Chương 7: Giới thiệu tổng quan về lập trình

Phần b: Mảng một chiều-Bài tập

Nhập môn lập trình

Trình bày: Nguyễn Sơn Hoàng Quốc
Email: nshquoc@fit.hcmus.edu.vn

Câu 1 – Nhập mảng thực

- Viết hàm nhập vào mảng số thực *n* phần tử.

Câu 1 – Nhập mảng thực

```
1. void Nhap(float a[100], int &n)
2. {
3.     cout << "n = ";
4.     cin >> n;
5.
6.     for (int i = 0; i < n; i++)
7.     {
8.         cout << "a[" << i << "]=";
9.         cin >> a[i];
10.    }
11.}
```

Câu 2 – Xuất mảng tam giác

- Viết các hàm cần thiết xuất mảng các tam giác (chứa các đỉnh).

Câu 2 – Xuất mảng tam giác

```
1. #include <iostream>
2. using namespace std;
3. struct DIEM
4. {
5.    float x;
6.    float y;
7. };

8. struct TAM_GIAC
9. {
10.   DIEM A;
11.   DIEM B;
12.   DIEM C;
13.};
```

Câu 2 – Xuất mảng tam giác

```
1. void Xuat(DIEM D)
2. {
3.    cout << "(" << D.x << "," << D.y << ")";
4. }

5. void Xuat(TAM_GIAC tg)
6. {
7.    cout << "Diem A" << endl;
8.    Xuat(tg.A);
9.    cout << "Diem B" << endl;
10.   Xuat(tg.B);
11.   cout << "Diem C" << endl;
12.   Xuat(tg.C);
13.}
```

Câu 2 – Xuất mảng tam giác

```
1. void Xuat(TAM_GIAC a[100], int n)
2. {
3.   for (int i = 0; i < n; i++)
4.   {
5.       cout<<"Tam giac thu "<<i;
6.       Xuat(a[i]);
7.   }
8. }
```

Câu 3 – Chỉ số nguyên tố đầu tiên

- Viết hàm tìm chỉ số của phần tử nguyên tố đầu tiên trong mảng nguyên. Trả về -1 nếu không tìm thấy.

Câu 3 – Chỉ số nguyên tố đầu tiên

```
1. bool LaNguyenTo(int x)
2. {
3.    bool laNguyenTo = true;
4.
5.    for (int i = 2; i < x - 1; i++)
6.         if (x % i == 0)
7.              laNguyenTo = false;
8.
9.    return laNguyenTo;
10.}
```

Câu 3 – Chỉ số nguyên tố đầu tiên

```
1. int TimNguyenToDauTien(int a[100], int n)
2. {
3.     int chiSo = -1;
4.
5.     for (int i = 0; i < n; i++)
6.         if (chiSo==-1 && LaNguyenTo(a[i]))
               chiSo = i;
7.     return chiSo;
8. }
```

Câu 4 – Chính phương cuối cùng

- Viết chương trình nhập vào một mảng số nguyên, tìm và xuất giá trị chính phương cuối cùng trong mảng số nguyên. Nếu không có trả về giá trị -1.

Câu 4 – Chính phương cuối cùng

```
1. #include <iostream>
2. using namespace std;

3. void Nhap(int a[100], int &n);
4. bool LaChinhPhuong(int x);
5. int TimChinhPhuong(int a[100], int n);
```

Câu 4 – Chính phương cuối cùng

```
1. void main()
2. {
3.     int a[100];
4.     int n, giaTri;
5.
6.     cout << "Nhap mang = " << endl;
7.     Nhap(a, n);
8.
9.     giaTri = TimChinhPhuong(a, n);
10.
11.    if (giaTri == -1)
12.          cout << "khong co chinh phuong";
13.    else
14.          cout << "Chinh phuong = " << giaTri;
15.}
```

Câu 4 – Chính phương cuối cùng

```
1. void Nhap(int a[100], int &n)
2. {
3.     cout << "n = ";
4.     cin >> n;
5.
6.     for (int i = 0; i < n; i++)
7.     {
8.         cout << "a[" << i << "]=";
9.         cin >> a[i];
10.    }
11.}
```

Câu 4 – Chính phương cuối cùng

```
1. bool LaChinhPhuong(int x)
2. {
3.    bool laChinhPhuong = false;
4.
5.    for (int i = 1; i <= x; i++)
6.         if (i * i == x)
7.              laChinhPhuong = true;
8.
9.    return LaChinhPhuong;
10.}
```

Câu 4 – Chính phương cuối cùng

```
1. int TimChinhPhuong(int a[100], int n)
2. {
3.    int ketQua = -1;
4.
5.    for (int i = 0; i < n; i++)
6.         if (LaChinhPhuong(a[i]))
7.              ketQua = a[i];
8.
9.    return ketQua;
10.}
```

Câu 5 – Có chứa số âm ?

- Viết hàm kiểm tra mảng có chứa số âm hay không

Câu 5 – Có chứa số âm ?

```
1. bool ChuaSoAm(float a[], int n)
2. {
3.    bool chuaSoAm = false;
4.
5.    for (int i = 0; i < n; i++)
6.         if (a[i] < 0)
7.              chuaSoAm = true;
8.
9.    return chuaSoAm;
10.}
```

Câu 6 – Kiểm tra tăng giảm?

- Viết hàm kiểm tra mảng có tăng và giảm nghiêm ngặt liên tục hay không (Ví dụ: mảng tăng giảm nghiêm ngặt liên tục {1, 4, 2, 3, -3, 6}). Ban đầu có thể tăng hoặc giảm tùy ý.

Câu 6 – Tăng giảm nghiêm ngặt?

```
1.  bool KiemTraTangGiam(int a[], int n){
2.      bool ketQua = true;
3.      int tang = 0; // -1:giam; 0:chua xac dinh; 1:tang
4.      for (int i = 1; i < n; i++)      {
5.          if (tang == 0) // chua xac dinh
6.              if (a[i] > a[i - 1]) tang = 1;
7.              else if (a[i] < a[i - 1]) tang = -1;
8.              else ketQua = false;
9.          if (tang == -1 && a[i] >= a[i - 1])
10.             ketQua = false;
11.         if (tang == 1 && a[i] <= a[i - 1])
12.             ketQua = false;
13.         if (tang == 1) tang = -1;
14.         if (tang == -1) tang = 1;
15.     }
16.     return ketQua;
17. }
```

Câu 7: Chia mảng chẵn lẻ

- Viết hàm chia mảng nguyên *a* cho trước thành hai mảng *b* chứa các số chẵn và mảng *c* chứa các số lẻ

Câu 7: Chia mảng chẵn lẻ

```
1.  void ChiaMang(      int a[100], int n,
                        int b[100], int &n1,
                        int c[100], int &n2){
2.      n1 = 0;
3.      n2 = 0;
4.      for (int i = 0; i < n; i++)
5.          if (a[i] % 2 == 0) {
6.              n1 = n1 + 1;
7.              b[n1 - 1] = a[i];
8.          }
9.          else {
10.             n2 = n2 + 1;
11.             c[n2 - 1] = a[i];
12.         }
13. }
```

Câu 8: Gộp mảng không trùng lắp

- Viết hàm gộp hai mảng *a* và *b* cho trước thành mảng *c* gồm các phần tử không trùng lắp nhau. Vị trí các phần tử mảng *a* ở đầu, các phần tử mảng *b* ở cuối.

Câu 8: Gộp mảng không trùng lắp

```
1. bool TrungLap(int a[100], int n, int x)
2. {
3.     bool trungLap = false;
4.
5.     for (int i = 0; i < n; i++)
6.         if (a[i] == x)
7.             trungLap = true;
8.
9.     return trungLap;
10.}
```

Câu 8: Gộp mảng không trùng lắp

```
1. void GopMang(int a[100], int n1,
      int b[100], int n2, int c[100], int &n){
2.    for (int i = 0; i < n1; i++)
3.          if (TrungLap(c, n, a[i]) == false){
4.                n = n + 1;
5.                c[n - 1] = a[i];
6.          }
7.    for (int i = 0; i < n2; i++)
8.          if (TrungLap(c, n, b[i]) == false){
9.                n = n + 1;
10.               c[n - 1] = b[i];
11.         }
12.}
```

Câu 9: Giá trị phân số nhỏ nhất

- Viết hàm tìm giá trị nhỏ nhất của một mảng các phân số.

Câu 9: Giá trị phân số nhỏ nhất

```
1. struct PHAN_SO
2. {
3.     int tu;
4.     int mau;
5. };

6. float LayGiaTri(PHAN_SO ps)
7. {
8.     float giaTri;
9.     giaTri = (float)ps.tu / ps.mau;
10.    return giaTri;
11.}
```

Câu 9: Giá trị phân số nhỏ nhất

```
1. float GiaTriNhoNhat(PHAN_SO a[100], int n)
2. {
3.     float nhoNhat = 999999, giaTri;
4.     for (int i = 0; i < n; i++)
5.     {
6.         giaTri = LayGiaTri(a[i]);
7.         if (giaTri < nhoNhat)
8.             nhoNhat = giaTri;
9.     }
10.    return nhoNhat;
11.}
```

Câu 10: Tam giác diện tích lớn nhất

- Viết hàm tìm chỉ số của tam giác có diện tích lớn nhất trong mảng tam giác.

Câu 10: Tam giác diện tích lớn nhất

```
1. struct DIEM
2. {
3.     float x;
4.     float y;
5. };

6. struct TAM_GIAC
7. {
8.     DIEM A;
9.     DIEM B;
10.    DIEM C;
11.};
```

Câu 10: Tam giác diện tích lớn nhất

```
1. float TinhKhoangCach(DIEM A, DIEM B)
2. {
3.     float khoangCach;
4.
5.     khoangCach=sqrt((A.x-B.y)*(A.x-B.x)+
                       (A.y-B.y)*(A.y-B.y));
6.
7.     return khoangCach;
8. }
```

Câu 10: Tam giác diện tích lớn nhất

```
1. float TinhDienTich(TAM_GIAC tg)
2. {
3.     float dienTich, p, a, b, c;
4.
5.     a = TinhKhoangCach(tg.B, tg.C);
6.     b = TinhKhoangCach(tg.A, tg.C);
7.     c = TinhKhoangCach(tg.A, tg.B);
8.
9.     p = (a + b + c) / 2;
10.
11.    dienTich=sqrt(p*(p-a)*(p-b)*(p-c));
12.
13.    return dienTich;
14.}
```

Câu 10: Tam giác diện tích lớn nhất

```
1. int LonNhat(TAM_GIAC a[100], int n)
2. {
3.     int chiSo = -1, lonNhat, dienTich;
4.     for (int i = 0; i < n; i++)
5.     {
6.         float dienTich = TinhDienTich(a[i]);
7.         if (chiSo==-1 || dienTich>lonNhat)
8.         {
9.             chiSo = i;
10.            lonNhat = dienTich;
11.        }
12.    }
13.    return chiSo;
14.}
```

Câu 11: Sắp giảm theo trị tuyệt đối

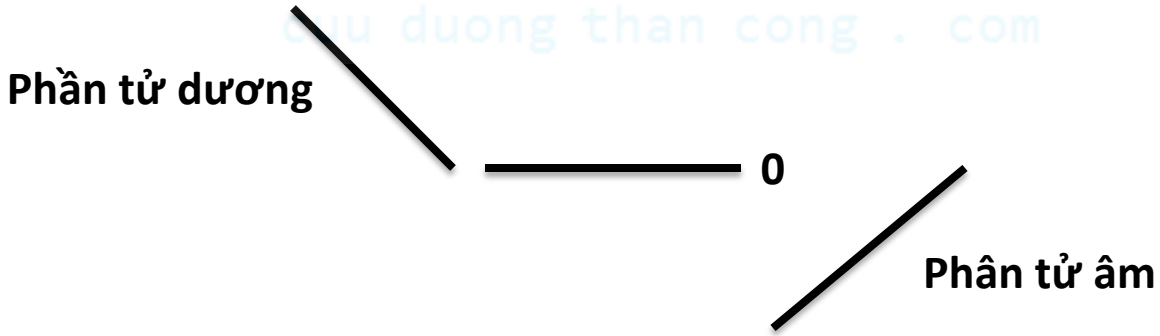
- Viết hàm sắp xếp các phần tử giảm dần theo trị tuyệt đối

Câu 11: Sắp giảm theo trị tuyệt đối

```
1. void SapGiam(int a[100], int n)
2. {
3.    for (int i = 0; i < n - 1; i++)
4.          for (int j = i + 1; j < n; j++)
5.              if (abs(a[i]) < abs(a[j]))
6.                  {
7.                      int tam = a[i];
8.                      a[i] = a[j];
9.                      a[j] = tam;
10.                 }
11.}
```

Câu 12: Sắp theo điều kiện

- Viết hàm sắp các phần tử dương ở đầu mảng giảm dần, phần tử 0 ở giữa, phần tử âm ở cuối mảng tăng dần.



Câu 12: Sắp theo điều kiện

```
1. void SapXep(int a[100], int n)
2. {
3.    for (int i = 0; i < n - 1; i++)
4.          for (int j = i + 1; j < n; j++)
5.                if (CanHoanVi(a[i], a[j]))
6.                {
7.                      int tam = a[i];
8.                      a[i] = a[j];
9.                      a[j] = tam;
10.               }
11.}
```

Câu 12: Sắp theo điều kiện

```
1. bool CanHoanVi(int v1, int v2)
2. {
3.     bool canHoanVi = false;
4.     if (v1 > 0 && v2 > 0 && v1 < v2)
5.         canHoanVi = true;
6.     if (v1<0 && v2<0 && v1>v2)
7.         canHoanVi = true;
8.     if (v1 <= 0 && v2>0)
9.         canHoanVi = true;
10.    if (v1<0 && v2 >= 0)
11.        canHoanVi = true;
12.    return canHoanVi;
13.}
```