃY FIBONACI – MINH HỌA ĐỆ QUY NHỊ PHÂN

Bài cơ sở 016. Tính số hạng thứ n của dãy của dãy

$$\text{fibonacci} \begin{cases} f_0 = 1 \\ f_1 = 1 \\ f_n = f_{n-1} + f_{n-2} \quad (n \geq 2) \end{cases}$$

– Ví dụ: Tính f_{10}.

n	0	1	2	3	4	5	6	7	8	9	10	11	12
f_n	1	1	2	3	5	8	13	21	34	55	89	134	223

– Định nghĩa hàm

```
01542. int Fibo(int n)
01543. {
01544.     if(n==0)
01545.         return 1;
01546.     if(n==1)
```

```
01547.        return 1;
01548.    int a = Fibo(n-1);
01549.    int b = Fibo(n-2);
01550.    return (a+b);
01551. }
```

– Một cách viết gọn hơn

```
01552. long Fibo(int n)
01553. {
01554.    if(n==0)
01555.       return 1;
01556.    if(n==1)
01557.       return 1;
01558.    return Fibo(n-1) + Fibo(n-2);
01559. }
```

03.06 TÍNH TOÁN – MINH HỌA ĐỆ QUY NHỊ PHÂN

Bài cơ sở 017. Viết hàm đệ quy tính tổng của biểu thức sau: $S(x,n) = x + x^2 + x^3 + \cdots + x^n$.

– Ta có:
$S(x,n) = x + x^2 + x^3 + \cdots + x^{n-2} + x^{n-1} + x^n.$

– Do đó:
+ $S(x,n) = x + x^2 + x^3 + \cdots + x^{n-2} + x^{n-1} + x^n.$
+ $S(x,n-1) = x + x^2 + x^3 + \cdots + x^{n-2} + x^{n-1}.$
+ $S(x,n-2) = x + x^2 + x^3 + \cdots + x^{n-2}.$

– Suy ra:
$S(x,n) = S(x,n-1) + x^n.$ (*)
$S(x,n-1) = S(x,n-2) + x^{n-1}.$ (**)

– Nhân hai vế của biểu thức (**) cho x ta được:
$xS(x,n-1) = x(S(x,n-2) + x^{n-1})$
$\Rightarrow xS(x,n-1) = xS(x,n-2) + x^n$
$\Rightarrow x^n = xS(x,n-1) - xS(x,n-2)$ (***)

– Thay (***) vào (*) ta được:
$S(x,n) = S(x,n-1) + x^n$
$S(x,n) = S(x,n-1) + xS(x,n-1) - xS(x,n-2)$
$S(x,n) = (1+x)S(x,n-1) - xS(x,n-2)$

– Công thức trên được gọi là công thức đệ quy nhị phân.
$S(x,n) = (1+x)S(x,n-1) - xS(x,n-2)$

– Điều kiện dừng:
+ $S(x,0) = 0.$
+ $S(x,1) = x.$

– Định nghĩa hàm

```
01560. float Tinh(float x,int n)
01561. {
01562.     if(n==0)
01563.        return 0;
01564.     if(n==1)
01565.        return x;
01566.     float a = Tinh(x,n-1);
01567.     float b = Tinh(x,n-2);
01568.     return ((1+x)*a -x*b);
01569. }
```

03.07 BÀI TẬP ĐỆ QUY NHỊ PHÂN

03.07.01 Tính số hạng thứ n của dãy số

Bài 048.Tính số hạng thứ n của dãy $\begin{cases} a_0 = -1 \\ a_1 = 3 \\ a_{n+1} = 5a_n + 6a_{n-1} (n \geq 1) \end{cases}$.

– Dữ liệu test

```
01570. Dữ liệu vào:   n = 3
01571. Dữ liệu ra:    63
01572.
01573. Dữ liệu vào:   n = 11
01574. Dữ liệu ra:    103,656,303
01575.
01576. Dữ liệu vào:   n = 100
01577. Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy.

```
01578. int Tinh(int n)
01579. {
01580.     if(n==0)
01581.        return -1;
01582.     if(n==1)
01583.        return 3;
01584.     int x = Tinh(n-1);
01585.     int y = Tinh(n-2);
01586.     return (5*x + 6*y);
01587. }
```

Bài 049.Tính số hạng thứ n của dãy $\begin{cases} a_0 = 1 \\ a_1 = 2 \\ a_{n+1} = 4a_n + a_{n-1} (n \geq 1) \end{cases}$.

– Dữ liệu test

```
01588. Dữ liệu vào:   n = 3
01589. Dữ liệu ra:    38
01590.
01591. Dữ liệu vào:   n = 11
01592. Dữ liệu ra:    3,940,598
01593.
01594. Dữ liệu vào:   n = 100
01595. Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy.

```
01596. int Tinh(int n)
01597. {
01598.    if(n==0)
01599.       return 1;
01600.    if(n==1)
01601.       return 2;
01602.    int x = Tinh(n-1);
01603.    int y = Tinh(n-2);
01604.    return (4*x + y);
01605. }
```

Bài 050.Tính số hạng thứ n của dãy $\begin{cases} a_0 = -1 \\ a_1 = 3 \\ a_n = 5a_{n-1} - a_{n-2} (n \geq 2) \end{cases}$

– Dữ liệu test

```
01606. Dữ liệu vào:   n = 3
01607. Dữ liệu ra:    77
01608.
01609. Dữ liệu vào:   n = 11
01610. Dữ liệu ra:    21,389,272
01611.
01612. Dữ liệu vào:   n = 100
01613. Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy.

```
01614. int Tinh(int n)
01615. {
01616.    if(n==0)
01617.       return -1;
```

```
01618.     if(n==1)
01619.        return 3;
01620.     int x = Tinh(n-1);
01621.     int y = Tinh(n-2);
01622.     return (5*x - y);
01623.  }
```

03.07.02 Tính toán

Bài 051. $S(n) = 1 + 1 \times 2 + \cdots + 1 \times 2 \times 3 \times \cdots \times n.$

- Ta có: $S(n) = 1! + 2! + \cdots + (n-1)! + n!$
- Do đó:
  - $S(n) = 1! + 2! + \cdots + (n-2)! + (n-1)! + n!$
  - $S(n-1) = 1! + 2! + \cdots + (n-2)! + (n-1)!$
  - $S(n-2) = 1! + 2! + \cdots + (n-2)!$
- Suy ra:
  - $S(n) = S(n-1) + n!$ (*)
  - $S(n-1) = S(n-2) + (n-1)!$ (**)
- Nhân hai vế của (**) cho n ta được:
  $nS(n-1) = n\,(S(n-2) + (n-1)!)$
  $\Rightarrow nS(n-1) = n\,S(n-2) + n!$
  $\Rightarrow nS(n-1) - n\,S(n-2) = n!$
  $\Rightarrow n! = nS(n-1) - n\,S(n-2)$ (***)
- Thế (***) vào (*) ta được
  $S(n) = S(n-1) + n!$
  $S(n) = S(n-1) + nS(n-1) - nS(n-2)$
  $S(n) = (1+n)S(n-1) - nS(n-2).$
- Công thức trên được gọi là công thức đệ quy nhị phân.
  $S(n) = (1+n)S(n-1) - n\,S(n-2).$
- Điều kiện dừng:
  - $S(0) = 0.$
  - $S(1) = 1.$
- Dữ liệu test

```
01624. Dữ liệu vào:   n = 3
01625. Dữ liệu ra:    9
01626.
01627. Dữ liệu vào:   n = 11
01628. Dữ liệu ra:    43,954,713
01629.
01630. Dữ liệu vào:   n = 100
01631. Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy.

```
01632. float Tinh(int n)
01633. {
01634.     if(n==0)
01635.        return 0;
01636.     if(n==1)
01637.        return 1;
01638.     float a = Tinh(n-1);
01639.     float b = Tinh(n-2);
01640.     return ((1+n)*a-n*b);
01641. }
```

Bài 052. $S(x,n) = x + x^2 + x^3 + \cdots + x^n$.

– Ta có:
$S(x,n) = x + x^2 + x^3 + \cdots + x^{n-2} + x^{n-1} + x^n.$

– Do đó:
+ $S(x,n) = x + x^2 + x^3 + \cdots + x^{n-2} + x^{n-1} + x^n.$
+ $S(x,n-1) = x + x^2 + x^3 + \cdots + x^{n-2} + x^{n-1}.$
+ $S(x,n-2) = x + x^2 + x^3 + \cdots + x^{n-2}.$

– Suy ra:
$S(x,n) = S(x,n-1) + x^n.$ (*)
$S(x,n-1) = S(x,n-2) + x^{n-1}.$ (**)

– Nhân hai vế của biểu thức (**) cho x ta được:
$xS(x,n-1) = x(S(x,n-2) + x^{n-1}).$
$\Rightarrow xS(x,n-1) = xS(x,n-2) + x^n.$
$\Rightarrow x^n = xS(x,n-1) - xS(x,n-2)$ (***)

– Thay (***) vào (*) ta được:
$S(x,n) = S(x,n-1) + x^n.$
$S(x,n) = S(x,n-1) + xS(x,n-1) - xS(x,n-2).$
$S(x,n) = (1+x)S(x,n-1) - xS(x,n-2).$

– Công thức trên được gọi là công thức đệ quy nhị phân.
$S(x,n) = (1+x)S(x,n-1) - xS(x,n-2).$

– Điều kiện dừng:
+ $S(x,0) = 0.$
+ $S(x,1) = x.$

– Dữ liệu test

```
Dữ liệu vào:   x = 3.1 n = 3
Dữ liệu ra:    42.50100

Dữ liệu vào:   x = 3.1 n = 11
Dữ liệu ra:    375,076.03990
```

```
01647.
01648. Dữ liệu vào:   x = 3.1 n = 100
01649. Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy.

```
01650. float Tinh(float x,int n)
01651. {
01652.    if(n==0)
01653.       return 0;
01654.    if(n==1)
01655.       return x;
01656.    float a = Tinh(x,n-1);
01657.    float b = Tinh(x,n-2);
01658.    return ((1+x)*a-x*b);
01659. }
```

Bài 053. $S(x,n) = 1 + x + x^2 + x^3 + \cdots + x^{n-1} + x^n$.

– Ta có:

$S(x,n) = 1 + x + x^2 + x^3 + \cdots + x^{n-2} + x^{n-1} + x^n.$

– Do đó:

+ $S(x,n) = 1 + x + x^2 + x^3 + \cdots + x^{n-2} + x^{n-1} + x^n.$
+ $S(x,n-1) = 1 + x + x^2 + x^3 + \cdots + x^{n-2} + x^{n-1}.$
+ $S(x,n-2) = 1 + x + x^2 + x^3 + \cdots + x^{n-2}.$

– Suy ra:

$S(x,n) = S(x,n-1) + x^n.$ (*)

$S(x,n-1) = S(x,n-2) + x^{n-1}.$ (**)

– Nhân hai vế của biểu thức (**) cho x ta được:

$xS(x,n-1) = x(S(x,n-2) + x^{n-1})$

$\Rightarrow xS(x,n-1) = xS(x,n-2) + x^n$

$\Rightarrow x^n = xS(x,n-1) - xS(x,n-2)$ (***)

– Thay (***) vào (*) ta được:

$S(x,n) = S(x,n-1) + x^n$

$S(x,n) = S(x,n-1) + xS(x,n-1) - xS(x,n-2)$

$S(x,n) = (1+x)S(x,n-1) - xS(x,n-2)$

– Công thức trên được gọi là công thức đệ quy nhị phân.

$S(x,n) = (1+x)S(x,n-1) - xS(x,n-2)$

– Điều kiện dừng:

+ $S(x,0) = 1.$
+ $S(x,1) = 1 + x.$

– Dữ liệu test

```
01660. Dữ liệu vào:   x = 3.1 n = 3
01661. Dữ liệu ra:    56.21100
```

```
01662.
01663. Dữ liệu vào:   x = 3.1 n = 11
01664. Dữ liệu ra:    496,068.31080
01665.
01666. Dữ liệu vào:   x = 3.1 n = 100
01667. Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy.

```
01668. float Tinh(float x,int n)
01669. {
01670.     if(n==0)
01671.        return 1;
01672.     if(n==1)
01673.        return (1+x);
01674.     float a = Tinh(x,n-1);
01675.     float b = Tinh(x,n-2);
01676.     return ((1+x)*a-x*b);
01677. }
```

Bài 054. $S(x,n) = x^2 + x^4 + \cdots + x^{2(n-1)} + x^{2n}$.

– Ta có:
$S(x,n) = x^2 + x^4 + \cdots + x^{2(n-2)} + x^{2(n-1)} + x^{2n}$.

– Do đó:
$S(x,n) = x^2 + x^4 + \cdots + x^{2(n-2)} + x^{2(n-1)} + x^{2n}$.
$S(x,n-1) = x^2 + x^4 + \cdots + x^{2(n-2)} + x^{2(n-1)}$.
$S(x,n-2) = x^2 + x^4 + \cdots + x^{2(n-2)}$.

– Suy ra:
$S(x,n) = S(x,n-1) + x^{2n}$. (*)
$S(x,n-1) = S(x,n-2) + x^{2(n-1)}$. (**)

– Nhân hai vế của biểu thức (**) cho x^2 ta được:
$x^2 S(x,n-1) = x^2\left(S(x,n-2) + x^{2(n-1)}\right)$
$\Rightarrow x^2 S(x,n-1) = x^2 S(x,n-2) + x^{2n}$
$\Rightarrow x^{2n} = x^2 S(x,n-1) - x^2 S(x,n-2)$ (***)

– Thay (***) vào (*) ta được:
$S(x,n) = S(x,n-1) + x^{2n}$
$S(x,n) = S(x,n-1) + x^2 S(x,n-1) - x^2 S(x,n-2)$
$S(x,n) = (1+x^2)S(x,n-1) - x^2 S(x,n-2)$

– Công thức trên được gọi là công thức đệ quy nhị phân.
$S(x,n) = (1+x^2)S(x,n-1) - x^2 S(x,n-2)$

– Điều kiện dừng:
+ $S(x,0) = 0$.
+ $S(x,1) = x^2$.

– Dữ liệu test

```
01678. Dữ liệu vào:   x = 3.1 n = 3
01679. Dữ liệu ra:    989.46567
01680.
01681. Dữ liệu vào:   x = 3.1 n = 11
01682. Dữ liệu ra:    72,057,219,623.21934
01683.
01684. Dữ liệu vào:   x = 3.1 n = 100
01685. Dữ liệu ra:    Tràn số, Tràn stack
```

– Định nghĩa hàm đệ quy.

```
01686. float Tinh(float x,int n)
01687. {
01688.    if(n==0)
01689.       return 0;
01690.    if(n==1)
01691.       return x*x;
01692.    float a = Tinh(x,n-1);
01693.    float b = Tinh(x,n-2);
01694.    return ((1+x*x)*a - x*x*b);
01695. }
```

Bài 055. $S(x,n) = x + x^3 + \cdots + x^{2(n-1)+1} + x^{2n+1}$.

– Ta có:
$S(x,n) = x + x^3 + \cdots + x^{2(n-2)+1} + x^{2(n-1)+1} + x^{2n+1}$.

– Do đó:
$S(x,n) = x + x^3 + \cdots + x^{2(n-2)+1} + x^{2(n-1)+1} + x^{2n+1}$.
$S(x,n-1) = x + x^3 + \cdots + x^{2(n-2)+1} + x^{2(n-1)+1}$.
$S(x,n-2) = x + x^3 + \cdots + x^{2(n-2)+1}$.

– Suy ra:
$S(x,n) = S(x,n-1) + x^{2n+1}$. (*)
$S(x,n-1) = S(x,n-2) + x^{2(n-1)+1}$. (**)

– Nhân hai vế của biểu thức (**) cho x^2 ta được:
$x^2S(x,n-1) = x^2\left(S(x,n-2) + x^{2(n-1)+1}\right)$
$\Rightarrow x^2S(x,n-1) = x^2S(x,n-2) + x^{2n+1}$
$\Rightarrow x^{2n+1} = x^2S(x,n-1) - x^2S(x,n-2)$ (***)

– Thay (***) vào (*) ta được:
$S(x,n) = S(x,n-1) + x^{2n+1}$
$S(x,n) = S(x,n-1) + x^2S(x,n-1) - x^2S(x,n-2)$
$S(x,n) = (1+x^2)S(x,n-1) - x^2S(x,n-2)$

– Công thức trên được gọi là công thức đệ quy nhị phân.
$S(x,n) = (1+x^2)S(x,n-1) - x^2S(x,n-2)$

- Điều kiện dừng:
  + $S(x,0) = x$.
  + $S(x,1) = x + x^3$.
- Dữ liệu test

```
01696. Dữ liệu vào:   x = 3.1 n = 3
01697. Dữ liệu ra:    3,070.44360
01698.
01699. Dữ liệu vào:   x = 3.1 n = 11
01700. Dữ liệu ra:    223,377,380,835.08001
01701.
01702. Dữ liệu vào:   x = 3.1 n = 100
01703. Dữ liệu ra:    Tràn số, Tràn stack
```

- Định nghĩa hàm đệ quy.

```
01704. long double Tinh(long double x,int n)
01705. {
01706.    if(n==0)
01707.       return x;
01708.    if(n==1)
01709.       return (x+x*x*x);
01710.    long double a = Tinh(x,n-1);
01711.    long double b = Tinh(x,n-2);
01712.    return ((1+x*x)*a - x*x*b);
01713. }
```

Bài 056. $S(n) = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \cdots + \frac{1}{1+2+3+\cdots+n}$.

- Ta có:
  $S(n) = 1 + \frac{1}{1+2} + \cdots + \frac{1}{1+2+\cdots+(n-2)} + \frac{1}{1+2+\cdots+(n-1)} + \frac{1}{1+2+3+\cdots+n}$.
- Do đó:
  $S(n) = 1 + \frac{1}{1+2} + \cdots + \frac{1}{1+2+\cdots+(n-2)} + \frac{1}{1+2+\cdots+(n-1)} + \frac{1}{1+2+3+\cdots+n}$.
  $S(n-1) = 1 + \frac{1}{1+2} + \cdots + \frac{1}{1+2+\cdots+(n-2)} + \frac{1}{1+2+\cdots+(n-1)}$.
  $S(n-2) = 1 + \frac{1}{1+2} + \cdots + \frac{1}{1+2+\cdots+(n-2)}$.
- Suy ra:
  $S(n) = S(n-1) + \frac{1}{1+2+3+\cdots+n}$. (*)
  $S(n-1) = S(n-2) + \frac{1}{1+2+\cdots+(n-1)}$. (**)
- Từ (**) ta được:

$$S(n-1)-S(n-2)=\frac{1}{1+2+\cdots+(n-1)}.$$
$$\Rightarrow \frac{1}{S(n-1)-S(n-2)}=1+2+\cdots+(n-1).$$
$$\Rightarrow \frac{1}{S(n-1)-S(n-2)}+n=1+2+\cdots+(n-1)+n.$$
$$\Rightarrow 1+2+\cdots+(n-1)+n=\frac{1}{S(n-1)-S(n-2)}+n. \qquad (***)$$

– Thay (***) vào (*) ta được:

$$S(n)=S(n-1)+\frac{1}{1+2+3+\cdots+n}.$$
$$S(n)=S(n-1)+\frac{1}{\frac{1}{S(n-1)-S(n-2)}+n}.$$

– Công thức trên được gọi là công thức đệ quy nhị phân.

$$S(n)=S(n-1)+\frac{1}{\frac{1}{S(n-1)-S(n-2)}+n}.$$

– Điều kiện dừng:

+ $S(0)=0.$
+ $S(1)=1.$

– Dữ liệu test

```
01714. Dữ liệu vào:   n = 3
01715. Dữ liệu ra:    1.50000
01716.
01717. Dữ liệu vào:   n = 11
01718. Dữ liệu ra:    1.83333
01719.
01720. Dữ liệu vào:   n = 100
01721. Dữ liệu ra:    Tràn số, tràn stack.
```

– Định nghĩa hàm đệ quy.

```
01722. float Tinh(int n)
01723. {
01724.    if(n==0)
01725.       return 0;
01726.    if(n==1)
01727.       return 1;
01728.    float a = Tinh(n-1);
01729.    float b = Tinh(n-2);
01730.    return a + 1/(n + 1/(a-b));
01731. }
```

Bài 057. $S(x,n)=x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots+\frac{x^n}{n!}.$

– Ta có:

$$S(x,n)=x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots+\frac{x^{(n-2)}}{(n-2)!}+\frac{x^{(n-1)}}{(n-1)!}+\frac{x^n}{n!}.$$

- Do đó:

$$S(x,n) = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^{(n-2)}}{(n-2)!} + \frac{x^{(n-1)}}{(n-1)!} + \frac{x^n}{n!}.$$

$$S(x,n-1) = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^{(n-2)}}{(n-2)!} + \frac{x^{(n-1)}}{(n-1)!}.$$

$$S(x,n-2) = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^{(n-2)}}{(n-2)!}.$$

- Suy ra:

$$S(x,n) = S(x,n-1) + \frac{x^n}{n!}. \quad (*)$$

$$S(x,n-1) = S(x,n-2) + \frac{x^{(n-1)}}{(n-1)!}. \quad (**)$$

- Nhân hai vế của biểu thức (**) cho $\frac{x}{n}$ ta được:

$$\frac{x}{n}S(x,n-1) = \frac{x}{n}\left(S(x,n-2) + \frac{x^{(n-1)}}{(n-1)!}\right).$$

$$\Rightarrow \frac{x}{n}S(x,n-1) = \frac{x}{n}S(x,n-2) + \frac{x^n}{n!}.$$

$$\Rightarrow \frac{x^n}{n!} = \frac{x}{n}S(x,n-1) - \frac{x}{n}S(x,n-2) \quad (***)$$

- Thay (***) vào (*) ta được:

$$S(x,n) = S(x,n-1) + \frac{x^n}{n!}.$$

$$S(x,n) = S(x,n-1) + \frac{x}{n}S(x,n-1) - \frac{x}{n}S(x,n-2).$$

$$S(x,n) = \left(1 + \frac{x}{n}\right)S(x,n-1) - \frac{x}{n}S(x,n-2).$$

- Công thức trên được gọi là công thức đệ quy nhị phân.

$$S(x,n) = \left(1 + \frac{x}{n}\right)S(x,n-1) - \frac{x}{n}S(x,n-2).$$

- Dữ liệu test

```
01732. Dữ liệu vào:   x = 3.1 n = 3
01733. Dữ liệu ra:
01734.
01735. Dữ liệu vào:   x = 3.1 n = 11
01736. Dữ liệu ra:
01737.
01738. Dữ liệu vào:   x = 3.1 n = 100
01739. Dữ liệu ra:
```

- Điều kiện dừng:
  - $S(x,0) = 0$.
  - $S(x,1) = x$.
- Định nghĩa hàm đệ quy.

```
01740. float Tinh(float x,int n)
01741. {
01742.    if(n==0)
01743.       return 0;
```

```
01744.     if(n==1)
01745.        return x;
01746.     float a = Tinh(x,n-1);
01747.     float b = Tinh(x,n-2);
01748.     return ((1+x/n)*a - x/n*b);
01749.  }
```

Bài 058. $S(x,n) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + \frac{x^{2n}}{(2n)!}$.

- Ta có:
  $$S(x,n) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + \frac{x^{2(n-2)}}{(2(n-2))!} + \frac{x^{2(n-1)}}{(2(n-1))!} + \frac{x^{2n}}{(2n)!}.$$
- Do đó:
  $$S(x,n) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + \frac{x^{2(n-2)}}{(2(n-2))!} + \frac{x^{2(n-1)}}{(2(n-1))!} + \frac{x^{2n}}{(2n)!}.$$
  $$S(x,n-1) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + \frac{x^{2(n-2)}}{(2(n-2))!} + \frac{x^{2(n-1)}}{(2(n-1))!}.$$
  $$S(x,n-2) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + \frac{x^{2(n-2)}}{(2(n-2))!}.$$
- Suy ra:
  $$S(x,n) = S(x,n-1) + \frac{x^{2n}}{(2n)!}. \quad (*)$$
  $$S(x,n-1) = S(x,n-2) + \frac{x^{2(n-1)}}{(2(n-1))!}. \quad (**)$$
- Nhân hai vế của biểu thức (**) cho $\frac{x^2}{(2n-1)(2n)}$ ta được:
  $$\frac{x^2}{(2n-1)(2n)} S(x,n-1) = \frac{x^2}{(2n-1)(2n)} \left(S(x,n-2) + \frac{x^{2(n-1)}}{(2(n-1))!} \right).$$
  $$\Rightarrow \frac{x^2}{(2n-1)(2n)} S(x,n-1) = \frac{x^2}{(2n-1)(2n)} S(x,n-2) + \frac{x^{2n}}{(2n)!}.$$
  $$\Rightarrow \frac{x^{2n}}{(2n)!} = \frac{x^2}{(2n-1)(2n)} S(x,n-1) - \frac{x^2}{(2n-1)(2n)} S(x,n-2) \ (***)$$
- Thay (***) vào (*) ta được:
  $$S(x,n) = S(x,n-1) + \frac{x^{2n}}{(2n)!}.$$
  $$S(x,n) = S(x,n-1) + \frac{x^2}{(2n-1)(2n)} S(x,n-1) - \frac{x^2}{(2n-1)(2n)} S(x,n-2).$$
  $$S(x,n) = \left(1 + \frac{x^2}{(2n-1)(2n)}\right) S(x,n-1) - \frac{x^2}{(2n-1)(2n)} S(x,n-2).$$
- Công thức trên được gọi là công thức đệ quy nhị phân.
  $$S(x,n) = \left(1 + \frac{x^2}{(2n-1)(2n)}\right) S(x,n-1) - \frac{x^2}{(2n-1)(2n)} S(x,n-2).$$
- Điều kiện dừng:
  + $S(x,0) = 1$.
  + $S(x,1) = 1 + \frac{x^2}{2!}$.
- Định nghĩa hàm đệ quy.

```
01750.  float Tinh(float x,int n)
01751.  {
01752.      if(n==0)
01753.         return 1;
01754.      if(n==1)
01755.         return 1+x*x/2;
01756.      float a = Tinh(x,n-1);
01757.      float b = Tinh(x,n-2);
01758.      float hs = x*x/(2*n-1)/(2*n);
01759.      return ((1 + hs)*a - hs*b);
01760.  }
```

Bài 059. $S(x,n) = 1 + x + \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + \frac{x^{2n+1}}{(2n+1)!}$.

- Ta có:

  $S(x,n) = 1 + x + \frac{x^3}{3!} + \cdots + \frac{x^{2(n-2)+1}}{(2(n-2)+1)!} + \frac{x^{2(n-1)+1}}{(2(n-1)+1)!} + \frac{x^{2n+1}}{(2n+1)!}$.
- Do đó:

  $S(x,n) = 1 + x + \frac{x^3}{3!} + \cdots + \frac{x^{2(n-2)+1}}{(2(n-2)+1)!} + \frac{x^{2(n-1)+1}}{(2(n-1)+1)!} + \frac{x^{2n+1}}{(2n+1)!}$.

  $S(x,n-1) = 1 + x + \frac{x^3}{3!} + \cdots + \frac{x^{2(n-2)+1}}{(2(n-2)+1)!} + \frac{x^{2(n-1)+1}}{(2(n-1)+1)!}$.

  $S(x,n-2) = 1 + x + \frac{x^3}{3!} + \cdots + \frac{x^{2(n-2)+1}}{(2(n-2)+1)!}$.
- Suy ra:

  $S(x,n) = S(x,n-1) + \frac{x^{2n+1}}{(2n+1)!}$. (*)

  $S(x,n-1) = S(x,n-2) + \frac{x^{2(n-1)+1}}{(2(n-1)+1)!}$. (**)
- Nhân hai vế của biểu thức (**) cho $\frac{x^2}{(2n)(2n+1)}$ ta được:

  $\frac{x^2}{(2n)(2n+1)} S(x,n-1) = \frac{x^2}{(2n)(2n+1)} \left(S(x,n-2) + \frac{x^{2(n-1)+1}}{(2(n-1)+1)!} \right)$.

  $\Rightarrow \frac{x^2}{(2n)(2n+1)} S(x,n-1) = \frac{x^2}{(2n)(2n+1)} S(x,n-2) + \frac{x^{2n+1}}{(2n+1)!}$.

  $\Rightarrow \frac{x^{2n+1}}{(2n+1)!} = \frac{x^2}{(2n)(2n+1)} S(x,n-1) - \frac{x^2}{(2n)(2n+1)} S(x,n-2)$ (***)
- Thay (***) vào (*) ta được:

  $S(x,n) = S(x,n-1) + \frac{x^{2n+1}}{(2n+1)!}$.

  $S(x,n) = S(x,n-1) + \frac{x^2}{(2n)(2n+1)} S(x,n-1) - \frac{x^2}{(2n)(2n+1)} S(x,n-2)$.

  $S(x,n) = \left(1 + \frac{x^2}{(2n)(2n+1)}\right) S(x,n-1) - \frac{x^2}{(2n)(2n+1)} S(x,n-2)$.
- Công thức trên được gọi là công thức đệ quy nhị phân.

  $S(x,n) = \left(1 + \frac{x^2}{(2n)(2n+1)}\right) S(x,n-1) - \frac{x^2}{(2n)(2n+1)} S(x,n-2)$.
- Điều kiện dừng:

+ $S(x,0) = 1 + x$.
+ $S(x,1) = 1 + x + \frac{x^3}{3!}$.

– Định nghĩa hàm đệ quy.

```
01761. float Tinh(float x,int n)
01762. {
01763.     if(n==0)
01764.        return (1 + x);
01765.     if(n==1)
01766.        return (1 + x + x*x*x/6);
01767.     float a = Tinh(x,n-1);
01768.     float b = Tinh(x,n-2);
01769.     float hs = x*x/(2*n)/(2*n+1);
01770.     return ((1 + hs)*a - hs*b);
01771. }
```

03.07.03 Thuật toán tìm kiếm nhị phân

Bài 060.Cho mảng một chiều các số nguyên tăng dần. Định nghĩa hàm đệ quy nhị phân tìm vị trí giá trị x trong mảng. Trong trường hợp không tìm thấy hàm trả về giá trị -1.

03.07.04 Thuật toán sắp xếp

Bài 061.Định nghĩa hàm đệ quy nhị phân sắp xếp mảng một chiều cách số thực tăng dần bằng thuật toán Quick sort.

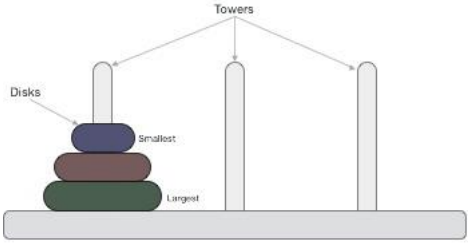
Bài 062.Định nghĩa hàm đệ quy nhị phân sắp xếp mảng một chiều cách số thực tăng dần bằng thuật toán Merge sort.

03.07.05 Bài toán cổ điển

Bài 063.Định nghĩa hàm đệ quy nhị phân giải bài toán Tháp Hà Nội.

– Trong một ngôi đền thiêng liêng có ba chiếc cọc bằng kim cương, được chôn chặt, thẳng đứng. Ông trời đã xếp 64 chiếc đĩa vàng có các đường kính khác nhau và có lỗ ở giữa vào một chiếc cọc theo thứ tự to trước nhỏ sau (vì vậy mới có tên là tháp). Làm thế nào chuyển từng chiếc một toàn bộ 64 cái đĩa đó sang chiếc cọc thứ ba thông qua chiếc cọc thứ hai với điều kiện trong quá tình chuyển ở cả ba cọc, các đĩa to không bao giờ được nằm trên các đĩa nhỏ hơn chúng, và số lần chuyển là bao nhiêu?

- Tổng quát hơn, ta có: cho trước ba chiếc cọc chôn thẳng đứng. Trên một cọc đã xếp n ($n = 1, 2 \ldots$) đĩa có đường kính khác nhau xuyên qua lỗ thủng ở giữa. Đĩa to nằm dưới, đĩa nhỏ hơn nằm trên. Vấn đề đặt ra là làm thế nào để chuyển từng chiếc một toàn bộ số đĩa đó sang chiếc cọc thứ ba thông qua chiếc cọc thứ hai với yêu cầu trong quá trình chuyển, ở cả ba cọc luôn có trật tự đĩa to nằm dưới đĩa bé nằm trên.



- Qui tắc trò chơi toán học Tháp Hà Nội (Tower of Hanoi)
  + Nhiệm vụ của trò chơi là di chuyển các đĩa có kích cỡ khác nhau sang cột khác sao cho vẫn đảm bảo thứ tự ban đầu của các đĩa: đĩa nhỏ nằm trên đĩa lớn.
  + Một lần chỉ được di chuyển một đĩa
  + Một đĩa chỉ có thể được đặt lên một đĩa lớn hơn
- Lời giải cho bài toán tổng quát
  + Với $n = 1$ thì ta chỉ việc
    - chuyển đĩa duy nhất sang cột thứ 3.
  + Với $n = 2$ thì ta chỉ việc
    - chuyển đĩa nhỏ sang cột thứ 2,
    - chuyển đĩa to sang cột thứ 3
    - rồi chuyển đĩa từ cột 2 sang cột thứ 3.
  + Với $n = 3$ ta có cách làm:
    - chuyển đĩa nhỏ nhất sang cột thứ 3,
    - chuyển đĩa nhỏ nhì sang cột thứ 2,
    - chuyển đĩa nhỏ nhất sang cột thứ 2,
    - chuyển đĩa to sang cột thứ 3,
    - chuyển đĩa nhỏ nhất sang cột thứ 1,
    - chuyển đĩa nhỏ nhì sang cột thứ 3,
    - chuyển đĩa nhỏ nhất sang cột thứ 3.
  + Điểm chung của hai cách chơi trên là:
    - ta đều phải chuyển được tất cả $(n - 1)$ đĩa nhỏ qua cột thứ hai,
    - chuyển đĩa to nhất qua cột thứ ba rồi
    - chuyển $(n - 1)$ đĩa từ cột thứ hai sang cột thứ ba. Trò chơi sẽ kết thúc!

- Gọi $S(n)$ là số bước để di chuyển n đĩa qua cột ta cần thì:
  - Bước 1: Chuyển $(n-1)$ đĩa bé hơn từ cột (1) sang cột (2). Chúng ta có $S(n-1)$ bước di chuyển.
  - Bước 2: Chuyển đĩa to nhất từ cột (1) sang cột (3). Chúng ta có 1 bước di chuyển.
  - Bước 3: Chuyển $(n-1)$ đĩa từ cột (2) sang cột (3). Chúng ta có $S(n-1)$ bước di chuyển.
- Vậy, số bước để di chuyển n đĩa từ cột (1) sang cột (3) chính là $S(n) = S(n-1) + 1 + S(n-1)$
  - $S(n) = 2S(n-1) + 1$.
  - $S(2) = 3$.
  - $S(3) = 7$.
- Ta dễ dàng rút ra công thức tổng quát: $S(n) = (2^n - 1)$ bước thực hiện.
- Định nghĩa hàm đệ quy.

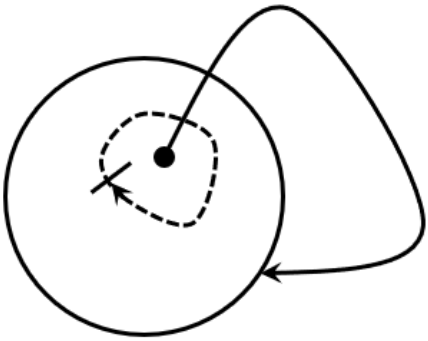
```
01772. void Tower(int n , char a, char b, char c )
01773. {
01774.    if(n==1)
01775.    {
01776.       cout<<"\t"<<a<< "-----"<<c<<endl;
01777.       return;
01778.    }
01779.    Tower(n-1,a,c,b);
01780.    Tower(1,a,b,c);
01781.    Tower(n-1,b,a,c);
01782. }
```

CHƯƠNG 04. ĐỆ QUY PHI TUYẾN

04.01 KHÁI NIỆM ĐỆ QUY PHI TUYẾN

- Khái niệm: Hàm được gọi là đệ quy phi tuyến (multiple recursion) nếu bên trong thân của hàm có lời gọi hàm lại chính nó được đặt bên trong thân vòng lặp hoặc số lời gọi hàm lại chính nó nhiều hơn hai lần.

04.02 HÌNH ẢNH ĐỆ QUY PHI TUYẾN



04.03 CẤU TRÚC HÀM ĐỆ QUY PHI TUYẾN

```
01783.  KDL TenHam(<ThamSo>)
01784.  {
01785.     if <điều kiện dừng>
01786.     {
01787.        ...
01788.        return <Giá Trị Trả Về>;
01789.     }
01790.     ...
01791.     while(<điều kiện lặp>)
01792.     {
01793.        ...
01794.        ...TenHam(<ThamSo>;
01795.        ...
```

```
01796.     }
01797.     ...
01798.  }
```

04.04 MINH HỌA ĐỆ QUY PHI TUYẾN

Bài cơ sở 018. Viết hàm đệ quy tính tổng số hạng thứ n của dãy sau:

$$\begin{cases} x_0 = 1 \\ x_n = n^2x_0 + (n-1)^2x_1 + \cdots + (n-i)^2x_i + \cdots + 2^2x_{n-2} + 1^2x_{n-1} \end{cases}$$

– Định nghĩa hàm

```
01799.  int Tinhxn(int n)
01800.  {
01801.     if(n==0)
01802.        return 1;
01803.     int s = 0;
01804.     for(int i=0;i<=n-1;i++)
01805.     { g
01806.        int xi = Tinhxn(i);
01807.        s = s + (n-i)*(n-i)*xi;
01808.     }
01809.     return s;
01810.  }
```

– Một cách định nghĩa hàm khác

```
01811.  long Tinhxn(int n)
01812.  {
01813.     if(n==0)
01814.        return 1;
01815.     long s = 0;
01816.     for(int i=n;i>=1;i--)
01817.     {
01818.        long xi = Tinhxn(n-i);
01819.        s = s + i*i*xi;
01820.     }
01821.     return s;
01822.  }
```

04.05 BÀI TẬP ĐỆ QUY PHI TUYẾN

Bài 064.Định nghĩa hàm phi tuyến giải bài toán Mã Đi Tuần.

Bài 065.Định nghĩa hàm phi tuyến giải bài toán Tám Quân Hậu.

Bài 066.Định nghĩa hàm phi tuyến giải bài toán phát sinh Chỉnh Hợp.

Bài 067.Định nghĩa hàm phi tuyến giải bài toán phát sinh Tổ Hợp.

CHƯƠNG 05. ĐỆ QUY TUYẾN TÍNH TRÊN MẢNG MỘT CHIỀU

05.01 KHÁI NIỆM ĐỆ QUY TUYẾN TÍNH

- Khái niệm: Một hàm được gọi là đệ quy tuyến tính nếu trong thân của hàm đó có duy nhất một lời gọi hàm gọi lại chính nó một cách tường minh.

05.02 KỸ THUẬT XUẤT MẢNG ĐỆ QUY

05.02.01 Bài cơ sở Kỹ thuật Xuất mảng Đệ quy

Bài cơ sở 019. Định nghĩa hàm xuất mảng một chiều các số nguyên ra màn hình.

- Khai báo hàm.

```
01823.  void Xuat(int [], int);
```

- Định nghĩa hàm đệ quy.

```
01824.  void Xuat(int a[], int n)
01825.  {
01826.     if(n==0)
01827.        return;
01828.     Xuat(a,n-1);
01829.     cout << setw(6) << a[n-1];
01830.  }
```

Bài cơ sở 020. Định nghĩa hàm xuất mảng một chiều các số thực ra màn hình.

- Khai báo hàm.

```
01831.  void Xuat(float [],int);
```

- Định nghĩa hàm đệ quy.

```
01832.  void Xuat(float a[], int n)
01833.  {
01834.     if(n==0)
01835.        return;
```

```
01836.     Xuat(a,n-1);
01837.     cout << setw(10) << setprecision(3) << a[n-1];
01838.  }
```

05.02.02 Bài tập Kỹ thuật Xuất mảng Đệ quy

Bài 068.Viết hàm xuất mảng một chiều các số thực.

– Khai báo hàm.

```
01839.  void Xuat(float [],int);
```

– Định nghĩa hàm đệ quy.

```
01840.  void LietKe(float a[], int n)
01841.  {
01842.     if (n == 0)
01843.        return;
01844.     LietKe(a, n - 1);
01845.     cout<<setw(10)<<setprecision(3)<<a[n-1];
01846.  }
```

Bài 069.Viết hàm xuất mảng một chiều các số nguyên.

– Khai báo hàm.

```
01847.  void Xuat(int [],int);
```

– Định nghĩa hàm đệ quy.

```
01848.  void Xuat(int a[], int n)
01849.  {
01850.     if (n == 0)
01851.        return;
01852.     Xuat(a, n - 1);
01853.     cout << setw(6) << a[n - 1];
01854.  }
```

05.01 KỸ THUẬT LIỆT KÊ ĐỆ QUY

05.01.01 Bài cơ sở Kỹ thuật Liệt kê Đệ quy

Bài cơ sở 021. 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.

```
01855. void LietKe(int [],int);
```

– Định nghĩa hàm đệ quy.

```
01856. void LietKe(int a[],int n)
01857. {
01858.    if(n==0)
01859.       return;
01860.    LietKe(a,n-1);
01861.    if(a[n-1]%2!=0)
01862.       cout << setw(4) << a[n-1];
01863. }
```

05.01.02 Bài tập Kỹ thuật Liệt kê Đệ quy

Bài 070.Viết hàm liệt kê các giá trị chẵn trong mảng một chiều các số nguyên.

– Khai báo hàm.

```
01864. void LietKe(int [],int);
```

– Định nghĩa hàm đệ quy.

```
01865. void LietKe(int a[], int n)
01866. {
01867.    if (n == 0)
01868.       return;
01869.    LietKe(a, n - 1);
01870.    if (a[n - 1] % 2 == 0)
01871.       cout << setw(6) << a[n - 1];
01872. }
```

Bài 071.Hãy liệt kê các số âm trong mảng một chiều các số thực.

– Khai báo hàm.

```
01873. void LietKe(float [],int);
```

– Định nghĩa hàm đệ quy.

```
01874. void LietKe(float a[], int n)
01875. {
01876.    if (n == 0)
01877.       return;
01878.    LietKe(a, n - 1);
01879.    if (a[n - 1] < 0)
```

```
01880.       cout<<setw(10)<<setprecision(3)<<a[n-1];
01881. }
```

Bài 072.Hãy liệt kê các số dương trong mảng một chiều các số thực.

– Khai báo hàm.

```
01882. void LietKe(float [],int);
```

– Định nghĩa hàm đệ quy.

```
01883. void LietKe(float a[], int n)
01884. {
01885.    if (n == 0)
01886.       return;
01887.    LietKe(a, n - 1);
01888.    if (a[n - 1] > 0)
01889.       cout<<setw(10)<<setprecision(3)<<a[n-1];
01890. }
```

Bài 073.Hãy liệt kê các giá trị trong mảng một chiều các số nguyên có chữ số đầu tiên là chữ số lẻ.

– Khai báo hàm.

```
01891. void LietKe(int [],int);
```

– Định nghĩa hàm đệ quy.

```
01892. void LietKe(int a[], int n)
01893. {
01894.    if (n == 0)
01895.       return;
01896.    LietKe(a, n - 1);
01897.    if (ChuSoDau(a[n - 1]) % 2 != 0)
01898.       cout << setw(6) << a[n - 1];
01899. }
```

Bài 074.Hãy liệt kê các giá trị trong mảng các số nguyên có chữ số đầu tiên là chữ số chẵn.

– Khai báo hàm.

```
01900. void LietKe(int [],int);
```

– Định nghĩa hàm đệ quy.

```
01901. void LietKe(int a[], int n)
01902. {
01903.    if (n == 0)
01904.       return;
01905.    LietKe(a, n - 1);
01906.    if (ChuSoDau(a[n - 1]) % 2 == 0)
```

```
01907.        cout << setw(5) << a[n - 1];
01908.  }
```

Bài 075.Hãy liệt kê các giá trị có toàn chữ số lẻ trong mảng một chiều các số nguyên.

– Khai báo hàm.

```
01909.  void LietKe(int[],int);
```

– Định nghĩa hàm đệ quy.

```
01910.  void LietKe(int a[], int n)
01911.  {
01912.     if (n == 0)
01913.        return;
01914.     LietKe(a, n - 1);
01915.     if (ktToanLe(a[n - 1]))
01916.        cout << setw(6) << a[n - 1];
01917.  }
```

Bài 076.Cho mảng một chiều các số nguyên. Hãy viết hàm liệt kê các giá trị trong mảng có dạng 3^k. Nếu mảng không tồn tại giá trị dạng 3^k thì hàm sẽ trả về giá trị 0.

– Khai báo hàm.

```
01918.  void LietKe(int [],int);
```

– Định nghĩa hàm đệ quy.

```
01919.  void LietKe(int a[], int n)
01920.  {
01921.     if (n == 0)
01922.        return;
01923.     LietKe(a, n - 1);
01924.     if (ktDang3m(a[n - 1]))
01925.        cout << setw(6) << a[n - 1];
01926.  }
```

Bài 077.Viết hàm liệt kê các vị trí mà giá trị tại đó là giá trị âm trong mảng một chiều các số thực.

– Khai báo hàm.

```
01927.  void LietKe(float [],int);
```

– Định nghĩa hàm đệ quy.

```
01928.  void LietKe(float a[], int n)
01929.  {
01930.     if (n == 0)
```

```
01931.       return;
01932.    LietKe(a, n - 1);
01933.    if (a[n - 1] < 0)
01934.       cout << setw(6) << n - 1;
01935. }
```

Bài 078.Hãy liệt kê các vị trí mà giá trị tại đó là số nguyên tố trong mảng một chiều các số nguyên.

– Khai báo hàm.

```
01936. void LietKe(int [],int);
```

– Định nghĩa hàm đệ quy.

```
01937. void LietKe(int a[], int n)
01938. {
01939.    if (n == 0)
01940.       return;
01941.    LietKe(a, n - 1);
01942.    if (ktNguyenTo(a[n - 1]))
01943.       cout << setw(6) <<n - 1;
01944. }
```

Bài 079.Hãy liệt kê các vị trí mà giá trị tại vị trí đó là số chính phương trong mảng một chiều các số nguyên.

– Khai báo hàm.

```
01945. void LietKe(int [],int);
```

– Định nghĩa hàm đệ quy.

```
01946. void LietKe(int a[], int n)
01947. {
01948.    if (n == 0)
01949.       return;
01950.    LietKe(a, n - 1);
01951.    if (ktChinhPhuong(a[n - 1]))
01952.       cout << setw(6) <<n - 1;
01953. }
```

Bài 080.Hãy liệt kê các giá trị trong mảng một chiều các số thực thuộc đoạn $[x, y]$ cho trước.

– Khai báo hàm.

```
01954. void LietKe(float [],int, float, float);
```

– Định nghĩa hàm đệ quy.

```
01955. void LietKe(float a[],int n,float x,float y)
01956. {
```

```
01957.     if (n == 0)
01958.        return;
01959.     LietKe(a, n - 1, x, y);
01960.     if (a[n - 1] >= x && a[n - 1] <= y)
01961.        cout << setw(10) << a[n - 1];
01962.  }
```

Bài 081.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 các số nguyên.

– Khai báo hàm.

```
01963.  void LietKe(int [],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
01964.  void LietKe(int a[], int n, int x, int y)
01965.  {
01966.     if (n == 0)
01967.        return;
01968.     LietKe(a, n - 1, x, y);
01969.     if (a[n-1]>=x && a[n-1]<=y && a[n-1]%2==0)
01970.        cout << setw(6) << a[n - 1];
01971.  }
```

Bài 082.(*) Hãy liệt kê các vị trí mà giá trị tại đó là giá trị lớn nhất trong mảng một chiều các số thực.

– Khai báo hàm.

```
01972.  void LietKe(float [],int);
```

– Định nghĩa hàm đệ quy.

```
01973.  void LietKe(float a[],int n)
01974.  {
01975.     if(n==0)
01976.        return ;
01977.     float lc = LonNhat(a,n-1);
01978.     if(lc < a[n-1])
01979.     {
01980.        cout << setw(6) << n-1;
01981.        return;
01982.     }
01983.     if(lc == a[n-1])
01984.        cout << setw(6) << n-1;
01985.     LietKe(a,n-1);
01986.  }
```

Bài 083.(*) Hãy liệt kê các vị trí mà giá trị tại đó là giá trị nhỏ nhất trong mảng một chiều các số thực.

– Khai báo hàm.

```
01987.  void LietKe(float [],int);
```

– Định nghĩa hàm đệ quy.

```
01988.  void LietKe(float a[],int n)
01989.  {
01990.     if(n==0)
01991.        return ;
01992.     float lc = NhoNhat(a,n-1);
01993.     if(lc > a[n-1])
01994.     {
01995.        cout << setw(6) << n-1;
01996.        return;
01997.     }
01998.     if(lc == a[n-1])
01999.        cout << setw(6) << n-1;
02000.     LietKe(a,n-1);
02001.  }
```

Bài 084.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.

```
02002.  void LietKe(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02003.  void LietKe(float a[], int n)
02004.  {
02005.     if (n <= 1)
02006.        return;
```

```
02007.     if(a[n-2] > abs(a[n-1]))
02008.        cout<<setw(10)<<setprecision(3)<<a[n-2];
02009.     LietKe(a, n - 1);
02010.  }
```

Bài 085.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.

```
02011.  void LietKe(float [],int);
```

- Định nghĩa hàm đệ quy.

```
02012.  void LietKe(float a[], int n)
02013.  {
02014.     if (n <= 2)
02015.        return;
02016.     if(a[n-2]>a[n-3] && a[n-2] < abs(a[n-1]))
02017.        cout<<setw(10)<<setprecision(3)<<a[n-2];
02018.     LietKe(a, n - 1);
02019.  }
```

05.02 KỸ THUẬT TÍNH TOÁN ĐỆ QUY

05.02.01 Bài cơ sở Kỹ thuật Tính toán Đệ quy

Bài cơ sở 022. 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.

```
02020. float TongAm(float [],int);
```

– Định nghĩa hàm

```
02021. float TongAm(float a[],int n)
02022. {
02023.    if(n==0)
02024.       return 0;
02025.    float s = TongAm(a,n-1);
02026.    if(a[n-1]<0)
02027.          s = s + a[n-1];
02028.    return s;
02029. }
```

– Định nghĩa hàm đệ quy (cách 02).

```
02030. float TongAm(float a[],int n)
02031. {
02032.    if(n==0)
02033.       return 0;
02034.    float s = TongAm(a,n-1);
02035.    if(a[n-1]<0)
02036.       return s + a[n-1];
02037.    return s;
02038. }
```

– Định nghĩa hàm đệ quy (cách 03).

```
02039. float TongAm(float a[],int n)
02040. {
02041.    if(n==0)
02042.       return 0;
02043.    if(a[n-1]<0)
02044.       return (TongAm(a,n-1)+ a[n-1]);
02045.    return TongAm(a,n-1);
02046. }
```

05.02.02 Bài tập Kỹ thuật Tính toán Đệ quy

Bài 086.Tính tổng các phần tử trong mảng một chiều các số thực.

– Khai báo hàm.

```
02047. float Tong(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02048.  float Tong(float a[],int n)
02049.  {
02050.     if(n==0)
02051.        return 0;
02052.     return Tong(a,n-1) + a[n-1];
02053.  }
```

Bài 087.Tính tổng các giá trị dương trong mảng một chiều các số thực.

– Khai báo hàm.

```
02054.  float TongDuong(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02055.  float TongDuong(float a[],int n)
02056.  {
02057.     if(n==0)
02058.        return 0;
02059.     float s = TongDuong(a,n-1);
02060.     if(a[n-1]>0)
02061.        s = s + a[n-1];
02062.     return s;
02063.  }
```

Bài 088.Tính tổng các giá trị có chữ số hàng chục là chữ số 5 có trong mảng các số nguyên.

– Khai báo hàm.

```
02064.  int TongGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02065.  int TongGiaTri(int a[],int n)
02066.  {
02067.     if(n==0)
02068.        return 0;
02069.     int s = TongGiaTri(a,n-1);
02070.     if(abs(a[n-1])/10%10==5)
02071.        s = s + a[n-1];
02072.     return s;
02073.  }
```

Bài 089.Tính tổng các giá trị chính phương trong mảng các số nguyên.

– Khai báo hàm.

```
02074.  int TongChinhPhuong(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02075.  int TongChinhPhuong(int a[],int n)
02076.  {
02077.     if(n==0)
02078.        return 0;
02079.     int s = TongChinhPhuong(a,n-1);
02080.     if(ktChinhPhuong(a[n-1]))
02081.        s = s + a[n-1];
02082.     return s;
02083.  }
```

Bài 090.Tính tổng các giá trị đối xứng trong mảng các số nguyên.

– Khai báo hàm.

```
02084.  int TongDoiXung(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02085.  int TongDoiXung(int a[],int n)
02086.  {
02087.     if(n==0)
02088.        return 0;
02089.     int s = TongDoiXung(a,n-1);
02090.     if(ktDoiXung(a[n-1]))
02091.        s = s + a[n-1];
02092.     return s;
02093.  }
```

Bài 091.Tính tổng các giá trị có chữ số đầu tiên là chữ số lẻ trong mảng một chiều các số nguyên.

– Khai báo hàm.

```
02094.  int TongGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02095.  int TongGiaTri(int a[],int n)
02096.  {
02097.     if(n==0)
02098.        return 0;
02099.     int s = TongGiaTri(a,n-1);
02100.     if(ChuSoDau(a[n-1])%2 != 0)
02101.        s = s + a[n-1];
02102.     return s;
02103.  }
```

Bài 092.Tính tổng các giá trị có chữ số đầu tiên là chữ số chẵn có trong mảng các số nguyên.

– Khai báo hàm.

```
02104. int TongGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02105. int TongGiaTri(int a[],int n)
02106. {
02107.    if(n==0)
02108.       return 0;
02109.    int s = TongGiaTri(a,n-1);
02110.    if(ChuSoDau(a[n-1])%2 == 0)
02111.       s = s + a[n-1];
02112.    return s;
02113. }
```

Bài 093.Tính tổng các giá trị lớn hơn giá trị đứng liền trước nó trong mảng một chiều các số thực.

– Khai báo hàm.

```
02114. float TongGiaTri(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02115. float TongGiaTri(float a[],int n)
02116. {
02117.    if(n<=1)
02118.       return 0;
02119.    float s = TongGiaTri(a,n-1);
02120.    if(a[n-1] > a[n-2])
02121.       s = s + a[n-1];
02122.    return s;
02123. }
```

Bài 094.Tính tổng các giá trị lớn hơn trị tuyệt đối của giá trị đứng liền sau nó trong mảng một chiều các số thực.

– Khai báo hàm.

```
02124. float TongGiaTri(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02125. float TongGiaTri(float a[],int n)
02126. {
02127.    if(n<=1)
02128.       return 0;
02129.    float s = TongGiaTri(a,n-1);
02130.    if(a[n-2] > abs(a[n-1])
02131.       s = s + a[n-2];
02132.    return s;
```

```
02133. }
```

05.03 KỸ THUẬT ĐẾM ĐỆ QUY

05.03.01 Bài cơ sở Kỹ thuật Đếm Đệ quy

> Bài cơ sở 023. 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.

```
02134. int DemNguyenTo(int [],int);
```

- Cách 01: Định nghĩa hàm

```
02135. int DemNguyenTo(int a[],int n)
02136. {
02137.     if(n==0)
02138.        return 0;
02139.     int dem = DemNguyenTo(a,n-1);
02140.     if(ktNguyenTo(a[n-1]))
02141.           dem++;
02142.     return dem;
02143. }
```

05.03.02 Kỹ thuật Đếm

> Bài 095.Đếm số lượng giá trị chẵn có trong mảng các số nguyên.

- Khai báo hàm.

```
02144. int  DemChan(int [],int);
```

- Định nghĩa hàm đệ quy.

```
02145. int  DemChan(int a[],int n)
02146. {
02147.     if(n==0)
02148.        return 0;
02149.     int dem = DemChan(a,n-1);
02150.     if(a[n-1]%2==0)
02151.        dem++;
02152.     return dem;
```

```
02153. }
```

Bài 096.Đếm số lượng giá trị dương chia hết cho 7 trong mảng một chiều các số nguyên.

– Khai báo hàm.

```
02154. int  DemGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02155. int  DemGiaTri(int a[],int n)
02156. {
02157.    if(n==0)
02158.       return 0;
02159.    int dem = DemGiaTri(a,n-1);
02160.    if(a[n-1]>0 && a[n-1]%7==0)
02161.       dem++;
02162.    return dem;
02163. }
```

Bài 097.Hãy đếm số lượng giá trị có chữ số tận cùng bằng 5 trong mảng các số nguyên.

– Khai báo hàm.

```
02164. int  DemGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02165. int  DemGiaTri(int a[],int n)
02166. {
02167.    if(n==0)
02168.       return 0;
02169.    int dem = DemGiaTri(a,n-1);
02170.    if(abs(a[n-1])%10==5)
02171.       dem++;
02172.    return dem;
02173. }
```

Bài 098.Đếm số lần xuất hiện của giá trị x trong mảng các số thực.

– Khai báo hàm.

```
02174. int  TanSuat(float [],int,float);
```

– Định nghĩa hàm đệ quy.

```
02175. int  TanSuat(float a[],int n,float x)
02176. {
02177.    if(n==0)
02178.       return 0;
02179.    int dem = TanSuat(a,n-1,x);
```

```
02180.     if(a[n-1]==x)
02181.        dem++;
02182.     return dem;
02183.  }
```

Bài 099.Đếm số lượng giá trị đối xứng trong mảng các số nguyên.

– Khai báo hàm.

```
02184.  int  DemDoiXung(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02185.  int DemDoiXung(int a[],int n)
02186.  {
02187.     if(n==0)
02188.        return 0;
02189.     int dem = DemDoiXung(a,n-1);
02190.     if(ktDoiXung(a[n-1]))
02191.        dem++;
02192.     return dem;
02193.  }
```

Bài 100.Hãy đếm số lượng "số nguyên tố" có trong mảng một chiều các số nguyên.

– Khai báo hàm.

```
02194.  int  DemNguyenTo(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02195.  int DemNguyenTo(int a[],int n)
02196.  {
02197.     if(n==0)
02198.        return 0;
02199.     int dem = DemNguyenTo(a,n-1);
02200.     if(ktNguyenTo (a[n-1]))
02201.        dem++;
02202.     return dem;
02203.  }
```

Bài 101.Hãy đếm số lượng "số hoàn thiện" có trong mảng một chiều các số nguyên.

– Khai báo hàm.

```
02204.  int  DemHoanThien(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02205.  int DemHoanThien(int a[],int n)
02206.  {
```

```
02207.     if(n==0)
02208.        return 0;
02209.     int dem = DemHoanThien(a,n-1);
02210.     if(ktHoanThien(a[n-1]))
02211.        dem++;
02212.     return dem;
02213.  }
```

Bài 102.(*) Hãy đếm số lượng các giá trị lớn nhất có trong mảng một chiều các số thực.

– Khai báo hàm.

```
02214.  int  DemLonNhat(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02215.  int  DemLonNhat(float a[],int n)
02216.  {
02217.     if(n==0)
02218.        return 0;
02219.     float lc = LonNhat(a,n-1);
02220.     if(lc < a[n-1])
02221.        return 1;
02222.     int dem = DemLonNhat(a,n-1);
02223.     if(lc == a[n-1])
02224.        dem++;
02225.     return dem;
02226.  }
```

05.04 KỸ THUẬT TÌM KIẾM ĐỆ QUY

05.04.01 Bài cơ sở Kỹ thuật Đếm Đệ quy

Bài cơ sở 024. Định nghĩa hàm đệ quy 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.

```
02227.  float LonNhat(float [],int);
```

– Định nghĩa hàm

```
02228.  float LonNhat(float a[],int n)
```

```
02229. {
02230.    if(n==1)
02231.       return a[0];
02232.    float lc = LonNhat(a,n-1);
02233.    if(a[n-1]>lc)
02234.          lc = a[n-1];
02235.    return lc;
02236. }
```

Bài cơ sở 025. 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.

– Khai báo hàm.

```
02237. float DuongDau(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02238. float DuongDau(float a[],int n)
02239. {
02240.    if(n==0)
02241.       return 0;
02242.    float lc = DuongDau(a,n-1);
02243.    if(lc!=0)
02244.       return lc;
02245.    if(a[n-1]>0)
02246.       return a[n-1];
02247.    return 0;
02248. }
```

Bài cơ sở 026. 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.

– Khai báo hàm.

```
02249. int ChanCuoi(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02250. int ChanCuoi(int a[],int n)
02251. {
02252.    if(n==0)
02253.       return -1;
02254.    if(a[n-1]%2==0)
02255.       return a[n-1];
02256.    return ChanCuoi(a,n-1);
```

```
02257. }
```

Bài cơ sở 027. 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.

– Khai báo hàm.

```
02258. int ViTriDau(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02259. int ViTriDau(int a[],int n)
02260. {
02261.    if(n==0)
02262.       return -1;
02263.    int lc = ViTriDau(a,n-1);
02264.    if(lc!=-1)
02265.       return lc;
02266.    if(a[n-1]%2==0)
02267.       return n-1;
02268.    return -1;
02269. }
```

Bài cơ sở 028. 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.

– Khai báo hàm.

```
02270. int ViTriCuoi(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02271. int ViTriCuoi(int a[],int n)
02272. {
02273.    if(n==0)
02274.       return -1;
02275.    if(ktHoanThien(a[n-1]))
02276.       return n-1;
02277.    return ViTriCuoi(a,n-1);
02278. }
```

Bài cơ sở 029. 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.

– Khai báo hàm.

```
02279.  float DuongNhoNhat(float [],int);
```

– Định nghĩa hàm

```
02280.  float DuongNhoNhat(float a[],int n)
02281.  {
02282.     if(n==0)
02283.        return 0;
02284.     float lc = DuongNhoNhat(a,n-1);
02285.     if(a[n-1]<=0)
02286.        return lc;
02287.     if(lc==0)
02288.        return a[n-1];
02289.     if(a[n-1] < lc)
02290.        lc = a[n-1];
02291.     return lc;
02292.  }
```

Bài cơ sở 030. 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.

– Khai báo hàm.

```
02293.  int TimViTri(float [],int);
```

– Định nghĩa hàm

```
02294.  int TimViTri(float a[],int n)
02295.  {
02296.     if(n==0)
02297.        return -1;
02298.     int lc = TimViTri(a,n-1);
02299.     if(a[n-1]<=0)
02300.        return lc;
02301.     if(lc==-1)
02302.        return n-1;
02303.     if(a[n-1] < a[lc])
02304.        lc = n-1;
02305.     return lc;
02306.  }
```

05.04.02 Kỹ thuật Tìm kiếm – Kỹ thuật Đặt lính canh

Bài 103.(Hạt nhân) Viết hàm tìm "giá trị lớn nhất" trong mảng một chiều số thực.

– Khai báo hàm.

```
02307. float LonNhat(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02308. float LonNhat(float a[],int n)
02309. {
02310.    if(n==1)
02311.       return a[0];
02312.    float lc = LonNhat(a,n-1);
02313.    if(a[n-1]>lc)
02314.       lc = a[n-1];
02315.    return lc;
02316. }
```

Bài 104.Tìm "giá trị nhỏ nhất" trong mảng một chiều số thực.

– Khai báo hàm.

```
02317. float NhoNhat(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02318. float NhoNhat(float a[],int n)
02319. {
02320.    if(n==1)
02321.       return a[0];
02322.    float lc = NhoNhat(a,n-1);
02323.    if(a[n-1]<lc)
02324.       lc = a[n-1];
02325.    return lc;
02326. }
```

Bài 105.Tìm "một vị trí mà giá trị tại vị trí đó là giá trị nhỏ nhất" trong mảng một chiều các số thực.

– Khai báo hàm.

```
02327. int TimViTri(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02328. int TimViTri(float a[],int n)
02329. {
02330.    if(n==1)
02331.       return 0;
```

```
02332.    int lc = TimViTri(a,n-1);
02333.    if(a[n-1]<a[lc])
02334.       lc = n-1;
02335.    return lc;
02336. }
```

Bài 106.Hãy tìm giá trị trong mảng các số thực "xa giá trị x nhất".

– Khai báo hàm.

```
02337. float XaNhat(float [],int,float);
```

– Định nghĩa hàm đệ quy.

```
02338. float XaNhat(float a[],int n,float x)
02339. {
02340.    if(n==1)
02341.       return a[0];
02342.    float lc = XaNhat(a,n-1,x);
02343.    if(abs(a[n-1]-x)>abs(lc-x))
02344.       lc = a[n-1];
02345.    return lc;
02346. }
```

Bài 107.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".

– Khai báo hàm.

```
02347. int TimViTri(float [],int,float);
```

– Định nghĩa hàm đệ quy.

```
02348. int TimViTri(float a[],int n,float x)
02349. {
02350.    if(n==1)
02351.       return 0;
02352.    int lc = TimViTri(a,n-1,x);
02353.    if(abs(a[n-1]-x) < abs(a[lc]-x))
02354.       lc = n-1;
02355.    return lc;
02356. }
```

Bài 108.Cho mảng một chiều các số thực hãy tìm đoạn $[x, y]$ sao cho đoạn này chứa tất cả các giá trị trong mảng.

– Khai báo hàm.

```
02357. void TimDoan(float [],int,float&,float&);
```

– Định nghĩa hàm đệ quy.

```
02358. void TimDoan(float a[],int n,float&x,float&y)
```

```
02359. {
02360.    if(n==1)
02361.    {
02362.       x = a[0];
02363.       y = a[0];
02364.       return;
02365.    }
02366.    TimDoan(a,n-1,x,y);
02367.    if(a[n-1] < x)
02368.       x = a[n-1];
02369.    if(a[n-1] > y)
02370.       y = a[n-1];
02371. }
```

Bài 109.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.

– Khai báo hàm.

```
02372. float TimX(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02373. float TimX(float a[],int n)
02374. {
02375.    if(n==1)
02376.       return abs(a[0]);
02377.    float lc = TimX(a,n-1);
02378.    if(abs(a[n-1]) > lc)
02379.       lc = abs(a[n-1]);
02380.    return lc;
02381. }
```

Bài 110.(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.

– Khai báo hàm.

```
02382. float DuongDau(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02383. float DuongDau(float a[],int n)
02384. {
02385.    if(n==0)
02386.       return 0;
02387.    float lc = DuongDau(a,n-1);
02388.    if(lc!=0)
```

```
02389.       return lc;
02390.    if(a[n-1]>0)
02391.       return a[n-1];
02392.    return 0;
02393. }
```

Bài 111.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.

– Khai báo hàm.

```
02394. float AmDau(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02395. float AmDau(float a[],int n)
02396. {
02397.    if(n==0)
02398.       return 0;
02399.    float lc = AmDau(a,n-1);
02400.    if(lc!=0)
02401.       return lc;
02402.    if(a[n-1]<0)
02403.       return a[n-1];
02404.    return 0;
02405. }
```

Bài 112.Hãy tìm giá trị đầu tiên lớn hơn giá trị 2003 trong mảng thực. Nếu mảng không có giá trị thỏa điều kiện thì hàm trả về giá trị là 0.

– Khai báo hàm.

```
02406. float DauTien(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02407. float DauTien(float a[],int n)
02408. {
02409.    if(n==0)
02410.       return 0;
02411.    float lc = DauTien(a,n-1);
02412.    if(lc!=0)
02413.       return lc;
02414.    if(a[n-1]>2003)
02415.       return a[n-1];
02416.    return 0;
02417. }
```

Bài 113.Viết hàm tìm "số chẵn đầu tiên" trong mảng nguyên. Nếu mảng không có giá trị chẵn thì hàm sẽ trả về giá trị không chẵn là -1.

– Khai báo hàm.

```
02418.  int ChanDau(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02419.  int ChanDau(int a[],int n)
02420.  {
02421.     if(n==0)
02422.        return -1;
02423.     int lc = ChanDau(a,n-1);
02424.     if(lc!=-1)
02425.        return lc;
02426.     if(a[n-1]%2==0)
02427.        return a[n-1];
02428.     return -1;
02429.  }
```

Bài 114.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.

– Khai báo hàm.

```
02430.  int DauTien(int [],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
02431.  int DauTien(int a[],int n,int x,int y)
02432.  {
02433.     if(n==0)
02434.        return x;
02435.     int lc = DauTien(a,n-1,x,y);
02436.     if(lc!=x)
02437.        return lc;
02438.     if(a[n-1]>x && a[n-1]<y)
02439.        return a[n-1];
02440.     return x;
02441.  }
```

Bài 115.(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.

– Khai báo hàm.

```
02442.  int ChanCuoi(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02443.  int ChanCuoi(int a[],int n)
02444.  {
02445.     if(n==0)
02446.        return -1;
02447.     if(a[n-1]%2==0)
02448.        return a[n-1];
02449.     return ChanCuoi(a,n-1);
02450.  }
```

Bài 116.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.

– Khai báo hàm.

```
02451.  float DuongCuoi(float[],int);
```

– Định nghĩa hàm đệ quy.

```
02452.  float DuongCuoi(float a[],int n)
02453.  {
02454.     if(n==0)
02455.        return -1;
02456.     if(a[n-1]>0)
02457.        return a[n-1];
02458.     return DuongCuoi(a,n-1);
02459.  }
```

Bài 117.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.

– Khai báo hàm.

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

– Định nghĩa hàm đệ quy.

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

Bài 118.Tìm số chính phương đầu trong mảng một chiều các số nguyên. Nếu mảng không có số chính phương thì hàm trả về giá trị là -1.

– Khai báo hàm.

```
02469.  int ChinhPhuongDau(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02470.  int ChinhPhuongDau(int a[],int n)
02471.  {
02472.     if(n==0)
02473.        return -1;
02474.     int lc = ChinhPhuongDau(a,n-1);
02475.     if(lc!=-1)
02476.        return lc;
02477.     if(ktChinhPhuong(a[n-1]))
02478.        return a[n-1];
02479.     return -1;
02480.  }
```

Bài 119.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.

– Khai báo hàm.

```
02481.  int NguyenToDau(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02482.  int NguyenToDau(int a[],int n)
02483.  {
02484.     if(n==0)
02485.        return -1;
02486.     int lc = NguyenToDau(a,n-1);
02487.     if(lc!=-1)
02488.        return lc;
02489.     if(ktNguyenTo(a[n-1]))
02490.        return a[n-1];
02491.     return -1;
02492.  }
```

Bài 120.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.

– Khai báo hàm.

```
02493.  int HoanThienDau(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02494.  int HoanThienDau(int a[],int n)
02495.  {
02496.     if(n==0)
02497.        return -1;
02498.     int lc = HoanThienDau(a,n-1);
02499.     if(lc!=-1)
02500.        return lc;
02501.     if(ktHoanThien(a[n-1]))
02502.        return a[n-1];
02503.     return -1;
02504.  }
```

Bài 121.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.

– Khai báo hàm.

```
02505.  int DoiXungDau(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02506.  int DoiXungDau(int a[],int n)
02507.  {
02508.     if(n==0)
02509.        return 10;
02510.     int lc = DoiXungDau(a,n-1);
02511.     if(lc!=10)
02512.        return lc;
02513.     if(ktDoiXung(a[n-1]))
02514.        return a[n-1];
02515.     return 10;
02516.  }
```

Bài 122.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.

– Khai báo hàm.

```
02517.  int TimGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02518.  int TimGiaTri(int a[],int n)
02519.  {
02520.     if(n==0)
02521.        return 0;
02522.     int lc = TimGiaTri(a,n-1);
```

```
02523.     if(lc!=0)
02524.        return lc;
02525.     if(ChuSoDau(a[n-1])%2!=0)
02526.        return a[n-1];
02527.     return 0;
02528.  }
```

Bài 123.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.

– Khai báo hàm.

```
02529.  int TimGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02530.  int TimGiaTri(int a[],int n)
02531.  {
02532.     if(n==0)
02533.        return 0;
02534.     int lc = TimGiaTri(a,n-1);
02535.     if(lc!=0)
02536.        return lc;
02537.     if(ktDang2m(a[n-1]))
02538.        return a[n-1];
02539.     return 0;
02540.  }
```

Bài 124.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.

– Khai báo hàm.

```
02541.  int NguyenToCuoi(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02542.  int NguyenToCuoi(int a[],int n)
02543.  {
02544.     if(n==0)
02545.        return -1;
02546.     if(ktNguyenTo(a[n-1]))
02547.        return a[n-1];
02548.     return NguyenToCuoi(a,n-1);
02549.  }
```

Bài 125.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.

– Khai báo hàm.

```
02550. int HoanThienCuoi(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02551. int HoanThienCuoi(int a[],int n)
02552. {
02553.    if(n==0)
02554.       return -1;
02555.    if(ktHoanThien(a[n-1]))
02556.       return a[n-1];
02557.    return HoanThienCuoi(a,n-1);
02558. }
```

Bài 126.(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.

– Khai báo hàm.

```
02559. int ViTriDau(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02560. int ViTriDau(int a[],int n)
02561. {
02562.    if(n==0)
02563.       return -1;
02564.    int lc = ViTriDau(a,n-1);
02565.    if(lc!=-1)
02566.       return lc;
02567.    if(a[n-1]%2==0)
02568.       return n-1;
02569.    return -1;
02570. }
```

Bài 127.(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.

– Khai báo hàm.

```
02571. int ViTriCuoi(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02572. int ViTriCuoi(int a[],int n)
```

```
02573. {
02574.    if(n==0)
02575.       return -1;
02576.    if(ktHoanThien(a[n-1]))
02577.       return n-1;
02578.    return ViTriCuoi(a,n-1);
02579. }
```

Bài 128.(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.

– Khai báo hàm.

```
02580. float DuongNhoNhat(float [],int);
```

– Định nghĩa hàm

```
02581. float DuongNhoNhat(float a[],int n)
02582. {
02583.    if(n==0)
02584.       return 0;
02585.    float lc = DuongNhoNhat(a,n-1);
02586.    if(a[n-1]<=0)
02587.       return lc;
02588.    if(lc==0)
02589.       return a[n-1];
02590.    if(a[n-1] < lc)
02591.       lc = a[n-1];
02592.    return lc;
02593. }
```

Bài 129.(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.

– Khai báo hàm.

```
02594. int TimViTri(float [],int);
```

– Định nghĩa hàm

```
02595. int TimViTri(float a[],int n)
02596. {
02597.    if(n==0)
02598.       return -1;
02599.    int lc = TimViTri(a,n-1);
02600.    if(a[n-1]<=0)
```

```
02601.       return lc;
02602.    if(lc==-1)
02603.       return n-1;
02604.    if(a[n-1] < a[lc])
02605.       lc = n-1;
02606.    return lc;
02607. }
```

Bài 130.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.

– Khai báo hàm.

```
02608. float AmNhoNhat(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02609. float AmNhoNhat(float a[],int n)
02610. {
02611.    if(n==0)
02612.       return 0;
02613.    float lc = AmNhoNhat(a,n-1);
02614.    if(a[n-1]>=0)
02615.       return lc;
02616.    if(lc==0)
02617.       return a[n-1];
02618.    if(a[n-1] > lc)
02619.       lc = a[n-1];
02620.    return lc;
02621. }
```

Bài 131.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.

– Khai báo hàm.

```
02622. int TimGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02623. int TimGiaTri(int a[],int n)
02624. {
02625.    if(n==0)
02626.       return 0;
02627.    int lc = TimGiaTri(a,n-1);
02628.    if(!ktNguyenTo(a[n-1]))
02629.       return lc;
02630.    if(lc==0)
02631.       return a[n-1];
```

```
02632.     if(a[n-1] > lc)
02633.        lc = a[n-1];
02634.     return lc;
02635.  }
```

Bài 132.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.

– Khai báo hàm.

```
02636.  int TimGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02637.  int TimGiaTri(int a[],int n)
02638.  {
02639.     if(n==0)
02640.        return -1;
02641.     int lc = TimGiaTri(a,n-1);
02642.     if(!ktHoanThien(a[n-1]))
02643.        return lc;
02644.     if(lc==-1)
02645.        return a[n-1];
02646.     if(a[n-1] < lc)
02647.        lc = a[n-1];
02648.     return lc;
02649.  }
```

Bài 133.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.

– Khai báo hàm.

```
02650.  int ChanNhoNhat(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02651.  int ChanNhoNhat(int a[],int n)
02652.  {
02653.     if(n==0)
02654.        return -1;
02655.     int lc = ChanNhoNhat (a,n-1);
02656.     if(a[n-1]%2!=0)
02657.        return lc;
02658.     if(lc==-1)
02659.        return a[n-1];
02660.     if(a[n-1] < lc)
```

```
02661.        lc = a[n-1];
02662.     return lc;
02663.  }
```

Bài 134.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.

– Khai báo hàm.

```
02664.  int TimViTri(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02665.  int TimViTri(float a[],int n)
02666.  {
02667.     if(n==0)
02668.        return -1;
02669.     int lc = TimViTri(a,n-1);
02670.     if(a[n-1]>=0)
02671.        return lc;
02672.     if(lc==-1)
02673.        return n-1;
02674.     if(a[n-1] > a[lc])
02675.        lc = n-1;
02676.     return lc;
02677.  }
```

Bài 135.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).

– Khai báo hàm.

```
02678.  int TimGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02679.  int TimGiaTri(int a[],int n)
02680.  {
02681.     if(n==0)
02682.        return 0;
02683.     int lc = TimGiaTri(a,n-1);
02684.     if(!ktToanLe(a[n-1]))
02685.        return lc;
02686.     if(lc==0)
02687.        return a[n-1];
02688.     if(a[n-1] > lc)
02689.        lc = a[n-1];
02690.     return lc;
```

```
02691. }
```

Bài 136.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 (168).

– Khai báo hàm.

```
02692. int TimGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02693. int TimGiaTri(int a[],int n)
02694. {
02695.    if(n==0)
02696.       return 0;
02697.    int lc = TimGiaTri(a,n-1);
02698.    if(!ktDang2m(a[n-1]))
02699.       return lc;
02700.    if(lc==0)
02701.       return a[n-1];
02702.    if(a[n-1] > lc)
02703.       lc = a[n-1];
02704.    return lc;
02705. }
```

Bài 137.(*) Hãy tìm một giá trị có số lần xuất hiện nhiều nhất trong mảng các số nguyên.

– Khai báo hàm.

```
02706. int TimGiaTri(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02707. int TimGiaTri(int a[],int n)
02708. {
02709.    if(n==1)
02710.       return a[0];
02711.    int lc = TimGiaTri(a,n-1);
02712.    if(TanSuat(a,n,a[n-1])>TanSuat(a,n,lc))
02713.       lc = a[n-1];
02714.    return lc;
02715. }
```

Bài 138.Cho mảng một chiều các số nguyên dương. Hãy viết hàm tìm ước chung lớn nhất của tất cả các phần tử trong mảng.

– Khai báo hàm.

```
02716. int TimUCLN(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02717.  int TimUCLN(int a[],int n)
02718.  {
02719.     if(n==1)
02720.        return a[0];
02721.     int lc = TimUCLN(a,n-1);
02722.     lc = ucln(lc,a[n-1]);
02723.     return lc;
02724.  }
```

Bài 139. Cho mảng một chiều các số nguyên dương. Hãy viết hàm tìm bội chung nhỏ nhất của tất cả các phần tử trong mảng.

– Khai báo hàm.

```
02725.  int TimBCNN(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02726.  int TimBCNN(int a[],int n)
02727.  {
02728.     if(n==1)
02729.        return a[0];
02730.     int lc = TimBCNN(a,n-1);
02731.     lc = bcnn(lc,a[n-1]);
02732.     return lc;
02733.  }
```

05.05 KỸ THUẬT ĐẶT CỜ HIỆU ĐỆ QUY

05.05.01 Bài cơ sở Kỹ thuật Đếm Đệ quy

Bài cơ sở 031. 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.

```
02734. int ktTonTaiChan(int [],int);
```

– Định nghĩa hàm

```
02735. int ktTonTaiChan(int a[],int n)
02736. {
02737.    if(n==0)
02738.       return 0;
02739.    if(a[n-1]%2==0 && a[n-1]<2004)
02740.          return 1;
02741.    return ktTonTaiChan(a,n-1);
02742. }
```

Bài cơ sở 032. 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.

– Khai báo hàm.

```
02743. int ktTinhChat(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02744. int ktTinhChat(int a[],int n)
02745. {
02746.    if(n==0)
02747.       return 1;
02748.    if(ktHoanThien(a[n-1]) && a[n-1]>256)
02749.       return 0;
02750.    return ktTinhChat(a,n-1);
02751. }
```

Bài cơ sở 033. Hãy cho biết mảng các số nguyên có toàn số chẵn hay không? Nếu mảng có tồn tại giá trị lẻ trả về giá trị 0, ngược lại trả về 1.

– Khai báo hàm.

```
02752. int ktToanChan(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02753. int ktToanChan(int a[],int n)
02754. {
02755.    if(n==0)
02756.       return 0;
02757.    if(n==1)
```

```
02758.     {
02759.        if(a[n-1]%2==0)
02760.           return 1;
02761.        return 0;
02762.     }
02763.     if(a[n-1]%2==0 && ktToanChan(a,n-1)==1)
02764.        return 1;
02765.     return 0;
02766.  }
```

05.05.02 Kỹ thuật Đặt cờ hiệu

Bài 140.(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.

– Khai báo hàm.

```
02767.  int ktKhong(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02768.  int ktKhong(int a[],int n)
02769.  {
02770.     if(n==0)
02771.        return 0;
02772.     if(a[n-1]==0)
02773.        return 1;
02774.     return ktKhong(a,n-1);
02775.  }
```

Bài 141.Hãy kiểm tra mảng số nguyên có tồn tại hai giá trị không liên tiếp hay không?

– Khai báo hàm.

```
02776.  int ktTonTai(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02777.  int ktTonTai(int a[],int n)
02778.  {
02779.     if(n<=1)
02780.        return 0;
02781.     if(a[n-1]==0 && a[n-2]==0)
02782.        return 1;
02783.     return ktTonTai(a,n-1);
02784.  }
```

Bài 142.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.

– Khai báo hàm.

```
02785. int ktTonTaiChan(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02786. int ktTonTaiChan(int a[],int n)
02787. {
02788.    if(n==0)
02789.       return 0;
02790.    if(a[n-1]%2==0)
02791.       return 1;
02792.    return ktTonTaiChan(a,n-1);
02793. }
```

Bài 143.Hãy kiểm tra mảng số nguyên có tồn tại số nguyên tố hay không? Nếu có trả về 1, nếu không trả về 0.

– Khai báo hàm.

```
02794. int ktTonTaiNguyenTo(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02795. int ktTonTaiNguyenTo(int a[],int n)
02796. {
02797.    if(n==0)
02798.       return 0;
02799.    if(ktNguyenTo(a[n-1]))
02800.       return 1;
02801.    return ktTonTaiNguyenTo(a,n-1);
02802. }
```

Bài 144.(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.

– Khai báo hàm.

```
02803. int ktTinhChat(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02804. int ktTinhChat(int a[],int n)
02805. {
02806.    if(n==0)
02807.       return 1;
02808.    if(ktHoanThien(a[n-1]) && a[n-1]>256)
```

```
02809.        return 0;
02810.     return ktTinhChat(a,n-1);
02811.  }
```

Bài 145.(Hạt nhân) Hãy cho biết mảng các số nguyên có toàn số chẵn hay không? Nếu mảng có tồn tại giá trị lẻ trả về giá trị 0, ngược lại trả về 1.

– Khai báo hàm.

```
02812.  int ktToanChan(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02813.  int ktToanChan(int a[],int n)
02814.  {
02815.     if(n==0)
02816.        return 0;
02817.     if(n==1)
02818.     {
02819.        if(a[n-1]%2==0)
02820.           return 1;
02821.        return 0;
02822.     }
02823.     if(a[n-1]%2==0 && ktToanChan(a,n-1)==1)
02824.        return 1;
02825.     return 0;
02826.  }
```

Bài 146.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?

– Khai báo hàm.

```
02827.  int ktChanLe(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02828.  int ktChanLe(int a[],int n)
02829.  {
02830.     if(n==0)
02831.        return 0;
02832.     if(n==1)
02833.        return 0;
02834.     if(n==2)
02835.     {
02836.        if((a[0]+a[1])%2!=0)
02837.           return 1;
```

```
02838.        return 0;
02839.     }
02840.     if((a[n-1]+a[n-2])%2!=0)
02841.        if(ktChanLe(a,n-1)==1)
02842.           return 1;
02843.     return 0;
02844. }
```

Bài 147.Hãy kiểm tra mảng có tăng dần hay không?

– Khai báo hàm.

```
02845. int ktTang(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02846. int ktTang(int a[],int n)
02847. {
02848.    if(n==0)
02849.       return 0;
02850.    if(n==1)
02851.       return 1;
02852.    if((a[n-2]<=a[n-1]) && ktTang(a,n-1)==1)
02853.       return 1;
02854.    return 0;
02855. }
```

Bài 148.Hãy kiểm tra mảng có giảm dần hay không?

– Khai báo hàm.

```
02856. int ktGiam(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02857. int ktGiam(int a[],int n)
02858. {
02859.    if(n==0)
02860.       return 0;
02861.    if(n==1)
02862.       return 1;
02863.    if((a[n-2]>=a[n-1]) && ktGiam(a,n-1)==1)
02864.       return 1;
02865.    return 0;
02866. }
```

Bài 149.Hãy cho biết các phần tử trong mảng có bằng nhau không?

– Khai báo hàm.

```
02867. int ktBang(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02868.  int ktBang(int a[],int n)
02869.  {
02870.     if(n==0)
02871.        return 0;
02872.     if(n==1)
02873.        return 1;
02874.     if((a[n-2]==a[n-1]) && ktBang(a,n-1)==1)
02875.        return 1;
02876.     return 0;
02877.  }
```

05.06 KỸ THUẬT XÂY DỰNG MẢNG ĐỆ QUY

05.06.01 Bài cơ sở Kỹ thuật Đếm Đệ quy

> Bài cơ sở 034. 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.

```
02878.  void XayDung(int [],int,int [],int&);
```

– Định nghĩa hàm

```
02879.  void XayDung(int a[],int n, int b[],int &k)
02880.  {
02881.     if(n==0)
02882.     {
02883.        k = 0;
02884.        return;
02885.     }
02886.     XayDung(a,n-1,b,k);
02887.     if(ktDoiXung(a[n-1]))
02888.        b[k++] = a[n-1];
02889.  }
```

05.06.02 Kỹ thuật Xây dựng mảng

Bài 150.(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.

– Khai báo hàm.

```
02890. void XayDung(int [],int,int [],int&);
```

– Định nghĩa hàm đệ quy.

```
02891. void XayDung(int a[],int n, int b[],int &k)
02892. {
02893.    if(n==0)
02894.    {
02895.       k = 0;
02896.       return;
02897.    }
02898.    XayDung(a,n-1,b,k);
02899.    if(a[n-1]%2!=0)
02900.       b[k++] = a[n-1];
02901. }
```

Bài 151.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.

– Khai báo hàm.

```
02902. void XayDung(float [],int,float [],int&);
```

– Định nghĩa hàm đệ quy.

```
02903. void XayDung(float a[],int n,float b[],int&k)
02904. {
02905.    if(n==0)
02906.    {
02907.       k = 0;
02908.       return;
02909.    }
02910.    XayDung(a,n-1,b,k);
02911.    if(a[n-1]<0)
02912.       b[k++] = a[n-1];
02913. }
```

Bài 152.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*.

– Khai báo hàm.

```
02914. void XayDung(int [],int,int [],int&);
```

– Định nghĩa hàm đệ quy.

```
02915. void XayDung(int a[],int n, int b[],int &k)
02916. {
02917.    if(n==0)
02918.    {
02919.       k = 0;
02920.       return;
02921.    }
02922.    XayDung(a,n-1,b,k);
02923.    if(ktNguyenTo(a[n-1]))
02924.       b[k++] = a[n-1];
02925. }
```

05.07 KỸ THUẬT SẮP XẾP ĐỆ QUY

05.07.01 Bài cơ sở Kỹ thuật Đếm Đệ quy

Bài cơ sở 035. Định nghĩa hàm sắp xếp mảng một chiều các số thực tăng dần bằng phương pháp đệ quy.

– 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.

```
02926. void SapTang(float [],int);
```

– Định nghĩa hàm

```
02927. void SapTang(float a[],int n)
02928. {
02929.    if(n==1)
02930.       return;
02931.    for(int i=0;i<=n-2;i++)
02932.       if(a[i]>a[n-1])
02933.          HoanVi(a[i],a[n-1]);
02934.    SapTang(a,n-1);
02935. }
```

Bài cơ sở 036. 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.

– Khai báo hàm.

```
02936.  void ViTriLeTang(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02937.  void ViTriLeTang(float a[],int n)
02938.  {
02939.     if(n==1)
02940.        return;
02941.     for(int i=0;i<=n-2;i++)
02942.        if(i%2!=0 && (n-1)%2!=0 && a[i]>a[n-1])
02943.           HoanVi(a[i],a[n-1]);
02944.     ViTriLeTang(a,n-1);
02945.  }
```

05.07.02 Kỹ thuật Sắp xếp

Bài 153.(Hạt nhân) Hãy sắp xếp các giá trị trong mảng các số thực tăng dần.

– Khai báo hàm.

```
02946.  void SapTang(float [],int);
```

– Định nghĩa hàm đệ quy.

```
02947.  void SapTang(float a[],int n)
02948.  {
02949.     if(n==1)
02950.        return;
02951.     for(int i=0;i<=n-2;i++)
02952.        if(a[i]>a[n-1])
02953.           HoanVi(a[i],a[n-1]);
02954.     SapTang(a,n-1);
02955.  }
```

Bài 154.Hãy sắp xếp các giá trị trong mảng các số nguyên giảm dần.

– Khai báo hàm.

```
02956.  void SapGiam(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02957.  void SapGiam(float a[],int n)
02958.  {
02959.     if(n==1)
02960.        return;
02961.     for(int i=0;i<=n-2;i++)
02962.        if(a[i]<a[n-1])
```

```
02963.          HoanVi(a[i],a[n-1]);
02964.     SapGiam(a,n-1);
02965.  }
```

Bài 155.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.

– Khai báo hàm.

```
02966.  void ViTriLeTang(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02967.  void ViTriLeTang(float a[],int n)
02968.  {
02969.     if(n==1)
02970.        return;
02971.     for(int i=0;i<=n-2;i++)
02972.        if(i%2!=0 && (n-1)%2!=0 && a[i]>a[n-1])
02973.           HoanVi(a[i],a[n-1]);
02974.     ViTriLeTang(a,n-1);
02975.  }
```

Bài 156.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.

– Khai báo hàm.

```
02976.  void NguyenToTang(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02977.  void NguyenToTang(int a[],int n)
02978.  {
02979.     if(n==1)
02980.        return;
02981.     for(int i=0;i<=n-2;i++)
02982.        if(ktNguyenTo(a[i]) &&
02983.           ktNguyenTo(a[n-1]) &&
02984.           a[i]>a[n-1])
02985.           HoanVi(a[i],a[n-1]);
02986.     NguyenToTang(a,n-1);
02987.  }
```

Bài 157.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.

– Khai báo hàm.

```
02988.  void HoanThienGiam(int [],int);
```

– Định nghĩa hàm đệ quy.

```
02989.  void HoanThienGiam(int a[],int n)
02990.  {
02991.     if(n==1)
02992.        return;
02993.     for(int i=0;i<=n-2;i++)
02994.        if(ktHoanThien(a[i]) &&
02995.           ktHoanThien(a[n-1]) &&
02996.           a[i]<a[n-1])
02997.           HoanVi(a[i],a[n-1]);
02998.     HoanThienGiam(a,n-1);
02999.  }
```

Bài 158.Hãy sắp xếp các số dương trong mảng các số thực tăng dần các số âm giữ nguyên vị trí của chúng trong mảng.

– Khai báo hàm.

```
03000.  void DuongTang(float [],int);
```

– Định nghĩa hàm đệ quy.

```
03001.  void DuongTang(float a[],int n)
03002.  {
03003.     if(n==1)
03004.        return;
03005.     for(int i=0;i<=n-2;i++)
03006.        if(a[i]>0 && a[n-1]>0 && a[i]>a[n-1])
03007.           HoanVi(a[i],a[n-1]);
03008.     DuongTang(a,n-1);
03009.  }
```

Bài 159.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.

– Khai báo hàm.

```
03010.  void ChanTangLeTang(int [],int);
```

– Định nghĩa hàm đệ quy.

```
03011.  void ChanTangLeTang(int a[],int n)
03012.  {
03013.     if(n==1)
03014.        return;
03015.     for(int i=0;i<=n-2;i++)
03016.     {
03017.        if(a[i]%2==0&&a[n-1]%2==0&&a[i]>a[n-1])
03018.           HoanVi(a[i],a[n-1]);
```

```
03019.        if(a[i]%2!=0&&a[n-1]%2!=0&&a[i]>a[n-1])
03020.           HoanVi(a[i],a[n-1]);
03021.     }
03022.     ChanTangLeTang(a,n-1);
03023.  }
```

Bài 160.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.

– Khai báo hàm.

```
03024.  void DuongTangAmGiam(float [],int);
```

– Định nghĩa hàm đệ quy.

```
03025.  void DuongTangAmGiam(float a[],int n)
03026.  {
03027.     if(n==1)
03028.        return;
03029.     for(int i=0;i<=n-2;i++)
03030.     {
03031.        if(a[i]>0 && a[n-1]>0 && a[i]>a[n-1])
03032.           HoanVi(a[i],a[n-1]);
03033.        if(a[i]<0 && a[n-1]<0 && a[i]<a[n-1])
03034.           HoanVi(a[i],a[n-1]);
03035.     }
03036.     DuongTangAmGiam(a,n-1);
03037.  }
```

05.08 KỸ THUẬT XÓA ĐỆ QUY

05.08.01 Bài cơ sở Kỹ thuật Đếm Đệ quy

Bài cơ sở 037. Định nghĩa hàm đệ quy xóa phần tử tại vị trí vt trong mảng một chiều các số thực.

– 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.

```
03038.  void XoaViTri(int [],int &,int);
```

– Định nghĩa hàm

```
03039.  void XoaViTri(int a[],int &n,int vt)
03040.  {
03041.     if(vt==(n-1))
03042.     {
03043.        n--;
03044.        return;
03045.     }
03046.     a[vt] = a[vt+1];
03047.     XoaViTri(a,n,vt+1);
03048.  }
```

05.08.02 Kỹ thuật Xóa

Bài 161.Xóa các phần tử có chỉ số *vt* trong mảng một chiều các số thực.

- 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.

```
03049.  void XoaViTri(float [],int &,int);
```

- Định nghĩa hàm

```
03050.  void XoaViTri(float a[],int &n,int vt)
03051.  {
03052.     if(vt==(n-1))
03053.     {
03054.        n--;
03055.        return;
03056.     }
03057.     a[vt] = a[vt+1];
03058.     XoaViTri(a,n,vt+1);
03059.  }
```

Bài 162.(Hạt nhân) Hãy xóa tất cả số âm trong mảng các số thực (273).

- 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.

```
03060.  void XoaAm(float [], int &);
```

– Định nghĩa hàm xóa các giá trị âm trong mảng.

```
03061.  void XoaAm(float a[], int &n)
03062.  {
03063.     if(n==0)
03064.        return;
03065.  }
```

05.09 KỸ THUẬT THÊM ĐỆ QUY

05.09.01 Bài cơ sở Kỹ thuật Đếm Đệ quy

Bài cơ sở 038. Hãy thêm một phần tử có giá trị x vào mảng tại vị trí vt.

– 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.

```
03066.  void ThemViTri(int [],int &,int,int);
```

– Định nghĩa hàm

```
03067.  void ThemViTri(int a[],int &n,int x,int vt)
03068.  {
03069.     if(vt==n)
03070.     {
03071.        a[n] = x;
03072.        n++;
03073.        return;
03074.     }
03075.     HoanVi(a[vt],x);
03076.     ThemViTri(a,n,x,vt+1);
03077.  }
```

05.09.02 Kỹ thuật Thêm

Bài 163.Hãy thêm một phần tử có giá trị x vào mảng tại vị trí vt.

– Khai báo hàm.

```
03078. void ThemViTri(int [],int &,int,int);
```

– Định nghĩa hàm

```
03079. void ThemViTri(int a[],int &n,int x,int vt)
03080. {
03081.    if(vt==n)
03082.    {
03083.       a[n] = x;
03084.       n++;
03085.       return;
03086.    }
03087.    HoanVi(a[vt],x);
03088.    ThemViTri(a,n,x,vt+1);
03089. }
```

Bài 164.Hãy thêm một giá trị x vào trong mảng tăng mà vẫn giữ nguyên tính đơn điệu tăng của mảng.

– Khai báo hàm.

```
03090. void ThemBaoToan(float [],int&,float);
```

– Định nghĩa hàm đệ quy.

```
03091. void ThemBaoToan(float a[],int &n,float x)
03092. {
03093.    if(n==0)
03094.    {
03095.       a[0] = x;
03096.       n++;
03097.       return;
03098.    }
03099.    if(x >= a[n-1])
03100.    {
03101.       a[n] = x;
03102.       n++;
03103.       return;
03104.    }
03105.    a[n] = a[n-1];
03106.    n--;
03107.    ThemBaoToan(a,n,x);
03108.    n++;
03109. }
```

05.10 KỸ THUẬT XỬ LÝ MẢNG CON ĐỆ QUY

05.10.01 Bài cơ sở Kỹ thuật Đếm Đệ quy

> **Bài cơ sở 039.** Định nghĩa hàm đệ quy 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 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

- 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

- Kết quả: 761, 2230, 232, 227.
- Khai báo hàm.

```
03110.  void XuatCon(int [],int,int,int);
```

- Định nghĩa hàm đệ quy xuất mảng con có độ dài l bắt đầu tại vị trí vt.

```
03111.  void XuatCon(int a[],int n,int vt,int l)
03112.  {
03113.     if(l==0)
03114.        return;
03115.     XuatCon(a,n,vt,l-1);
03116.     cout << setw(4) << a[vt+l-1];
03117.  }
```

> **Bài cơ sở 040.** Định nghĩa hàm đệ quy 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,int);
void XuatCon(int [],int,int);
```

- Định nghĩa hàm đệ quy xuất mảng con có độ dài l bắt đầu tại vị trí vt không đệ quy.

```
void XuatCon(int a[],int n,int vt,int l)
{
    if(l==0)
        return;
    XuatCon(a,n,vt,l-1);
    cout << setw(4) << a[vt+l-1];
}
```

- Định nghĩa hàm đệ quy xuất tất cả các mảng con có độ dài l đệ quy.

```
void XuatCon(int a[],int n,int l)
{
    if(n<l)
        return;
    XuatCon(a,n-1,l);
    XuatCon(a,n,n-l,l);
}
```

Bài cơ sở 041. (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

- 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.

```
03134.  void XuatCon(int [],int,int,int);
03135.  void XuatCon(int [],int,int);
03136.  void XuatCon(int [],int);
```

- Định nghĩa hàm đệ quy xuất mảng con có độ dài l bắt đầu tại vị trí vt không đệ quy.

```
03137.  void XuatCon(int a[],int n,int vt,int l)
03138.  {
03139.     if(l==0)
03140.        return;
03141.     XuatCon(a,n,vt,l-1);
03142.     cout << setw(4) << a[vt+l-1];
03143.  }
```

- Định nghĩa hàm đệ quy xuất tất cả các mảng con có độ dài l đệ quy.

```
03144.  void XuatCon(int a[],int n,int l)
03145.  {
03146.     if(n<l)
03147.        return;
03148.     XuatCon(a,n-1,l);
03149.     XuatCon(a,n,n-l,l);
03150.  }
```

- Định nghĩa hàm đệ quy xuất các mảng con trong mảng một chiều.

```
03151.  void XuatCon(int a[],int n)
03152.  {
03153.     if(n==0)
03154.        return;
03155.     XuatCon(a,n-1);
03156.     for(int l=1;l<=n;l++)
```

```
03157.        XuatCon(a,n,n-1,l);
03158.  }
```

05.10.02 Kỹ thuật Xử lý Mảng con

Bài 165.Liệt kê tất cả các mảng con trong mảng một chiều các số nguyên.

- Khai báo hàm.

```
03159.  void LietKe(int [],int);
```

- Định nghĩa hàm đệ quy.

```
03160.
03161.
```

Bài 166.Liệt kê tất cả các mảng con có độ dài lớn hơn 2 trong mảng một chiều các số nguyên.

- Khai báo hàm.

```
03162.  void LietKe(int [],int);
```

- Định nghĩa hàm đệ quy.

```
03163.
03164.
```

Bài 167.Liệt kê các dãy con tăng trong mảng một chiều các số thực.

- Khai báo hàm.

```
03165.  void LietKe(float [],int);
```

- Định nghĩa hàm đệ quy.

```
03166.
03167.
```

Bài 168.Xuất và tính tổng từng mảng con tăng trong mảng một chiều các số thực.

- Khai báo hàm.

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

- Định nghĩa hàm đệ quy.

```
03169.
03170.
```

Bài 169.Liệt kê các dãy con toàn dương có độ dài lớn hơn 1 trong mảng một chiều số thực.

- Khai báo hàm.

```
03171. void LietKe(float [],int);
```

– Định nghĩa hàm đệ quy.

```
03172.
03173.
```

Bài 170.Đếm số lượng mảng con tăng có độ dài lớn hơn 1 trong mảng một chiều các số thực.

– Khai báo hàm.

```
03174. int DemConTang(float [],int);
```

– Định nghĩa hàm đệ quy.

```
03175.
03176.
```

Bài 171.Đếm số lượng mảng con giảm trong mảng một chiều các số thực.

– Khai báo hàm.

```
03177. int DemConGiam(float [],int);
```

Bài 172.Cho hai mảng a và b. Hãy cho biết mảng a có phải là mảng con trong mảng b hay không?

– Khai báo hàm.

```
03178. int ktCon(int [],int,int [],int);
```

– Định nghĩa hàm đệ quy.

```
03179.
03180.
```

Bài 173.Cho hai mảng a và b. Hãy đếm số lần xuất hiện của mảng a trong mảng b.

– Khai báo hàm.

```
03181. int DemCon(int [],int,int [],int);
```

– Định nghĩa hàm đệ quy.

```
03182.
03183.
```

05.11 BÀI TẬP ĐỆ QUY TUYẾN TÍNH TRÊN MẢNG MỘT CHIỀU

05.11.01 Kỹ thuật Xử lý trên mảng

Bài 174.Hãy nguyên hóa mảng bằng cách thay tất cả các phần tử trong mảng bằng số nguyên gần nó nhất.

– Khai báo hàm.

```
03184.  void NguyenHoa(int [],int);
```

– Định nghĩa hàm đệ quy.

```
03185.  void NguyenHoa(float a[], int n)
03186.  {
03187.     if (n == 0)
03188.        return;
03189.     NguyenHoa(a, n - 1);
03190.     if (a[n-1]>0)
03191.        a[n-1] = (float)int(a[n - 1] + 0.5);
03192.     else
03193.        a[n-1] = (float)int(a[n - 1] - 0.5);
03194.  }
```

Bài 175.Hãy đưa số một về đầu mảng.

– Khai báo hàm.

```
03195.  void MotVeDau(int [],int);
```

– Định nghĩa hàm đệ quy.

```
03196.  void MotVeDau(int a[], int n)
03197.  {
03198.     if(n==0)
03199.        return;
03200.     if(a[n-1]!=1)
03201.     {
03202.        MotVeDau(a,n-1);
03203.        return;
03204.     }
03205.     for(int i=0; i<=n-2; i++)
03206.        if(a[i]!=1)
03207.        {
03208.           HoanVi(a[n - 1], a[i]);
03209.           break;
03210.        }
03211.     MotVeDau(a,n-1);
```

```
03212. }
```

Bài 176.Hãy đưa các số chia hết cho 3 về đầu mảng.

– Khai báo hàm.

```
03213. void DuaVeDau(int [],int);
```

– Định nghĩa hàm đệ quy.

```
03214. void DuaVeDau(int a[], int n)
03215. {
03216.    if(n==0)
03217.       return;
03218.    if(a[n-1]%3!=0)
03219.    {
03220.       DuaVeDau(a,n-1);
03221.       return;
03222.    }
03223.    for(int i=0; i<=n-2; i++)
03224.       if(a[i]%3!=0)
03225.       {
03226.          HoanVi(a[n - 1], a[i]);
03227.          break;
03228.       }
03229.    DuaVeDau(a,n-1);
03230. }
```

Bài 177.Hãy đưa các số nguyên tố về cuối mảng.

– Khai báo hàm.

```
03231. void DuaVeCuoi(int [],int);
```

– Định nghĩa hàm đệ quy.

```
03232. void DuaVeCuoi(int a[], int n)
03233. {
03234.    if(n==0)
03235.       return;
03236.    if(ktNguyenTo(a[n-1])==1)
03237.    {
03238.       DuaVeCuoi(a,n-1);
03239.       return;
03240.    }
03241.    for(int i=0; i<=n-2; i++)
03242.       if(ktNguyenTo(a[i]))
03243.       {
03244.          HoanVi(a[n - 1], a[i]);
```

```
03245.          break;
03246.       }
03247.    DuaVeCuoi(a,n-1);
03248. }
```

Bài 178.Hãy "dịch trái xoay vòng" các phần tử trong mảng.

– Khai báo hàm.

```
03249. void DichTrai(int [],int);
```

– Định nghĩa hàm đệ quy.

```
03250. void DichTrai(int a[], int n)
03251. {
03252.    if (n<=1)
03253.       return;
03254.    DichTrai(a, n-1);
03255.    swap(a[n-2], a[n-1]);
03256. }
```

Bài 179.Hãy "dịch phải xoay vòng" các phần tử trong mảng.

– Khai báo hàm.

```
03257. void DichPhai(int [],int);
```

– Định nghĩa hàm đệ quy.

```
03258. void DichPhai(int a[], int n)
03259. {
03260.    if (n<=1)
03261.       return;
03262.    swap(a[n-2], a[n-1]);
03263.    DichPhai(a, n-1);
03264. }
```

05.11.02 Thử thách nhỏ

Bài 180. (*) 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

– Kết quả:

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

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

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

+ $(29,19)$.
+ $(19,29)$.

– Khai báo hàm.

```
03265. void XuatBo2(float, float);
03266. void LietKe(float [],int);
```

– Định nghĩa hàm xuất cặp.

```
03267. void XuatBo2(float x, float y)
03268. {
03269.    cout<<setw(10)<<setprecision(3);
03270.    cout<<"("<<x<<","<<y<<")"<<endl;
03271. }
```

– Định nghĩa hàm đệ quy.

```
03272. void LietKe(float a[],int n)
03273. {
03274.    if(n==1)
03275.       return;
03276.    for(int i=0;i<=n-2;i++)
03277.    {
03278.       XuatBo2(a[i],a[n-1]);
03279.       XuatBo2(a[n-1],a[i]);
03280.    }
03281.    LietKe(a,n-1);
03282. }
```

Bài 181. (*) 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 $x \leq y$.

– Khai báo hàm.

```
03283. void XuatBo2(float, float);
03284. void LietKe(float [],int);
```

– Định nghĩa hàm xuất cặp.

```
03285. void XuatBo2(float x, float y)
03286. {
03287.    cout<<setw(10)<<setprecision(3);
03288.    cout<<"("<<x<<","<<y<<")"<<endl;
```

```
03289. }
```

– Định nghĩa hàm đệ quy.

```
03290. void LietKe(float a[],int n)
03291. {
03292.    if(n==1)
03293.       return;
03294.    for(int i=0;i<=n-2;i++)
03295.    {
03296.       if(a[i]<=a[n-1])
03297.          XuatBo2(a[i],a[n-1]);
03298.       if(a[n-1]<=a[i])
03299.          XuatBo2(a[n-1],a[i]);
03300.    }
03301.    LietKe(a,n-1);
03302. }
```

Bài 182.(*) 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).

– Khai báo hàm.

```
03303. void XuatBo3(float, float, float);
03304. void LietKe(float [],int);
```

– Định nghĩa hàm xuất cặp.

```
03305. void XuatBo3(float x, float y,float z)
03306. {
03307.    cout<<setw(10)<<setprecision(3);
03308.    cout<<"("<<x<<","<<y<<","<<z<<")"<<endl;
03309. }
```

– Định nghĩa hàm đệ quy.

```
03310. void LietKe(float a[],int n)
03311. {
03312.    if(n<=2)
03313.       return;
03314.    for(int i=0;i<=n-3;i++)
03315.       for(int j=i+1;j<=n-2;j++)
03316.       {
03317.          if(a[i] == (a[j] + a[n-1]))
03318.          {
03319.             XuatBo3(a[i],a[j],a[n-1]);
03320.             XuatBo3(a[i],a[n-1],a[j]);
```

```
            }
            if(a[j] == (a[i] + a[n-1]))
            {
               XuatBo3(a[j],a[i],a[n-1]);
               XuatBo3(a[j],a[n-1],a[i]);
            }
            if(a[n-1] == (a[i] + a[j]))
            {
               XuatBo3(a[n-1],a[i],a[j]);
               XuatBo3(a[n-1],a[j],a[i]);
            }
         }
      LietKe(a,n-1);
   }
```

Bài 183.(*) Hãy xác định số lượng các phần tử kề nhau mà cả hai đều chẵn.

Bài 184.(*) 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

– Cách đếm:

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

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.

```
00001.  int  DemPhanBiet(int [],int);
```

– Định nghĩa hàm đệ quy.

```
00002.  int DemPhanBiet(int a[], int n)
00003.  {
00004.     if(n==0)
00005.        return 0;
00006.     if(n==1)
00007.        return 1;
00008.     int dem = DemPhanBiet(a,n-1);
00009.
00010.     int flag = 1;
00011.     for (int i = 0; i <= n-2; i++)
00012.        if(a[i]==a[n-1])
00013.           flag  = 0;
00014.     if(flag==1)
00015.        dem++;
00016.     return dem;
00017.  }
```

Bài 185. (*) Hãy đếm số lượng các giá trị phân biệt có trong mảng một chiều các số thực.

– Khai báo hàm.

```
00018.  int  DemPhanBiet(float [],int);
```

– Định nghĩa hàm đệ quy.

```
00019.  int DemPhanBiet(float a[], int n)
00020.  {
00021.     if(n==0)
00022.        return 0;
00023.     if(n==1)
00024.        return 1;
00025.     int dem = DemPhanBiet(a,n-1);
00026.
00027.     int flag = 1;
00028.     for (int i = 0; i <= n-2; i++)
```

```
00029.        if(a[i]==a[n-1])
00030.           flag  = 0;
00031.     if(flag==1)
00032.        dem++;
00033.     return dem;
00034.  }
```

Bài 186. (*) Hãy đếm số lượng số nguyên tố phân biệt trong mảng các số nguyên.

– Khai báo hàm.

```
00035.  int  DemNguyenToPhanBiet(int [],int);
```

– Định nghĩa hàm đệ quy.

```
00036.  int DemNguyenToPhanBiet(int a[], int n)
00037.  {
00038.     if(n==0)
00039.        return 0;
00040.     int dem = DemNguyenToPhanBiet(a,n-1);
00041.
00042.     int flag = 1;
00043.     for (int i = 0; i <= n-2; i++)
00044.        if(a[i]==a[n-1])
00045.           flag  = 0;
00046.     if(flag==1 && ktNguyenTo(a[n-1]) )
00047.        dem++;
00048.     return dem;
00049.  }
```

Bài 187. (*) 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	19	99	29	12	19	88	11

– Cách đếm:

19	29	62	19	99	29	12	19	88	**11**

19	29	62	19	99	29	12	19	**88**	11

19	29	62	19	99	29	12	**19**	88	11

19	29	62	19	99	29	**12**	19	88	11

19	29	62	19	99	**29**	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

19	29	62	19	99	29	12	19	88	11

– Kết quả: số lượng giá trị trong mảng thỏa tính chất: “lớn hơn tất cả các giá trị đứng đằng trước nó” là 3.

19	29	62	19	99	29	12	19	88	11

– Khai báo hàm.

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

– Định nghĩa hàm đệ quy.

```
int DemGiaTri(int [],int)
{
    if(n==0)
        return 0;
    if(n==1)
        return 0;
    int dem = DemGiaTri(a,n-1);
    int flag = 1;
    for (int i = 0; i <= n-2; i++)
        if(a[i]>=a[n-1])
            flag  = 0;
    if(flag==1)
        dem++;
    return dem;
}
```

Bài 188.(*) Hãy cho biết tất cả các phần tử trong mảng *a* có nằm trong mảng *b* hay không?

– Khai báo hàm.

```
int ktThuoc(int [],int,int [],int);
```

– Định nghĩa hàm đệ quy.

```
int ktThuoc(int a[],int n,int b[],int k)
{
    if(n==1)
    {
        if(TanSuat(b,k,a[0])==0)
            return 0;
        return 1;
```

```
00074.     }
00075.     if(TanSuat(b,k,a[n-1])>0 &&
00076.        ktThuoc(a,n-1,b,k)==1)
00077.        return 1;
00078.     return 0;
00079.  }
```

Bài 189.(*) 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 (181).

– 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.

```
00080.  void LietKe(int [],int);
```

– Định nghĩa hàm đệ quy.

```
00081.
```

Bài 190.(*) Cho hai mảng các số nguyên *a* và *b*. Hãy cho biết số lần xuất hiện của mảng *a* trong mảng *b*.

– Khai báo hàm.

```
00082.  int  DemXuatHien(int [],int,int [],int);
```

– Định nghĩa hàm đệ quy.

```
00083.  A
```

Bài 191.(*) 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.

– Khai báo hàm.

```
00084.  int GanNhat(float [],int,float&,float&);
```

– Định nghĩa hàm đệ quy.

```
00085.
00086.
```

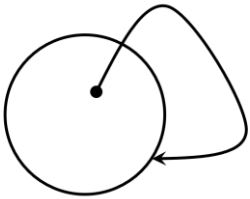
CHƯƠNG 06. ĐỆ QUY TUYẾN TÍNH TRÊN MA TRẬN

06.01 KHÁI NIỆM ĐỆ QUY TUYẾN TÍNH

- Khái niệm: Một hàm được gọi là đệ quy tuyến tính nếu trong thân của hàm đó có duy nhất một lời gọi hàm gọi lại chính nó một cách tường minh.

06.02 HÌNH ẢNH

- Hình vẽ minh họa



- Trong hình vẽ minh họa trên ta có thể hiểu như sau:
  + Hàm là vòng tròn.
  + Lời gọi hàm vòng cung có mũi tên.
  + Lời gọi hàm bắt đầu tại một điểm trong vòng tròn và kết thúc với mũi tên tại biên vòng tròn.

06.03 CẤU TRÚC HÀM ĐỆ QUY TUYẾN TÍNH

```
00087.  KDL TenHam(<ThamSo>)
00088.  {
00089.     if <điều kiện dừng>
00090.     {
00091.        ...
00092.        return <Giá Trị Trả Về>;
00093.     }
00094.     ...
00095.     ...TenHam(<ThamSo>;
00096.     ...
00097.  }
```

06.01 KỸ THUẬT LIỆT KÊ ĐỆ QUY

06.01.01 Không gian liệt kê là toàn bộ ma trận

Bài cơ sở 042. Định nghĩa hàm liệt kê các số nguyên tố trong ma trận các số nguyên.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12	31	-42	-73	18
1	-87	121	25	46	7
2	79	-82	-11	-27	-96

+ Dữ liệu ra: 31, 07, 79.

– Khai báo hàm.

```
00098. void LietKe(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00099. void LietKe(int a[][100],int m,int n)
00100. {
00101.    if(m==0)
00102.       return;
00103.    LietKe(a,m-1,n);
00104.    for(int j=0;j<n;j++)
00105.       if(ktNguyenTo(a[m-1][j]))
00106.          cout << setw(4) << a[m-1][j];
00107. }
```

06.01.02 Không gian liệt kê là một dòng trong ma trận

Bài cơ sở 043. Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các giá trị lẻ trên một dòng.

– Ví dụ:

+ Dữ liệu vào:

0	12	38	-42	-73	18	-83	-87
1	-87	121	19	46	17	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	93	13	47	35	42

+ Dữ liệu ra: $-87, 121, 19, 17, 37$.

– Hình vẽ minh họa:

0	1	…	j	…	$n-2$	$n-1$

d	-87	121	+19	+46	+17	+24	+37

– Các phần tử trên dòng d của ma trận a là: $a[d][0]$, $a[d][1]$, $\cdots, a[d][j], \cdots, a[d][n-2]$, $a[d][n-1]$.

– Khai báo hàm.

```
00108.  void LietKe(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00109.  void LietKe(int a[][100],int m,int n,int d)
00110.  {
00111.     if(n==0)
00112.        return;
00113.     LietKe(a,m,n-1,d);
00114.     if(a[d][n-1]%2!=0)
00115.        cout << setw(4) << a[d][n-1];
00116.  }
```

06.01.03 Không gian liệt kê là một cột trong ma trận

Bài cơ sở 044. Cho ma trận các số nguyên. Hãy định nghĩa hàm liệt kê các số hoàn thiện trên một cột.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	6	46	7	24	37
2	79	-82	28	-27	-96	42	-38
3	12	29	32	13	47	35	42

+ Dữ liệu ra: 6, 28.

– Hình vẽ minh họa.

			c			
0			-73			
1			46			
…			-27			
i			13			
…			88			
$m-2$			12			
$m-1$			1			

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

```
00117. void LietKe(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00118. void LietKe(int a[][100],int m,int n,int c)
00119. {
00120.    if(m==0)
00121.       return;
00122.    LietKe(a,m-1,n,c);
00123.    if(ktHoanThien(a[m-1][c]))
00124.       cout << setw(4) << a[m-1][c];
00125. }
```

06.02 KỸ THUẬT TÍNH TOÁN ĐỆ QUY

06.02.01 Không gian tính toán là toàn bộ ma trận

Bài cơ sở 045. Viết hàm tính tổng các giá trị âm trong ma trận các số thực.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	0	67.31	0	7.45
2	79.21	-82.56	-11.12	-27.45	66.89

+ Dữ liệu ra:

+ Thông báo (nếu cần thiết): Tổng các giá trị âm là:

– Khai báo hàm.

```
00126. float TongAm(float [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00127. float TongAm(float a[][100],int m,int n)
00128. {
00129.    if(m==0)
00130.       return 0;
00131.    float s = TongAm(a,m-1,n);
00132.    for(int j=0;j<n;j++)
00133.       if(a[m-1][j]<0)
00134.          s = s + a[m-1][j];
00135.    return s;
00136. }
```

– Định nghĩa hàm đệ quy.

```
00137.  float TongAm(float a[][100],int m,int n)
00138.  {
00139.     if(n==0)
00140.        return 0;
00141.     float s = TongAm(a,m,n-1);
00142.     for(int i=0;i<m;i++)
00143.        if(a[i][n-1]<0)
00144.           s = s + a[i][n-1];
00145.     return s;
00146.  }
```

06.03 KỸ THUẬT ĐẾM ĐỆ QUY

06.03.01 Không gian đếm là toàn bộ ma trận

Bài cơ sở 046. Viết hàm đếm số lượng giá trị toàn chữ số chẵn trong ma trận các số nguyên?

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12	22	-421	-73	82
1	248	834	0	461	7
2	222	-57	-11	848	-961

+ Dữ liệu ra: 6.

+ Thông báo (nếu cần thiết): Số lượng giá trị toàn chữ số chẵn là 6.

– Khai báo hàm.

```
00147.  int DemToanChan(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00148.  int DemToanChan(int a[][100], int m,int n)
00149.  {
00150.     if(m==0)
00151.        return 0;
00152.     int dem = DemToanChan(a,m-1,n);
00153.     for(int j=0;j<n;j++)
00154.        if(ktToanChan(a[m-1][j]))
00155.           dem = dem + 1;
00156.     return dem;
00157.  }
```

– Định nghĩa hàm đệ quy.

```
00158.  int DemToanChan(int a[][100], int m,int n)
00159.  {
00160.     if(n==0)
00161.        return 0;
00162.     int dem = DemToanChan(a,m,n-1);
00163.     for(int i=0;i<m;i++)
00164.        if(ktToanChan(a[i][n-1]))
00165.           dem = dem + 1;
00166.     return dem;
00167.  }
```

06.03.02 Không gian đếm là một dòng trong ma trận

Bài cơ sở 047. Cho ma trận các số nguyên. Hãy định nghĩa hàm đếm số lượng giá trị có dạng 2^m trên một dòng.

– Ví dụ:

+ Dữ liệu vào:

0	12	38	-42	-73	18	-83	-87
1	-87	121	19	1024	17	64	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	93	13	47	35	42

+ Dữ liệu ra: $1024, 64$.

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Các phần tử trên dòng d của ma trận a là: $a[d][0], a[d][1], \cdots, a[d][j], \cdots, a[d][n-2], a[d][n-1]$.

– Khai báo hàm.

```
00168.  int DemDang2m(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00169.  int DemDang2m(int a[][100],int m,int n,int d)
00170.  {
00171.     if(n==0)
00172.        return 0;
00173.     int dem = DemDang2m(a,m,n-1,d);
00174.     if(ktDang2m(a[d][n-1]))
```

```
00175.           dem = dem + 1;
00176.     return dem;
00177.  }
```

06.03.03 Không gian đếm là một cột trong ma trận

Bài cơ sở 048. Cho ma trận các số nguyên. Hãy định nghĩa hàm đếm số lượng giá trị đối xứng trên một cột.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-424	-73	18	-83	-87
1	-87	121	6	46	7	24	37
2	79	-82	28	-27	-96	42	-38
3	12	29	3223	13	47	35	42

+ Dữ liệu ra: 4.

– Hình vẽ minh họa.

	c
0	-73
1	46
…	-27
i	13
…	88
$m-2$	12
$m-1$	1

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

```
00178.  int DemDoiXung(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00179.  int DemDoiXung(int a[][100],int m,int n,
00180.                 int c)
00181.  {
00182.     if(m==0)
00183.        return 0;
00184.     int dem = DemDoiXung(a,m-1,n,c);
00185.     if(ktDoiXung(a[m-1][c]))
00186.        dem = dem + 1;
00187.     return dem;
00188.  }
```

06.04 KỸ THUẬT TÌM KIẾM ĐỆ QUY

06.04.01 Không gian tìm kiếm là toàn bộ ma trận

> Bài cơ sở 049. Viết hàm tìm giá trị lớn nhất trong ma trận các số thực.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	0	67.31	0	7.45
2	79.21	-82.56	-11.12	-27.45	66.89

+ Dữ liệu ra: 79.21.

+ Thông báo (nếu cần thiết): Giá trị lớn nhất trong ma trận 79.21.

– Khai báo hàm.

```
00189. float LonNhat(float [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00190. float LonNhat(float a[][100], int m,int n)
00191. {
00192.    if(m==1)
00193.    {
00194.       float lc = a[0][0];
00195.       for(int j=0;j<n;j++)
00196.          if(a[0][j]>lc)
00197.             lc = a[0][j];
00198.       return lc;
00199.    }
00200.    float lc = LonNhat(a,m-1,n);
00201.    for(int j=0;j<n;j++)
00202.       if(a[m-1][j]>lc)
00203.             lc = a[m-1][j];
00204.    return lc;
00205. }
```

– Định nghĩa hàm đệ quy.

```
00206. float LonNhat(float a[][100], int m,int n)
00207. {
00208.    if(n==1)
00209.    {
00210.       float lc = a[0][0];
00211.       for(int i=0;i<m;i++)
```

```
00212.          if(a[i][0]>lc)
00213.             lc = a[i][0];
00214.       return lc;
00215.    }
00216.    float lc = LonNhat(a,m,n-1);
00217.    for(int i=0;i<m;i++)
00218.       if(a[i][n-1]>lc)
00219.             lc = a[i][n-1];
00220.    return lc;
00221. }
```

06.04.02 Không gian tìm kiếm là một dòng trong ma trận

Bài cơ sở 050. Tìm giá trị lớn nhất trên một dòng trong ma trận các số thực.

– Ví dụ:

+ Dữ liệu vào:

0	12	38	-42	-73	18	-83	-87
1	-87	121	19	46	17	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	93	13	47	35	42

+ Dữ liệu ra: $+121$.

+ Lớn nhất trên dòng 1 là: $+121$.

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Các phần tử trên dòng d của ma trận a là: $a[d][0], a[d][1], \cdots, a[d][j], \cdots, a[d][n-2], a[d][n-1]$.

– Khai báo hàm.

```
00222. int LonNhatDong(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00223. float LonNhatDong(float a[][100],int m,int n,
00224.                   int d)
00225. {
00226.    if(n==1)
00227.       return a[d][0];
00228.    float lc = LonNhatDong(a,m,n-1,d);
```

```
00229.     if(a[d][n-1]>lc)
00230.        lc = a[d][n-1];
00231.     return lc;
00232.  }
```

06.04.03 Không gian tìm kiếm là một cột trong ma trận

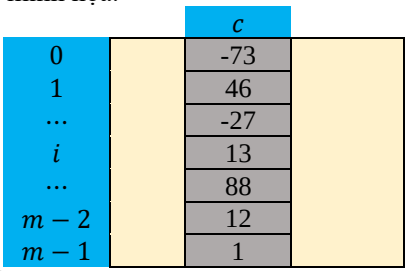
Bài cơ sở 051. Tìm giá trị nhỏ nhất trên một cột trong ma trận các số thực (373).

– Ví dụ:

	0	1	2	3	4	5	6
0	12	38	42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	32	13	47	35	42

+ Giá trị nhỏ nhất trên cột 2 là: -11

– Hình vẽ minh họa.



– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c], a[m-1][c]$.

– Khai báo hàm.

```
00233.  int NhoNhatCot(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00234.  float NhoNhatCot(float a[][100],int m,int n,
00235.                   int c)
00236.  {
00237.     if(m==1)
00238.        return a[0][c];
00239.     float lc = NhoNhatCot(a,m-1,n,c);
00240.     if(a[m-1][c]<lc)
00241.        lc = a[m-1][c];
00242.     return lc;
00243.  }
```

06.04.04 Tìm kiếm phần tử thỏa điều kiện

Bài cơ sở 052. Tìm số chẵn đầu tiên trong ma trận số nguyên. Nếu ma trận không có giá trị chẵn thì hàm sẽ trả về giá trị không chẵn là -1.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3
0	121	8271	-42	-731
1	-87	1218	5	46
2	7	-82	212	-7

  - + Dữ liệu ra: -42.
- Khai báo hàm.

```
00244. int ChanDau(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00245. int ChanDau(int a[][100],int m,int n)
00246. {
00247.    if(m==0)
00248.       return -1;
00249.    int lc = ChanDau(a,m-1,n);
00250.    if(lc!=-1)
00251.       return lc;
00252.    for(int j=0;j<n;j++)
00253.       if(a[m-1][j]%2==0)
00254.          return a[m-1][j];
00255.    return -1;
00256. }
```

Bài cơ sở 053. Định nghĩa hàm tìm số lẻ lớn nhất trong ma trận các số nguyên. Trong trường hợp hàm ko có giá trị lẻ thì hàm trả về giá trị ko lẻ là 0.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3
0	121	8271	-42	-731
1	-87	9218	5	46
2	7	-82	212	-7

  - + Dữ liệu ra: 8271.
- Khai báo hàm.

```
00257. int LeLonNhat(int [][100],int,int);
```

– Định nghĩa hàm lẻ lớn nhất đệ quy.

```
00258.  int LeLonNhat(int a[][100], int m,int n)
00259.  {
00260.     if(m==0)
00261.        return 0;
00262.     int lc = LeLonNhat(a,m-1,n);
00263.     for(int j=0;j<n;j++)
00264.        if(a[m-1][j]%2!=0)
00265.           if(lc==0 || a[m-1][j]>lc)
00266.              lc = a[m-1][j];
00267.     return lc;
00268.  }
```

06.05 KỸ THUẬT ĐẶT CỜ HIỆU ĐỆ QUY

06.05.01 Kiểm tra trên toàn bộ ma trận

Bài cơ sở 054. Viết hàm kiểm tra trong ma trận các số nguyên có tồn tại giá trị chẵn nhỏ hơn 2004 hay không?

– Ví dụ 01:

+ Dữ liệu vào:

	0	1	2	3	4
0	12	121	-42	-73	81
1	25	121	0	46	7
2	49	-82	-11	1	-96

+ Dữ liệu ra: +1.

+ Thông báo (nếu cần thiết): ma trận tồn tại giá trị chẵn.

– Ví dụ 02:

+ Dữ liệu vào:

	0	1	2	3	4
0	97	121	-83	-73	8168
1	25	121	9124	57	7
2	49	-821	-11	1	-95

+ Dữ liệu ra: 0.

+ Thông báo (nếu cần thiết): ma trận ko tồn tại giá trị chẵn.

– Khai báo hàm.

```
00269.  int TonTaiChan(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00270.  int TonTaiChan(int a[][100], int m,int n)
00271.  {
00272.     if(m==0)
```

```
00273.        return 0;
00274.     for(int j=0;j<n;j++)
00275.        if(a[m-1][j]%2==0 && a[m-1][j]<2004)
00276.              return 1;
00277.     return TonTaiChan(a,m-1,n);
00278.  }
```

– Định nghĩa hàm đệ quy.

```
00279.  int TonTaiChan(int a[][100], int m,int n)
00280.  {
00281.     if(n==0)
00282.        return 0;
00283.     for(int i=0;i<m;i++)
00284.        if(a[i][n-1]%2==0 && a[i][n-1]<2004)
00285.              return 1;
00286.     return TonTaiChan(a,m,n-1);
00287.  }
```

06.05.02 Kiểm tra trên dòng

Bài cơ sở 055. Viết hàm kiểm tra một dòng trong ma trận các số nguyên có tồn tại giá trị chẵn không?

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83

+ Dữ liệu ra: $+1$.

+ Thông báo (nếu cần thiết): Dòng 1 có tồn tại giá trị chẵn.

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Các phần tử trên dòng d của ma trận a là: $a[d][0], a[d][1], \cdots, a[d][j], \cdots, a[d][n-2], a[d][n-1]$.

– Khai báo hàm.

```
00288.  int ktTonTai(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00289.  int ktTonTai(int a[][100],int m,int n,int d)
00290.  {
00291.     if(n==0)
00292.        return 0;
00293.     if(a[d][n-1]%2==0)
00294.        return 1;
00295.     return ktTonTai(a,m,n-1,d);
00296.  }
```

06.05.03 Kiểm tra trên cột

Bài cơ sở 056. Cho ma trận các số nguyên. Hãy định nghĩa hàm kiểm tra một cột có toàn giá trị chẵn không.

– Ví dụ:

+ Dữ liệu vào

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	254	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	32	13	47	35	42

+ Dữ liệu ra: 0.

– Hình vẽ minh họa.

	c
0	-73
1	46
…	-27
i	13
…	88
$m-2$	12
$m-1$	1

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

```
00297.  int ktToanChan(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00298.  int ktToanChan(int a[][100],int m,int n,
00299.                 int c)
00300.  {
00301.     if(m==0)
00302.        return 0;
```

```
00303.     if(m==1)
00304.     {
00305.        if(a[0][c]%2==0)
00306.           return 1;
00307.        return 0;
00308.     }
00309.     if(a[m-1][c]%2!=0)
00310.        return 0;
00311.     return ktToanChan(a,m-1,n,c);
00312.  }
```

– Định nghĩa hàm đệ quy.

```
00313.  int ktToanChan(int a[][100],int m,int n,
00314.                   int c)
00315.  {
00316.     if(m==0)
00317.        return 0;
00318.     if(m==1)
00319.     {
00320.        if(a[0][c]%2==0)
00321.           return 1;
00322.        return 0;
00323.     }
00324.     if(a[m-1][c]%2==0&&ktToanChan(a,m-1,n,c))
00325.        return 1;
00326.     return 0;
00327.  }
```

06.06 KỸ THUẬT XÓA ĐỆ QUY

06.06.01 Kỹ thuật xóa dòng trong ma trận

Bài cơ sở 057. Định nghĩa hàm xóa một dòng trong ma trận?

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	98.23	-82.56	-11.12	-27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4	18.89	56.54	92.3	11.04	67.89

+ Dữ liệu ra:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	79.21	52.54	83.56	43.21	23.67
3	18.89	56.54	92.3	11.04	67.89

– Hướng dẫn.

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	98.23	-82.56	-11.12	-27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4	18.89	56.54	92.3	11.04	67.89

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	79.21	-82.56	-11.12	-27.45	66.89
3	18.89	52.54	83.56	43.21	23.67
4		56.54	92.3	11.04	67.89

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	79.21	52.54	-11.12	-27.45	66.89
3	18.89	56.54	83.56	43.21	23.67
4			92.3	11.04	67.89

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	79.21	52.54	83.56	-27.45	66.89
3	18.89	56.54	92.3	43.21	23.67
4				11.04	67.89

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	79.21	52.54	83.56	43.21	66.89
3	18.89	56.54	92.3	11.04	23.67
4					67.89

0	1	2	3	4

0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	79.21	52.54	83.56	43.21	23.67
3	18.89	56.54	92.3	11.04	67.89
4					

– Khai báo hàm.

```
00328. void XoaDong(int [][100],int &,int,int);
```

– Định nghĩa hàm đệ quy.

```
00329. void XoaDong(int a[][100],int &m,int n,int d)
00330. {
00331.
00332. }
```

06.06.02 Kỹ thuật xóa cột trong ma trận

Bài cơ sở 058. Định nghĩa hàm xóa một cột trong ma trận?

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	98.23	-82.56	-11.12	-27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4	18.89	56.54	92.3	11.04	67.89

+ Dữ liệu ra:

	0	1	2	3
0	12.28	38.41	-73.21	18.18
1	-5.82	11.48	46.19	7.45
2	98.23	-82.56	-27.45	66.89
3	79.21	52.54	43.21	23.67
4	18.89	56.54	11.04	67.89

– Hướng dẫn làm.

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	98.23	-82.56	-11.12	-27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4	18.89	56.54	92.3	11.04	67.89

0	1	2	3	4

0	12.28	38.41	-73.21	18.18	
1	-5.82	11.48	67.31	46.19	7.45
2	98.23	-82.56	-11.12	-27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4	18.89	56.54	92.3	11.04	67.89

	0	1	2	3	4
0	12.28	38.41	-73.21	18.18	
1	-5.82	11.48	46.19	7.45	
2	98.23	-82.56	-11.12	-27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4	18.89	56.54	92.3	11.04	67.89

	0	1	2	3	4
0	12.28	38.41	-73.21	18.18	
1	-5.82	11.48	46.19	7.45	
2	98.23	-82.56	-27.45	66.89	
3	79.21	52.54	83.56	43.21	23.67
4	18.89	56.54	92.3	11.04	67.89

	0	1	2	3	4
0	12.28	38.41	-73.21	18.18	
1	-5.82	11.48	46.19	7.45	
2	98.23	-82.56	-27.45	66.89	
3	79.21	52.54	43.21	23.67	
4	18.89	56.54	92.3	11.04	67.89

	0	1	2	3	4
0	12.28	38.41	-73.21	18.18	
1	-5.82	11.48	46.19	7.45	
2	98.23	-82.56	-27.45	66.89	
3	79.21	52.54	43.21	23.67	
4	18.89	56.54	11.04	67.89	

– Khai báo hàm.

```
00333.  void XoaCot(float [][100],int,int &,int);
```

– Định nghĩa hàm đệ quy.

```
00334.  void XoaCot(float a[][100],int m,int &n,
00335.                int c)
00336.  {
00337.  }
```

06.07 KỸ THUẬT THÊM ĐỆ QUY

06.07.01 Kỹ thuật thêm dòng

Bài cơ sở 059. Định nghĩa hàm thêm một dòng toàn giá trị -1 tại vị trí dòng d trong ma trận các số thực?

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

  - + Dữ liệu ra:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	−1	−1	−1	−1	−1
3	79.21	−82.56	−11.12	−27.45	66.89
4	79.21	52.54	83.56	43.21	23.67

- Hướng dẫn.

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4					

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	−1	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4	79.21				

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	−1	−1	−11.12	−27.45	66.89
3	79.21	−82.56	83.56	43.21	23.67
4	79.21	52.54			

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	−1	−1	−1	−27.45	66.89
3	79.21	−82.56	−11.12	43.21	23.67
4	79.21	52.54	83.56		

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	−1	−1	−1	−1	66.89
3	79.21	−82.56	−11.12	−27.45	23.67
4	79.21	52.54	83.56	43.21	

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	−1	−1	−1	−1	−1
3	79.21	−82.56	−11.12	−27.45	66.89
4	79.21	52.54	83.56	43.21	23.67

– Khai báo hàm.

```
00338.  void ThemDong(int [][100],int &,int,int);
```

– Định nghĩa hàm thêm dòng toàn giá trị −1 đệ quy.

```
00339.  void ThemDong(int a[][100],int &m,int n,
00340.                int d)
00341.  {
00342.  }
```

06.07.02 Kỹ thuật thêm cột

Bài cơ sở 060. Định nghĩa hàm thêm một cột toàn giá trị 0 vào trong ma trận các số thực tại vị trí cột *c*?

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

+ Dữ liệu ra:

	0	1	2	3	4	5
0	12.28	38.41	0	−42.43	−73.21	18.18
1	−5.82	11.48	0	67.31	46.19	7.45
2	79.21	−82.56	0	−11.12	−27.45	66.89
3	79.21	52.54	0	83.56	43.21	23.67

+ Hướng dẫn.

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

	0	1	2	3	4	5
0	12.28	38.41	0	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45	
2	79.21	−82.56	−11.12	−27.45	66.89	
3	79.21	52.54	83.56	43.21	23.67	

	0	1	2	3	4	5
0	12.28	38.41	0	−42.43	−73.21	18.18
1	−5.82	11.48	0	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89	
3	79.21	52.54	83.56	43.21	23.67	

	0	1	2	3	4	5
0	12.28	38.41	0	−42.43	−73.21	18.18
1	−5.82	11.48	0	67.31	46.19	7.45
2	79.21	−82.56	0	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67	

	0	1	2	3	4	5
0	12.28	38.41	0	−42.43	−73.21	18.18
1	−5.82	11.48	0	67.31	46.19	7.45
2	79.21	−82.56	0	−11.12	−27.45	66.89
3	79.21	52.54	0	83.56	43.21	23.67

– Khai báo hàm.

```
00343.  void ThemCot(int [][100],int,int&,int c);
```

– Định nghĩa hàm đệ quy.

```
00344.  void ThemCot(int a[][100],int m,int &n,int c)
00345.  {
00346.  }
```

06.08 KỸ THUẬT XÂY DỰNG MA TRẬN ĐỆ QUY

06.08.01 Xây dựng ma trận đệ quy

> Bài cơ sở 061. Viết hàm xây dựng ma trận các số nguyên B từ ma trận A các số thực với nguyên tắc như sau: Kích thước ma trận B bằng ma trận A, phần tử `B[i][j]=1` nếu `A[i][j]>0`, `B[i][j]=-1` nếu `A[i][j]<0` và `B[i][j]=0` nếu `A[i][j]=0`.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	0	67.31	0	7.45
2	79.21	-82.56	-11.12	-27.45	66.89

+ Dữ liệu ra:

	0	1	2	3	4
0	+1	+1	−1	−1	+1
1	−1	+0	+1	+0	+1
2	+1	−1	−1	−1	+1

– Khai báo hàm.

```
00347. void XayDung(  float [][100],int,int,
00348.               int [][100],int &,int &);
```

– Định nghĩa hàm đệ quy.

```
00349. void XayDung(  float a[][100],int m,int n,
00350.               int b[][100],int &k,int &l)
00351. {
00352.    if(m==0)
00353.    {
00354.       k = 0;
00355.       l = n;
00356.       return;
00357.    }
00358.    XayDung(a,m-1,n,b,k,l);
00359.    l = n;
00360.    for(int j=0;j<l;j++)
00361.          if(a[m-1][j]>0)
00362.             b[m-1][j] = 1;
00363.          else if(a[m-1][j]<0)
00364.             b[m-1][j] = -1;
00365.          else
```

```
00366.              b[m-1][j] = 0;
00367.     k++;
00368.  }
```

06.09 BÀI TẬP MA TRẬN ĐỆ QUY

06.09.01 Kỹ thuật liệt kê đệ quy

Bài 192.Cho ma trận các số nguyên. Hãy định nghĩa hàm liệt kê các giá trị chẵn.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38

  - + Dữ liệu ra: 12, 38, -42, 18, 46, 24, -82, -96, 42, -38.
- Khai báo hàm.

```
00369.  void LietKe(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00370.  void LietKe(int a[][100],int m,int n)
00371.  {
00372.     if(m==0)
00373.        return;
00374.     LietKe(a,m-1,n);
00375.     for(int j=0;j<n;j++)
00376.        if(a[m-1][j]%2==0)
00377.           cout << setw(4) << a[m-1][j];
00378.  }
```

Bài 193.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các giá trị lẻ trên các dòng chỉ số chẵn.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38

  - + Dữ liệu ra: -73, -83, -87, 79, -11, -27.
- Khai báo hàm.

```
00379.  void LietKe(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00380. void LietKe(int a[][100],int m,int n)
00381. {
00382.    if(m==0)
00383.       return;
00384.    LietKe(a,m-1,n);
00385.    for(int j=0;j<n;j++)
00386.       if(a[m-1][j]%2!=0)
00387.          cout << setw(4) << a[m-1][j];
00388. }
```

Bài 194.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các số chính phương.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	0	-73
1	-87	121	25	46
2	79	-82	-11	49

+ Dữ liệu ra: $0, 121, 25, 49$.

– Khai báo hàm.

```
00389. void LietKe(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00390. void LietKe(int a[][100],int m,int n)
00391. {
00392.    if(m==0)
00393.       return;
00394.    LietKe(a,m-1,n);
00395.    for(int j=0;j<n;j++)
00396.       if(ktChinhPhuong(a[m-1][j]))
00397.          cout << setw(4) << a[m-1][j];
00398. }
```

Bài 195.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các số nguyên tố.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	11	38	-42	73
1	-87	121	29	46
2	79	-82	-11	53

+ Dữ liệu ra: $11, 73, 29, 53$.

– Khai báo hàm.

```
00399. void LietKe(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00400. void LietKe(int a[][100],int m,int n)
00401. {
00402.    if(m==0)
00403.       return;
00404.    LietKe(a,m-1,n);
00405.    for(int j=0;j<n;j++)
00406.       if(ktNguyenTo(a[m-1][j]))
00407.          cout << setw(4) << a[m-1][j];
00408. }
```

Bài 196.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các số nguyên toàn chữ số chẵn.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	-42	-73
1	-87	121	25	46
2	79	-82	-11	-27

+ Dữ liệu ra: -42, 46, -82.

– Khai báo hàm.

```
00409. void LietKe(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00410. void LietKe(int a[][100],int m,int n)
00411. {
00412.    if(m==0)
00413.       return;
00414.    LietKe(a,m-1,n);
00415.    for(int j=0;j<n;j++)
00416.       if(ktToanChan(a[m-1][j]))
00417.          cout << setw(4) << a[m-1][j];
00418. }
```

Bài 197.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các số nguyên có dạng 2^m.

– Ví dụ:

+ Dữ liệu vào:

0	1	2	3

0	12	64	-42	-73
1	-87	121	25	32
2	79	-82	1	-27

+ Dữ liệu ra: 64, 32, 1.

– Khai báo hàm.

```
00419. void LietKe(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00420. void LietKe(int a[][100],int m,int n)
00421. {
00422.    if(m==0)
00423.       return;
00424.    LietKe(a,m-1,n);
00425.    for(int j=0;j<n;j++)
00426.       if(ktDang2m(a[m-1][j]))
00427.          cout << setw(4) << a[m-1][j];
00428. }
```

Bài 198.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các giá trị chẵn trên một dòng.

– Ví dụ:

+ Dữ liệu vào:

0	12	38	-42	-73	18	-83	-87
1	-87	121	19	46	17	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	93	13	47	35	42

+ Dữ liệu ra: 46, 24.

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Các phần tử trên dòng d của ma trận a là: $a[d][0], a[d][1], \cdots, a[d][j], \cdots, a[d][n-2], a[d][n-1]$.

– Khai báo hàm.

```
00429. void LietKe(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00430. void LietKe(int a[][100],int m,int n,int d)
00431. {
00432.    if(n==0)
```

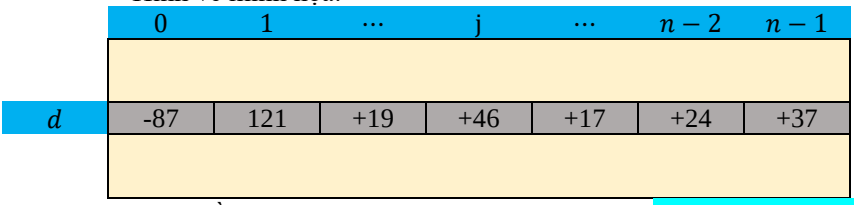
```
00433.        return;
00434.     LietKe(a,m,n-1,d);
00435.     if(a[d][n-1]%2==0)
00436.        cout << setw(4) << a[d][n-1];
00437.  }
```

Bài 199.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các số nguyên có dạng 3^m trên dòng.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3
0	12	81	-42	-73
1	81	121	85	3
2	243	-82	-11	-27

  - + Dữ liệu ra: 81, 3.
- Hình vẽ minh họa:



- Các phần tử trên dòng d của ma trận a là: $a[d][0], a[d][1], \cdots, a[d][j], \cdots, a[d][n-2], a[d][n-1]$.
- Khai báo hàm.

```
00438.  void LietKe(int [][100],int,int,int);
```

- Định nghĩa hàm đệ quy.

```
00439.  void LietKe(int a[][100],int m,int n,int d)
00440.  {
00441.     if(n==0)
00442.        return;
00443.     LietKe(a,m,n-1,d);
00444.     if(ktDang3m(a[d][n-1]))
00445.        cout << setw(4) << a[d][n-1];
00446.  }
```

Bài 200.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các số hoàn thiện trên một dòng.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3
0	12	6	-42	-73

1	-87	121	6	46
2	79	-82	28	-27

+ Dữ liệu ra: $+6$.

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Các phần tử trên dòng d của ma trận a là: $a[d][0]$, $a[d][1]$, $\cdots, a[d][j], \cdots, a[d][n-2]$, $a[d][n-1]$.

– Khai báo hàm.

```
00447.  void LietKe(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00448.  void LietKe(int a[][100],int m,int n,int d)
00449.  {
00450.     if(n==0)
00451.        return;
00452.     LietKe(a,m,n-1,d);
00453.     if(ktHoanThien(a[d][n-1]))
00454.        cout << setw(4) << a[d][n-1];
00455.  }
```

Bài 201. Cho ma trận các số nguyên. Hãy định nghĩa liệt kê các số chẵn trên một cột.

– Ví dụ:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	32	13	47	35	42

+ Tính tích các số chẵn trên cột 2: $-42 + 32 = 10$.

– Hình vẽ minh họa.

		c	
0		-73	
1		46	
⋯		-27	
i		13	
⋯		88	
$m-2$		12	
$m-1$		1	

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

```
00456. void LietKe(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00457. void LietKe(int a[][100],int m,int n,int c)
00458. {
00459.    if(m==0)
00460.       return;
00461.    LietKe(a,m-1,n,c);
00462.    if(a[m-1][c]%2==0)
00463.       cout << setw(4) << a[m-1][c];
00464. }
```

Bài 202.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các số đối xứng trên một cột.

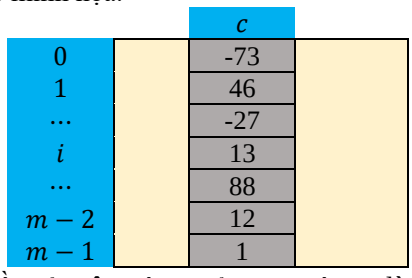
– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	-8	-73
1	-878	121	25	46
2	79	-82	-98	22

+ Dữ liệu ra: -8, -878, 22.

– Hình vẽ minh họa.



– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

```
00465. void LietKe(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00466. void LietKe(int a[][100],int m,int n,int c)
00467. {
00468.    if(m==0)
00469.       return;
```

```
00470.     LietKe(a,m-1,n,c);
00471.     if(ktDoiXung(a[m-1][c]))
00472.        cout << setw(4) << a[m-1][c];
00473.  }
```

Bài 203.Cho ma trận các số nguyên. Hãy định nghĩa các hàm liệt kê các số nguyên có dạng 5^m trên một cột.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3
0	12	1	-42	-73
1	-87	121	25	46
2	625	-82	-11	-27
	76	4	625	-11

  - + Dữ liệu ra: 25, 625.
- Các phần tử trên cột c của ma trận a là: $a[0][c], a[1][c], \cdots, a[i][c], \cdots, a[m-2][c], a[m-1][c]$.
- Khai báo hàm.

```
00474.  void LietKe(int [][100],int,int,int);
```

- Định nghĩa hàm đệ quy.

```
00475.  void LietKe(int a[][100],int m,int n,int c)
00476.  {
00477.     if(m==0)
00478.        return;
00479.     LietKe(a,m-1,n,c);
00480.     if(ktDang5m(a[m-1][c]))
00481.        cout << setw(4) << a[m-1][c];
00482.  }
```

06.09.02 Kỹ thuật tính toán đệ quy

Bài 204.Cho ma trận các số nguyên. Hãy định nghĩa hàm tính tổng giá trị chẵn.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3
0	12	38	-42	-73
1	-87	121	25	46
2	79	-82	-11	-27

  - + Dữ liệu ra: -28.
- Khai báo hàm.

```
00483.  int TongChan(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00484.  int TongChan(int a[][100],int m,int n)
00485.  {
00486.     if(m==0)
00487.        return 0;
00488.     int s = TongChan(a,m-1,n);
00489.     for(int j=0;j<n;j++)
00490.        if(a[m-1][j]%2==0)
00491.           s = s + a[m-1][j];
00492.     return s;
00493.  }
```

Bài 205.Cho ma trận các số nguyên. Hãy định nghĩa hàm tính tích giá giá trị lẻ tại các vị trí có chỉ số dòng là chỉ số chẵn.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	-42	-73
1	-87	11	25	46
2	72	-82	-11	-24

+ Dữ liệu ra: $(-73) \times (-11) = +803$

– Khai báo hàm.

```
00494.  int TichLe(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00495.  int TichLe(int a[][100],int m,int n)
00496.  {
00497.     if(m==0)
00498.        return 1;
00499.     int T = TichLe(a,m-1,n);
00500.     for(int j=0;j<n;j++)
00501.        if(a[m-1][j]%2!=0 && (m-1)%2==0)
00502.           T = T * a[m-1][j];
00503.     return T;
00504.  }
```

Bài 206.Cho ma trận các số nguyên. Hãy định nghĩa hàm tính tổng các số chính phương nằm trên các cột có chỉ số lẻ.

– Ví dụ:

+ Dữ liệu vào:

0	1	2	3

0	28	36	-42	-73
1	1	120	25	46
2	79	-82	-11	-27

+ Dữ liệu ra: 62.

– Các phần tử trên cột c của ma trận a là: $a[0][c], a[1][c], \cdots, a[i][c], \cdots, a[m-2][c], a[m-1][c]$.

– Khai báo hàm.

```
00505.  int TongChinhPhuong(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00506.  int TongChinhPhuong(int a[][100],int m,int n)
00507.  {
00508.     if(m==0)
00509.        return 0;
00510.     int s = TongChinhPhuong(a,m-1,n);
00511.     for(int j=0;j<n;j++)
00512.        if(ktChinhPhuong(a[m-1][j])&& j%2!=0)
00513.           s = s + a[m-1][j];
00514.     return s;
00515.  }
```

Bài 207.Cho ma trận các số nguyên. Hãy định nghĩa hàm tính tổng các số nguyên tố.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	-42	73
1	31	121	25	46
2	79	-82	11	-27

+ Dữ liệu ra: $73 + 31 + 11 = 115$.

– Khai báo hàm.

```
00516.  int TongNguyenTo(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00517.  int TongNguyenTo(int a[][100],int m,int n)
00518.  {
00519.     if(m==0)
00520.        return 0;
00521.     int s = TongNguyenTo(a,m-1,n);
00522.     for(int j=0;j<n;j++)
00523.        if(ktNguyenTo(a[m-1][j]))
00524.           s = s + a[m-1][j];
00525.     return s;
```

```
00526. }
```

Bài 208.Cho ma trận các số nguyên. Hãy định nghĩa hàm tính tổng các số nguyên toàn chữ số chẵn.

- Ví dụ:
  - Dữ liệu vào:

	0	1	2	3
0	12	38	-42	-73
1	-87	121	25	46
2	79	-82	-11	-27

  - Dữ liệu ra: -78.
- Khai báo hàm.

```
00527. int TongToanChan(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00528. int TongToanChan(int a[][100],int m,int n)
00529. {
00530.    if(m==0)
00531.       return 0;
00532.    int s = TongToanChan(a,m-1,n);
00533.    for(int j=0;j<n;j++)
00534.       if(ktToanChan(a[m-1][j]))
00535.          s = s + a[m-1][j];
00536.    return s;
00537. }
```

Bài 209.Cho ma trận các số nguyên. Hãy định nghĩa hàm tính tổng các số nguyên có dạng 3^m.

- Ví dụ:
  - Dữ liệu vào:

	0	1	2	3
0	12	16	-42	-73
1	-87	121	25	128
2	79	-82	-11	-27

  - Dữ liệu ra: $+144$.
- Khai báo hàm.

```
00538. int Tong3m(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00539. int Tong3m(int a[][100],int m,int n)
00540. {
00541.    if(m==0)
00542.       return 0;
```

```
00543.     int s = Tong3m(a,m-1,n);
00544.     for(int j=0;j<n;j++)
00545.        if(ktDang3m(a[m-1][j]))
00546.           s = s + a[m-1][j];
00547.     return s;
00548.  }
```

Bài 210.Cho ma trận các số thực. Hãy định nghĩa các hàm tính tổng các số dương.

- Ví dụ:
  - Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

  - Dữ liệu ra: 629.59
- Khai báo hàm.

```
00549.  float TongDuong(float [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00550.  float TongDuong(float a[][100],int m,int n)
00551.  {
00552.     if(m==0)
00553.        return 0;
00554.     float s = TongDuong(a,m-1,n);
00555.     for(int j=0;j<n;j++)
00556.        if(a[m-1][j]>0)
00557.           s = s + a[m-1][j];
00558.     return s;
00559.  }
```

Bài 211.Cho ma trận các số thực. Hãy định nghĩa các hàm tính tổng các số âm.

- Ví dụ:
  - Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

  - Dữ liệu ra: −242.59

– Khai báo hàm.

```
00560.  float TongAm(float [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00561.  float TongAm(float a[][100],int m,int n)
00562.  {
00563.     if(m==0)
00564.        return 0;
00565.     float s = TongAm(a,m-1,n);
00566.     for(int j=0;j<n;j++)
00567.        if(a[m-1][j]<0)
00568.           s = s + a[m-1][j];
00569.     return s;
00570.  }
```

Bài 212.Cho ma trận các số thực. Hãy định nghĩa các hàm tính tổng các giá trị nằm trong đoạn $[x, y]$.

– Ví dụ:

+ Dữ liệu vào: $[x, y] \equiv [-10, 10]$

	0	1	2	3	4
0	12.28	38.41	−0.43	−73.21	18.18
1	−15.82	11.48	67.31	46.19	17.45
2	0.21	−82.56	−11.12	−7.45	+8.89
3	79.21	52.54	83.56	3.21	23.67

+ Dữ liệu ra: +4.43

– Khai báo hàm.

```
00571.  float Tong(float [][100],int,int,
00572.             float,float);
```

– Định nghĩa hàm đệ quy.

```
00573.  float Tong(float a[][100],int m,int n,
00574.             float x,float y)
00575.  {
00576.     if(m==0)
00577.        return 0;
00578.     float s = Tong(a,m-1,n,x,y);
00579.     for(int j=0;j<n;j++)
00580.        if(a[m-1][j]>x && a[m-1][j]<y)
00581.           s = s + a[m-1][j];
00582.     return s;
00583.  }
```

Bài 213. Cho ma trận các số nguyên. Hãy định nghĩa các hàm tính tổng số chẵn trên một dòng.

– Ví dụ:

+ Dữ liệu vào:

0	12	38	-42	-73	18	-83	-87
1	-87	121	19	46	17	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	93	13	47	35	42

+ Dữ liệu ra: $+70$.

+ Thông báo (nếu cần thiết): Tổng số chẵn trên dòng 1 là: $46 + 24 = 70$.

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Khai báo hàm.

```
00584.  int TongDong(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00585.  int TongDong(int a[][100],int m,int n,int d)
00586.  {
00587.      if(n==0)
00588.          return 0;
00589.      int s = TongDong(a,m,n-1,d);
00590.      if(a[d][n-1]%2==0)
00591.          s = s + a[d][n-1];
00592.      return s;
00593.  }
```

Bài 214. Cho ma trận các số nguyên. Hãy định nghĩa các hàm tính tổng số nguyên tố trên một dòng.

– Ví dụ:

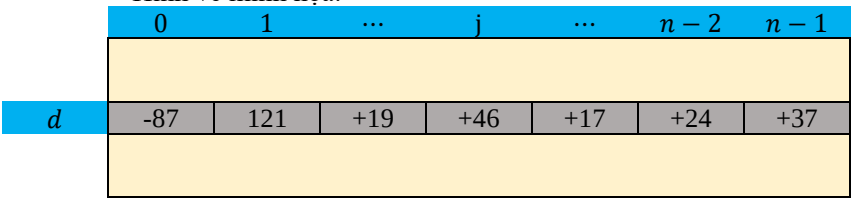
+ Dữ liệu vào:

0	12	38	-42	-73	18	-83	-87
1	-87	121	19	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	93	13	47	35	42

+ Dữ liệu ra: $+63$.

+ Thông báo (nếu cần thiết): Tổng số nguyên tố trên dòng 1 là: $19 + 7 + 37 = 63$.

– Hình vẽ minh họa:



– Khai báo hàm.

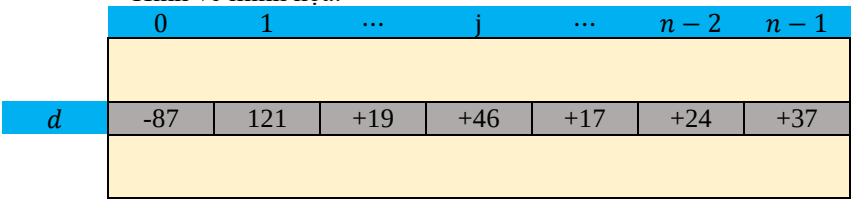
```
00594.  int TongDong(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00595.  int TongDong(int a[][100],int m,int n,int d)
00596.  {
00597.     if(n==0)
00598.        return 0;
00599.     int s = TongDong(a,m,n-1,d);
00600.     if(ktNguyenTo(a[d][n-1]))
00601.        s = s + a[d][n-1];
00602.     return s;
00603.  }
```

Bài 215.Cho ma trận các số nguyên. Hãy định nghĩa các hàm tính tổng các số đối xứng trên một dòng.

– Hình vẽ minh họa:



– Khai báo hàm.

```
00604.  int TongDong(int [][100],int,int,int);
```

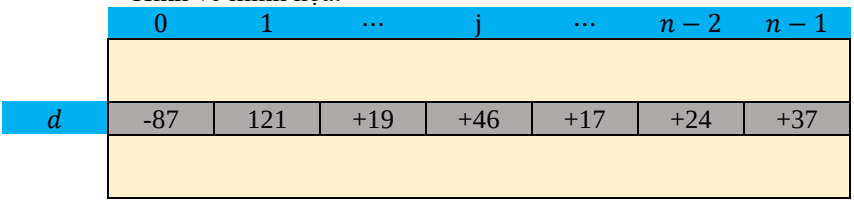
– Định nghĩa hàm đệ quy.

```
00605.  int TongDong(int a[][100],int m,int n,int d)
00606.  {
00607.     if(n==0)
00608.        return 0;
00609.     int s = TongDong(a,m,n-1,d);
00610.     if(ktDoiXung(a[d][n-1]))
00611.        s = s + a[d][n-1];
00612.     return s;
```

```
00613.  }
```

Bài 216.Cho ma trận các số thực. Hãy định nghĩa các hàm tính tổng các giá trị trên một dòng.

– Hình vẽ minh họa:



– Khai báo hàm.

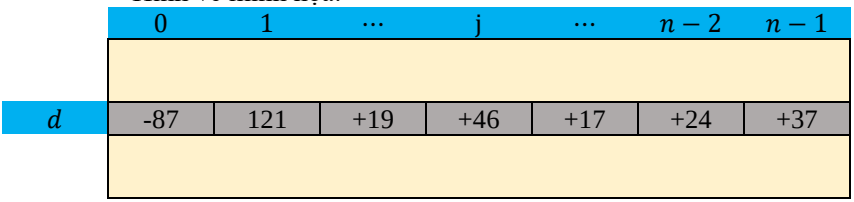
```
00614.  float TongDong(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00615.  float TongDong(float a[][100],int m,int n,
00616.                       int d)
00617.  {
00618.     if(n==0)
00619.        return 0;
00620.     float s = TongDong(a,m,n-1,d);
00621.     if(ktDoiXung(a[d][n-1]))
00622.        s = s + a[d][n-1];
00623.     return s;
00624.  }
```

Bài 217.Cho ma trận các số thực. Hãy định nghĩa các hàm tính tổng các giá trị dương trên một dòng.

– Hình vẽ minh họa:



– Khai báo hàm.

```
00625.  float TongDong(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00626.  float TongDong(float a[][100],int m,int n,
00627.                       int d)
00628.  {
00629.     if(n==0)
```

```
00630.        return 0;
00631.     float s = TongDong(a,m,n-1,d);
00632.     if(a[d][n-1]>0)
00633.        s = s + a[d][n-1];
00634.     return s;
00635.  }
```

Bài 218.Cho ma trận các số nguyên. Hãy định nghĩa hàm tính tổng các số nguyên có dạng 5^m trên một dòng.

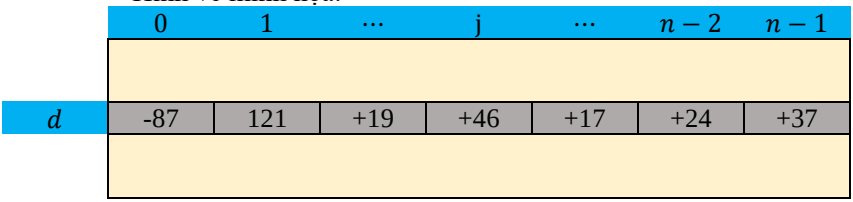
– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	27	38	-42	-73
1	81	121	1	46
2	79	81	-11	-27

+ Dữ liệu ra: $+89(81+1)$.

– Hình vẽ minh họa:



– Khai báo hàm.

```
00636.  int Tong5m(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00637.  int Tong5m(int a[][100],int m,int n,        int d)
00638.  {
00639.     if(n==0)
00640.        return 0;
00641.     float s = TongDong(a,m,n-1,d);
00642.     if(a[d][n-1]>0)
00643.        s = s + a[d][n-1];
00644.     return s;
00645.  }
```

Bài 219.Cho ma trận các số nguyên. Hãy định nghĩa các hàm tính tích các số chẵn trên một cột.

– Ví dụ:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37

2	79	-82	-11	-27	-96	42	-38
3	12	29	32	13	47	35	42

+ Tính tích các số chẵn trên cột 2: $-42 \times 32 = -1344$

– Hình vẽ minh họa.

		c	
0		-73	
1		46	
…		-27	
i		13	
…		88	
$m-2$		12	
$m-1$		1	

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c], a[m-1][c]$.

– Khai báo hàm.

```
00646.  int TichCot(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00647.  int TichCot(int a[][100],int m,int n,int c)
00648.  {
00649.     if(m==0)
00650.        return 1;
00651.     int T = TichCot(a,m-1,n,c);
00652.     if(a[m-1][c]%2==0)
00653.        T = T * a[m-1][c];
00654.     return T;
00655.  }
```

Bài 220. Cho ma trận các số nguyên. Hãy định nghĩa các hàm tính tổng các giá trị lẻ trên một cột.

– Ví dụ:

+ Dữ liệu vào:

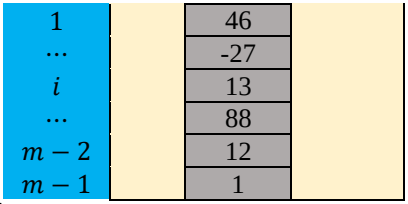
	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	32	13	47	35	42

+ Dữ liệu ra: $+14$.

+ Thông báo (nếu cần thiết): Tổng các giá trị lẻ trên cột 2: $25 + (-11) = 14$.

– Hình vẽ minh họa.

	c	
0	-73	



– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

```
00656.  int TongCot(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00657.  int TongCot(int a[][100],int m,int n,int c)
00658.  {
00659.      if(m==0)
00660.         return 0;
00661.      int s = TongCot(a,m-1,n,c);
00662.      if(a[m-1][c]%2!=0)
00663.         s = s + a[m-1][c];
00664.      return s;
00665.  }
```

Bài 221. Cho ma trận các số nguyên. Hãy định nghĩa các hàm tính tổng các số chính phương trên một cột.

– Ví dụ:

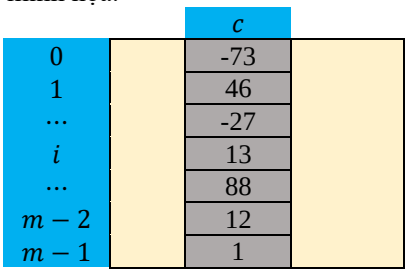
+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	32	13	47	35	42

+ Dữ liệu ra: $+25$.

+ Thông báo (nếu cần thiết): Tổng các số chính phương trên cột 2: 25.

– Hình vẽ minh họa.



– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

```
00666.  int TongCot(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00667.  int TongCot(int a[][100],int m,int n,int c)
00668.  {
00669.      if(m==0)
00670.          return 0;
00671.      int s = TongCot(a,m-1,n,c);
00672.      if(ktChinhPhuong(a[m-1][c]))
00673.          s = s + a[m-1][c];
00674.      return s;
00675.  }
```

Bài 222. Cho ma trận các số nguyên. Hãy định nghĩa các hàm tính tổng các giá trị có dạng 2^m trên một cột.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	32	13	47	35	42

+ Dữ liệu ra: $+32$.

+ Thông báo (nếu cần thiết): Tổng các giá trị có dạng 2^m trên cột 2: 32.

– Hình vẽ minh họa.

		c	
0		-73	
1		46	
…		-27	
i		13	
…		88	
$m-2$		12	
$m-1$		1	

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

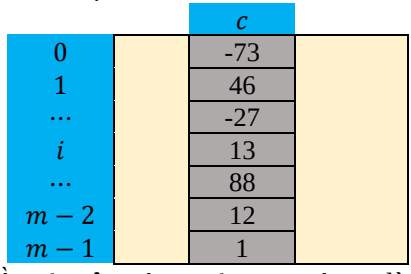
```
00676.  int TongCot(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00677.  int TongCot(int a[][100],int m,int n,int c)
00678.  {
00679.     if(m==0)
00680.        return 0;
00681.     int s = TongCot(a,m-1,n,c);
00682.     if(ktDang2m(a[m-1][c]))
00683.        s = s + a[m-1][c];
00684.     return s;
00685.  }
```

Bài 223.Cho ma trận các số thực. Hãy định nghĩa các hàm tính tích các giá trị dương trên một cột.

- Hình vẽ minh họa.



- Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c], a[m-1][c]$.
- Khai báo hàm.

```
00686.  float TichCot(float [][100],int,int,int);
```

- Định nghĩa hàm đệ quy.

```
00687.  float TichCot(float a[][100],int m,int n,
00688.                int c)
00689.  {
00690.     if(m==0)
00691.        return 1;
00692.     float T = TichCot(a,m-1,n,c);
00693.     if(a[m-1][c]>0)
00694.        T = T * a[m-1][c];
00695.     return T;
00696.  }
```

Bài 224.Cho ma trận các số thực. Hãy định nghĩa các hàm tính tích các giá trị thuộc đoạn $[-1,0]$ trên một cột.

- Hình vẽ minh họa.

0		-73	
1		46	
…		-27	
i		13	
…		88	
$m-2$		12	
$m-1$		1	

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

```
00697. float TichCot(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00698. float TichCot(float a[][100],int m,int n,
00699.                int c)
00700. {
00701.    if(m==0)
00702.       return 1;
00703.    float T = TichCot(a,m-1,n,c);
00704.    if(a[m-1][c]>=-1 && a[m-1][c]<=0)
00705.       T = T * a[m-1][c];
00706.    return T;
00707. }
```

06.09.03 Kỹ thuật đếm đệ quy

Bài 225.Cho ma trận các số nguyên. Hãy định nghĩa hàm đếm số lượng số chính phương.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	0	-73
1	-87	121	25	46
2	79	-82	-11	49

+ Dữ liệu ra: 4.

– Khai báo hàm.

```
00708. int DemChinhPhuong(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00709. int DemChinhPhuong(int a[][100], int m,int n)
00710. {
00711.    if(m==0)
00712.       return 0;
```

```
00713.     int dem = DemChinhPhuong(a,m-1,n);
00714.     for(int j=0;j<n;j++)
00715.        if(ktChinhPhuong(a[m-1][j]))
00716.           dem = dem + 1;
00717.     return dem;
00718.  }
```

Bài 226.Viết hàm đếm số lượng số dương trong ma trận các số thực.

- Ví dụ:
  + Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

  + Dữ liệu ra:
- Khai báo hàm.

```
00719.  int DemDuong(float [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00720.  int DemDuong(float a[][100], int m,int n)
00721.  {
00722.     if(m==0)
00723.        return 0;
00724.     int dem = DemDuong(a,m-1,n);
00725.     for(int j=0;j<n;j++)
00726.        if(a[m-1][j]>0)
00727.           dem = dem + 1;
00728.     return dem;
00729.  }
```

Bài 227.Cho ma trận các số nguyên. Hãy định nghĩa hàm đếm số lượng giá trị chẵn.

- Khai báo hàm.

```
00730.  int DemChan(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00731.  int DemChan(int a[][100], int m,int n)
00732.  {
00733.     if(m==0)
00734.        return 0;
00735.     int dem = DemChan(a,m-1,n);
00736.     for(int j=0;j<n;j++)
```

```
00737.        if(a[m-1][j]%2==0)
00738.           dem = dem + 1;
00739.     return dem;
00740.  }
```

Bài 228.Cho ma trận các số nguyên. Hãy định nghĩa hàm đếm số lượng số hoàn thiện.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3
0	6	38	-42	28
1	-87	121	28	46
2	79	-82	-11	6

  - + Dữ liệu ra: 4.
- Khai báo hàm.

```
00741.  int DemHoanThien(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00742.  int DemHoanThien(int a[][100], int m,int n)
00743.  {
00744.     if(m==0)
00745.        return 0;
00746.     int dem = DemHoanThien(a,m-1,n);
00747.     for(int j=0;j<n;j++)
00748.        if(ktHoanThien(a[m-1][j]))
00749.           dem = dem + 1;
00750.     return dem;
00751.  }
```

Bài 229.Cho ma trận các số nguyên. Hãy định nghĩa hàm đếm số lượng số nguyên toàn chữ số chẵn.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3
0	6	38	-42	282
1	-87	127	171	46
2	79	-82	-19	-8

  - + Dữ liệu ra: 6.
- Khai báo hàm.

```
00752.  int DemToanChan(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00753.  int DemToanChan(int a[][100], int m,int n)
```

```
00754. {
00755.    if(m==0)
00756.       return 0;
00757.    int dem = DemToanChan(a,m-1,n);
00758.    for(int j=0;j<n;j++)
00759.       if(ktToanChan(a[m-1][j]))
00760.          dem = dem + 1;
00761.    return dem;
00762. }
```

Bài 230.Cho ma trận các số nguyên. Hãy định nghĩa hàm đếm số lượng số nguyên có dạng 2^m.

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3
0	537	64	-42	282
1	1	127	171	46
2	79	-82	1024	-8

  - + Dữ liệu ra: 3.
- Khai báo hàm.

```
00763. int DemDang2m(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00764. int DemDang2m(int a[][100], int m,int n)
00765. {
00766.    if(m==0)
00767.       return 0;
00768.    int dem = DemDang2m(a,m-1,n);
00769.    for(int j=0;j<n;j++)
00770.       if(ktDang2m(a[m-1][j]))
00771.          dem = dem + 1;
00772.    return dem;
00773. }
```

Bài 231.Đếm tần suất xuất hiện của một giá trị x trong ma trận các số thực (336).

- Ví dụ:
  - + Dữ liệu vào: $x = 38.41$.

	0	1	2	3	4
0	12.28	38.41.	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	38.41
2	79.21	−82.56	−11.12	−27.45	66.89
3	38.41	52.54	83.56	43.21	23.67

+ Dữ liệu ra: 3.

– Khai báo hàm.

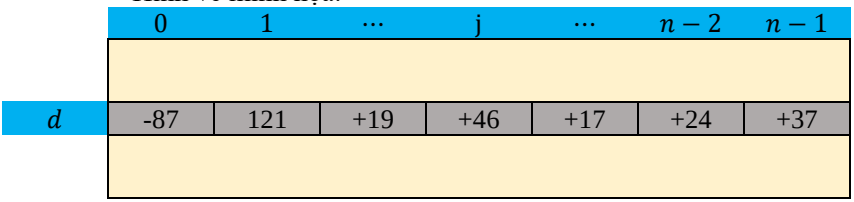
```
00774.  int TanSuat(float [][100],int,int,float);
```

– Định nghĩa hàm đệ quy.

```
00775.  int TanSuat(float a[][100], int m,int n,
00776.              float x)
00777.  {
00778.     if(m==0)
00779.        return 0;
00780.     int dem = TanSuat(a,m-1,n,x);
00781.     for(int j=0;j<n;j++)
00782.        if(a[m-1][j]==x)
00783.           dem = dem + 1;
00784.     return dem;
00785.  }
```

Bài 232.Đếm số lượng số dương trên một hàng trong ma trận các số thực.

– Hình vẽ minh họa:



– Khai báo hàm.

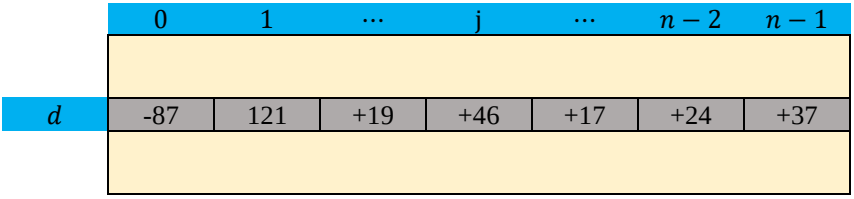
```
00786.  int DemDuongDong(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00787.  int DemDuongDong(float a[][100],int m,int n,
00788.                   int d)
00789.  {
00790.     if(n==0)
00791.        return 0;
00792.     int dem = DemDuongDong(a,m,n-1,d);
00793.     if(a[d][n-1]>0)
00794.        dem = dem + 1;
00795.     return dem;
00796.  }
```

Bài 233.Đếm số lượng số hoàn thiện trên một dòng trong ma trận các số nguyên.

– Hình vẽ minh họa:



– Khai báo hàm.

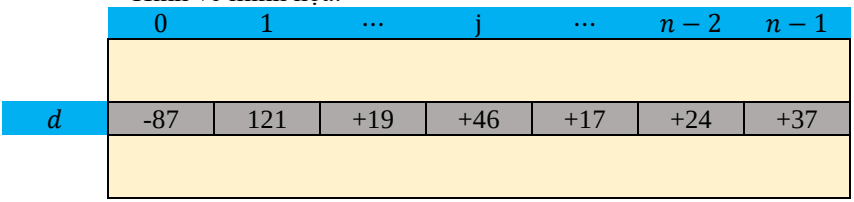
```
00797.  int DemHoanThien(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00798.  int DemHoanThien(int a[][100],int m,int n,
00799.                   int d)
00800.  {
00801.     if(n==0)
00802.        return 0;
00803.     int dem = DemHoanThien(a,m,n-1,d);
00804.     if(ktHoanThien(a[d][n-1]))
00805.        dem = dem + 1;
00806.     return dem;
00807.  }
```

Bài 234.Cho ma trận các số nguyên. Hãy định nghĩa hàm đếm số lượng số nguyên toàn chữ số lẻ trên một dòng.

– Hình vẽ minh họa:



– Khai báo hàm.

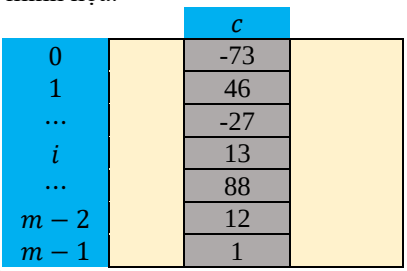
```
00808.  int DemToanLe(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00809.  int DemToanLe(int a[][100],int m,int n,
00810.                int d)
00811.  {
00812.     if(n==0)
00813.        return 0;
00814.     int dem = DemToanLe(a,m,n-1,d);
00815.     if(ktToanLe(a[d][n-1]))
00816.        dem = dem + 1;
00817.     return dem;
00818.  }
```

Bài 235.Đếm số lượng số âm trên một cột trong ma trận các số thực.

– Hình vẽ minh họa.



– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

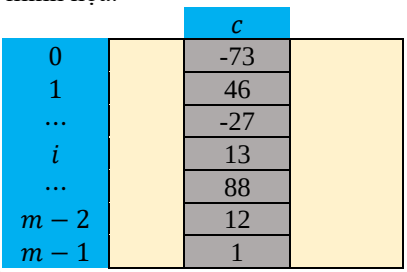
```
00819.  int DemAm(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00820.  int DemAm(float a[][100],int m,int n,int c)
00821.  {
00822.      if(m==0)
00823.         return 0;
00824.      int dem = DemAm(a,m-1,n,c);
00825.      if(a[m-1][c]<0)
00826.         dem = dem + 1;
00827.      return dem;
00828.  }
```

Bài 236.Đếm số lượng số nguyên tố trên một cột trong ma trận các số nguyên.

– Hình vẽ minh họa.



– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Khai báo hàm.

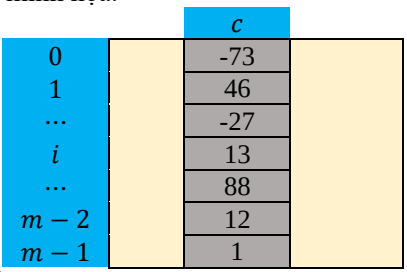
```
00829.  int DemNguyenTo(int [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00830.  int DemNguyenTo(int a[][100],int m,int n,
00831.                  int c)
00832.  {
00833.     if(m==0)
00834.        return 0;
00835.     int dem = DemNguyenTo(a,m-1,n,c);
00836.     if(ktNguyenTo(a[m-1][c]))
00837.        dem = dem + 1;
00838.     return dem;
00839.  }
```

Bài 237.Đếm số lượng số nguyên có chữ số đầu tiên là chữ số chẵn trên một cột trong ma trận các số nguyên.

- Hình vẽ minh họa.



- Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.
- Khai báo hàm.

```
00840.  int DemGiaTri(int [][100],int,int,int);
```

- Định nghĩa hàm đệ quy.

```
00841.  int DemGiaTri(int a[][100],int m,int n,int c)
00842.  {
00843.     if(m==0)
00844.        return 0;
00845.     int dem = DemGiaTri(a,m-1,n,c);
00846.     if(ChuSoDau(a[m-1][c])%2!=0)
00847.        dem = dem + 1;
00848.     return dem;
00849.  }
```

Bài 238.Đếm số lượng chữ số trong ma trận các số nguyên.

- Ví dụ:
  + Dữ liệu vào:

0	1	2	3

0	121	8	-42	-731
1	-87	1218	5	46
2	7	-82	212	-7

+ Dữ liệu ra: 31.

– Khai báo hàm.

```
00850. int DemChuSo(int);
00851. int DemChuSo(int [][100],int,int);
```

– Định nghĩa hàm đếm số lượng chữ số của một số nguyên.

```
00852. int DemChuSo(int a[][100],int m,int n)
00853. {
00854.    if(m==0)
00855.       return 0;
00856.    int dem = DemChuSo(a,m-1,n);
00857.    for(int j=0;j<n;j++)
00858.       dem = dem + DemChuSo(a[m-1][j]);
00859.    return dem;
00860. }
```

06.09.04 Kỹ thuật tìm kiếm đệ quy

Bài 239.Tìm giá trị lớn nhất trong ma trận (367).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

+ Dữ liệu ra: +83.56.

– Khai báo hàm.

```
00861. float LonNhat(float [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00862. float LonNhat(float a[][100], int m,int n)
00863. {
00864.    if(m==1)
00865.    {
00866.       float lc = a[0][0];
00867.       for(int j=0;j<n;j++)
00868.          if(a[0][j]>lc)
00869.             lc = a[0][j];
00870.       return lc;
```

```
00871.     }
00872.     float lc = LonNhat(a,m-1,n);
00873.     for(int j=0;j<n;j++)
00874.        if(a[m-1][j]>lc)
00875.              lc = a[m-1][j];
00876.     return lc;
00877.  }
```

Bài 240.Tìm số chẵn đầu tiên trong ma trận số nguyên. Nếu ma trận không có giá trị chẵn thì hàm sẽ trả về giá trị không chẵn là −1.

- Ví dụ:
  + Dữ liệu vào:

	0	1	2	3
0	121	827	-42	-731
1	-87	1218	5	46
2	7	-82	212	-7

  + Dữ liệu ra: −42.
- Khai báo hàm.

```
00878.  int ChanDau(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
00879.  int ChanDau(int a[][100],int m,int n)
00880.  {
00881.     if(m==0)
00882.        return -1;
00883.     int lc = ChanDau(a,m-1,n);
00884.     if(lc!=-1)
00885.        return lc;
00886.     for(int j=0;j<n;j++)
00887.        if(a[m-1][j]%2==0)
00888.           return a[m-1][j];
00889.     return -1;
00890.  }
```

Bài 241.Tìm giá trị dương đầu tiên trong ma trận. Nếu ma trận không có giá trị dương thì hàm sẽ trả về giá trị không dương là −1.

- Ví dụ:
  + Dữ liệu vào:

	0	1	2	3	4
0	−68.43	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

+ Dữ liệu ra: 38.41.

– Khai báo hàm.

```
00891.  float DuongDau(float [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00892.  float DuongDau(float a[][100],int m,int n)
00893.  {
00894.     if(m==0)
00895.        return -1;
00896.     int lc = DuongDau(a,m-1,n);
00897.     if(lc!=-1)
00898.        return lc;
00899.     for(int j=0;j<n;j++)
00900.        if(a[m-1][j]>0)
00901.           return a[m-1][j];
00902.     return -1;
00903.  }
```

Bài 242.Tìm số nguyên tố đầu tiên trong ma trận các số nguyên (374).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	11	38	-42	73
1	-87	121	29	46
2	79	-82	-11	53

+ Dữ liệu ra: 11.

– Khai báo hàm.

```
00904.  int NguyenToDau(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00905.  int NguyenToDau(int a[][100],int m,int n)
00906.  {
00907.     if(m==0)
00908.        return -1;
00909.     int lc = NguyenToDau(a,m-1,n);
00910.     if(lc!=-1)
00911.        return lc;
00912.     for(int j=0;j<n;j++)
00913.        if(ktNguyenTo(a[m-1][j]))
00914.           return a[m-1][j];
00915.     return -1;
00916.  }
```

Bài 243.Tìm giá trị lớn nhất trên một dòng trong ma trận các số thực.

– Ví dụ:

+ Dữ liệu vào:

0	12	38	-42	-73	18	-83	-87
1	-87	121	19	46	17	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	93	13	47	35	42

+ Dữ liệu ra: $+121$.

+ Lớn nhất trên dòng 1 là: $+121$.

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Các phần tử trên dòng d của ma trận a là: $a[d][0], a[d][1], \cdots, a[d][j], \cdots, a[d][n-2], a[d][n-1]$.

– Khai báo hàm.

```
00917.  float LonNhatDong(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00918.  float LonNhatDong(float a[][100],int m,int n,
00919.                    int d)
00920.  {
00921.     if(n==1)
00922.        return a[d][0];
00923.     float lc = LonNhatDong(a,m,n-1,d);
00924.     if(a[d][n-1]>lc)
00925.        lc = a[d][n-1];
00926.     return lc;
00927.  }
```

Bài 244.Tìm giá trị nhỏ nhất trên một cột trong ma trận các số thực (373).

– Ví dụ:

	0	1	2	3	4	5	6
0	12	38	42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	32	13	47	35	42

+ Giá trị nhỏ nhất trên cột 2 là: -11

– Hình vẽ minh họa.

	c
0	-73
1	46
…	-27
i	13
…	88
$m-2$	12
$m-1$	1

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c], a[m-1][c]$.

– Khai báo hàm.

```
00928. float NhoNhatCot(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
00929. float NhoNhatCot(float a[][100],int m,int n,
00930.                  int c)
00931. {
00932.    if(m==1)
00933.       return a[0][c];
00934.    float lc = NhoNhatCot(a,m-1,n,c);
00935.    if(a[m-1][c]<lc)
00936.       lc = a[m-1][c];
00937.    return lc;
00938. }
```

Bài 245.Tìm giá trị âm lớn nhất trong ma trận.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	+5.82	+11.48	+67.31	+46.19	+7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	66.28	52.54	83.56	43.21	23.67

+ Dữ liệu ra: −11.12.

– Khai báo hàm.

```
00939. float AmLonNhat(float [][100],int,int);
```

– Định nghĩa hàm âm lớn nhất.

```
00940. float AmLonNhat(float a[][100], int m,int n)
00941. {
00942.    if(m==0)
00943.       return 0;
00944.    float lc = AmLonNhat(a,m-1,n);
```

```
00945.     for(int j=0;j<n;j++)
00946.        if(a[m-1][j]<0)
00947.           if(lc==0 || a[m-1][j]>lc)
00948.              lc = a[m-1][j];
00949.     return lc;
00950.  }
```

Bài 246.Tìm số chẵn lớn nhất trong ma trận các số nguyên (375).

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	12	29	32	13	47	35	42

  - + Dữ liệu ra: +46.
- Khai báo hàm.

```
00951.  int ChanLonNhat(int [][100],int,int);
```

- Định nghĩa hàm chẵn lớn nhất.

```
00952.  int ChanLonNhat(int a[][100], int m,int n)
00953.  {
00954.     if(m==0)
00955.        return -1;
00956.     int lc = ChanLonNhat(a,m-1,n);
00957.     for(int j=0;j<n;j++)
00958.        if(a[m-1][j]%2==0)
00959.           if(lc==-1 || a[m-1][j]>lc)
00960.              lc = a[m-1][j];
00961.     return lc;
00962.  }
```

Bài 247.Tìm giá trị dương nhỏ nhất trong ma trận các số thực (376).

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	+5.82	+11.48	+67.31	+46.19	+7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	66.28	52.54	83.56	43.21	23.67

  - + Dữ liệu ra: +5.82.
- Khai báo hàm.

```
00963. float DuongNhoNhat(float [][100],int,int);
```

– Định nghĩa hàm dương nhỏ nhất.

```
00964. float DuongNhoNhat(float a[][100],int m,int n)
00965. {
00966.    if(m==0)
00967.       return 0;
00968.    float lc = DuongNhoNhat(a,m-1,n);
00969.    for(int j=0;j<n;j++)
00970.       if(a[m-1][j]>0)
00971.          if(lc==0 || a[m-1][j]<lc)
00972.             lc = a[m-1][j];
00973.    return lc;
00974. }
```

Bài 248.Tìm số nguyên tố lớn nhất trong ma trận các số nguyên (377).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	11	38	-42	73
1	-87	121	29	46
2	79	-82	-11	53

+ Dữ liệu ra: 73.

– Khai báo hàm.

```
00975. int NguyenToLonNhat(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
00976. int NguyenToLonNhat(int a[][100],int m,int n)
00977. {
00978.    if(m==0)
00979.       return -1;
00980.    int lc = NguyenToLonNhat(a,m-1,n);
00981.    for(int j=0;j<n;j++)
00982.       if(ktNguyenTo(a[m-1][j]))
00983.          if(lc==-1 || a[m-1][j]>lc)
00984.             lc = a[m-1][j];
00985.    return lc;
00986. }
```

Bài 249.Tìm số chính phương lớn nhất trong ma trận các số nguyên.

– Ví dụ:

+ Dữ liệu vào:

0	1	2	3

0	12	38	0	-73
1	-87	121	25	46
2	79	-82	-11	49

+ Dữ liệu ra: 121.

– Khai báo hàm.

```
00987.  int ChinhPhuongLonNhat(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

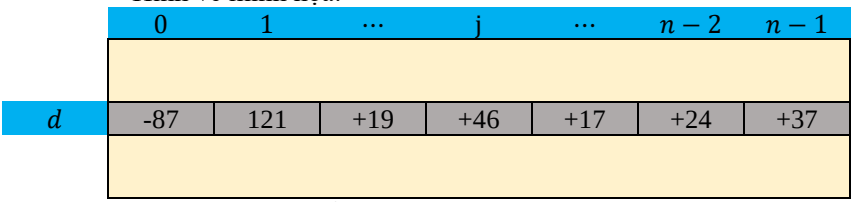
```
00988.  int ChinhPhuongLonNhat(int a[][100],
00989.                                          int m,int n)
00990.  {
00991.     if(m==0)
00992.        return -1;
00993.     int lc = ChinhPhuongLonNhat a,m-1,n);
00994.     for(int j=0;j<n;j++)
00995.        if(ktChinhPhuong(a[m-1][j]))
00996.           if(lc==-1 || a[m-1][j]>lc)
00997.              lc = a[m-1][j];
00998.     return lc;
00999.  }
```

Bài 250.Tìm tổng dòng lớn nhất trong ma trận các số thực.

– Khai báo hàm.

```
01000.  float TongDong(float [][100],int,int,int);
01001.  float TongLonNhat(float [][100],int,int);
```

– Hình vẽ minh họa:



– Định nghĩa hàm tổng dòng.

```
01002.  float TongDong(float a[][100],int m,int n,
01003.                    int d)
01004.  {
01005.     if(n==0)
01006.        return 0;
01007.     return TongDong(a,m,n-1,d) + a[d][n-1];
01008.  }
```

– Định nghĩa hàm tổng dòng lớn nhất.

```
01009.  float TongLonNhat(float a[][100],int m,int n)
```

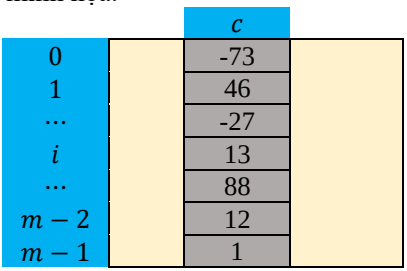
```
01010. {
01011.    if(m==1)
01012.       return TongDong(a,m,n,0);
01013.    float lc = TongLonNhat(a,m-1,n);
01014.    if(TongDong(a,m,n,m-1)>lc)
01015.          lc = TongDong(a,m,n,m-1);
01016.    return lc;
01017. }
```

Bài 251.Tìm tổng cột nhỏ nhất trong ma trận các số thực.

– Khai báo hàm.

```
01018. float TongCot(float [][100],int,int,int);
01019. float TongNhoNhat(float [][100],int,int);
```

– Hình vẽ minh họa.



– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c]$, $a[m-1][c]$.

– Định nghĩa hàm tổng cột.

```
01020. float TongCot(float a[][100],int m,int n,
01021.                  int c)
01022. {
01023.    if(m==0)
01024.       return 0;
01025.    return TongCot(a,m-1,n,c) + a[m-1][c];
01026. }
```

– Định nghĩa hàm tổng cột nhỏ nhất.

```
01027. float TongNhoNhat(float a[][100],int m,int n)
01028. {
01029.    if(n==1)
01030.       return TongCot(a,m,n,0);
01031.    float lc = TongNhoNhat(a,m,n-1);
01032.    if(TongCot(a,m,n,n-1)>lc)
01033.          lc = TongCot(a,m,n,n-1);
01034.    return lc;
```

```
01035. }
```

06.09.05 Kỹ thuật đặt cờ hiệu

Bài 252.Kiểm tra ma trận có tồn tại số dương hay không (348).

- Ví dụ:
  - + Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	83.56	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

  - + Dữ liệu ra: +1.
- Khai báo hàm.

```
01036. int TonTaiDuong(float [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
01037. int TonTaiDuong(float a[][100], int m,int n)
01038. {
01039.    if(m==0)
01040.       return 0;
01041.    for(int j=0;j<n;j++)
01042.       if(a[m-1][j]>0)
01043.             return 1;
01044.    return TonTaiDuong(a,m-1,n);
01045. }
```

Bài 253.Kiểm tra ma trận có tồn tại số lẻ hay không (350).

- Khai báo hàm.

```
01046. int TonTaiLe(int [][100],int,int);
```

- Định nghĩa hàm đệ quy.

```
01047. int TonTaiLe(int a[][100], int m,int n)
01048. {
01049.    if(m==0)
01050.       return 0;
01051.    for(int j=0;j<n;j++)
01052.       if(a[m-1][j]%2!=0)
01053.             return 1;
01054.    return TonTaiLe(a,m-1,n);
01055. }
```

Bài 254.Kiểm tra ma trận có tồn tại số hoàn thiện hay không (349).

– Khai báo hàm.

```
01056.  int TonTaiHoanThien(int [][100],int,int);
```

– Định nghĩa hàm đệ quy.

```
01057.  int TonTaiHoanThien(int a[][100],int m,int n)
01058.  {
01059.     if(m==0)
01060.        return 0;
01061.     for(int j=0;j<n;j++)
01062.        if(ktHoanThien(a[m-1][j]))
01063.              return 1;
01064.     return TonTaiLe(a,m-1,n);
01065.  }
```

Bài 255.Kiểm tra ma trận có toàn dương hay không (351).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	83.56	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

+ Dữ liệu ra: 0.

– Khai báo hàm.

```
01066.  int ktToanDuong(float [][100],int,int);
```

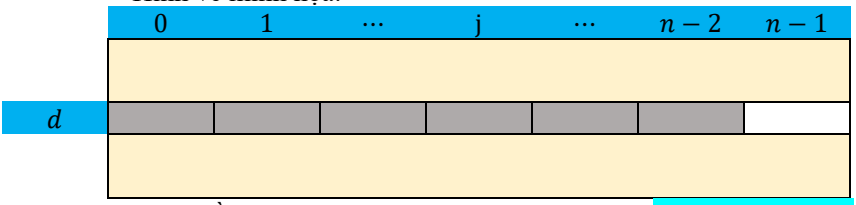
– Định nghĩa hàm đệ quy.

```
01067.  int ktToanDuong(float a[][100],int m,int n)
01068.  {
01069.     if(m==0)
01070.        return 0;
01071.     if(m==1)
01072.     {
01073.        int flag = 1;
01074.        for(int j=0;j<n;j++)
01075.           if(a[0][j]<=0)
01076.              flag = 0;
01077.        return flag;
01078.     }
01079.     for(int j=0;j<n;j++)
01080.        if(a[m-1][j]<=0)
01081.              return 0;
```

```
01082.     return ktToanDuong(a,m-1,n);
01083.  }
```

Bài 256.Kiểm tra một hàng ma trận có tăng dần hay không (352).

- Hình vẽ minh họa:



- Các phần tử trên dòng d của ma trận a là: $a[d][0], a[d][1], \cdots, a[d][j], \cdots, a[d][n-2], a[d][n-1]$.
- Dòng d tăng dần khi: $a[d][0] \le a[d][1] \le \cdots \le a[d][j] \le \cdots \le a[d][n-2] \le a[d][n-1]$.
- Khai báo hàm.

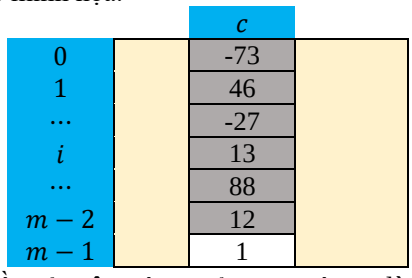
```
01084.  int ktDongTang(float [][100],int,int,int);
```

- Định nghĩa hàm đệ quy.

```
01085.  int ktDongTang(float a[][100],int m,int n,
01086.                    int d)
01087.  {
01088.     if(n==1)
01089.        return 1;
01090.     if(a[d][n-2]<=a[d][n-1] &&
01091.        ktDongTang(a,m,n-1,d)==1)
01092.        return 1;
01093.     return 0;
01094.  }
```

Bài 257.Kiểm tra một cột trong ma trận có giảm dần hay không (353).

- Hình vẽ minh họa.



- Các phần tử trên cột c của ma trận a là: $a[0][c], a[1][c], \cdots, a[i][c], \cdots, a[m-2][c], a[m-1][c]$.

– Cột c giảm dần khi: $a[0][c] \geq a[1][c] \geq \cdots \geq a[i][c] \geq \cdots \geq a[m-2][c] \geq a[m-1][c]$.

– Khai báo hàm.

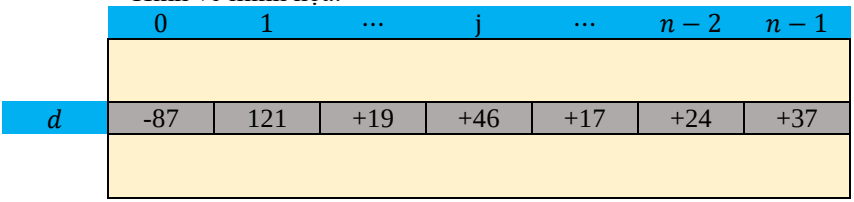
```
01095. int ktCotGiam(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
01096. int ktCotGiam(float a[][100],int m,int n,
01097.                 int c)
01098. {
01099.    if(m==1)
01100.       return 1;
01101.    if(a[m-2][c]>=a[m-1][c] &&
01102.       ktCotGiam(a,m-1,n,c)==1)
01103.       return 1;
01104.    return 0;
01105. }
```

Bài 258.Liệt kê các dòng có chứa giá trị chẵn trong ma trận các số nguyên (358).

– Hình vẽ minh họa:



– Khai báo hàm.

```
01106. int ktDong(int[][100],int,int,int);
01107. void LietKe(int[][100],int,int);
```

– Định nghĩa hàm kiểm tra dòng có chứa giá trị chẵn hay không?

```
01108. int ktDong(int a[][100],int m,int n,int d)
01109. {
01110.    if(n==0)
01111.       return 0;
01112.    if(a[d][n-1]%2==0)
01113.       return 1;
01114.    return ktDong(a,m,n-1,d);
01115. }
```

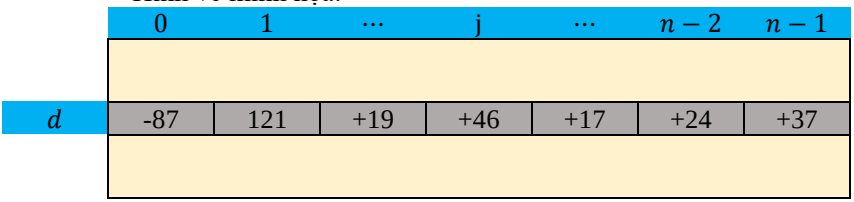
– Định nghĩa hàm liệt kê các dòng có chứa giá trị chẵn

```
01116. void LietKe(int a[][100],int m,int n)
01117. {
01118.    if(m==0)
```

```
01119.       return;
01120.    LietKe(a,m-1,n);
01121.    if(ktDong(a,m,n,m-1)==1)
01122.       cout << setw(4) << (m-1);
01123. }
```

Bài 259.Liệt kê các dòng có chứa giá trị âm trong ma trận các số thực.

– Hình vẽ minh họa:



– Khai báo hàm.

```
01124. int ktDong(float [][100],int,int,int);
01125. void LietKe(float [][100],int,int);
```

– Định nghĩa hàm kiểm tra dòng có chứa giá trị âm hay không?

```
01126. int ktDong(float a[][100],int m,int n,int d)
01127. {
01128.    if(n==0)
01129.       return 0;
01130.    if(a[d][n-1]<0)
01131.       return 1;
01132.    return ktDong(a,m,n-1,d);
01133. }
```

– Định nghĩa hàm liệt kê các dòng có chứa giá trị âm.

```
01134. void LietKe(float a[][100],int m,int n)
01135. {
01136.    if(m==0)
01137.       return;
01138.    LietKe(a,m-1,n);
01139.    if(ktDong(a,m,n,m-1)==1)
01140.       cout << setw(4) << (m-1);
01141. }
```

Bài 260.Liệt kê các dòng có chứa số nguyên tố trong ma trận các số nguyên.

– Ví dụ:

+ Dữ liệu vào:

0	1	2	3

0	11	38	-42	73
1	-87	121	29	46
2	79	-82	-11	53

+ Dữ liệu ra: $0, 1, 2$.

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Khai báo hàm.

```
01142.  int ktDong(int [][100],int,int,int);
01143.  void LietKe(int [][100],int,int);
```

– Định nghĩa hàm kiểm tra dòng có chứa số nguyên tố hay không?

```
01144.  int ktDong(int a[][100],int m,int n,int d)
01145.  {
01146.     if(n==0)
01147.        return 0;
01148.     if(a[d][n-1]<0)
01149.        return 1;
01150.     return ktDong(a,m,n-1,d);
01151.  }
```

– Định nghĩa hàm liệt kê các dòng có chứa số nguyên tố.

```
01152.  void LietKe(int a[][100],int m,int n)
01153.  {
01154.     if(m==0)
01155.        return;
01156.     LietKe(a,m-1,n);
01157.     if(ktDong(a,m,n,m-1)==1)
01158.        cout << setw(4) << (m-1);
01159.  }
```

Bài 261.Liệt kê các dòng trong ma trận các số thực thỏa mãn đồng thời các điều kiện sau: dòng có chứa giá trị âm, giá trị dương và giá trị 0 (*phần tử trung hòa*).

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Khai báo hàm.

```
01160.  int ktDong(float [][100],int,int,int);
01161.  void LietKe(float [][100],int,int);
```

– Định nghĩa hàm kiểm tra dòng có chứa giá trị âm, giá trị dương và giá trị 0 hay không?

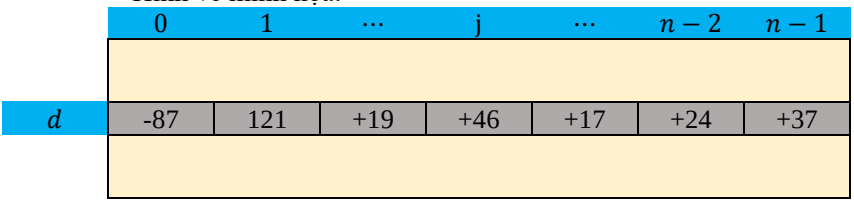
```
01162.  int ktDong1(float a[][100],int m,int n,int d)
01163.  {
01164.     if(n==0)
01165.        return 0;
01166.     if(a[d][n-1]<0)
01167.        return 1;
01168.     return ktDong1(a,m,n-1,d);
01169.  }
01170.  int ktDong2(float a[][100],int m,int n,int d)
01171.  {
01172.     if(n==0)
01173.        return 0;
01174.     if(a[d][n-1]>0)
01175.        return 1;
01176.     return ktDong2(a,m,n-1,d);
01177.  }
01178.  int ktDong3(float a[][100],int m,int n,int d)
01179.  {
01180.     if(n==0)
01181.        return 0;
01182.     if(a[d][n-1]==0)
01183.        return 1;
01184.     return ktDong3(a,m,n-1,d);
01185.  }
```

– Định nghĩa hàm liệt kê các dòng có chứa giá trị âm, giá trị dương và giá trị 0.

```
01186.  void LietKe(float a[][100],int m,int n)
01187.  {
01188.     if(m==0)
01189.        return;
01190.     LietKe(a,m-1,n);
01191.     if(ktDong1(a,m,n,m-1)==1) &&
01192.        ktDong2(a,m,n,m-1)==1) &&
01193.        ktDong3(a,m,n,m-1)==1))
01194.        cout << setw(4) << (m-1);
01195.  }
```

Bài 262.Liệt kê các dòng toàn âm trong ma trận các số thực (355).

– Hình vẽ minh họa:



– Khai báo hàm.

```
01196.  int ktDong(float [][100],int,int,int);
01197.  void LietKe(float [][100],int,int);
```

– Định nghĩa hàm kiểm tra dòng có toàn âm hay không?

```
01198.  int ktDong(float a[][100],int m,int n,int d)
01199.  {
01200.     if(n==0)
01201.        return 0;
01202.     if(n==1)
01203.     {
01204.        if(a[d][0]<0)
01205.           return 1;
01206.        else
01207.           return 0;
01208.     }
01209.     if(a[d][n-1]<0 && ktDong(a,m,n-1,d)==1)
01210.        return 1;
01211.     return 0;
01212.  }
```

– Định nghĩa hàm liệt kê các dòng toàn âm.

```
01213.  void LietKe(float a[][100],int m,int n)
01214.  {
01215.     if(m==0)
01216.        return;
01217.     LietKe(a,m-1,n);
01218.     if(ktDong(a,m,n,m-1)==1)
01219.        cout << setw(4) << (m-1);
01220.  }
```

Bài 263.Liệt kê chỉ số các dòng chứa toàn giá trị chẵn trong ma trận các số nguyên (356).

– Hình vẽ minh họa:

d	-87	121	+19	+46	+17	+24	+37

– Khai báo hàm.

```
01221.  int ktDong(int [][100],int,int,int);
01222.  void LietKe(int [][100],int,int);
```

– Định nghĩa hàm kiểm tra dòng có toàn chẵn hay không?

```
01223.  int ktDong(int a[][100],int m,int n,int d)
01224.  {
01225.     if(n==0)
01226.        return 0;
01227.     if(n==1)
01228.     {
01229.        if(a[d][0]%2==0)
01230.           return 1;
01231.        else
01232.           return 0;
01233.     }
01234.     if(a[d][n-1]%2==0 && ktDong(a,m,n-1,d)==1)
01235.        return 1;
01236.     return 0;
01237.  }
```

– Định nghĩa hàm liệt kê các dòng toàn chẵn.

```
01238.  void LietKe(int a[][100],int m,int n)
01239.  {
01240.     if(m==0)
01241.        return;
01242.     LietKe(a,m-1,n);
01243.     if(ktDong(a,m,n,m-1)==1)
01244.        cout << setw(4) << (m-1);
01245.  }
```

Bài 264. Liệt kê các cột trong ma trận các số nguyên có chứa số chính phương (360).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	0	-73
1	-87	121	25	46
2	79	-82	-11	49

+ Dữ liệu ra: 1, 2, 3.

– Hình vẽ minh họa.

	c
0	-73
1	46
⋯	-27
i	13
⋯	88
$m-2$	12
$m-1$	1

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c], a[m-1][c]$.

– Khai báo hàm.

```
01246.  int ktCot(int [][100],int,int,int);
01247.  void LietKe(int [][100],int,int);
```

– Định nghĩa hàm kiểm tra cột có số chính phương hay không?

```
01248.  int ktCot(int a[][100],int m,int n,int c)
01249.  {
01250.     if(m==0)
01251.        return 0;
01252.     if(ktChinhPhuong(a[m-1][c]))
01253.        return 1;
01254.     return ktCot(a,m-1,n,c);
01255.  }
```

– Định nghĩa hàm liệt kê các cột có chứa số chính phương.

```
01256.  void LietKe(int a[][100],int m,int n)
01257.  {
01258.     if(n==0)
01259.        return;
01260.     LietKe(a,m,n-1);
01261.     if(ktCot(a,m,n,n-1)==1)
01262.        cout << setw(4) << (n-1);
01263.  }
```

Bài 265. Liệt kê các dòng giảm dần trong ma trận số thực (362).

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Khai báo hàm.

```
01264.  int ktDongGiam(float [][100],int,int,int);
01265.  void LietKe(float [][100],int,int);
```

– Định nghĩa hàm kiểm tra dòng có giảm dần hay không?

```
01266.  int ktDongGiam(float a[][100],int m,int n,
01267.                  int d)
01268.  {
01269.     if(n==1)
01270.        return 1;
01271.     if(a[d][n-2]>=a[d][n-1] &&
01272.        ktDongGiam(a,m,n-1,d)==1)
01273.        return 1;
01274.     return 0;
01275.  }
```

– Định nghĩa hàm liệt kê các dòng giảm dần.

```
01276.  void LietKe(float a[][100],int m,int n)
01277.  {
01278.     if(m==0)
01279.        return;
01280.     LietKe(a,m-1,n);
01281.     if(ktDongGiam(a,m,n,m-1)==1)
01282.        cout << setw(4) << (m-1);
01283.  }
```

Bài 266.Liệt kê các cột tăng dần trong ma trận số thực (363).

– Hình vẽ minh họa.

	c
0	-73
1	46
…	-27
i	13
…	88
$m-2$	12
$m-1$	1

– Các phần tử trên cột c của ma trận a là: $a[0][c]$, $a[1][c]$, $\cdots, a[i][c], \cdots, a[m-2][c], a[m-1][c]$.

– Khai báo hàm.

```
01284.  int ktCotTang(float [][100],int,int,int);
01285.  void LietKe(float [][100],int,int);
```

– Định nghĩa hàm kiểm tra cột có tăng dần hay không?

```
01286.  int ktCotTang(float a[][100],int m,int n,
```

```
01287.                  int c)
01288. {
01289.    if(m==1)
01290.       return 1;
01291.    if(a[m-2][c]<=a[m-1][c] &&
01292.       ktCotTang(a,m-1,n,c)==1)
01293.       return 1;
01294.    return 0;
01295. }
01296.
```

– Định nghĩa hàm liệt kê các cột tăng dần.

```
01297. void LietKe(float a[][100],int m,int n)
01298. {
01299.    if(n==0)
01300.       return;
01301.    LietKe(a,m,n-1);
01302.    if(ktCotTang(a,m,n,n-1)==1)
01303.       cout << setw(4) << (n-1);
01304. }
```

06.09.06 Kỹ thuật xây dựng ma trận

Bài 267.Cho ma trận các số thực $A(m \times n)$. Hãy xây dựng ma trận $B(m \times n)$ từ ma trận A sao cho B$[i][j]$ = abs(A$[i][j]$) (426).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	83.56	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

+ Dữ liệu ra:

	0	1	2	3	4
0	12.28	38.41	+42.43	+73.21	18.18
1	+5.82	11.48	67.31	83.56	7.45
2	79.21	+82.56	+11.12	+27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

– Khai báo hàm.

```
01305. void XayDung(float [][100],int,int
01306.              float [][100],int &,int &);
```

– Định nghĩa hàm đệ quy.

```
01307.  void XayDung(  float a[][100],int m, int n,
01308.                 float b[][100],int &k,int &l)
01309.  {
01310.     if(m==0)
01311.     {
01312.        k = 0;
01313.        l = n;
01314.        return;
01315.     }
01316.     XayDung(a,m-1,n,b,k,l);
01317.     for(int j=0;j<l;j++)
01318.        b[m-1][j] = abs(a[m-1][j]);
01319.     k++;
01320.  }
```

06.09.07 Kỹ thuật sắp xếp

Bài 268.Hoán vị hai dòng trên ma trận.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	78	23	8	74	38	82	83
2	79	-82	-11	-27	-96	42	-38
3	-87	121	25	46	7	24	37
4	42	35	92	52	39	12	99

– Hình vẽ minh họa:

	0	1	…	j	…	$n-2$	$n-1$
$d1$	-87	121	+19	+46	+17	+24	+37
$d2$	-87	121	+19	+46	+17	+24	+37

– Khai báo hàm.

```
01321.  void HoanViDong(float [][100],int,int,int,
01322.                       int);
```

– Định nghĩa hàm đệ quy.

```
01323.  void HoanViDong(float a[][100],int m,int n,
01324.                       int d1,int d2)
01325.  {
01326.     if(n==0)
01327.        return;
01328.     HoanViDong(a,m,n-1,d1,d2);
01329.     HoanVi(a[d1][n-1],a[d2][n-1]);
01330.  }
```

Bài 269.Hoán vị hai cột trên ma trận.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	12	38	18	-73	-42	-83	-87
1	-87	121	7	46	25	24	37
2	79	-82	-96	-27	-11	42	-38
3	78	23	38	74	8	82	83
4	42	35	39	52	92	12	99

– Khai báo hàm.

```
01331.  void HoanViCot(float [][100],int,int,int,
01332.                    int);
```

– Định nghĩa hàm đệ quy.

```
01333.  void HoanViCot(   float a[][100],int m,int n,
01334.                       int c1,int c2)
01335.  {
01336.     if(m==0)
01337.        return;
01338.     HoanViDong(a,m-1,n,c1,c2);
01339.     HoanVi(a[m-1][c1],a[m-1][c2]);
```

```
01340. }
```

Bài 270.Định nghĩa hàm sắp xếp các phần tử trên một dòng tăng dần từ trái sang phải (407).

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	-96	-82	-38	-27	-11	42	79
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

– Khai báo hàm.

```
01341. void SapDongTang(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
01342. void SapDongTang(float a[][100],int m,int n,
01343.                  int d)
01344. {
01345.    if(n==1)
01346.       return;
01347.    for(int j=0;j<=n-2;j++)
01348.       if(a[d][j] > a[d][n-1])
01349.          HoanVi(a[d][j],a[d][n-1]);
01350.    SapDongTang(a,m,n-1,d);
01351. }
```

Bài 271.Viết hàm sắp xếp các phần tử trên một dòng giảm dần từ trái sang phải (408).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	42	-11	-27	-38	-82	-96
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

– Hình vẽ minh họa:

	0	1	⋯	j	⋯	$n-2$	$n-1$
d	-87	121	+19	+46	+17	+24	+37

– Khai báo hàm.

```
01352. void SapDongGiam(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
01353. void SapDongGiam(float a[][100],int m,int n,
01354.                  int d)
01355. {
01356.    if(n==1)
01357.       return;
01358.    for(int j=0;j<=n-2;j++)
01359.       if(a[d][j] < a[d][n-1])
01360.          HoanVi(a[d][j],a[d][n-1]);
01361.    SapDongGiam(a,m,n-1,d);
01362. }
01363.
```

Bài 272. Viết hàm sắp xếp các phần tử trên một cột tăng dần từ trên xuống dưới (409).

– Ví dụ:

+ Dữ liệu vào:

0	1	2	3	4	5	6

0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	-27	7	24	37
2	79	-82	-11	46	-96	42	-38
3	78	23	8	52	38	82	83
4	42	35	92	74	39	12	99

– Khai báo hàm.

```
01364. void SapCotTang(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
01365. void SapCotTang(float a[][100],int m,int n,
01366.                 int c)
01367. {
01368.    if(m==1)
01369.       return;
01370.    for(int i=0;i<=m-2;i++)
01371.       if(a[i][c] > a[m-1][c])
01372.            HoanVi(a[i][c],a[m-1][c]);
01373.    SapCotTang(a,m-1,n,c);
01374. }
```

Bài 273.Viết hàm sắp xếp các phần tử trên một cột giảm dần từ trên xuống dưới (410).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	12	38	-42	74	18	-83	-87
1	-87	121	25	52	7	24	37
2	79	-82	-11	46	-96	42	-38
3	78	23	8	-27	38	82	83

4	42	35	92	-73	39	12	99

– Khai báo hàm.

```
01375.  void SapCotGiam(float [][100],int,int,int);
```

– Định nghĩa hàm đệ quy.

```
01376.  void SapCotGiam(float a[][100],int m,int n,
01377.                  int c)
01378.  {
01379.     if(m==1)
01380.        return;
01381.     for(int i=0;i<=m-2;i++)
01382.        if(a[i][c] < a[m-1][c])
01383.              HoanVi(a[i][c],a[m-1][c]);
01384.     SapCotGiam(a,m-1,n,c);
01385.  }
```

Bài 274.Viết hàm sắp xếp các phần tử trong ma trận theo yêu cầu sau:
- Dòng có chỉ số chẵn tăng dần.
- Dòng có chỉ số lẻ giảm dần.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

+ Các dòng có chỉ số chẵn và chỉ số lẻ.

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	-87	-83	-73	-42	12	18	38
1	121	46	37	25	24	7	-87
2	-96	-82	-38	-26	-11	42	79
3	83	82	78	74	38	23	8
4	12	35	39	42	52	92	99

– Khai báo hàm.

```
01386.  void SapDongTang(float [][100],int,int,int);
```

```
01387.  void SapDongGiam(float [][100],int,int,int);
01388.  void SapXep(float [][100],int,int);
```

– Định nghĩa hàm sắp dòng tăng đệ quy.

```
01389.  void SapDongTang(float a[][100],int m,int n,
01390.                   int d)
01391.  {
01392.     if(n==1)
01393.        return;
01394.     for(int j=0;j<=n-2;j++)
01395.        if(a[d][j] > a[d][n-1])
01396.           HoanVi(a[d][j],a[d][n-1]);
01397.     SapDongTang(a,m,n-1,d);
01398.  }
```

– Định nghĩa hàm sắp dòng giảm đệ quy.

```
01399.  void SapDongGiam(float a[][100],int m,int n,
01400.                   int d)
01401.  {
01402.     if(n==1)
01403.        return;
01404.     for(int j=0;j<=n-2;j++)
01405.        if(a[d][j] < a[d][n-1])
01406.           HoanVi(a[d][j],a[d][n-1]);
01407.     SapDongGiam(a,m,n-1,d);
01408.  }
```

– Định nghĩa hàm sắp xếp đệ quy.

```
01409.  void SapXep(int a[][100],int m,int n)
01410.  {
01411.     if(m==1)
01412.        return;
01413.     SapXep(a,m-1,n);
01414.     if((m-1)%2==0)
01415.        SapDongTang(a,m,n,m-1);
01416.     else
01417.        SapDongGiam(a,m,n,m-1);
01418.  }
```

Bài 275.Viết hàm sắp xếp các phần tử trong ma trận theo yêu cầu sau:
- Cột có chỉ số chẵn giảm dần từ trên xuống.
- Cột có chỉ số lẻ tăng dần từ trên xuống (414).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

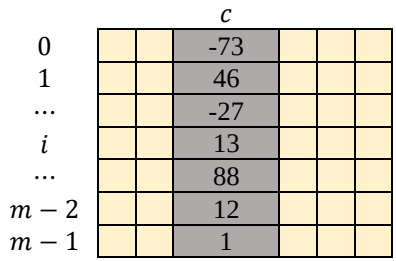
+ Các cột có chỉ số chẵn và chỉ số lẻ.

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	78	23	8	74	38	82	83
4	42	35	92	52	39	12	99

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	79	-82	92	-73	39	-83	99
1	78	23	25	-27	38	12	83
2	42	35	8	46	18	24	37
3	12	38	-11	52	7	42	-38
4	-87	121	-42	74	-96	82	-87

– Hình vẽ minh họa.



– Khai báo hàm.

```
01419.  void SapCotTang(float [][100],int,int,int);
01420.  void SapCotGiam(float [][100],int,int,int);
01421.  void SapXep(float [][100],int,int);
```

– Định nghĩa hàm sắp cột tăng đệ quy.

```
01422.  void SapCotTang(float a[][100],int m,int n,
01423.                   int c)
01424.  {
01425.     if(m==1)
01426.        return;
01427.     for(int i=0;i<=m-2;i++)
01428.        if(a[i][c] > a[m-1][c])
01429.              HoanVi(a[i][c],a[m-1][c]);
```

```
01430.    SapCotTang(a,m-1,n,c);
01431. }
```

– Định nghĩa hàm sắp cột giảm đệ quy.

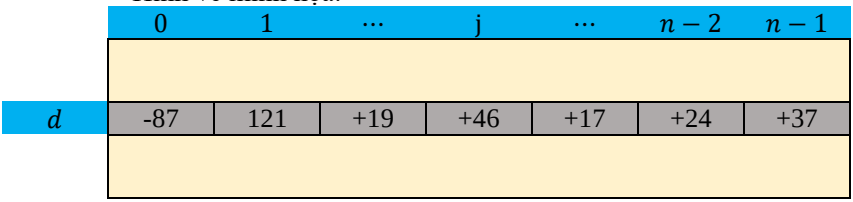
```
01432. void SapCotGiam(float a[][100],int m,int n,
01433.                   int c)
01434. {
01435.    if(m==1)
01436.       return;
01437.    for(int i=0;i<=m-2;i++)
01438.       if(a[i][c] < a[m-1][c])
01439.            HoanVi(a[i][c],a[m-1][c]);
01440.    SapCotGiam(a,m-1,n,c);
01441. }
```

– Định nghĩa hàm sắp xếp đệ quy.

```
01442. void SapXep(int a[][100],int m,int n)
01443. {
01444.    if(n==1)
01445.       return;
01446.    if((n-1)%2==0)
01447.       SapCotGiam(a,m,n,n-1);
01448.    else
01449.       SapCotTang(a,m,n,n-1);
01450. }
```

Bài 276.Hãy sắp xếp các dòng trong ma trận theo tiêu chuẩn sau: dòng có tổng dòng nhỏ hơn nằm ở trên và dòng có tổng dòng lớn hơn bằng nằm ở dưới.

– Hình vẽ minh họa:



– Khai báo hàm.

```
01451. void HoanViDong(float [][100],int,int,int,
01452.                    int);
01453. float TongDong(float [][100],int,int,int,
01454.                   int);
01455. void SapXep(float [][100],int,int);
```

– Định nghĩa hàm hoán vị đệ quy.

```
01456. void HoanViDong(float a[][100],int m,int n,
01457.                 int d1,int d2)
01458. {
01459.    if(n==0)
01460.       return;
01461.    HoanViDong(a,m,n-1,d1,d2);
01462.    HoanVi(a[d1][n-1],a[d2][n-1]);
01463. }
```

– Định nghĩa hàm tổng dòng đệ quy.

```
01464. float TongDong(   float a[][100],int m,int n,
01465.                   int d)
01466. {
01467.    if(n==0)
01468.       return 0;
01469.    return TongDong(a,m,n-1,d) + a[d][n-1];
01470. }
```

– Định nghĩa hàm sắp tăng theo tổng dòng đệ quy.

```
01471. void SapTang(float a[][100],int m,int n)
01472. {
01473.    if(m==1)
01474.       return;
01475.    for(int i=0;i<=m-2;i++)
01476.       if(TongDong(a,m,n,i)>
01477.          TongDong(a,m,n,m-1))
01478.          HoanViDong(a,m,n,i,m-1);
01479.    SapTang(a,m-1,n);
01480. }
```

06.09.08 Kỹ thuật xóa

Bài 277.Xóa một dòng trong ma trận.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	98.23	-82.56	-11.12	-27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4	18.89	56.54	92.3	11.04	67.89

+ Dữ liệu ra:

0	1	2	3	4

0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	79.21	52.54	83.56	43.21	23.67
3	18.89	56.54	92.3	11.04	67.89

– Khai báo hàm.

```
01481. void XoaDong(float [][100],int &,int,int);
```

– Định nghĩa hàm đệ quy.

```
01482. void XoaDong(float a[][100],int &m,int n,
01483.                 int d)
01484. {
01485.    if(n==0)
01486.    {
01487.       m--;
01488.       return;
01489.    }
01490.    XoaDong(a,m,n-1,d);
01491.    for(int i=d;i<=m-2;i++)
01492.       a[i][n-1] = a[i+1][n-1];
01493. }
```

Bài 278.Xóa một cột trong ma trận.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	98.23	-82.56	-11.12	-27.45	66.89
3	79.21	52.54	83.56	43.21	23.67
4	18.89	56.54	92.3	11.04	67.89

+ Dữ liệu ra:

	0	1	2	3
0	12.28	38.41	-73.21	18.18
1	-5.82	11.48	46.19	7.45
2	98.23	-82.56	-27.45	66.89
3	79.21	52.54	43.21	23.67
4	18.89	56.54	11.04	67.89

– Khai báo hàm.

```
01494. void XoaCot(float [][100],int,int &,int);
```

– Định nghĩa hàm đệ quy.

```
01495. void XoaCot(float a[][100],int m,int &n,
01496.               int c)
01497. {
```

```
01498.     if(m==0)
01499.     {
01500.        n--;
01501.        return;
01502.     }
01503.     XoaCot(a,m-1,n,c);
01504.     for(int j=c;j<=n-2;j++)
01505.        a[m-1][j] = a[m-1][j+1];
01506.  }
```

06.09.09 Kỹ thuật thêm

Bài 279.Thêm một dòng toàn giá trị $+1$ tại vị trí dòng d.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	12.28	38.41	−42.43	−73.21	18.18
3	−5.82	11.48	67.31	46.19	7.45

+ Dữ liệu ra:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	+1	+1	+1	+1	+1
3	12.28	38.41	−42.43	−73.21	18.18
4	−5.82	11.48	67.31	46.19	7.45

– Khai báo hàm.

```
01507.  void ThemDong(int [][100],int &,int,int d);
```

– Định nghĩa hàm đệ quy.

```
01508.  void ThemDong(int a[][100],int &m,int n,
01509.                  int d)
01510.  {
01511.     if(n==0)
01512.     {
01513.        m++;
01514.        return;
01515.     }
01516.     ThemDong(a,m,n-1,d);
01517.     for(int i=m;i>d;i--)
01518.        a[i][n-1] = a[i-1][n-1];
```

```
01519.     a[d][n-1] = 1;
01520.  }
```

Bài 280.Thêm một cột toàn giá trị 0 tại vị trí cột *c*.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	−42.43	−73.21	18.18
1	−5.82	11.48	67.31	46.19	7.45
2	79.21	−82.56	−11.12	−27.45	66.89
3	79.21	52.54	83.56	43.21	23.67

+ Dữ liệu ra:

	0	1	2	3	4	
0	12.28	38.41	0	−42.43	−73.21	18.18
1	−5.82	11.48	0	67.31	46.19	7.45
2	79.21	−82.56	0	−11.12	−27.45	66.89
3	79.21	52.54	0	83.56	43.21	23.67

– Khai báo hàm.

```
01521.  void ThemCot(int [][100],int,int&,int);
```

– Định nghĩa hàm đệ quy.

```
01522.  void ThemCot(int a[][100],int m,int &n,
01523.                  int c)
01524.  {
01525.     if(m==0)
01526.     {
01527.        n++;
01528.        return;
01529.     }
01530.     ThemCot(a,m-1,n,c);
01531.     for(int j=n;j>c;j--)
01532.        a[m-1][j] = a[m-1][j-1];
01533.     a[m-1][c] = 0;
01534.  }
```

Bài 281.Thêm một dòng vào sau dòng cuối cùng của ma trận sao cho mỗi phần tử trên dòng là phần tử lớn nhất của cột mà nó thuộc về.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	-42	-73
1	-87	121	25	46
2	19	-82	-11	-27

+ Dữ liệu ra:

	0	1	2	3
0	12	38	-42	-73
1	-87	121	25	46
2	19	-82	-11	-27
3	19	121	25	46

– Khai báo hàm.

```
01535. float LonNhatCot(float[][100],int,int,int);
01536. void ThemDong(float [][100],int&,int);
```

– Định nghĩa hàm tìm giá trị lớn nhất cột.

```
01537. float LonNhatCot(float a[][100],int m,int n,
01538.                  int c)
01539. {
01540.    if(m==1)
01541.       return a[0][c];
01542.    float lc = LonNhatCot(a,m-1,n,c);
01543.    if(a[m-1][c]>lc)
01544.       lc = a[m-1][c];
01545.    return lc;
01546. }
```

– Định nghĩa hàm thêm dòng.

```
01547. void ThemDong(float a[][100],int &m,int n)
01548. {
01549.    if(n==0)
01550.    {
01551.       m++;
01552.       return;
01553.    }
01554.    ThemDong(a,m,n-1);
01555.    a[m-1][n-1] = LonNhatCot(a,m-1,n,n-1);
01556. }
```

Bài 282. Thêm một cột vào sau cột cuối cùng của ma trận sao cho mỗi phần tử trên cột là phần tử nhỏ nhất của dòng mà nó thuộc về.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	-42	-73
1	-87	121	25	46
2	19	-82	-11	-27

+ Dữ liệu ra:

	0	1	2	3	4
0	12	38	-42	-73	-73
1	-87	121	25	46	-87
2	19	-82	-11	-27	-82

– Khai báo hàm.

```
01557. float NhoNhatDong(float[][100],int,int,int);
01558. void ThemCot(float [][100],int,int&);
```

– Định nghĩa hàm tìm giá trị nhỏ nhất dòng.

```
01559. float NhoNhatDong(float a[][100],int m,int n,
01560.                   int d)
01561. {
01562.    if(n==1)
01563.       return a[d][0];
01564.    float lc = NhoNhatDong(a,m,n-1,d);
01565.    if(a[d][n-1]>lc)
01566.       lc = a[d][n-1];
01567.    return lc;
01568. }
```

– Định nghĩa hàm thêm cột.

```
01569. void ThemCot(float a[][100],int m,int &n)
01570. {
01571.    if(m==0)
01572.    {
01573.       n++;
01574.       return;
01575.    }
01576.    ThemCot(a,m-1,n);
01577.    a[m-1][n-1] = NhoNhatDong(a,m,n-1,m-1);
01578. }
```

Bài 283.Thêm một dòng vào sau dòng cuối cùng của ma trận sao cho mỗi phần tử trên dòng là tổng tất cả các phần tử trên cột mà nó thuộc về.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	-42	-73
1	-87	121	25	46
2	19	-82	-11	-27

+ Dữ liệu ra:

	0	1	2	3
0	12	38	-42	-73

1	-87	121	25	46
2	19	-82	-11	-27
3	-56	77	-28	-54

– Khai báo hàm.

```
01579. float TongCot(float[][100],int,int,int);
01580. void ThemDong(float [][100],int&,int);
```

– Định nghĩa hàm tính tổng dòng.

```
01581. float TongCot(float a[][100],int m,int n,
01582.                int c)
01583. {
01584.    if(m==0)
01585.       return 0;
01586.    return TongCot(a,m-1,n,c) + a[m-1][c];
01587. }
```

– Định nghĩa hàm thêm cột.

```
01588. void ThemDong(float a[][100],int &m,int n)
01589. {
01590.    if(n==0)
01591.    {
01592.       m++;
01593.       return;
01594.    }
01595.    ThemDong(a,m,n-1);
01596.    a[m-1][n-1] = TongCot(a,m-1,n,n-1);
01597. }
```

Bài 284. Thêm một cột vào sau cột cuối cùng của ma trận sao cho mỗi phần tử trên cột là tích tất cả các phần tử trên hàng mà nó thuộc về.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3
0	12	38	-42	-73
1	-87	121	25	46
2	19	-82	-11	-27

+ Dữ liệu ra:

	0	1	2	3	4
0	12	38	-42	-73	1.398.096
1	-87	121	25	46	-12.106.050
2	19	-82	-11	-27	-462.726

– Khai báo hàm.

```
01598. float TichDong(float[][100],int,int,int);
```

```
01599. void ThemCot(float [][100],int,int&);
```

– Định nghĩa hàm tính tích các phần tử trên một cột.

```
01600. float TichDong(float a[][100],int m,int n,
01601.                int d)
01602. {
01603.    if(n==1)
01604.       return a[d][0];
01605.    return TichDong(a,m,n-1,d) * a[d][n-1];
01606. }
```

– Định nghĩa hàm đệ quy.

```
01607. void ThemCot(float a[][100],int m,int &n)
01608. {
01609.    if(m==0)
01610.    {
01611.       n++;
01612.       return;
01613.    }
01614.    ThemCot(a,m-1,n);
01615.    a[m-1][n-1] = TichDong(a,m,n-1,m-1);
01616. }
```

06.09.10 Kỹ thuật xử lý trên ma trận

Bài 285.Hãy biến đổi ma trận bằng cách thay các giá trị âm bằng trị tuyệt đối của nó.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	79.21	-82.56	-11.12	-27.45	66.89

+ Dữ liệu ra:

	0	1	2	3	4
0	12.28	38.41	42.43	73.21	18.18
1	5.82	11.48	67.31	46.19	7.45
2	79.21	82.56	11.12	27.45	66.89

– Khai báo hàm.

```
01617. void BienDoi(float [][100],int,int);
```

– Định nghĩa hàm biến đổi ma trận.

```
01618. void BienDoi(float a[][100],int m,int n)
01619. {
```

```
01620.     if(m==0)
01621.        return;
01622.     BienDoi(a,m-1,n);
01623.     for(int j=0;j<n;j++)
01624.        if(a[m-1][j]<0)
01625.           a[m-1][j] = -a[m-1][j];
01626.  }
```

Bài 286.Hãy biến đổi ma trận bằng cách thay các giá trị bằng giá trị nguyên gần nó nhất.

- Ví dụ:
  - Dữ liệu vào:

	0	1	2	3	4
0	12.28	38.41	-42.43	-73.21	18.18
1	-5.82	11.48	67.31	46.19	7.45
2	79.21	-82.56	-11.12	-27.45	66.89

  - Dữ liệu ra:

	0	1	2	3	4
0	12	38	-42	-73	18
1	-6	11	67	46	7
2	79	83	-11	-27	67

- Khai báo hàm.

```
01627.  void NguyenHoa(float [][100],int,int);
```

- Định nghĩa hàm biến đổi ma trận bằng cách thay các giá trị bằng giá trị nguyên gần nó nhất.

```
01628.  void NguyenHoa(float a[][100], int m,int n)
01629.  {
01630.     if(m==0)
01631.        return;
01632.     NguyenHoa(a,m-1,n);
01633.     for(int j=0;j<n;j++)
01634.        if(a[m-1][j]>0)
01635.           a[m-1][j] = int(a[m-1][j]+0.5);
01636.        else
01637.           a[m-1][j] = int(a[m-1][j]-0.5);
01638.  }
```

Bài 287.Dịch xuống xoay vòng các hàng trong ma trận.

- Ví dụ:
  - Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87

1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	56	47	58	-89	42	-12	76

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	56	47	58	-89	42	-12	76
1	12	38	-42	-73	18	-83	-87
2	-87	121	25	46	7	24	37
3	79	-82	-11	-27	-96	42	-38

– Khai báo hàm.

```
01639.  void DichXuongCot(float [][100],int,int,int);
01640.  void DichXuong(float [][100],int,int);
```

– Định nghĩa hàm dịch xuống một cột.

```
01641.  void DichXuongCot(float a[][100],
01642.                                    int m,int n,int c)
01643.  {
01644.     float temp = a[m-1][c];
01645.     for(int i=m-1;i>=1;i--)
01646.        a[i][c] = a[i-1][c];
01647.     a[0][c] = temp;
01648.  }
```

– Định nghĩa hàm dịch xuống xoay vòng các hàng trong ma trận

```
01649.  void DichXuong(float a[][100],int m,int n)
01650.  {
01651.     if(n==0)
01652.        return;
01653.     DichXuong(a,m,n-1);
01654.     DichXuongCot(a,m,n,n-1);
01655.  }
```

Bài 288.Dịch lên xoay vòng các hàng trong ma trận.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	56	47	58	-89	42	-12	76

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	-87	121	25	46	7	24	37
1	79	-82	-11	-27	-96	42	-38

2	56	47	58	-89	42	-12	76
3	12	38	-42	-73	18	-83	-87

– Khai báo hàm.

```
01656.  void DichLen(float [][100],int,int);
```

– Định nghĩa hàm dịch lên xoay vòng một cột.

```
01657.  void DichLenCot(float a[][100],
01658.                                  int m,int n,int c)
01659.  {
01660.     float temp = a[0][c];
01661.     for(int i=0;i<=m-2;i++)
01662.        a[i][c] = a[i+1][c];
01663.     a[m-1][c] = temp;
01664.  }
```

– Định nghĩa hàm dịch lên xoay vòng các hàng trong ma trận.

```
01665.  void DichLen(float a[][100],int m,int n)
01666.  {
01667.     if(n==0)
01668.        return;
01669.     DichLen(a,m,n-1);
01670.     DichLenCot(a,m,n,n-1);
01671.  }
```

Bài 289.Dịch trái xoay vòng các cột trong ma trận.

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	56	47	58	-89	42	-12	76

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	38	-42	-73	18	-83	-87	12
1	121	25	46	7	24	37	-87
2	-82	-11	-27	-96	42	-38	79
3	47	58	-89	42	-12	76	56

– Khai báo hàm.

```
01672.  void DichTrai(float [][100],int,int);
```

– Định nghĩa hàm dịch trái xoay vòng một dòng trong ma trận.

```
01673.  void DichTraiDong(float a[][100],
01674.                                  int m,int n,int d)
```

```
01675. {
01676.    float temp = a[d][0];
01677.    for(int j=0;j<=n-2;j++)
01678.       a[d][j] = a[d][j+1];
01679.    a[d][n-1] = temp;
01680. }
```

– Định nghĩa hàm dịch trái xoay vòng các cột trong ma trận.

```
01681. void DichTrai(float a[][100],int m,int n)
01682. {
01683.    if(m==0)
01684.       return;
01685.    DichTrai(a,m-1,n);
01686.    DichTraiDong(a,m,n,m-1);
01687. }
```

Bài 290.Dịch phải xoay vòng các cột trong ma trận (397).

– Ví dụ:

+ Dữ liệu vào:

	0	1	2	3	4	5	6
0	12	38	-42	-73	18	-83	-87
1	-87	121	25	46	7	24	37
2	79	-82	-11	-27	-96	42	-38
3	56	47	58	-89	42	-12	76

+ Dữ liệu ra:

	0	1	2	3	4	5	6
0	-87	12	38	-42	-73	18	-83
1	37	-87	121	25	46	7	24
2	-38	79	-82	-11	-27	-96	42
3	76	56	47	58	-89	42	-12

– Khai báo hàm.

```
01688. void DichPhai(float [][100],int,int);
```

– Định nghĩa hàm dịch phải xoay vòng một dòng trong ma trận.

```
01689. void DichPhaiDong(float a[][100],
01690.                                  int m,int n,int d)
01691. {
01692.    float temp = a[d][n-1];
01693.    for(int j=n-1;j>=1;j--)
01694.       a[d][j] = a[d][j-1];
01695.    a[d][0] = temp;
01696. }
```

– Định nghĩa hàm dịch phải xoay vòng các cột trong ma trận.

```
01697.  void DichPhai(float a[][100],int m,int n)
01698.  {
01699.     if(m==0)
01700.        return;
01701.     DichPhai(a,m-1,n);
01702.     DichPhaiDong(a,m,n,m-1);
01703.  }
```