Khoa Công nghệ thông tin
Đại học Khoa học tự nhiên TP HCM

BÁO CÁO ĐỒ ÁN 1

Cấu trúc dữ liệu và giải thuật

1712078 - Ngô Phan Nhật Lâm
1712212 - Lý Thiên Ân

MỤC LỤC

Mức độ hoàn thành 3

Phân chia công việc 4

Mô tả các chức năng và cách thực hiện 5

- Hàm Bool 5
- Làm phép toán trên vector 7
- Làm phép toán trên ma trận 8

Video hướng dẫn sử dụng 9

1 Mức độ hoàn thành

Nội dung	Thành phần	Mức độ hoàn thành
Hàm Bool	Tìm tất cả các công thức đa thức tối thiểu của hàm Bool	100%
Vector	Phép toán cộng hai vector	100%
	Phép toán nhân vector với một số alpha	100%
Ma Trận	Tìm định thức của ma trận	100%
	Nghịch đảo của ma trận	100%
	Tích hai ma trận	100%
	Tìm hạng của ma trận	100%
	Hệ phương trình tuyến tính	100%

2 Phân chia công việc

MSSV	Họ Tên	Công việc thực hiện

1712078	Ngô Phan Nhật Lâm	• Bài 1: Tìm công thức tối tiểu của hàm bool 4 biến:
		• Bài 2: Thực hiện phép toán trên vector:
		  ○ Cộng 2 vector
		  ○ Nhân vector với 1 số alpha

1712212	Lý Thiên Ân	• Bài 3: Thực hiện phép toán trên ma trận:
		  ○ Tìm định thức của ma trận
		  ○ Nghịch đảo của ma trận
		  ○ Tích hai ma trận
		  ○ Tìm hạng của ma trận
		  ○ Hệ phương trình tuyến tính

3 Mô tả các chức năng và cách thực hiện

- Hiển thị menu gồm 4 options cho người dùng chọn:
  - Chọn 1: Tìm đa thức tối tiểu của hàm Bool
  - Chọn 2: Tính toán trên Vector
  - Chọn 3: Tính toán trên Ma trận
  - Chọn -1: Exit

3.1 Tìm đa thức tối tiểu của hàm Bool

- Cho người dùng nhập vào dãy các số nguyên là vị trí của các đơn thức trong hàm bool. Chỉ xét trường hợp 4 biến, các trường hợp khác tương tự.
- Đọc từ màn hình dãy số nguyên, sau đó chuyển sang ma trận nhị phân để thực hiện tìm đa thức tối tiểu.
- Xét từng vị trí trong ma trận nhị phân, nếu bằng 0 thì bỏ qua, nếu bằng 1 thì:
  - Tìm bộ 2 đơn thức, nếu không tìm thấy thì lưu trữ 1 ô trên vào vector, ngược lại:
    - Tìm bộ 4 đơn thức, nếu không tồn tại thì lưu trữ bộ 2 vào vector, ngược lại:
      - Tìm bộ 8 đơn thức
        - Nếu không tồn tại thì ghi nhận bộ 4 phía trên vào vector và thoát ra.
        - Nếu tồn tại thì ghi nhận bộ 8 vào vector lưu trữ và thoát ra.
- Sau khi thực hiện bước trên, ta được 1 vector gồm các bộ đơn thức (1/2/4/8 ô), ta thực hiện sắp xếp các bộ theo kích thước giảm dần, sau đó kiểm tra các bộ này, nếu trùng nhau hoặc chứa trong thì loại bỏ những bộ nhỏ hơn.
- Kết hợp các bộ còn lại với nhau, ta được đa thức tối tiểu của hàm bool.

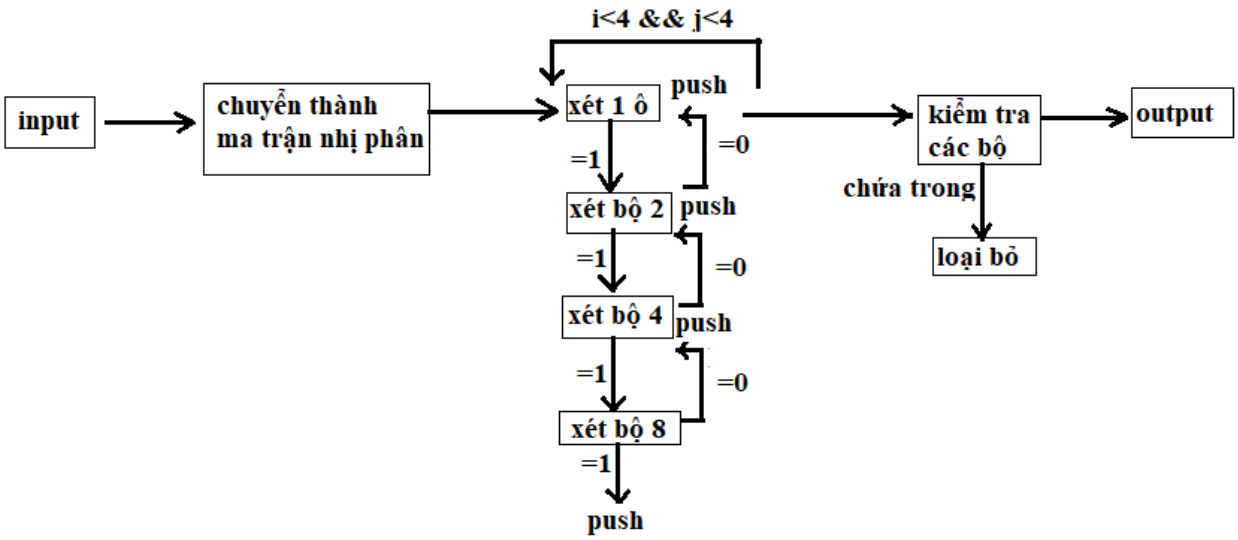
Class KMap thực hiện tạo ma trận nhị phân từ dữ liệu nhập vào:

```
class KMap {
private:
    // Mảng 4x4 lưu trữ các vị trí đã nhập vào
    int matrix[4][4];

    // Đọc line và đưa vào mảng
    int parseToArray(std::string line, int numbers[16]);

public:

    // Constructor
    KMap();

    // Đọc dãy số từ console
    void parseLine(std::string line);

    // Sao chép ma trận
    void getMatrix(int m[4][4]);

    // Vẽ bản đồ Karnaugh
    void drawMap();
};
// Hàm tổng hợp các chức năng
void Bool();
```

Class Minimize thực hiện tìm đa thức tối tiểu dựa trên ma trận nhị phân:

```
// Lưu trữ giá trị của 1 cặp
struct Pairs {
    int i, j;
};

class Minimize
{
private:
    int matrix[4][4];
    KMap* map;
    string rows[4];
    string columns[4];
    // Tìm các bộ đơn thức của hàm Bool
    void pushUnsortedGroups(vector<vector <Pairs>> &groups, int i, int j);
    // Sắp xếp lại các bộ này theo kích thước giảm dần
    void sortUnsortedGroup(vector <Pairs> &group);
    // Xóa những bộ trùng hoặc bị chứa trong
    void removeDuplicateGroups(vector<vector <Pairs>> &groups);
    // Convert thành kết quả
    string generateStringFromGroup(vector <Pairs> &group);
    // In kết quả
    void printGroups(vector<vector <Pairs>> &groups);
    // Kiểm tra 1 bộ có bị chứa trong bởi bộ lớn hơn
    bool isSubsetOf(vector <Pairs> &group1, vector <Pairs> &group2);
    // Kiểm tra bộ trùng
    bool hasCommonPair(Pairs pair, vector <vector <Pairs>> &groups, int skip);
public:
    Minimize(KMap *map);
    ~Minimize(void);
    // Convert thành kiểu string
    string getParsedExpression();
    // Kết hợp các bộ lại thành đa thức tối tiểu
    string getSimplifiedExpression();
};
```

Lưu đồ thuật toán:



Chạy thử chương trình:

```
1. BOOL FUNCTION.
2. VECTOR.
3. MATRIX.
-1. EXIT.
-------------------------
your option: 1
```

```
1. BOOL FUNCTION.
2. VECTOR.
3. MATRIX.
-1. EXIT.
-------------------------
your option: 1
Enter positions of Karnaugh Map:  0 4 5 6 7

1 0 0 0
1 1 1 1
0 0 0 0
0 0 0 0

Input: abcd + abcD + aBcD + ABcD + AbcD

2:   0 4
4:   4 5 7 6

Output: abc + cD
Press any key to continue . . .
```

3.2 Tính toán trên Vector

- Vector được lưu trữ bằng cấu trúc dữ liệu danh sách liên kết đơn.

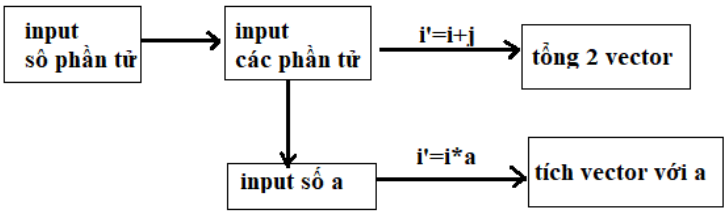
```
struct LinkedList {
    int data;
    struct LinkedList *next;
};
typedef struct LinkedList* node;
```

- Cho phép người dùng nhập vào số lượng phần tử của vector, nếu số lượng phần tử của 2 vector khác nhau thì nhập lại.
- Nhập các phần tử của cả 2 vector.
- Cộng 2 vector bằng cách tạo ra 1 vector mới, mỗi phần tử của vector mới là tổng của phần tử ở vị trí tương ứng của 2 vector đã nhập vào.
- Nhập vào số nguyên alpha. Nhân vector 1 với số alpha bằng cách tạo vector mới, mỗi phần tử của vector mới là tích của phần tử tương ứng trong vector 1 và số alpha.

Các hàm phụ trợ:

```
//Tạo node mới từ data
node CreateNode(int);
// Thêm vào cuối dslk
node AddTail(node, int );
// Lấy node có data=k
int Get(node , int );
// Tìm kiếm node có data=k
int Search(node , int e);
// Duyệt dslk đơn
void Traverser(node);
//Khởi tạo dslk
node InitHead();
// Tính độ dài hiện tại của dslk
int Length(node );
// Hàm cộng 2 vector
node Plus_2_Vector(node , node );
// Hàm nhân vector với alpha
node Multi_Vector(node , int );
// Hàm nhập các phần tử của dslk
node Input(int);
// Hàm tổng hợp các chức năng với vector
void Vector();
```

Lưu đồ thuật toán:



Chạy thử chương trình:

```
1. BOOL FUNCTION.
2. VECTOR.
3. MATRIX.
-1. EXIT.
-------------------------
your option: 2
```

```
1. BOOL FUNCTION.
2. VECTOR.
3. MATRIX.
-1. EXIT.
-------------------------
your option: 2
Enter the number members of vector 1:   4
enter value of vector:  1
enter value of vector:  4
enter value of vector:  2
enter value of vector:  7
Enter the number members of vector 2:   3
Enter the number members of vector 2:   4
enter value of vector:  7
enter value of vector:  5
enter value of vector:  -2
enter value of vector:  11

vector 1:
1  4  2  7
vector 2:
7  5  -2  11
Sum of 2 vectors:
8  9  0  18
enter number:   2

Multi of vector:
2  8  4  14
Press any key to continue . . .
```

3.3 Tính toán trên ma trận

Class MaTrix có các thuộc tính và constructor:

```
class MaTrix
{
    public:
        int n;
        double **a;
    public:
        MaTrix();
        MaTrix(int n);
        MaTrix(int n,double **a);
        MaTrix(const MaTrix &a);
```

class lưu ma trận vuông có kích thước n.

và các giá trị trong ma trận là số thực được lưu trong **a;

- Phương thức:

Phương thức nhập xuất:

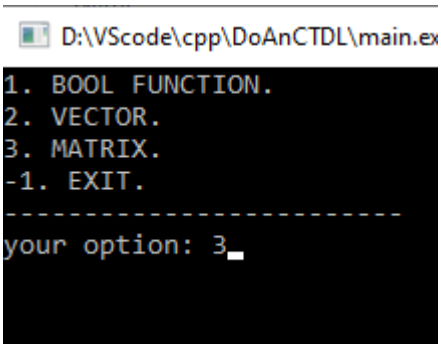
```
  // nhập ma trận
  // argument nếu có sẽ là tên file và sẽ đọc từ file, không có thì mặc định là nhập từ bàn phím
  void input(string filename="");
  // in ma trận
  // argument nếu có sẽ là tên file và sẽ in ra file, không có thì mặc định là in ra màn hình
  void printMaTrix(string filename="");
```

Các phương thức khác:

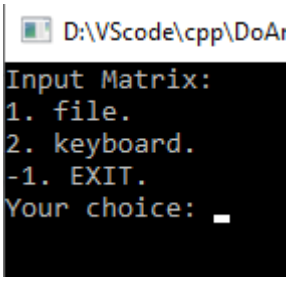
```
MaTrix& operator =(const MaTrix &a);
MaTrix& operator*(const MaTrix &a);

//định thức của ma trận
double det();

//tạo ma trận mới mà dòng I,cột J đã bị xóa đi trong this
MaTrix& deleteIJ(int I,int J);

//định thức của ma trận sau khi xóa dòng i, cột j
double A(int i,int j);

// tạo ra ma trận nghịch đảo
MaTrix& inverse();

//tạo ra ma trận mới mà dòng i, dòng j đổi chổ với nhau
MaTrix& swapIJ(int i,int j);

//tạo ra ma trận mới mà giá trị (dòng j) = (dòng j) + k*(dòng i)
MaTrix& addkItoJ(int I,int J,double k);

//hạng của ma trận
int rank();

//tạo ra ma trận mới mà thay đổi các giá trị cột j thành các giá trị mảng b theo thứ tự
MaTrix& changeJ(int J, double *b);

//giải hệ phương trình tuyến tính
double * solve(int nb,double *B);
```

Chạy thử chương trình:

Từ màn hình chính chọn **3** để vào sử dụng các chức năng xử lý trên ma trận

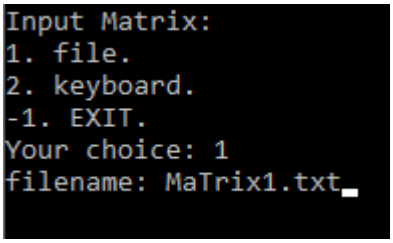


Tại đây bắt đầu nhập ma trận đầu tiên:

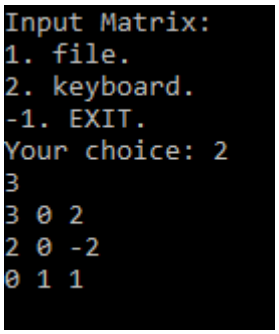


Ở đây có 2 lựa chọn:

1 là đọc từ file:



2 là đọc từ bàn phím:



khi đó màn hình các tính năng sẽ hiện ra:

```
FEATURE:
1. det.
2. inverse.
3. multiply with another Matrix.
4. rank.
5. solve.
-1. EXIT
Your option: _
```

chọn các tính năng mà bạn muốn và được sử lý trên ma trận khi nãy bạn nhập:

1.

```
FEATURE:
1. det.
2. inverse.
3. multiply with another Matrix.
4. rank.
5. solve.
-1. EXIT
Your option: 1
dinh thuc cua ma tran la: 10
Press any key to continue . . . _
```

2.

```
FEATURE:
1. det.
2. inverse.
3. multiply with another Matrix.
4. rank.
5. solve.
-1. EXIT
Your option: 2
nghich dao cua ma tran la:
0.20 0.20 -0.00
-0.20 0.30 1.00
0.20 -0.30 0.00
Press any key to continue . . . _
```

3. nhập thêm 1 ma trận nữa để thực hiện phép nhân: và sau khi nhập xong màn hình kết quả sẽ hiện ra:

```
ket qua khi nhan 2 ma tran:
3.00 0.00 2.00
2.00 0.00 -2.00
0.00 1.00 1.00
va:
0.20 0.20 0.00
-0.20 0.30 1.00
0.20 -0.30 0.00
la:
1.00 0.00 0.00
0.00 1.00 0.00
0.00 0.00 1.00
Press any key to continue . . .
```

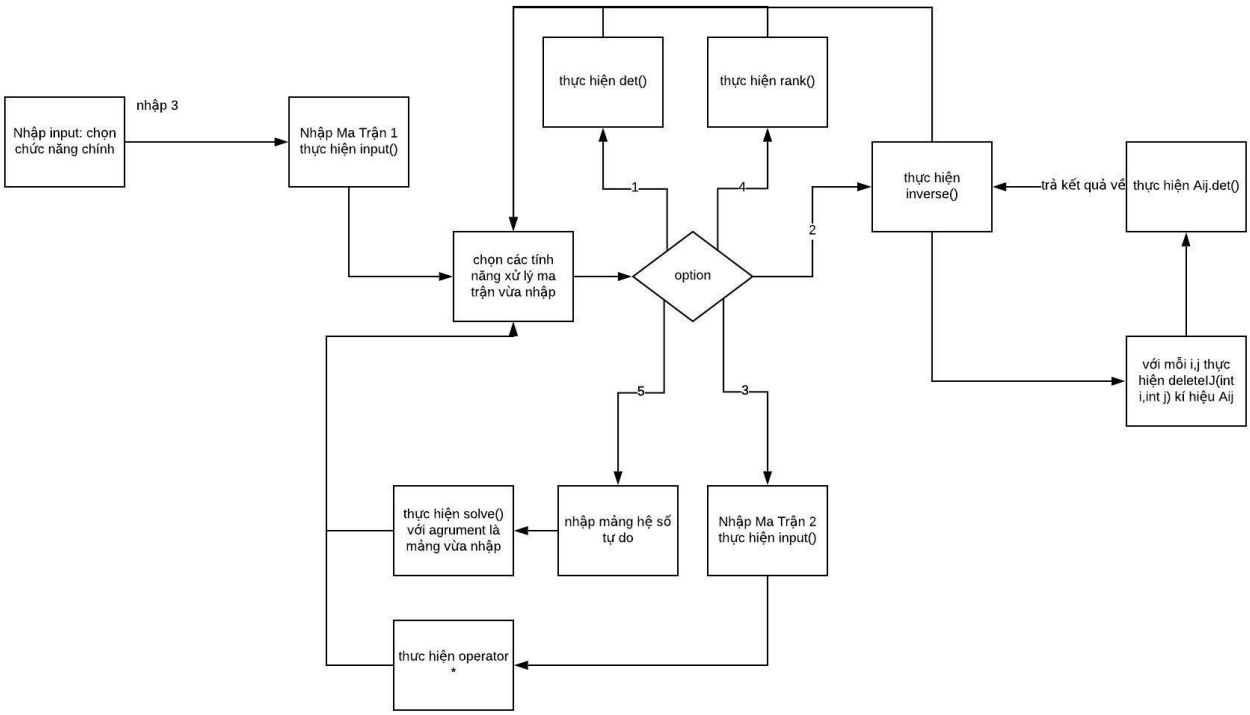
4.

```
FEATURE:
1. det.
2. inverse.
3. multiply with another Matrix.
4. rank.
5. solve.
-1. EXIT
Your option: 4
hang cua ma tran la: 3
Press any key to continue . . .
```

5. nhập thêm mảng hệ số tự do

```
FEATURE:
1. det.
2. inverse.
3. multiply with another Matrix.
4. rank.
5. solve.
-1. EXIT
Your option: 5
nhap mang he so tu do (n so):
11 4 4
he phuong trinh co nghiem duy nhat
Loi giai:
x [ 1 ] = 3
x [ 2 ] = 3
x [ 3 ] = 1
Press any key to continue . . .
```

Lưu đồ Thuật Toán:



4 Video hướng dẫn sử dụng

https://www.youtube.com/watch?v=iAMu0Hsl4uw