ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

BÙI NGỌC THIÊN THANH - 23521436

MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
CLASSWORK 2 AND HOMEWORK 1

OBJECT-ORIENTED PROGRAMMING
BTTL 2 VÀ BTVN 1

CỬ NHÂN NGÀNH KHOA HỌC MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN
CN. NGUYỄN NGỌC QUÍ

TP. HỒ CHÍ MINH, THÁNG 10 NĂM 2024

Mục lục

1. Bài tập 1 - BTVN Lab 1

Để tìm phân số lớn nhất và nhỏ nhất trong một mảng phân số, trước tiên, cần nhập vào số lượng phân số n. Sau đó, lần lượt nhập tử số và mẫu số của từng phân số. Khi mẫu số của bất kỳ phân số nào được nhập bằng 0, người dùng phải nhập lại giá trị hợp lệ cho mẫu số. Sau khi hoàn tất việc nhập liệu, kết quả sẽ xuất ra hai phân số có giá trị nhỏ nhất và lớn nhất theo định dạng: tử số nhỏ nhất / mẫu số nhỏ nhất và tử số lớn nhất / mẫu số lớn nhất. Lưu ý rằng trong bài toán này, không cần thiết phải rút gọn các phân số.

1.1 Mô tả đầu vào và đầu ra

- **Đầu vào:**
  - Một số nguyên n đại diện cho số lượng phân số cần nhập.
  - Sau đó, nhập n phân số với tử số và mẫu số. Mẫu số phải khác 0, nếu không, yêu cầu người dùng nhập lại mẫu số hợp lệ.
- **Đầu ra:**
  - Phân số có giá trị nhỏ nhất và phân số có giá trị lớn nhất (không cần rút gọn), in dưới dạng tử số / mẫu số.

1.2 Ý tưởng

- Sử dụng struct Fraction để lưu trữ các phân số với thuộc tính tử và mẫu.
- Hàm inputFraction() yêu cầu nhập tử số và mẫu số, đồng thời kiểm tra mẫu số hợp lệ (khác 0).
- Các phân số được chuẩn hóa để mẫu số luôn dương bằng cách thay đổi cả tử và mẫu nếu mẫu âm.
- Dùng hàm fractionValue() để so sánh giá trị các phân số bằng cách chia tử số cho mẫu số.
- Duyệt qua mảng phân số để tìm phân số nhỏ nhất và lớn nhất.
- In ra phân số nhỏ nhất và lớn nhất.

1.3 Thực thi code

BTVN 1 - Lab 1

```cpp
#include <iostream>
#include <vector>
using namespace std;

// Struct de luu phan so
struct Fraction {
    int tu;
    int mau;

    // Ham chuan hoa phan so de dam bao mau luon duong
    void normalize() {
        if (mau < 0) {
            tu = -tu;
            mau = -mau;
        }
    }
};

// Ham de so sanh 2 phan so
double fractionValue(const Fraction& frac) {
    return static_cast<double>(frac.tu) / frac.mau;
}

// Ham nhap phan so
Fraction inputFraction() {
    Fraction frac;
    cout << "Nhap tu so: ";
    cin >> frac.tu;
    do {
        cout << "Nhap mau so: ";
        cin >> frac.mau;
        if (frac.mau == 0) {
            cout << "Mau so khong hop le, vui long nhap lai." << endl;
        }
    } while (frac.mau == 0);
    frac.normalize(); // Chuan hoa phan so ngay sau khi nhap
    return frac;
}

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

    vector<Fraction> fractions(n);
    for (int i = 0; i < n; ++i) {
        cout << "Nhap phan so thu " << i + 1 << ": " << endl;
        fractions[i] = inputFraction();
```

```
    }

    // Khoi tao gia tri nho nhat va lon nhat
    Fraction minFrac = fractions[0];
    Fraction maxFrac = fractions[0];

    // Tim phan so nho nhat va lon nhat
    for (int i = 1; i < n; ++i) {
        if (fractionValue(fractions[i]) < fractionValue(minFrac)) {
            minFrac = fractions[i];
        }
        if (fractionValue(fractions[i]) > fractionValue(maxFrac)) {
            maxFrac = fractions[i];
        }
    }

    // In ket qua
    cout << "Phan so nho nhat: " << minFrac.tu << "/" << minFrac.mau << endl;
    cout << "Phan so lon nhat: " << maxFrac.tu << "/" << maxFrac.mau << endl;

    return 0;
}
```

1.4 Kiểm thử

- Testcase 1:

```
PS D:\THANH\HK3\OOP> cd "
Nhap so luong phan so: 3
Nhap phan so thu 1:
Nhap tu so: -1
Nhap mau so: 4
Nhap phan so thu 2:
Nhap tu so: -8
Nhap mau so: -9
Nhap phan so thu 3:
Nhap tu so: 6
Nhap mau so: -7
Phan so nho nhat: -6/7
Phan so lon nhat: 8/9
```

- Testcase 2:

```
PS D:\THANH\HK3\OOP> cd "d:
Nhap so luong phan so: 5
Nhap phan so thu 1:
Nhap tu so: -1
Nhap mau so: 4
Nhap phan so thu 2:
Nhap tu so: -2
Nhap mau so: 3
Nhap phan so thu 3:
Nhap tu so: -5
Nhap mau so: -6
Nhap phan so thu 4:
Nhap tu so: -7
Nhap mau so: 9
Nhap phan so thu 5:
Nhap tu so: 121
Nhap mau so: -24
Phan so nho nhat: -121/24
Phan so lon nhat: 5/6
```

- Testcase 3:

```
PS D:\THANH\HK3\OOP> cd "d:
Nhap so luong phan so: 4
Nhap phan so thu 1:
Nhap tu so: 144
Nhap mau so: -12
Nhap phan so thu 2:
Nhap tu so: -255
Nhap mau so: -15
Nhap phan so thu 3:
Nhap tu so: 2
Nhap mau so: -9
Nhap phan so thu 4:
Nhap tu so: 8
Nhap mau so: 13
Phan so nho nhat: -144/12
Phan so lon nhat: 255/15
```

2. Bài tập 2 - BTVN Lab 1

Để giải bài toán tìm phân số lớn thứ k và bé thứ k trong một mảng gồm n phân số, trước tiên, ta cần nhập số lượng phân số n và số nguyên dương k từ bàn phím. Sau đó, các phân số được nhập vào, có thể nhập trên cùng một dòng hoặc mỗi phân số trên một dòng riêng. Sau khi hoàn thành việc nhập liệu, chương trình sẽ tìm và xuất ra phân số lớn thứ k và phân số bé thứ k trong mảng, theo định dạng: tử số của phân số lớn thứ k / mẫu số của phân số lớn thứ k và tử số của phân số bé thứ k / mẫu số của phân số bé thứ k. Nếu không có phân số lớn thứ k hoặc bé thứ k, chương trình sẽ không xuất ra gì. Lưu ý rằng không cần rút gọn các phân số khi nhập vào.

2.1 Mô tả đầu vào và đầu ra

- **Đầu vào:**
  - Một số nguyên n (số lượng phân số cần nhập)
  - Một số nguyên dương k (chỉ số thứ tự phân số lớn thứ k và bé thứ k).
  - n phân số, mỗi phân số được nhập dưới dạng tử số và mẫu số. Yêu cầu mẫu số phải khác 0, nếu không sẽ yêu cầu nhập lại mẫu số hợp lệ.
- **Đầu ra:**
  - Phân số bé thứ k dưới dạng: tử số / mẫu số.
  - Phân số lớn thứ k dưới dạng: tử số / mẫu số.
  - Nếu k không hợp lệ (lớn hơn số lượng phân số), chương trình không xuất ra gì và in thông báo không có phân số thứ k.

2.2 Ý tưởng

- Struct Fraction: Lưu trữ tử số và mẫu số. Hàm normalize() đảm bảo mẫu số luôn dương.
- Hàm fractionValue(): Tính giá trị thực của phân số để so sánh.
- Hàm inputFraction(): Nhập phân số từ người dùng, kiểm tra mẫu số hợp lệ.

- Hàm compareFractions(): So sánh phân số dựa trên giá trị thực để sắp xếp.
- sort(): Sắp xếp các phân số theo giá trị.
- Xuất kết quả: Nếu k hợp lệ, xuất phân số lớn thứ k và bé thứ k.

2.3 Thực thi code

BTVN 2 - Lab 1

```
    #include <iostream>
#include <vector>
#include <algorithm> // de su dung ham sort
using namespace std;

// Struct de luu phan so
struct Fraction {
    int tu;
    int mau;

    // Ham chuan hoa phan so de dam bao mau so luon duong
    void normalize() {
        if (mau < 0) {
            tu = -tu;
            mau = -mau;
        }
    }
};

// Ham de so sanh hai phan so bang gia tri thuc
double fractionValue(const Fraction& frac) {
    return static_cast<double>(frac.tu) / frac.mau;
}

// Ham nhap phan so
Fraction inputFraction() {
    Fraction frac;
    cout << "Nhap tu so: ";
    cin >> frac.tu;
    do {
        cout << "Nhap mau so: ";
        cin >> frac.mau;
        if (frac.mau == 0) {
            cout << "Mau so khong hop le, vui long nhap lai." << endl;
        }
    } while (frac.mau == 0);
    frac.normalize(); // Chuan hoa phan so ngay sau khi nhap
    return frac;
}

```

```cpp
// Ham so sanh de sap xep cac phan so
bool compareFractions(const Fraction& a, const Fraction& b) {
    return fractionValue(a) < fractionValue(b);
}

int main() {
    int n, k;
    cout << "Nhap so luong phan so n: ";
    cin >> n;
    cout << "Nhap so nguyen duong k: ";
    cin >> k;

    vector<Fraction> fractions(n);
    for (int i = 0; i < n; ++i) {
        cout << "Nhap phan so thu " << i + 1 << ": " << endl;
        fractions[i] = inputFraction();
    }

    // Sap xep cac phan so theo gia tri thuc
    sort(fractions.begin(), fractions.end(), compareFractions);

    // Kiem tra neu k nam trong pham vi hop le
    if (k > 0 && k <= n) {
        // Xuat phan so be thu k
        // k - 1 vi chi so mang bat dau tu 0
        Fraction kthMin = fractions[k - 1];
        cout << "Phan so be thu " << k << ": " << kthMin.tu << "/"
        << kthMin.mau << endl;

        // Xuat phan so lon thu k
        // n - k de lay phan so lon thu k
        Fraction kthMax = fractions[n - k];
        cout << "Phan so lon thu " << k << ": " << kthMax.tu << "/"
        << kthMax.mau << endl;
    } else {
        cout << "Khong co phan so thu " << k << endl;
    }

    return 0;
}
```

2.4 Kiểm thử

- Testcase 1:

```
PS D:\THANH\HK3\OOP> cd "d:
Nhap so luong phan so n: 5
Nhap so nguyen duong k: 3
Nhap phan so thu 1:
Nhap tu so: 19
Nhap mau so: -5
Nhap phan so thu 2:
Nhap tu so: 2
Nhap mau so: 3
Nhap phan so thu 3:
Nhap tu so: -4
Nhap mau so: 37
Nhap phan so thu 4:
Nhap tu so: 4
Nhap mau so: 5
Nhap phan so thu 5:
Nhap tu so: 26
Nhap mau so: 5
Phan so be thu 3: 2/3
Phan so lon thu 3: 2/3
```

- Testcase 2:

```
PS D:\THANH\HK3\OOP> cd "d:
Nhap so luong phan so n: 4
Nhap so nguyen duong k: 2
Nhap phan so thu 1:
Nhap tu so: 4
Nhap mau so: 5
Nhap phan so thu 2:
Nhap tu so: 26
Nhap mau so: 5
Nhap phan so thu 3:
Nhap tu so: 9
Nhap mau so: 17
Nhap phan so thu 4:
Nhap tu so: 8
Nhap mau so: -13
Phan so be thu 2: 9/17
Phan so lon thu 2: 4/5
```

- Testcase 3:

```
PS D:\THANH\HK3\OOP> cd "d:
Nhap so luong phan so n: 6
Nhap so nguyen duong k: 2
Nhap phan so thu 1:
Nhap tu so: 19
Nhap mau so: -5
Nhap phan so thu 2:
Nhap tu so: 2
Nhap mau so: 3
Nhap phan so thu 3:
Nhap tu so: -4
Nhap mau so: 37
Nhap phan so thu 4:
Nhap tu so: 4
Nhap mau so: 5
Nhap phan so thu 5:
Nhap tu so: 9
Nhap mau so: 17
Nhap phan so thu 6:
Nhap tu so: 8
Nhap mau so: -13
Phan so be thu 2: -8/13
Phan so lon thu 2: 2/3
```

- Testcase 4:

```
PS D:\THANH\HK3\OOP> cd "d:\
 ($?) { .\BT2 }
Nhap so luong phan so n: 3
Nhap so nguyen duong k: 2
Nhap phan so thu 1:
Nhap tu so: 2
Nhap mau so: 3
Nhap phan so thu 2:
Nhap tu so: 1
Nhap mau so: -4
Nhap phan so thu 3:
Nhap tu so: -3
Nhap mau so: 8
Phan so be thu 2: -1/4
Phan so lon thu 2: -1/4
```

- Testcase 5:

```
PS D:\THANH\HK3\OOP> cd "d:
Nhap so luong phan so n: 9
Nhap so nguyen duong k: 2
Nhap phan so thu 1:
Nhap tu so: 2
Nhap mau so: 3
Nhap phan so thu 2:
Nhap tu so: -5
Nhap mau so: 9
Nhap phan so thu 3:
Nhap tu so: -6
Nhap mau so: -17
Nhap phan so thu 4:
Nhap tu so: 12
Nhap mau so: 8
Nhap phan so thu 5:
Nhap tu so: 23
Nhap mau so: 19
Nhap phan so thu 6:
Nhap tu so: -1
Nhap mau so: 4
Nhap phan so thu 7:
Nhap tu so: 2
Nhap mau so: 6
Nhap phan so thu 8:
Nhap tu so: 15
Nhap mau so: 8
Nhap phan so thu 9:
Nhap tu so: 21
Nhap mau so: 13
Phan so be thu 2: -1/4
Phan so lon thu 2: 21/13
```

3. Bài tập 3 - BTVN Lab 1

Cho một mảng gồm n phân số, mỗi phân số có dạng a_i / b_i với i từ 1 đến m. Nhiệm vụ của bạn là tìm một tập hợp con các phân số sao cho tích của các phân số trong tập hợp con bằng một phân số đích a_k / b_k đã cho. Trong trường hợp có nhiều tập hợp con thoả mãn điều kiện, bạn cần chọn tập hợp con có số lượng phần tử ít nhất. Nếu không có tập hợp con nào thoả mãn, không xuất ra kết quả nào. Để giải bài toán, bạn cần thực hiện các bước sau: nhập số lượng phân số n, nhập các phân số, và nhập phân số đích a_k / b_k. Sau đó, tìm và xuất tập hợp con có tích bằng phân số đích, sắp xếp các phân số trong tập hợp con theo thứ tự từ bé đến lớn. Lưu ý rằng không cần rút gọn các phân số khi nhập vào.

3.1 Mô tả đầu vào và đầu ra

- **Đầu vào:**
  - Số nguyên n (số lượng phân số cần nhập).
  - n phân số, mỗi phân số gồm tử số và mẫu số, yêu cầu mẫu số khác 0.
  - Một phân số đích a_k / b_k cần tìm tập hợp con có tích bằng phân số này.
- **Đầu ra:**
  - Tập hợp con của các phân số có tích bằng phân số đích, sắp xếp theo thứ tự giá trị thực từ bé đến lớn.
  - Nếu không có tập hợp con nào thỏa mãn, chương trình sẽ không xuất ra gì và in thông báo.

3.2 Ý tưởng

- Struct Fraction: Lưu trữ tử số và mẫu số của phân số. Hàm normalize() đảm bảo mẫu số luôn dương.
- Hàm inputFraction(): Nhập phân số từ người dùng và kiểm tra mẫu số hợp lệ.

- Sử dụng bitmask: Để tìm tất cả các tập hợp con của phân số bằng cách duyệt qua tất cả các trường hợp có thể.
- Tính tích: Đối với mỗi tập hợp con, tính tích của các phân số trong tập hợp.
- Kiểm tra: Nếu tích của tập hợp con bằng phân số đích, lưu tập hợp con đó. Chọn tập hợp con có ít phần tử nhất.
- Sắp xếp và in kết quả: Sắp xếp các phân số trong tập hợp con theo giá trị thực từ bé đến lớn.

3.3 Thực thi code

BTVN 3 - Lab 1

```cpp
#include <bits/stdc++.h>
using namespace std;

// Struct de luu phan so
struct Fraction {
    int tu;
    int mau;

    // Ham chuan hoa phan so de dam bao mau so luon duong
    void normalize() {
        if (mau < 0) {
            tu = -tu;
            mau = -mau;
        }
    }
};

// Ham de so sanh hai phan so bang gia tri thuc
double fractionValue(const Fraction& frac) {
    return static_cast<double>(frac.tu) / frac.mau;
}

// Ham nhap phan so
Fraction inputFraction() {
    Fraction frac;
    cout << "Nhap tu so: ";
    cin >> frac.tu;
    do {
        cout << "Nhap mau so: ";
        cin >> frac.mau;
        if (frac.mau == 0) {
            cout << "Mau so khong hop le, vui long nhap lai." << endl;
        }
    } while (frac.mau == 0);
```

```
35      frac.normalize(); // Chuan hoa phan so ngay sau khi nhap
36      return frac;
37  }
38
39  // Ham de tim tap hop con co tich bang phan so dich
40  bool findSubset(const vector<Fraction>& fractions, const Fraction& target,
41  vector<Fraction>& result) {
42      int n = fractions.size();
43      // So luong phan tu nho nhat trong tap hop con
44      int minSize = numeric_limits<int>::max();
45
46      // Su dung bitmask de tim tat ca cac tap hop con
47      for (int mask = 1; mask < (1 << n); ++mask) {
48          Fraction product = {1, 1}; // Khoi tao tich bang 1/1
49          vector<Fraction> subset;
50
51          for (int i = 0; i < n; ++i) {
52              if (mask & (1 << i)) { // Neu phan tu i nam trong tap hop con
53                  product.tu *= fractions[i].tu;
54                  product.mau *= fractions[i].mau;
55                  subset.push_back(fractions[i]);
56              }
57          }
58
59          // Kiem tra xem tich cua tap hop con co bang voi phan so dich khong
60          // Su dung dieu kien khong can rut gon phan so
61          if (product.tu * target.mau == target.tu * product.mau) {
62              if (subset.size() < minSize) {
63                  minSize = subset.size();
64                  result = subset; // Cap nhat tap hop con
65              }
66          }
67      }
68
69      return minSize != numeric_limits<int>::max();
70  }
71
72
73  int main() {
74      int n;
75      cout << "Nhap so luong phan so n: ";
76      cin >> n;
77
78      vector<Fraction> fractions(n);
79      for (int i = 0; i < n; ++i) {
80          cout << "Nhap phan so thu " << i + 1 << ": " << endl;
81          fractions[i] = inputFraction();
82      }
83
84      // Nhap phan so dich
85      cout << "Nhap phan so dich: " << endl ;
86      Fraction target = inputFraction();
```

```
87
88      vector<Fraction> result;
89      // Tim tap hop con co tich bang phan so dich
90      if (findSubset(fractions, target, result)) {
91          // Sap xep cac phan so trong tap hop con theo thu tu tu be den lon
92          sort(result.begin(), result.end(),
93          [](const Fraction& a, const Fraction& b) {
94              return fractionValue(a) < fractionValue(b);
95          });
96
97          // In ket qua
98          cout << "Tap hop con co tich bang " << target.tu << "/" << target.mau
99          << " la: " << endl;
100         for (const auto& frac : result) {
101             cout << frac.tu << "/" << frac.mau << " ";
102         }
103         cout << endl;
104     } else {
105         cout << "Khong co tap hop con nao thoat man yeu cau." << endl;
106     }
107
108     return 0;
109 }
```

3.4 Kiểm thử

- Testcase 1:

```
PS D:\THANH\HK3\OOP> cd "d:\THANH
Nhap so luong phan so n: 4
Nhap phan so thu 1:
Nhap tu so: 2
Nhap mau so: 4
Nhap phan so thu 2:
Nhap tu so: 3
Nhap mau so: 6
Nhap phan so thu 3:
Nhap tu so: 4
Nhap mau so: 4
Nhap phan so thu 4:
Nhap tu so: 5
Nhap mau so: 5
Nhap phan so dich:
Nhap tu so: 1
Nhap mau so: 1
Tap hop con co tich bang 1/1 la:
4/4
```

- Testcase 2:

```
PS D:\THANH\HK3\OOP> cd "d:\THANH
Nhap so luong phan so n: 5
Nhap phan so thu 1:
Nhap tu so: 1
Nhap mau so: 2
Nhap phan so thu 2:
Nhap tu so: 2
Nhap mau so: 3
Nhap phan so thu 3:
Nhap tu so: 4
Nhap mau so: 6
Nhap phan so thu 4:
Nhap tu so: 1
Nhap mau so: 1
Nhap phan so thu 5:
Nhap tu so: 3
Nhap mau so: 1
Nhap phan so dich:
Nhap tu so: 3
Nhap mau so: 2
Tap hop con co tich bang 3/2 la:
1/2 3/1
```

- Testcase 3:

```
PS D:\THANH\HK3\OOP> cd "d:\THANH\
Nhap so luong phan so n: 6
Nhap phan so thu 1:
Nhap tu so: 1
Nhap mau so: 3
Nhap phan so thu 2:
Nhap tu so: 1
Nhap mau so: 2
Nhap phan so thu 3:
Nhap tu so: 9
Nhap mau so: 4
Nhap phan so thu 4:
Nhap tu so: 3
Nhap mau so: 2
Nhap phan so thu 5:
Nhap tu so: 3
Nhap mau so: 1
Nhap phan so thu 6:
Nhap tu so: 2
Nhap mau so: 4
Nhap phan so dich:
Nhap tu so: 3
Nhap mau so: 4
Tap hop con co tich bang 3/4 la:
1/3 9/4
```

- Testcase 4:

```
PS D:\THANH\HK3\OOP> cd "d:\THANH
Nhap so luong phan so n: 9
Nhap phan so thu 1:
Nhap tu so: 1
Nhap mau so: 3
Nhap phan so thu 2:
Nhap tu so: -5
Nhap mau so: 6
Nhap phan so thu 3:
Nhap tu so: 8
Nhap mau so: -19
Nhap phan so thu 4:
Nhap tu so: -4
Nhap mau so: -5
Nhap phan so thu 5:
Nhap tu so: 13
Nhap mau so: 7
Nhap phan so thu 6:
Nhap tu so: 6
Nhap mau so: -2
Nhap phan so thu 7:
Nhap tu so: 16
Nhap mau so: 8
Nhap phan so thu 8:
Nhap tu so: 27
Nhap mau so: 3
Nhap phan so thu 9:
Nhap tu so: 15
Nhap mau so: 9
Nhap phan so dich:
Nhap tu so: 4
Nhap mau so: 6
Tap hop con co tich bang 4/6 la:
1/3 16/8
```

4. Bài tập 4 - BTVN Lab 1

Arnold's Cat Map là một phép biến đổi ma trận đơn giản nhưng thú vị, có tác dụng chuyển đổi dữ liệu từ một dạng có quy luật thành một dạng có vẻ hỗn độn. Được định nghĩa trên một ma trận vuông kích thước m x m, phép biến đổi Arnold's Cat Map áp dụng công thức để xác định vị trí mới của mỗi phần tử trong ma trận, với (i,j) là chỉ số hàng và cột của phần tử ban đầu. Nhiệm vụ của bài toán là xác định hệ số chu kỳ k của phép biến đổi này, tức là số lần biến đổi cần thiết để ma trận trở về trạng thái ban đầu. Đầu vào của bài toán bao gồm kích thước ma trận và ma trận dữ liệu, và đầu ra là hệ số chu kỳ k. Để giải bài toán, bạn cần áp dụng phép biến đổi Arnold's Cat Map nhiều lần và đếm số lần biến đổi cho đến khi ma trận trở về trạng thái ban đầu.

4.1 Mô tả đầu vào và đầu ra

- **Đầu vào:**
  - Kích thước ma trận: Nhập hai số nguyên m và n (kích thước của ma trận, với m là số dòng và n là số cột).
  - Ma trận nhị phân: Nhập các phần tử của ma trận nhị phân có kích thước m x n, mỗi phần tử là 0 hoặc 1.
- **Đầu ra:**
  - Xuất danh sách các hình chữ nhật có kích thước tối thiểu 2 x 2, với định dạng [x, y, w, h], trong đó: beginitemize
  - x và y là tọa độ gốc (góc trên bên trái) của hình chữ nhật.
  - w và h là chiều rộng và chiều cao của hình chữ nhật.

4.2 Ý tưởng

- Struct matrix: Lưu trữ ma trận nhị phân và thực hiện các phép toán cần thiết, bao gồm nhập ma trận, xây dựng ma trận tổng và tính tổng của các phần tử trong một hình chữ nhật.

- Hàm init(): Nhập giá trị cho ma trận từ đầu vào.
- Hàm build(): Tính tổng của các phần tử trong ma trận để sử dụng cho việc kiểm tra các hình chữ nhật.
- Hàm get_sum(): Tính tổng của các phần tử trong một hình chữ nhật cho trước để kiểm tra xem có phải hình chữ nhật toàn 1 hay không.
- Vòng lặp: Duyệt qua tất cả các điểm trong ma trận, kiểm tra các hình chữ nhật có kích thước tối thiểu 2 x 2 bằng cách sử dụng các chỉ số.
- In kết quả: Nếu tìm thấy hình chữ nhật thỏa mãn điều kiện, in tọa độ và kích thước của hình chữ nhật đó.

4.3 Thực thi code

BTVN 4 - Lab 1

```cpp
#include <iostream>
#include <vector>
using namespace std;

// Class dai dien cho ma tran va phep bien doi Arnold's Cat Map
class Matrix {
private:
    vector<vector<int>> data; // Du lieu cua ma tran
    int size; // Kich thuoc ma tran

public:
    // Constructor de khoi tao ma tran voi kich thuoc n x n
    Matrix(int n) : size(n) {
        data.resize(n, vector<int>(n, 0));
    }

    // Phuong thuc nhap ma tran tu dau vao
    void input() {
        for (int i = 0; i < size; ++i) {
            for (int j = 0; j < size; ++j) {
                cin >> data[i][j]; // Nhap tung phan tu
            }
        }
    }

    // Phuong thuc thuc hien phep bien doi Arnold's Cat Map
    void applyArnoldTransform(Matrix &tempMatrix) {
        for (int i = 0; i < size; ++i) {
            for (int j = 0; j < size; ++j) {
                tempMatrix.data[i][j] = data[(2*i + j) % size][(i + j) % size];
```

```
            }
        }
        *this = tempMatrix;  // Cap nhat trang thai ma tran hien tai
    }

    // Phuong thuc so sanh hai ma tran xem chung co giong nhau khong
    bool isEqual(const Matrix &otherMatrix) const {
        for (int i = 0; i < size; ++i) {
            for (int j = 0; j < size; ++j) {
                if (data[i][j] != otherMatrix.data[i][j]) {
                    return false; // Khong giong nhau
                }
            }
        }
        return true; // Giong nhau
    }
};

int main() {
    ios::sync_with_stdio(false); // Cai dat toc do nhap xuat
    cin.tie(nullptr); // Giai phong luong nho nhap xuat

    int n;
    cin >> n;  // Kich thuoc cua ma tran

    // Khoi tao cac ma tran ban dau, trung gian, va trang thai hien tai
    Matrix initialMatrix(n), currentMatrix(n), tempMatrix(n);

    initialMatrix.input();  // Nhap ma tran ban dau
    currentMatrix = initialMatrix;

    int k = 0;  // Bien dem so lan bien doi
    do {
        ++k;
        currentMatrix.applyArnoldTransform(tempMatrix); // Ap dung phep bien doi
    } while (!currentMatrix.isEqual(initialMatrix));

    // Xuat ra chu ky k
    cout << k << endl;

    return 0; // Tra ve 0 de cho biet chay thanh cong
}
```

4.4 Kiểm thử

- Testcase 1:

```
PS D:\THANH\HK3\OOP>
 ($?) { .\BT4 }
2
1 2
3 4
3
```

- Testcase 2:

```
PS D:\THANH\HK3\OOP\LAB_1_BTVN>
 BT4 } ; if ($?) { .\BT4 }
3
1 2 3
4 5 6
7 8 9
4
```

- Testcase 3:

```
PS D:\THANH\HK3\OOP\LAB_1_BTVN>
 BT4 } ; if ($?) { .\BT4 }
4
16 15 14 13
12 11 10 9
8 7 6 5
4 3 2 1
3
```

- Testcase 4:

```
PS D:\THANH\HK3\OOP>
BT4.cpp -o BT4 } ; if
5
1 2 3 6 8
1 2 5 9 4
5 3 4 2 6
1 8 7 2 6
6 4 8 1 2
10
```

5. Bài tập 5 - BTVN Lab 1

Chúng ta có một bức ảnh nhị phân dưới dạng ma trận kích thước m x n, trong đó mỗi phần tử có giá trị 0 hoặc 1. Bức ảnh này chứa các đối tượng hình chữ nhật, với các hình chữ nhật có gốc tọa độ là góc trên bên trái và có tất cả các phần tử bên trong là 1. Nhiệm vụ của bạn là tìm tất cả các hình chữ nhật trong bức ảnh, với điều kiện kích thước tối thiểu là 2 x 2. Để giải quyết bài toán, bạn cần thực hiện các bước sau: đầu tiên, nhập kích thước ma trận mmm và nnn, sau đó nhập ma trận nhị phân kích thước m x n. Cuối cùng, xác định và xuất danh sách các hình chữ nhật theo định dạng [x, y, w, h], trong đó x và y là tọa độ gốc của hình chữ nhật, và w và h lần lượt là chiều rộng và chiều cao của hình chữ nhật. Danh sách các hình chữ nhật cần được sắp xếp theo thứ tự từ trái sang phải và từ trên xuống dưới

5.1 Mô tả đầu vào và đầu ra

- **Đầu vào:**
  - Kích thước ma trận: Bạn sẽ nhập hai số nguyên, m và n, lần lượt đại diện cho số cột và số hàng của ma trận nhị phân.
  - Ma trận nhị phân: Sau khi nhập kích thước, bạn sẽ nhập m * n số nguyên (chỉ có giá trị 0 hoặc 1), mỗi số đại diện cho một phần tử trong ma trận. Ma trận này sẽ được điền theo thứ tự từ trái qua phải và từ trên xuống dưới.
- **Đầu ra:**
  - Chương trình sẽ in ra tất cả các hình chữ nhật có kích thước tối thiểu là 2 x 2, định dạng là [x, y, w, h] trong đó:
    - x: Tọa độ hàng (0-indexed) của góc trên bên trái của hình chữ nhật.
    - y: Tọa độ cột (0-indexed) của góc trên bên trái của hình chữ nhật.
    - w: Chiều rộng của hình chữ nhật.
    - h: Chiều cao của hình chữ nhật.

5.2 Ý tưởng

- Tạo ma trận tích lũy: Hàm build() tạo ra một ma trận tích lũy để có thể tính tổng số 1 trong bất kỳ hình chữ nhật nào một cách hiệu quả.
- Duyệt qua ma trận: Chương trình duyệt qua tất cả các điểm trong ma trận để xác định các hình chữ nhật:
  - Sử dụng hai vòng lặp lồng nhau để xác định các góc trên bên trái (x, y) và góc dưới bên phải (px, py) của hình chữ nhật.
  - Kiểm tra xem tổng số phần tử trong hình chữ nhật có phải là tất cả là 1 không bằng cách sử dụng hàm get_sum().

5.3 Thực thi code

BTVN 5 - Lab 1

```
#include<bits/stdc++.h>
using namespace std;

// Khai bao kich thuoc ma tran
int n, m;

// Struct de luu ma tran
struct matrix {
    int A[2002][2002]; // Mang 2 chieu

    // Ham khoi tao ma tran
    void init() {
        for(int i = 1; i <= n; i++) {
            for(int j = 1; j <= m; j++) {
                cin >> this->A[i][j]; // Nhap gia tri cho ma tran
            }
        }
    }

    // Ham xay dung ma tran luu tru tong
    void build() {
        for(int i = 1; i <= n + 1; i++) {
            for(int j = 1; j <= m + 1; j++) {
                this->A[i][j] += this->A[i - 1][j] + this->A[i][j - 1] -
                this->A[i - 1][j - 1]; // Tinh tong
            }
        }
    }

    // Ham lay tong cac phan tu trong hinh chu nhat
```

```
    int get_sum(int a, int b, int c, int d) {
        return this->A[c][d] - (a - 1 <= 0 ? 0 : this->A[a - 1][d]) -
        (b - 1 <= 0 ? 0 : this->A[c][b - 1]) +
        (a - 1 <= 0 || b - 1 <= 0 ? 0 : this->A[a - 1][b - 1]);
    }
};

// Khai bao ma tran A
matrix A;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    // Nhap kich thuoc ma tran
    cin >> m >> n;
    A.init(); // Khoi tao ma tran
    A.build(); // Xay dung ma tran luu tru tong

    // Duyet qua tat ca cac diem trong ma tran
    for(int x = 1; x <= n; x++) {
        for(int y = 1; y <= m; y++) {
            for(int px = x + 1; px <= n; px++) {
                for(int py = y + 1; py <= m; py++) {
                    // Kiem tra xem HCN co phai la HCN toan 1 khong
                    if (A.get_sum(x - 1, y - 1, px + 1, py + 1) ==
                        (px - x + 1) * (py - y + 1) &&
                        A.get_sum(x, y, px, py) == (px - x + 1) * (py - y + 1)){
                        // In ra vi tri va kich thuoc hinh chu nhat
                        cout << "[" << y - 1 << "," << x - 1 << ","
                            << (py - 1) - (y - 1) + 1 << ","
                            << (px - 1) - (x - 1) + 1 << "]" << '\n';
                    }
                }
            }
        }
    }
}
```

5.4 Kiểm thử

- Testcase 1:

```
PS D:\THANH\HK3\OOP>
 BT5.cpp -o BT5 } ;
5 5
1 1 0 0 1
1 1 0 1 1
0 0 0 1 1
1 1 1 1 1
0 0 0 1 1
[0,0,2,2]
```

- Testcase 2:

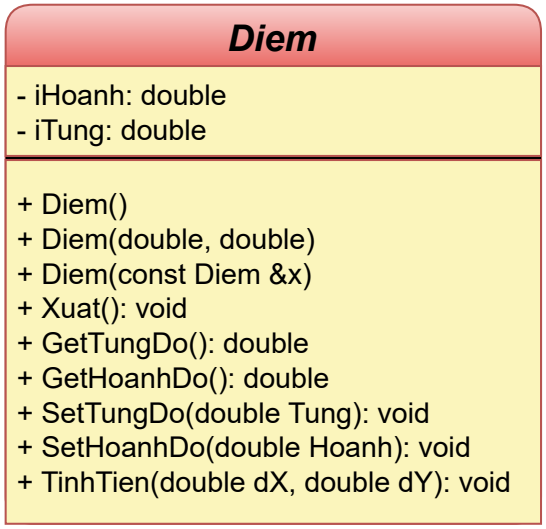
```
PS D:\THANH\HK3\OOP>
10 10
0 0 1 0 0 0 0 0 0 1
0 0 1 0 1 0 0 1 1 0
0 0 0 0 0 0 0 1 1 0
1 1 1 0 0 0 0 0 1 0
1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 1 1 0 0 0
[0,3,3,2]
[5,8,2,2]
```

6. Bài tập 1 - BTTL Lab 2

Xây dựng lớp điểm:

- **Thuộc tính:** iHoanh (Hoành độ), iTung (Tung độ)
- **Phương thức:** Diem(), Diem(int Hoanh, int Tung), Diem(const Diem &x), Xuat(), GetTungDo(), GetTungDo(), SetTungDo(), SetHoanhDo(), TinhTien()

6.1 Biểu đồ lớp



Hình 6.1: *Sơ đồ khối bài 1*

6.2 Mô tả đầu vào và đầu ra

- **Đầu vào:** Dữ liệu là tọa độ của điểm hoặc các thông số để tịnh tiến điểm.
- **Đầu ra:** Tọa độ của điểm được in ra hoặc thay đổi (sau khi cập nhật hoặc tịnh tiến)

6.3 Ý tưởng

- **Diem.h**
  - Định nghĩa lớp Diem với các thuộc tính và khai báo các phương thức cần thiết.
  - Đảm bảo sử dụng guard (#ifndef, #define, #endif) để tránh việc định nghĩa lại lớp khi file bị include nhiều lần.
- **Diem.cpp**
  - Thực hiện định nghĩa cụ thể cho các phương thức của lớp Diem, bao gồm constructor, getter, setter, và các phương thức xử lý khác (ví dụ: TinhTien, Xuat).
- **Main.cpp**
  - Sử dụng lớp Diem để tạo đối tượng, thử nghiệm các phương thức như khởi tạo đối tượng bằng các constructor, cập nhật tọa độ bằng SetHoanhDo, SetTungDo, và in tọa độ ra bằng Xuat.
  - Viết logic để kiểm tra các chức năng như tịnh tiến điểm bằng TinhTien.

6.4 Thực thi code

Link code: Bài 1

- **Diem.h**

```
#ifndef DIEM_H
#define DIEM_H

class Diem {
private:
    double iHoanh;
    double iTung;

public:
    // Constructor mac dinh
    Diem();

    // Constructor co tham so
    Diem(double hoanh, double tung);

    // Constructor sao chep
    Diem(const Diem &x);

```

```
19      // Getter
20      double GetTungDo() const;
21      double GetHoanhDo() const;
22
23      // Setter
24      void SetTungDo(double tung);
25      void SetHoanhDo(double hoanh);
26
27      // Xuat toa do
28      void Xuat() const;
29
30      // Tinh tien toa do
31      void TinhTien(double dX, double dY);
32  };
33
34  #endif
```

- **Diem.cpp**

```
1   #include <iostream>
2   #include "Diem.h"
3
4   using namespace std;
5
6   // Constructor mac dinh
7   Diem::Diem() : iHoanh(0), iTung(0) {}
8
9   // Constructor co tham so
10  Diem::Diem(double hoanh, double tung) : iHoanh(hoanh), iTung(tung) {}
11
12  // Constructor sao chep
13  Diem::Diem(const Diem &x) {
14      iHoanh = x.iHoanh;
15      iTung = x.iTung;
16  }
17
18  // Getter
19  double Diem::GetTungDo() const {
20      return iTung;
21  }
22
23  double Diem::GetHoanhDo() const {
24      return iHoanh;
25  }
26
27  // Setter
28  void Diem::SetTungDo(double tung) {
29      iTung = tung;
30  }
31
32  void Diem::SetHoanhDo(double hoanh) {
```

```
33      iHoanh = hoanh;
34  }
35
36  // Xuat toa do
37  void Diem::Xuat() const {
38      cout << "(" << iHoanh << ", " << iTung << ")" << endl;
39  }
40
41  // Tinh tien toa do
42  void Diem::TinhTien(double dX, double dY) {
43      iHoanh += dX;
44      iTung += dY;
45  }
```

6.5 Kiểm thử

- Testcase1:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai1> g++ Main.cpp Diem.cpp -o run
PS D:\THANH\HK3\OOP\LAB_2\Bai1> ./run
Nhap toa do diem A (hoanh, tung): 1.5 2.5
Nhap toa do diem B (hoanh, tung): 3.0 4.0
Toa do diem A: (1.5, 2.5)
Toa do diem B: (3, 4)
Toa do diem C (sao chep tu diem B): (3, 4)
Nhap toa do moi cho diem A (hoanh, tung): 5.5 7.8
Toa do diem A sau khi set: (5.5, 7.8)
Nhap gia tri tinh tien cho diem B (dX, dY): 1.5 -2.0
Toa do diem B sau khi tinh tien: (4.5, 2)
```

- Testcase2:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai1> g++ Main.cpp Diem.cpp -o run
PS D:\THANH\HK3\OOP\LAB_2\Bai1> ./run
Nhap toa do diem A (hoanh, tung): 2.0 3.0
Nhap toa do diem B (hoanh, tung): -1.0 4.5
Toa do diem A: (2, 3)
Toa do diem B: (-1, 4.5)
Toa do diem C (sao chep tu diem B): (-1, 4.5)
Nhap toa do moi cho diem A (hoanh, tung): 0.0 0.0
Toa do diem A sau khi set: (0, 0)
Nhap gia tri tinh tien cho diem B (dX, dY): 2.5 1.5
Toa do diem B sau khi tinh tien: (1.5, 6)
```

- Testcase3:

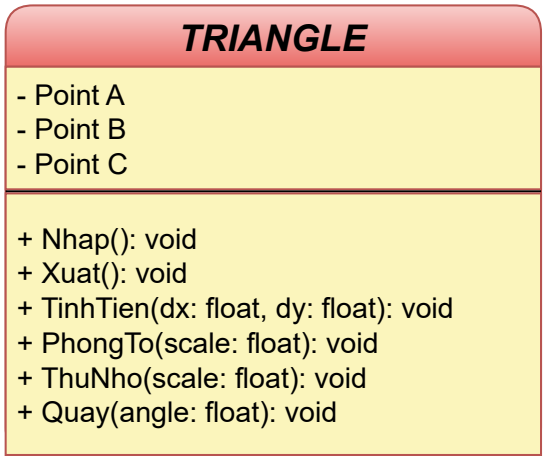
```
PS D:\THANH\HK3\OOP\LAB_2\Bai1> g++ Main.cpp Diem.cpp -o run
PS D:\THANH\HK3\OOP\LAB_2\Bai1> ./run
Nhap toa do diem A (hoanh, tung): 2.0 3.0
Nhap toa do diem B (hoanh, tung): -1.0 4.5
Toa do diem A: (2, 3)
Toa do diem B: (-1, 4.5)
Toa do diem C (sao chep tu diem B): (-1, 4.5)
Nhap toa do moi cho diem A (hoanh, tung): 1.5 6.0
Toa do diem A sau khi set: (1.5, 6)
Nhap gia tri tinh tien cho diem B (dX, dY): 2.4 3.6
Toa do diem B sau khi tinh tien: (1.4, 8.1)
```

7. Bài tập 2 - BTTL Lab 2

Xây dựng lớp tam giác:

- **Thuộc tính:** Đỉnh A, B, C
- **Phương thức:** Nhap(), Xuat(), TinhTien, PhongTo(), ThuNho(), Quay()

7.1 Biểu đồ lớp



Hình 7.2: *Sơ đồ khối bài 2*

7.2 Mô tả đầu vào và đầu ra

- **Đầu vào:** Tọa độ 3 đỉnh của tam giác: A(x1, y1), B(x2, y2), C(x3, y3).
- Tham số cho các thao tác:
  - Tọa độ tịnh tiến: (dx, dy).
  - Tỷ lệ phóng to, thu nhỏ: tiLe.
  - Góc quay: goc (đơn vị: độ).
- **Đầu ra:** Xuất tọa độ của tam giác sau khi thực hiện các thao tác:
  - Ban đầu.

- Sau tịnh tiến.
- Sau phóng to/ thu nhỏ.
- Sau khi quay quanh gốc tọa độ.

7.3 Ý tưởng

- TinhTien(dx, dy): Dịch chuyển cả 3 đỉnh của tam giác theo vector (dx, dy).
- PhongTo(tiLe): Nhân tọa độ các đỉnh với tỷ lệ tiLe để phóng to.
- ThuNho(tiLe): Chia tọa độ các đỉnh cho tiLe để thu nhỏ (hoặc gọi PhongTo với tỷ lệ nghịch).
- Quay(goc): Sử dụng công thức biến đổi tọa độ 2D để quay tam giác quanh gốc tọa độ với góc đã cho.

7.4 Thực thi code

Link code: Bài 2

- **Triangle.h**

```
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include <iostream>
#include <cmath>

// Lop Point dai dien cho toa do diem trong khong gian 2D
class Point {
public:
    float x, y;

    Point();
    Point(float x_val, float y_val);

    void Nhap();
    void Xuat() const;
};

// Lop Triangle dai dien cho tam giac voi cac dinh A, B, C
class Triangle {
private:
    Point A, B, C;
```

```
23
24 public:
25     Triangle();
26     Triangle(Point a, Point b, Point c);
27
28     void Nhap();
29     void Xuat() const;
30
31     void TinhTien(float dx, float dy);
32     void PhongTo(float scale);
33     void ThuNho(float scale);
34     void Quay(float angle);  // Quay quanh goc toa do theo goc angle (radian)
35 };
36
37 #endif
```

- **Triangle.cpp**

```
1  #include "Triangle.h"
2
3  // Implementation cua cac phuong thuc cua lop Point
4  Point::Point() : x(0), y(0) {}
5  Point::Point(float x_val, float y_val) : x(x_val), y(y_val) {}
6
7  void Point::Nhap() {
8      std::cout << "Nhap toa do x: ";
9      std::cin >> x;
10     std::cout << "Nhap toa do y: ";
11     std::cin >> y;
12 }
13
14 void Point::Xuat() const {
15     std::cout << "(" << x << ", " << y << ")";
16 }
17
18 // Implementation cua cac phuong thuc cua lop Triangle
19 Triangle::Triangle() : A(), B(), C() {}
20
21 Triangle::Triangle(Point a, Point b, Point c) : A(a), B(b), C(c) {}
22
23 void Triangle::Nhap() {
24     std::cout << "Nhap toa do dinh A:\n";
25     A.Nhap();
26     std::cout << "Nhap toa do dinh B:\n";
27     B.Nhap();
28     std::cout << "Nhap toa do dinh C:\n";
29     C.Nhap();
30 }
31
32 void Triangle::Xuat() const {
33     std::cout << "Dinh A: "; A.Xuat(); std::cout << "\n";
```

```
34      std::cout << "Dinh B: "; B.Xuat(); std::cout << "\n";
35      std::cout << "Dinh C: "; C.Xuat(); std::cout << "\n";
36  }
37
38  void Triangle::TinhTien(float dx, float dy) {
39      A.x += dx; A.y += dy;
40      B.x += dx; B.y += dy;
41      C.x += dx; C.y += dy;
42  }
43
44  void Triangle::PhongTo(float scale) {
45      A.x *= scale; A.y *= scale;
46      B.x *= scale; B.y *= scale;
47      C.x *= scale; C.y *= scale;
48  }
49
50  void Triangle::ThuNho(float scale) {
51      if (scale != 0) {
52          A.x /= scale; A.y /= scale;
53          B.x /= scale; B.y /= scale;
54          C.x /= scale; C.y /= scale;
55      }
56  }
57
58  void Triangle::Quay(float angle) {
59      float cosA = cos(angle);
60      float sinA = sin(angle);
61
62      // Quay diem A
63      float x_new = A.x * cosA - A.y * sinA;
64      float y_new = A.x * sinA + A.y * cosA;
65      A.x = x_new; A.y = y_new;
66
67      // Quay diem B
68      x_new = B.x * cosA - B.y * sinA;
69      y_new = B.x * sinA + B.y * cosA;
70      B.x = x_new; B.y = y_new;
71
72      // Quay diem C
73      x_new = C.x * cosA - C.y * sinA;
74      y_new = C.x * sinA + C.y * cosA;
75      C.x = x_new; C.y = y_new;
76  }
```

7.5 Kiểm thử

- Testcase 1:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai2> g++ ma
PS D:\THANH\HK3\OOP\LAB_2\Bai2> ./run
Nhap thong tin tam giac:
Nhap toa do dinh A:
Nhap toa do x: 1
Nhap toa do y: 2
Nhap toa do dinh B:
Nhap toa do x: 3
Nhap toa do y: 4
Nhap toa do dinh C:
Nhap toa do x: 5
Nhap toa do y: 6

Tam giac vua nhap:
Dinh A: (1, 2)
Dinh B: (3, 4)
Dinh C: (5, 6)

Nhap gia tri tinh tien (dx, dy): 2 3

Tam giac sau khi tinh tien:
Dinh A: (3, 5)
Dinh B: (5, 7)
Dinh C: (7, 9)

Nhap he so phong to: 2

Tam giac sau khi phong to:
Dinh A: (6, 10)
Dinh B: (10, 14)
Dinh C: (14, 18)
```

```
Nhap he so thu nho: 2

Tam giac sau khi thu nho:
Dinh A: (3, 5)
Dinh B: (5, 7)
Dinh C: (7, 9)

Nhap goc quay (tinh bang radian): 90

Tam giac sau khi quay:
Dinh A: (-5.8142, 0.441622)
Dinh B: (-8.49834, 1.33347)
Dinh C: (-11.1825, 2.22531)
```

- Testcase 2:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai2> g++ main
PS D:\THANH\HK3\OOP\LAB_2\Bai2> ./run
Nhap thong tin tam giac:
Nhap toa do dinh A:
Nhap toa do x: 0
Nhap toa do y: 0
Nhap toa do dinh B:
Nhap toa do x: 2
Nhap toa do y: 0
Nhap toa do dinh C:
Nhap toa do x: 1
Nhap toa do y: 3

Tam giac vua nhap:
Dinh A: (0, 0)
Dinh B: (2, 0)
Dinh C: (1, 3)

Nhap gia tri tinh tien (dx, dy): -1 2

Tam giac sau khi tinh tien:
Dinh A: (-1, 2)
Dinh B: (1, 2)
Dinh C: (0, 5)
```

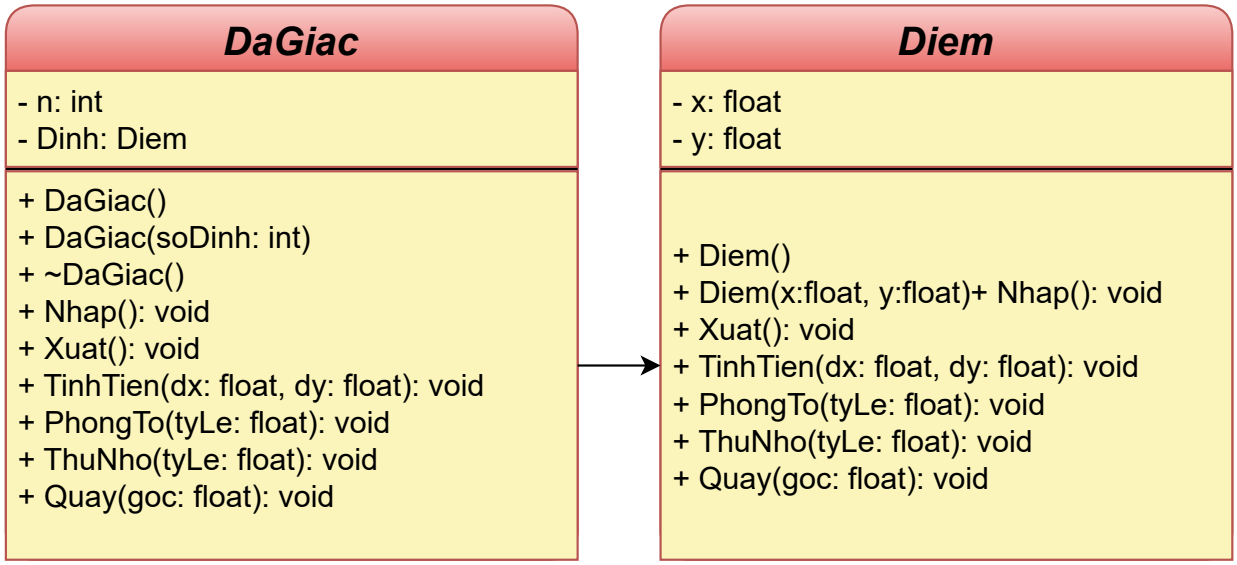
```
Nhap he so phong to: 1.5

Tam giac sau khi phong to:
Dinh A: (-1.5, 3)
Dinh B: (1.5, 3)
Dinh C: (0, 7.5)

Nhap he so thu nho: 1.5

Tam giac sau khi thu nho:
Dinh A: (-1, 2)
Dinh B: (1, 2)
Dinh C: (0, 5)

Nhap goc quay (tinh bang radian): pi /4

Tam giac sau khi quay:
Dinh A: (-1, 2)
Dinh B: (1, 2)
Dinh C: (0, 5)
```

8. Bài tập 3 - BTTL Lab 2

Xây dựng lớp đa giác:

- **Thuộc tính:** n (số đỉnh đa giác), Diem Dinh
- **Phương thức:** Nhap(), Xuat(), TinhTien, PhongTo(), ThuNho(), Quay()

8.1 Biểu đồ lớp



Hình 8.3: *Sơ đồ khối bài 3*

8.2 Mô tả đầu vào và đầu ra

- **Đầu vào:** Người dùng sẽ nhập số lượng đỉnh (n) của đa giác.
- **Đầu ra:** Sau khi thực hiện các phương thức biến đổi, đầu ra sẽ là tọa độ mới của các đỉnh sau khi dịch chuyển, phóng to, thu nhỏ hoặc quay.

8.3 Ý tưởng

- Tạo lớp Diem với các thuộc tính để lưu trữ tọa độ và các phương thức để nhập, xuất, và thực hiện các phép toán hình học như dịch chuyển, phóng to, thu nhỏ và quay.

- Tạo lớp DaGiac để quản lý một mảng các đối tượng Diem. Lớp này sẽ có các phương thức để nhập và xuất thông tin về đa giác cũng như thực hiện các biến đổi cho tất cả các đỉnh.
- Sử dụng toán tử hình học như dịch chuyển, phóng to, thu nhỏ và quay để thay đổi tọa độ của các đỉnh.

8.4 Thực thi code

Link code: Bài 3

- **DaGiac.h**

```
#ifndef DAGIAC_H
#define DAGIAC_H

#include <bits/stdc++.h>
using namespace std;

class Diem {
public:
    float x, y;

    Diem(float x = 0, float y = 0);

    void Nhap();
    void Xuat() const;
    void TinhTien(float dx, float dy);
    void Quay(float alpha);
};


class DaGiac {
private:
    int n;
    Diem* Dinh;

public:
    DaGiac(int soDinh = 3);
    ~DaGiac();

    void Nhap();
    void Xuat() const;
    void TinhTien(float dx, float dy);
    void PhongTo(float k);
    void ThuNho(float k);
    void Quay(float alpha);
};

#endif
```

- **DaGiac.cpp**

```cpp
#include "DaGiac.h"
#include <bits/stdc++.h>


Diem::Diem(float x, float y) : x(x), y(y) {}

void Diem::Nhap() {
    cout << "Nhap toa do (x, y): ";
    cin >> x >> y;
}

void Diem::Xuat() const {
    cout << "(" << x << ", " << y << ")";
}

void Diem::TinhTien(float dx, float dy) {
    x += dx;
    y += dy;
}

void Diem::Quay(float alpha) {
    float x_new = x * cos(alpha) - y * sin(alpha);
    float y_new = x * sin(alpha) + y * cos(alpha);
    x = x_new;
    y = y_new;
}

DaGiac::DaGiac(int soDinh) : n(soDinh) {
    Dinh = new Diem[n];
}

DaGiac::~DaGiac() {
    delete[] Dinh;
}

void DaGiac::Nhap() {
    cout << "Nhap so dinh da giac: ";
    cin >> n;
    Dinh = new Diem[n];
    for (int i = 0; i < n; ++i) {
        cout << "Nhap dinh thu " << i + 1 << ": ";
        Dinh[i].Nhap();
    }
}

void DaGiac::Xuat() const {
    for (int i = 0; i < n; ++i) {
        Dinh[i].Xuat();
        cout << " ";
    }
```

```
51      cout << '\n';
52  }
53
54  void DaGiac::TinhTien(float dx, float dy) {
55      for (int i = 0; i < n; ++i) {
56          Dinh[i].TinhTien(dx, dy);
57      }
58  }
59
60  void DaGiac::PhongTo(float k) {
61      for (int i = 0; i < n; ++i) {
62          Dinh[i].x *= k;
63          Dinh[i].y *= k;
64      }
65  }
66
67  void DaGiac::ThuNho(float k) {
68      for (int i = 0; i < n; ++i) {
69          Dinh[i].x /= k;
70          Dinh[i].y /= k;
71      }
72  }
73
74  void DaGiac::Quay(float alpha) {
75      for (int i = 0; i < n; ++i) {
76          Dinh[i].Quay(alpha);
77      }
78  }
```

8.5 Kiểm thử

- Testcase 1:

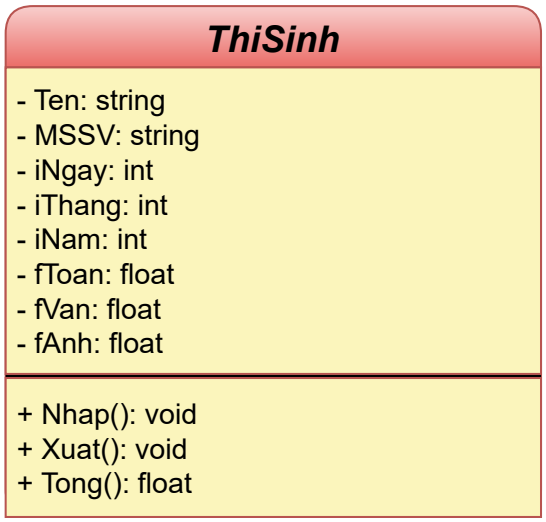
```
PS D:\THANH\HK3\OOP\LAB_2\Bai3> g++ main.cpp DaGiac.cpp -o run
PS D:\THANH\HK3\OOP\LAB_2\Bai3> ./run
Nhap so dinh da giac: 3
Nhap dinh thu 1: Nhap toa do (x, y): 0 0
Nhap dinh thu 2: Nhap toa do (x, y): 1 0
Nhap dinh thu 3: Nhap toa do (x, y): 0 1
Da giac: (0, 0) (1, 0) (0, 1)
Da giac sau khi tinh tien (2, 3): (2, 3) (3, 3) (2, 4)
Da giac sau khi phong to: (4, 6) (6, 6) (4, 8)
Da giac sau khi thu nho: (2, 3) (3, 3) (2, 4)
Da giac sau khi quay 45 do: (-0.707107, 3.53553) (0, 4.24264) (-1.41421, 4.24264)
```

9. Bài tập 4 - BTTL Lab 2

Xây dựng lớp thí sinh:

- **Thuộc tính:** Ten, MSSV, iNgay, iThang, iNam, fToan, fVan, fAnh
- **Phương thức:** Nhap(), Xuat(), Tong()

9.1 Biểu đồ lớp



Hình 9.4: *Sơ đồ khối bài 4*

9.2 Mô tả đầu vào và đầu ra

- **Đầu vào:**
  - Nhập số lượng thí sinh n.
  - Nhập thông tin cho mỗi thí sinh: tên, MSSV, ngày tháng năm sinh, điểm toán, văn, anh.
- **Đầu ra:**
  - In ra thông tin các thí sinh có tổng điểm lớn hơn 15.
  - In ra thông tin thí sinh có điểm cao nhất.

9.3 Ý tưởng

- Tạo lớp ThiSinh với các thuộc tính và phương thức như đã nêu.
- Sử dụng mảng động để lưu trữ thông tin thí sinh.
- Duyệt qua mảng để kiểm tra và in ra thông tin thí sinh theo yêu cầu.

9.4 Thực thi code

Link code: Bài 4

- **ThiSinh.h**

```
#ifndef THISINH_H
#define THISINH_H

#include <string>
using namespace std;

class ThiSinh {
private:
    string Ten;
    string MSSV;
    int iNgay, iThang, iNam;
    float fToan, fVan, fAnh;

public:
    // Nhap thong tin thi sinh
    void Nhap();

    // Xuat thong tin thi sinh
    void Xuat() const;

    // Tinh tong diem
    float Tong() const;

    // Lay MSSV
    string GetMSSV() const;
};

#endif // THISINH_H
```

- **ThiSinh.cpp**

```cpp
#include <iostream>
#include "ThiSinh.h"
using namespace std;

// Nhap thong tin thi sinh
void ThiSinh::Nhap() {
    cout << "Nhap ten: ";
    cin.ignore();
    getline(cin, Ten);
    cout << "Nhap MSSV: ";
    cin >> MSSV;
    cout << "Nhap ngay, thang, nam: ";
    cin >> iNgay >> iThang >> iNam;
    cout << "Nhap diem Toan, Van, Anh: ";
    cin >> fToan >> fVan >> fAnh;
}

// Xuat thong tin thi sinh
void ThiSinh::Xuat() const {
    cout << "Ten: " << Ten << ", MSSV: " << MSSV
            << ", Ngay sinh: " << iNgay << "/" << iThang << "/" << iNam
            << ", Diem Toan: " << fToan << ", Diem Van: " << fVan
            << ", Diem Anh: " << fAnh << ", Tong: " << Tong() << endl;
}

// Tinh tong diem
float ThiSinh::Tong() const {
    return fToan + fVan + fAnh;
}

// Lay MSSV
string ThiSinh::GetMSSV() const {
    return MSSV;
}
```

9.5 Kiểm thử

- Testcase 1:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai4> g++ main.cpp ThiSinh.cpp -o run
PS D:\THANH\HK3\OOP\LAB_2\Bai4> ./run
Nhap so luong thi sinh: 3
Nhap thong tin thi sinh 1:
Nhap ten: Shinichi Kudo
Nhap MSSV: 1109
Nhap ngay, thang, nam: 25 9 2000
Nhap diem Toan, Van, Anh: 10 8 10
Nhap thong tin thi sinh 2:
Nhap ten: Ran Mori
Nhap MSSV: 1234
Nhap ngay, thang, nam: 4 12 2000
Nhap diem Toan, Van, Anh: 9.5 9.5 10
Nhap thong tin thi sinh 3:
Nhap ten: Miyano Shiho
Nhap MSSV: 1436
Nhap ngay, thang, nam: 17 09 1998
Nhap diem Toan, Van, Anh: 10 9.75 10

Thi sinh co tong diem lon hon 15:
Ten: Shinichi Kudo, MSSV: 1109, Ngay sinh: 25/9/2000, Diem Toan: 10, Diem Van: 8, Diem Anh: 10, Tong: 28
Ten: Ran Mori, MSSV: 1234, Ngay sinh: 4/12/2000, Diem Toan: 9.5, Diem Van: 9.5, Diem Anh: 10, Tong: 29
Ten: Miyano Shiho, MSSV: 1436, Ngay sinh: 17/9/1998, Diem Toan: 10, Diem Van: 9.75, Diem Anh: 10, Tong: 29.75

Thi sinh co diem cao nhat:
Ten: Miyano Shiho, MSSV: 1436, Ngay sinh: 17/9/1998, Diem Toan: 10, Diem Van: 9.75, Diem Anh: 10, Tong: 29.75
```

- Testcase 2:

```
Nhap so luong thi sinh: 5
Nhap thong tin thi sinh 1:
Nhap ten: Doremon
Nhap MSSV: 1234
Nhap ngay, thang, nam: 11 1 2010
Nhap diem Toan, Van, Anh: 10 5 7
Nhap thong tin thi sinh 2:
Nhap ten: Nobita
Nhap MSSV: 1785
Nhap ngay, thang, nam: 1 2 2010
Nhap diem Toan, Van, Anh: 5 4 6
Nhap thong tin thi sinh 3:
Nhap ten: Shizuka
Nhap MSSV: 4512
Nhap ngay, thang, nam: 14 2 2010
Nhap diem Toan, Van, Anh: 8.5 9.5 7.78
Nhap thong tin thi sinh 4:
Nhap ten: Chaian
Nhap MSSV: 5112
Nhap ngay, thang, nam: 6 8 2010
Nhap diem Toan, Van, Anh: 4 3.5 5.5
Nhap thong tin thi sinh 5:
Nhap ten: Suneo
Nhap MSSV: 4312
Nhap ngay, thang, nam: 7 6 2010
Nhap diem Toan, Van, Anh: 8
 6.5
9

Thi sinh co tong diem lon hon 15:
Ten: Doremon, MSSV: 1234, Ngay sinh: 11/1/2010, Diem Toan: 10, Diem Van: 5, Diem Anh: 7, Tong: 22
Ten: Shizuka, MSSV: 4512, Ngay sinh: 14/2/2010, Diem Toan: 8.5, Diem Van: 9.5, Diem Anh: 7.78, Tong: 25.78
Ten: Suneo, MSSV: 4312, Ngay sinh: 7/6/2010, Diem Toan: 8, Diem Van: 6.5, Diem Anh: 9, Tong: 23.5

Thi sinh co diem cao nhat:
Ten: Shizuka, MSSV: 4512, Ngay sinh: 14/2/2010, Diem Toan: 8.5, Diem Van: 9.5, Diem Anh: 7.78, Tong: 25.78
```

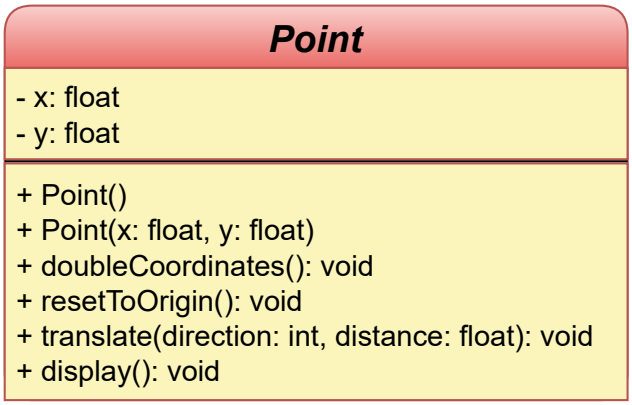
10. Bài tập 5 - BTTL Lab 2

Nhập vào một điểm trong mặt phẳng với hai thành phần là hoành độ và tung độ. Sau đó cho phép người dùng nhập n là số lượng chỉ thị cho chương trình, yêu cầu nhập giá trị các chỉ thị x.

- Nếu x = 1 thì nhân đôi tung độ và hoành độ.
- Nếu x = 2 thì gán điểm về gốc tọa độ.
- Nếu x = 3 thì yêu cầu người dùng nhập hướng tịnh tiến k (k = 0 tịnh tiến theo trục x, k khác 0 tịnh tiến theo trục y) và độ tịnh tiến d.
- Nếu x khác 1,2,3 thì không làm gì cả.

Sau khi thực hiện hết chỉ thị thì thoát chương trình. Cuối cùng là xuất ra thông tin điểm dưới dạng (a,b).

10.1 Biểu đồ lớp



Hình 10.5: *Sơ đồ khối bài 5*

10.2 Mô tả đầu vào và đầu ra

- **Đầu vào:** Một cặp số thực là tọa độ điểm trên mặt phẳng 2D (hoành độ và tung độ).

- Một số nguyên n là số lượng chỉ thị (n = 0).
- n chỉ thị (mỗi chỉ thị là một số nguyên) và các tham số bổ sung tùy theo loại chỉ thị:
  - Nếu chỉ thị là 3, nhập thêm hướng tịnh tiến (0 hoặc 1) và độ tịnh tiến là một số thực.
- **Đầu ra:**Tọa độ điểm cuối cùng dưới dạng (a, b), với a là hoành độ và b là tung độ.

10.3 Ý tưởng

- Bắt đầu với điểm có tọa độ ban đầu nhập từ người dùng.
- Dựa trên từng chỉ thị:
  - Nếu x = 1: Nhân đôi hoành độ và tung độ.
  - Nếu x = 2: Đặt điểm về gốc tọa độ (0, 0).
  - Nếu x = 3: Tịnh tiến điểm theo hướng nhập từ người dùng với khoảng cách d.
  - Nếu x khác 1, 2, 3: Bỏ qua chỉ thị.
- Cuối cùng, xuất ra tọa độ điểm sau khi thực hiện xong tất cả các chỉ thị.

10.4 Thực thi code

Link code: Bài 5

- **Point.h**

```
#ifndef POINT_H
#define POINT_H

class Point {
private:
    float x, y; // Hoanh do va tung do

public:
    Point(); // Constructor mac dinh
    Point(float x, float y); // Constructor co tham so
    void doubleCoordinates(); // Nhan doi tung do va hoanh do
    void resetToOrigin(); // Gan diem ve goc toa do
    void translate(int direction, float distance); // Tinh tien diem
    void display() const; // Xuat ra thong tin diem
};

#endif // POINT_H
```

- **Point.cpp**

```cpp
#include "Point.h"
#include <iostream>
using namespace std;

Point::Point() : x(0), y(0) {} // Constructor mac dinh gan ve (0,0)

Point::Point(float x, float y) : x(x), y(y) {} // Constructor co tham so

void Point::doubleCoordinates() {
    x *= 2;
    y *= 2;
}

void Point::resetToOrigin() {
    x = 0;
    y = 0;
}

void Point::translate(int direction, float distance) {
    if (direction == 0) {
        x += distance; // Tinh tien theo truc x
    } else {
        y += distance; // Tinh tien theo truc y
    }
}

void Point::display() const {
    cout << "(" << x << "," << y << ")" << endl;
}
```

10.5 Kiểm thử

- Testcase 1:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai5>
PS D:\THANH\HK3\OOP\LAB_2\Bai5>
4 6
3
1
2
3 0 6
(6,0)
```

- Testcase 2:

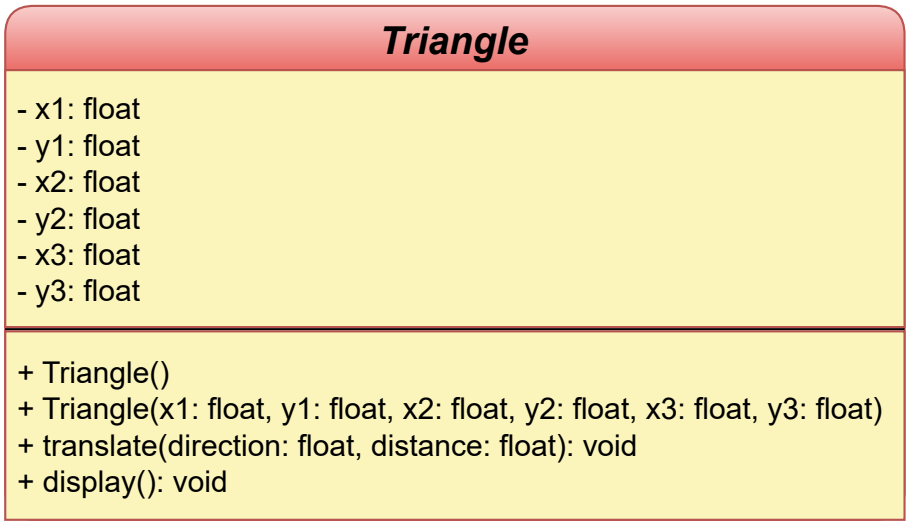
```
PS D:\THANH\HK3\OOP\LAB_2\Bai5>
PS D:\THANH\HK3\OOP\LAB_2\Bai5>
2.5 3.5
2
1
1
(10,14)
```

11. Bài tập 6 - BTTL Lab 2

Class Tam giác

Chúng ta cho người dùng nhập vào tọa độ 3 điểm của tam giác, bao gồm 6 biến (x1, y1, x2, y2, x3, y3) trên cùng một dòng. Sau đó nhập hướng tịnh tiến (đơn vị là độ - gốc tọa độ là trục dương x) và độ dài tịnh tiến. Cuối cùng là xuất ra thông tin 3 điểm của tam giác sau khi được tịnh tiến.
Lưu ý: chọn kiểu dữ liệu cho các điểm của tam giác là float để test case không bị sai do thừa phần thập phân. Chọn PI = 3.14

11.1 Biểu đồ lớp



Hình 11.6: *Sơ đồ khối bài 6*

11.2 Mô tả đầu vào và đầu ra

- **Đầu vào:** Nhập tọa độ 3 điểm của tam giác và hướng tịnh tiến (theo độ) cùng với độ dài tịnh tiến.
- **Đầu ra:** Xuất ra tọa độ của 3 điểm sau khi tịnh tiến.

11.3 Ý tưởng

- Tịnh tiến tam giác theo hướng đã cho, tính toán bằng công thức tịnh tiến sử dụng hàm cos và sin để thay đổi tọa độ theo trục x và y.

11.4 Thực thi code

Link code: Bài 6

- **Triangle.h**

```
#ifndef TRIANGLE_H
#define TRIANGLE_H

class Triangle {
private:
    // Toa do 3 diem cua tam giac
    float x1, y1, x2, y2, x3, y3;

public:
    Triangle(); // Constructor mac dinh
    // Constructor co tham so
    Triangle(float x1, float y1, float x2, float y2, float x3, float y3);
    // Tinh tien tam giac
    void translate(float direction, float distance);
    void display() const;
};

#endif // TRIANGLE_H
```

- **Triangle.cpp**

```
#include "Triangle.h"
#include <iostream>
#include <cmath>
using namespace std;

const float PI = 3.14;

Triangle::Triangle() : x1(0), y1(0), x2(0), y2(0), x3(0), y3(0) {}

Triangle::Triangle(float x1, float y1, float x2, float y2, float x3, float y3)
    : x1(x1), y1(y1), x2(x2), y2(y2), x3(x3), y3(y3) {}

void Triangle::translate(float direction, float distance) {
    // Chuyen doi tu do sang radian
    float rad = direction * PI / 180.0;
```

```
16
17      // Tinh cac gia tri tinh tien theo x y
18      float dx = distance * cos(rad);
19      float dy = distance * sin(rad);
20
21      // Tinh tien ca 3 diem
22      x1 += dx;
23      y1 += dy;
24      x2 += dx;
25      y2 += dy;
26      x3 += dx;
27      y3 += dy;
28  }
29
30  void Triangle::display() const {
31      cout << "(" << x1 << "," << y1 << ")" << endl;
32      cout << "(" << x2 << "," << y2 << ")" << endl;
33      cout << "(" << x3 << "," << y3 << ")" << endl;
34  }
```

11.5 Kiểm thử

- Testcase 1: Tịnh tiến theo trục x

```
PS D:\THANH\HK3\OOP\LAB_2\Bai6> ./run
2 2 4 2 3 3
0 5
(7,2)
(9,2)
(8,3)
```

- Testcase 2: Tịnh tiến theo trục y

```
PS D:\THANH\HK3\OOP\LAB_2\Bai6> ./run
-1 0 -2 2 0 3
90 3
(-0.997611,3)
(-1.99761,5)
(0.00238882,6)
```

- Testcase 3: Tịnh tiến theo góc 60 độ

```
PS D:\THANH\HK3\OOP\LAB_2\Bai6> ./run
1 1 2 1 1 2
60 2
(2.00092,2.73152)
(3.00092,2.73152)
(2.00092,3.73152)
```

- Testcase 4:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai6> g++ ma
PS D:\THANH\HK3\OOP\LAB_2\Bai6> ./run
0 0 1 0 0 1
45 4
(2.82955,2.8273)
(3.82955,2.8273)
```

- Testcase 5:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai6> ./run
0 0 3 0 0 4
45 5
(3.53694,3.53413)
(6.53694,3.53413)
(3.53694,7.53413)
```

- Testcase 6:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai6> ./run
1 1 4 1 1 5
90 2
(1.00159,3)
(4.00159,3)
(1.00159,7)
```

- Testcase 7:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai6> ./run
2 3 5 3 2 6
30 3
(4.59847,4.49931)
(7.59847,4.49931)
(4.59847,7.49931)
```

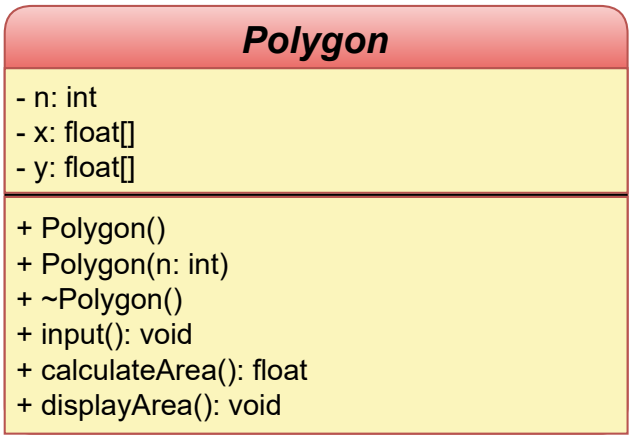
12. Bài tập 7 - BTTL Lab 2

Class Polygon

Nhập vào thông tin của một đa giác bao gồm số điểm n và thông tin n tọa độ trên n dòng. Sau đó xuất ra diện tích của hình này.

Lưu ý: số điểm phải lớn hơn 2, bởi vì 2 điểm trở lên thì mới tạo thành một đa giác có diện tích và đa giác ở đây luôn là đa giác lồi hoặc lõm.

12.1 Biểu đồ lớp



Hình 12.7: *Sơ đồ khối bài 7*

12.2 Mô tả đầu vào và đầu ra

- **Đầu vào:** Nhập số điểm và tọa độ các điểm của đa giác.
- **Đầu ra:** Xuất ra diện tích của đa giác.

12.3 Ý tưởng

- Sử dụng công thức shoelace để tính toán diện tích dựa trên tọa độ các đỉnh của đa giác.

12.4 Thực thi code

Link code: Bài 7

- **Polygon.h**

```h
#ifndef POLYGON_H
#define POLYGON_H

class Polygon {
private:
    int n; // So diem cua da giac
    float* x; // Mang chua hoanh do cac diem
    float* y; // Mang chua tung do cac diem

public:
    Polygon(); // Constructor mac dinh
    Polygon(int n); // Constructor co tham so
    ~Polygon(); // Destructor de giai phong bo nho
    void input(); // Nhap thong tin cac diem cua da giac
    float calculateArea() const; // Tinh dien tich da giac
    void displayArea() const; // Xuat dien tich da giac
};

#endif // POLYGON_H
```

- **Polygon.cpp**

```cpp
#include "Polygon.h"
#include <iostream>
#include <cmath> // Thu vien cho ham fabs (gia tri tuyet doi)
using namespace std;

// Constructor mac dinh
Polygon::Polygon() : n(0), x(nullptr), y(nullptr) {}

// Constructor voi tham so n la so diem
Polygon::Polygon(int n) : n(n) {
    x = new float[n];
    y = new float[n];
}

// Destructor giai phong bo nho
Polygon::~Polygon() {
    delete[] x;
    delete[] y;
}

// Nhap toa do cac diem cua da giac
```

```cpp
void Polygon::input() {
    for (int i = 0; i < n; ++i) {
        cin >> x[i] >> y[i];
    }
}

// Tinh dien tich da giac theo cong thuc shoelace
float Polygon::calculateArea() const {
    float area = 0.0;
    for (int i = 0; i < n - 1; ++i) {
        area += x[i] * y[i + 1] - x[i + 1] * y[i];
    }
    area += x[n - 1] * y[0] - x[0] * y[n - 1];
    return fabs(area) / 2.0; // Lay gia tri tuyet doi va chia doi
}

// Xuat dien tich cua da giac
void Polygon::displayArea() const {
    cout << calculateArea() << endl;
}
```

12.5 Kiểm thử

- Testcase 1:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai7> ./run
3
0 0
0 1
1 0
0.5
```

- Testcase 2:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai7> ./run
4
0 0
0 2
2 2
2 0
4
```

- Testcase 3:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai7> ./run
5
0 0
0 4
3 5
6 4
6 0
27
```

- Testcase 4:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai7> ./run
4
0 0
0 4
4 4
4 0
16
```

- Testcase 5:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai7> ./run
5
1 1
1 3
4 4
4 1
2 2
6
```

- Testcase 6:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai7> ./run
6
0 0
0 2
1 3
3 2
3 0
1 1
6
```

13. Bài tập 8 - BTTL Lab 2

Class List và vấn đề con trỏ

List là một class, kiểu dữ liệu đã được đóng gói sẵn trong C++, list có tác dụng quản lý mảng dễ dàng hơn so với việc khai bóa thông thường, ví dụ muốn xóa phần tử có giá trị 5 trong mảng, ta phải viết một vòng for để tìm; trong khi đó chỉ cần list.pop(5) là xong. List gồm 2 thuộc tính là một con trỏ quản lý mảng (*double) và một biến size (unsigned int) để truy xuất kích thước list.

B1. Chúng ta khởi tạo list rỗng.

B2. Cho người dùng nhập vào chỉ thị n.

- Nếu n = -1 thì nhảy đến B3.
- Nếu n = 0 thì cho người dùng nhập x và thêm x vào list.
- Nếu n = 1 thì cho người dùng nhập x và xóa phần tử đầu tiên có giá trị x ra khỏi list.
- Nếu n = 2 thì cho người dùng nhập x và xóa tất cả phần tử có giá trị x ra khỏi list.
- Nếu n = 3 thì cho người dùng nhập x, y và thay đổi phần tử thứ x bằng y, nếu x không hợp lệ thì không làm gì cả.

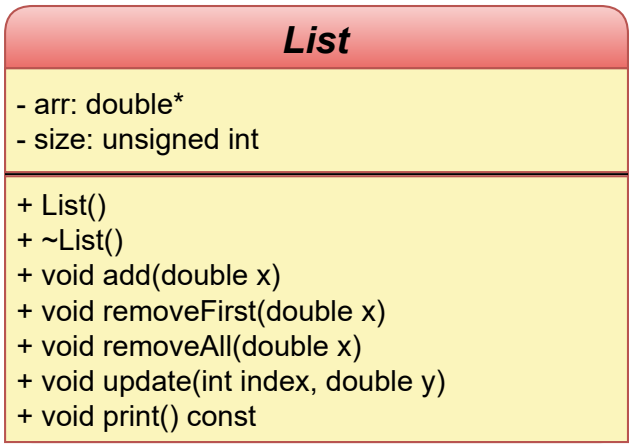
Quay trả lại B2.

B3. In ra màn hình list hiện tại theo mẫu [a,b,c,d,...].

B4. Kết thúc chương trình.

Lưu ý: nên hiểu rõ cách hoạt động của list và sau này chúng ta sẽ sử dụng list rất nhiều.

13.1 Biểu đồ lớp



Hình 13.8: *Sơ đồ khối bài 8*

13.2 Mô tả đầu vào và đầu ra

- **Đầu vào:** Nhập vào một số nguyên cho chỉ thị và tiếp theo là giá trị tương ứng cho các thao tác như đã mô tả.
- **Đầu ra:** Chương trình sẽ in ra danh sách hiện tại theo định dạng [a,b,c,d,...], trong đó a, b, c, và d là các giá trị trong danh sách.

13.3 Ý tưởng

- Khởi tạo danh sách: Khi người dùng khởi động chương trình, danh sách sẽ được khởi tạo rỗng.
- Nhập chỉ thị: Chương trình sẽ yêu cầu người dùng nhập một chỉ thị n và các giá trị tương ứng.
- Thao tác với danh sách:
  - Nếu chỉ thị là 0, một phần tử mới sẽ được thêm vào danh sách.
  - Nếu chỉ thị là 1, chương trình sẽ xóa phần tử đầu tiên có giá trị x trong danh sách.
  - Nếu chỉ thị là 2, tất cả các phần tử có giá trị x sẽ bị xóa khỏi danh sách.

  - Nếu chỉ thị là 3, chương trình sẽ cập nhật phần tử tại chỉ số x thành giá trị y (nếu chỉ số hợp lệ).

- Kết thúc chương trình: Khi người dùng nhập -1, chương trình sẽ in danh sách hiện tại và kết thúc.

13.4 Thực thi code

Link code: Bài 8

- **List.h**

```
#ifndef LIST_H
#define LIST_H

class List {
private:
    double* arr;            // Con tro quan ly mang
    unsigned int size;      // Kich thuoc cua list

public:
    List();                 // Constructor khoi tao list rong
    ~List();                // Destructor de giai phong bo nho

    void add(double x);     // Them phan tu vao list
    void removeFirst(double x); // Xoa phan tu dau tien co gia tri x
    void removeAll(double x);    // Xoa tat ca phan tu co gia tri x
    void update(int index, double y); // Cap nhat phan tu thu index thanh y
    void print() const;     // In ra list
};

#endif // LIST_H
```

- **List.cpp**

```
#include <iostream>
#include <iomanip>
#include "List.h"

List::List() : arr(nullptr), size(0) {}

List::~List() {
    delete[] arr; // Giai phong bo nho
}

void List::add(double x) {
    double* newArr = new double[size + 1]; // Tao mang moi lon hon
```

```
    for (unsigned int i = 0; i < size; ++i) {
        newArr[i] = arr[i]; // Sao chep du lieu cu
    }
    newArr[size] = x; // Them phan tu moi
    delete[] arr;     // Giai phong bo nho cu
    arr = newArr;     // Cap nhat con tro
    size++;           // Tang kich thuoc
}

void List::removeFirst(double x) {
    for (unsigned int i = 0; i < size; ++i) {
        if (arr[i] == x) {
            for (unsigned int j = i; j < size - 1; ++j) {
                arr[j] = arr[j + 1]; // Dich cac phan tu ben phai
            }
            size--; // Giam kich thuoc
            break; // Thoat sau khi da xoa
        }
    }
}

void List::removeAll(double x) {
    unsigned int i = 0;
    while (i < size) {
        if (arr[i] == x) {
            removeFirst(x); // Goi phuong thuc xoa phan tu dau tien
        } else {
            i++;
        }
    }
}

void List::update(int index, double y) {
    if (index >= 0 && index < size) {
        arr[index] = y; // Cap nhat phan tu
    }
}

void List::print() const {
    std::cout << "[";
    for (unsigned int i = 0; i < size; ++i) {
        std::cout << arr[i];
        if (i < size - 1) {
            std::cout << ",";
        }
    }
    std::cout << "]" << std::endl; // Ket thuc in ra
}
```

13.5 Kiểm thử

- Testcase 1: Thêm và Xóa Phần Tử

```
PS D:\THANH\HK3\OOP\LAB_2\Bai8> ./run
0 10
0 20
0 30
1 20
-1
[10,30]
```

- Testcase 2: Xóa Tất Cả Phần Tử và Cập Nhật

```
PS D:\THANH\HK3\OOP\LAB_2\Bai8> ./run
0 5
0 5
0 5
2 5
3 0 10
-1
[]
```

- Testcase 3: Cập Nhật và Thêm Phần Tử

```
PS D:\THANH\HK3\OOP\LAB_2\Bai8> ./run
0 1
0 2
3 1 3
0 4
-1
[1,3,4]
```

- Testcase 4:

```
PS D:\THANH\HK3\OOP\LAB_2\Bai8> ./run
0 1
0 2
0 6
2 2
3 1 4
-1
[1,4]
```