# 实验五 类和对象

## 题目1 图形类设计

### 实验描述：

设计三个图形类：Circle（圆）、Rectangle（矩形）、Triangle（三角形）；

1、Cirlce 类基本信息：圆心坐标、半径；

Rectangle 类基本信息：长、宽；

Triangle 类基本信息：三个顶点坐标；

其中：成员变量为 private 属性，成员函数为public 属性；

2、每个图形类有多个构造函数：缺省构造函数、带参数的构造函数；

3、每个图形类有计算图形的面积GetArea()，显示图形的基本信息函数Show()，修改基本信息的函数Set(形参)。以Circle 类为例：通过GetArea()计算圆的面积，Show()函数中显示圆心坐标、直径、周长、面积等基本信息；Set(int x,int y, int r)函数可以修改圆心坐标和半径。

### 实验代码：

/\*

 \* @Author: 王贤义

 \* @FilePath: \code\C+++C\c++\5\1pic.cpp

 \* @Description: 设计三个图形类：Circle（圆）、Rectangle（矩形）、Triangle（三角形）；

1、Cirlce 类基本信息：圆心坐标、半径；

Rectangle 类基本信息：长、宽；

Triangle 类基本信息：三个顶点坐标；

其中：成员变量为 private 属性，成员函数为public 属性；

2、每个图形类有多个构造函数：缺省构造函数、带参数的构造函数；

3、每个图形类有计算图形的面积GetArea()，显示图形的基本信息函数Show()，修改基本信息的函数Set(形参)。以Circle 类为例：通过GetArea()计算圆的面积，Show()函数中显示圆心坐标、直径、周长、面积等基本信息；Set(int x,int y, int r)函数可以修改圆心坐标和半径。

 \*/

#include <iostream>

#include <cmath>

using namespace std;

#define M\_PI 3.14159265358979323846

// 圆形类

class Circle

{

private:

    double x, y; // 圆心坐标

    double r;    // 半径

public:

    Circle()

    {

        x = y = r = 0; // 默认构造函数，将圆心坐标和半径初始化为0

    }

    Circle(double x, double y, double r)

    {

        this->x = x;

        this->y = y;

        this->r = r; // 构造函数，根据给定的圆心坐标和半径进行初始化

    }

    double GetArea()

    {

        return M\_PI \* r \* r; // 计算圆的面积

    }

    void Show()

    {

        cout << "Circle: Center=(" << x << "," << y << "), Radius=" << r << endl;

        cout << "Diameter=" << 2 \* r << ", Circumference=" << 2 \* M\_PI \* r << ", Area=" << GetArea() << endl;

        // 输出圆的信息，包括圆心、半径、直径、周长和面积

    }

    void Set(double x, double y, double r)

    {

        this->x = x;

        this->y = y;

        this->r = r; // 设置圆的圆心坐标和半径

    }

};

// 矩形类

class Rectangle

{

private:

    double length; // 长

    double width;  // 宽

public:

    Rectangle()

    {

        length = width = 0; // 默认构造函数，将长度和宽度初始化为0

    }

    Rectangle(double length, double width)

    {

        this->length = length;

        this->width = width; // 构造函数，根据给定的长度和宽度进行初始化

    }

    double GetArea()

    {

        return length \* width; // 计算矩形的面积

    }

    void Show()

    {

        cout << "Rectangle: Length=" << length << ", Width=" << width << endl;

        cout << "Perimeter=" << 2 \* (length + width) << ", Area=" << GetArea() << endl;

        // 输出矩形的信息，包括长度、宽度、周长和面积

    }

    void Set(double length, double width)

    {

        this->length = length;

        this->width = width; // 设置矩形的长度和宽度

    }

};

// 三角形类

class Triangle

{

private:

    double x1, y1; // 第一个顶点坐标

    double x2, y2; // 第二个顶点坐标

    double x3, y3; // 第三个顶点坐标

public:

    Triangle() {}

    bool isValid(double x1, double y1, double x2, double y2, double x3, double y3)

    {

        if (x1 == x2 && y1 == y2 || x1 == x3 && y1 == y3 || x2 == x3 && y2 == y3)

        {

            cerr << "Error: Invalid triangle!" << endl; // 输出无效三角形的错误信息

            return false;

        }

        return true;

    }

    Triangle(double x1, double y1, double x2, double y2, double x3, double y3)

    {

        if (!isValid(x1, y1, x2, y2, x3, y3))

        {

            return;

        }

        this->x1 = x1;

        this->y1 = y1;

        this->x2 = x2;

        this->y2 = y2;

        this->x3 = x3;

        this->y3 = y3; // 构造函数，根据给定的三个顶点坐标进行初始化

    }

    double GetArea()

    {

        return 0.5 \* abs(x1 \* y2 + x2 \* y3 + x3 \* y1 - x2 \* y1 - x3 \* y2 - x1 \* y3); // 计算三角形的面积

    }

    void Show()

    {

        if (!isValid(x1, y1, x2, y2, x3, y3))

        {

            return;

        }

        cout << "Triangle: (" << x1 << "," << y1 << ") to (" << x2 << "," << y2 << ") to (" << x3 << "," << y3 << ")" << endl;

        cout << "Area=" << GetArea() << endl;

        // 输出三角形的信息，包括三个顶点坐标和面积

    }

    void Set(double x1, double y1, double x2, double y2, double x3, double y3)

    {

        if (!isValid(x1, y1, x2, y2, x3, y3))

        {

            return;

        }

        this->x1 = x1;

        this->y1 = y1;

        this->x2 = x2;

        this->y2 = y2;

        this->x3 = x3;

        this->y3 = y3; // 设置三角形的三个顶点坐标

    }

};

int main()

{

    Circle circle;

    Rectangle rectangle;

    Triangle triangle;

    int choice;

    while (true)

    {

        cout << "Choose the shape to create (1: Circle, 2: Rectangle, 3: Triangle, 0: Exit): ";

        cin >> choice;

        if (choice == 0)

        {

            break;

        }

        switch (choice)

        {

        case 1:

        {

            double x, y, r;

            cout << "Enter the center coordinates (x, y) and radius: ";

            cin >> x >> y >> r;

            circle.Set(x, y, r);

            circle.Show();

            break;

        }

        case 2:

        {

            double length, width;

            cout << "Enter the length and width: ";

            cin >> length >> width;

            rectangle.Set(length, width);

            rectangle.Show();

            break;

        }

        case 3:

        {

            double x1, y1, x2, y2, x3, y3;

            cout << "Enter the coordinates of the three vertices (x1, y1), (x2, y2), (x3, y3): ";

            cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;

            triangle.Set(x1, y1, x2, y2, x3, y3);

            triangle.Show();

            break;

        }

        default:

            cout << "Invalid choice. Please try again." << endl;

            break;

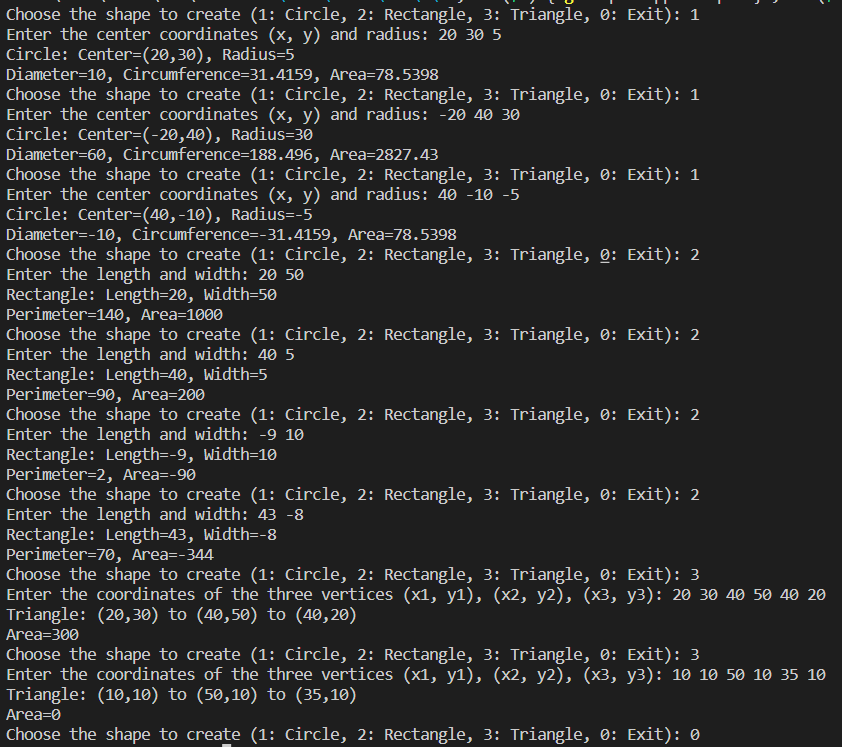
        }

    }

    return 0;

}

### 实验结果：



### 思考与扩展：

无

## 题目2 学生类设计

### 实验描述：

设计一个 Studnet(学生)类

1、基本信息：学号、姓名、性别、出生日期、年级、班级、院系、专业；

其中：基本信息为 private 属性，成员函数为public 属性；

2、Student 类有多个构造函数：缺省构造函数、带参数的构造函数、带默认参数的构造

函数

3、基本功能：

1）使用对象数组保存学生对象；

2）可以从键盘输入学生的基本信息；

3）SetInfo(形参表)函数可以修改学生的基本信息函数，例如：班级、专业等；

4）Show()函数显示学生信息；

### 实验代码：

/\*

 \* @Author: 王贤义

 \* @FilePath: \code\C+++C\c++\5\2student.cpp

 \* @Description: 设计一个 Studnet(学生)类

1、基本信息：学号、姓名、性别、出生日期、年级、班级、院系、专业；

其中：基本信息为 private 属性，成员函数为public 属性；

2、Student 类有多个构造函数：缺省构造函数、带参数的构造函数、带默认参数的构造

函数

3、基本功能：

1）使用对象数组保存学生对象；

2）可以从键盘输入学生的基本信息；

3）SetInfo(形参表)函数可以修改学生的基本信息函数，例如：班级、专业等；

4）Show()函数显示学生信息；

 \*/

#include <iostream>

#include <string>

using namespace std;

class Student

{

private:

    string id;        // 学生ID

    string name;      // 学生姓名

    string gender;    // 学生性别

    string birthdate; // 学生出生日期

    string grade;     // 学生年级

    string cls;       // 学生班级

    string dept;      // 学生所在系别

    string major;     // 学生专业

public:

    Student() {} // 默认构造函数

    // 构造函数，用于初始化学生对象的各个属性

    Student(string id, string name, string gender, string birthdate,

            string grade, string cls, string dept, string major)

        : id(id), name(name), gender(gender), birthdate(birthdate),

          grade(grade), cls(cls), dept(dept), major(major) {}

    // 显示学生信息的方法

    void show()

    {

        cout << "ID:\t\t" << id << endl;             // 输出学生ID

        cout << "Name:\t\t" << name << endl;         // 输出学生姓名

        cout << "Gender:\t\t" << gender << endl;     // 输出学生性别

        cout << "Birthdate:\t" << birthdate << endl; // 输出学生出生日期

        cout << "Grade:\t\t" << grade << endl;       // 输出学生年级

        cout << "Cls:\t\t" << cls << endl;           // 输出学生班级

        cout << "Dept:\t\t" << dept << endl;         // 输出学生所在系别

        cout << "Major:\t\t" << major << endl;       // 输出学生专业

    }

    // 设置学生班级和专业的方法

    void setInfo(string cls, string major)

    {

        this->cls = cls;     // 设置学生班级

        this->major = major; // 设置学生专业

    }

};

int main()

{

    Student students[5]; // 创建一个包含5个学生对象的数组

    for (int i = 0; i < 5; i++)

    {

        cout << "Input NO." << i + 1 << " Student's info " << endl;

        string id, name, gender, birthdate, grade, cls, dept, major;

        cout << "Input ID:\t\t";

        getline(cin, id); // 获取学生ID

        cout << "Input Name:\t\t";

        getline(cin, name); // 获取学生姓名

        cout << "Input Gender:\t\t";

        getline(cin, gender); // 获取学生性别

        cout << "Input Birthdate:\t";

        getline(cin, birthdate); // 获取学生出生日期

        cout << "Input Grade:\t\t";

        getline(cin, grade); // 获取学生年级

        cout << "Input Cls:\t\t";

        getline(cin, cls); // 获取学生班级

        cout << "Input Dept:\t\t";

        getline(cin, dept); // 获取学生所在系别

        cout << "Input Major:\t\t";

        getline(cin, major); // 获取学生专业

        students[i] = Student(id, name, gender, birthdate, grade, cls, dept, major); // 使用获取的信息创建学生对象并将其存储在数组中

    }

    for (int i = 0; i < 5; i++)

    {

        cout << "Showing the NO." << i + 1 << " student's info" << endl;

        students[i].show(); // 显示学生信息

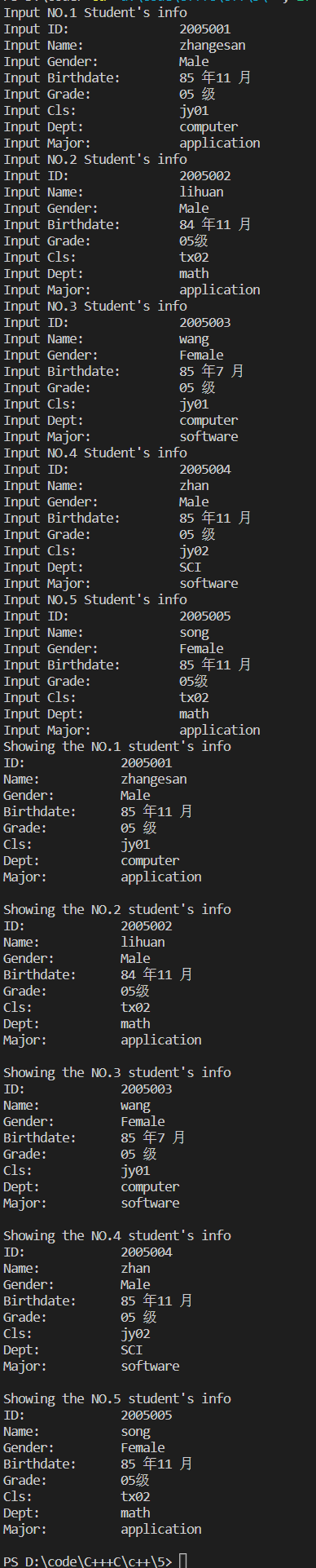
        cout << endl;

    }

    return 0;

}

### 实验结果：



### 思考与扩展：

采用什么方式可以让学号自动按输入顺序生成？

要实现学号自动按输入顺序生成，可以在Student类中添加一个静态成员变量，用于记录当前已创建的学生数量。然后在构造函数中将学号设置为当前学生数量加一。下面是修改后的代码：

#include <iostream>

#include <string>

using namespace std;

class Student

{

private:

    string id;        // 学生ID

    string name;      // 学生姓名

    string gender;    // 学生性别

    string birthdate; // 学生出生日期

    string grade;     // 学生年级

    string cls;       // 学生班级

    string dept;      // 学生所在系别

    string major;     // 学生专业

    static int count; // 当前已创建的学生数量

public:

    Student()

    {

        count++;                // 每创建一个学生对象，数量加一

        id = "S" + to\_string(count); // 设置学号为"S" + 学生数量

    }

    // 构造函数，用于初始化学生对象的各个属性

    Student(string name, string gender, string birthdate,

            string grade, string cls, string dept, string major)

        : name(name), gender(gender), birthdate(birthdate),

          grade(grade), cls(cls), dept(dept), major(major)

    {

        count++;                // 每创建一个学生对象，数量加一

        id = "S" + to\_string(count); // 设置学号为"S" + 学生数量

    }

    // 显示学生信息的方法

    void show()

    {

        cout << "ID:\t\t" << id << endl;             // 输出学生ID

        cout << "Name:\t\t" << name << endl;         // 输出学生姓名

        cout << "Gender:\t\t" << gender << endl;     // 输出学生性别

        cout << "Birthdate:\t" << birthdate << endl; // 输出学生出生日期

        cout << "Grade:\t\t" << grade << endl;       // 输出学生年级

        cout << "Cls:\t\t" << cls << endl;           // 输出学生班级

        cout << "Dept:\t\t" << dept << endl;         // 输出学生所在系别

        cout << "Major:\t\t" << major << endl;       // 输出学生专业

    }

};

int Student::count = 0; // 静态成员变量初始化为0

int main()

{

    Student students[5]; // 创建一个包含5个学生对象的数组

    for (int i = 0; i < 5; i++)

    {

        cout << "Input NO." << i + 1 << " Student's info " << endl;

        string name, gender, birthdate, grade, cls, dept, major;

        cout << "Input Name:\t\t";

        getline(cin, name); // 获取学生姓名

        cout << "Input Gender:\t\t";

        getline(cin, gender); // 获取学生性别

        cout << "Input Birthdate:\t";

        getline(cin, birthdate); // 获取学生出生日期

        cout << "Input Grade:\t\t";

        getline(cin, grade); // 获取学生年级

        cout << "Input Cls:\t\t";

        getline(cin, cls); // 获取学生班级

        cout << "Input Dept:\t\t";

        getline(cin, dept); // 获取学生所在系别

        cout << "Input Major:\t\t";

        getline(cin, major); // 获取学生专业

        students[i] = Student(name, gender, birthdate, grade, cls, dept, major); // 使用获取的信息创建学生对象并将其存储在数组中

    }

    for (int i = 0; i < 5; i++)

    {

        cout << "Showing the NO." << i + 1 << " student's info" << endl;

        students[i].show(); // 显示学生信息

        cout << endl;

    }

    return 0;

}

# 题目3 人员管理

### 实验描述：

设计某小型公司的 employee（人员）类

1、类

1）employee 类：

基本信息：编号、姓名、性别、出生日期、职位等；

出生日期使用自定义的 Date（日期）类；

其中：基本信息为 private 属性，成员函数为public 属性；

多个构造函数：缺省构造函数、带参数的构造函数、带默认参数的构造函数；可以从外部访问类成员的友员函数；

2）Date 类：

成员变量：年、月、日

成员函数：SetYear(int year)、SetMonth(int month)、SetDay(int day)

GetYear()、GetMonth()、GetDay()

2、基本功能：

1）职工信息的录入；

2）职工信息的显示；

3）用对象数组保存已输入的职工对象；

4）可以修改人员的基本信息，如：姓名、职位等；

5）可以通过编号或姓名进行人员查询；

### 实验代码：

/\*

 \* @Author: 王贤义

 \* @FilePath: \code\C+++C\c++\5\3employee.cpp

 \* @Description: 设计某小型公司的 employee（人员）类

1、类

1）employee 类：

基本信息：编号、姓名、性别、出生日期、职位等；

出生日期使用自定义的 Date（日期）类；

其中：基本信息为 private 属性，成员函数为public 属性；

多个构造函数：缺省构造函数、带参数的构造函数、带默认参数的构造函数；可以从外部访问类成员的友员函数；

2）Date 类：

成员变量：年、月、日

成员函数：SetYear(int year)、SetMonth(int month)、SetDay(int day)

GetYear()、GetMonth()、GetDay()

2、基本功能：

1）职工信息的录入；

2）职工信息的显示；

3）用对象数组保存已输入的职工对象；

4）可以修改人员的基本信息，如：姓名、职位等；

5）可以通过编号或姓名进行人员查询；

 \*/

#include <iostream>

#include <string>

using namespace std;

class Date

{

private:

    int year;  // 年份

    int month; // 月份

    int day;   // 日期

public:

    void SetYear(int year)

    {

        this->year = year;

    }

    void SetMonth(int month)

    {

        this->month = month;

    }

    void SetDay(int day)

    {

        this->day = day;

    }

    int GetYear()

    {

        return year;

    }

    int GetMonth()

    {

        return month;

    }

    int GetDay()

    {

        return day;

    }

};

class Employee

{

private:

    int number;      // 员工编号

    string name;     // 员工姓名

    string sex;      // 员工性别

    Date birth;      // 员工出生日期

    string position; // 员工职位

public:

    Employee() {} // 默认构造函数

    // 带参数的构造函数，用于初始化员工信息

    Employee(int num, string n, string s, int y, int m, int d, string pos)

    {

        number = num;

        name = n;

        sex = s;

        birth.SetYear(y);

        birth.SetMonth(m);

        birth.SetDay(d);

        position = pos;

    }

    // 带参数的构造函数，用于初始化员工信息（不包含性别和出生日期）

    Employee(int num, string n, string pos)

    {

        number = num;

        name = n;

        position = pos;

    }

    // 修改员工信息

    void ModifyInfo(string n, string pos)

    {

        name = n;

        position = pos;

    }

    // 显示员工信息

    void DisplayInfo()

    {

        cout << "====================" << endl;

        cout << "Employee Number: " << number << endl;

        cout << "Employee Name: " << name << endl;

        cout << "Employee Gender: " << sex << endl;

        cout << "Employee Date of Birth: " << birth.GetYear() << " / " << birth.GetMonth() << " / " << birth.GetDay() << endl;

        cout << "Employee Position: " << position << endl;

    }

    friend void SearchByNumber(Employee \*empList, int n, int num);

    friend void SearchByName(Employee \*empList, int n, string name);

    int getNumber()

    {

        return number;

    }

};

/\*\*

 \* @description: 友元函数，类外实现，按员工编号查找员工

 \* @param {Employee} \*empList   职工类指针，指向职工类数组

 \* @param {int} n               职工类数组长度

 \* @param {int} num             职工编号

 \* @return {\*}

 \*/

void SearchByNumber(Employee \*empList, int n, int num)

{

    for (int i = 0; i < n; i++)

    {

        if (empList[i].number == num)

        {

            empList[i].DisplayInfo();

            return;

        }

    }

    cout << "Employee with number " << num << " not found!" << endl;

}

/\*\*

 \* @description: 友元函数，类外实现，按员工姓名查找员工

 \* @param {Employee} \*empList   职工类指针，指向职工类数组

 \* @param {int} n               职工类数组长度

 \* @param {string} name         职工编号

 \* @return {\*}

 \*/

void SearchByName(Employee \*empList, int n, string name)

{

    for (int i = 0; i < n; i++)

    {

        if (empList[i].name == name)

        {

            empList[i].DisplayInfo();

            return;

        }

    }

    cout << "Employee with name " << name << " not found!" << endl;

}

int main()

{

    Employee empList[4];

    int n = 4;

    int choice = 0;

    int num;

    string name, s;

    int y, m, d;

    string pos;

    while (choice != 6)

    {

        cout << "Select an option:\n1. Enter employee information\n2. Display employee information\n3. Modify employee information\n4. Search by employee number\n5. Search by employee name\n6. Exit\n";

        cin >> choice;

        switch (choice)

        {

        case 1:

            for (int i = 0; i < n; i++)

            {

                cout << "Enter employee " << i + 1 << "'s information:" << endl;

                cout << "Enter employee number: ";

                cin >> num;

                cout << "Enter employee name: ";

                cin >> name;

                cout << "Enter employee gender: ";

                cin >> s;

                cout << "Enter employee birth year: ";

                cin >> y;

                cout << "Enter employee birth month: ";

                cin >> m;

                cout << "Enter employee birth day: ";

                cin >> d;

                cout << "Enter employee position: ";

                cin >> pos;

                empList[i] = Employee(num, name, s, y, m, d, pos);

            }

            break;

        case 2:

            for (int i = 0; i < n; i++)

            {

                empList[i].DisplayInfo();

            }

            break;

        case 3:

            cout << "Enter the employee number to modify: ";

            cin >> num;

            cout << "Enter the new employee name: ";

            cin >> name;

            cout << "Enter the new employee position: ";

            cin >> pos;

            for (int i = 0; i < n; i++)

            {

                if (empList[i].getNumber() == num)

                {

                    empList[i].ModifyInfo(name, pos);

                    break;

                }

            }

            break;

        case 4:

            cout << "Enter the employee number to search: ";

            cin >> num;

            SearchByNumber(empList, n, num);

            break;

        case 5:

            cout << "Enter the employee name to search: ";

            cin >> name;

            SearchByName(empList, n, name);

            break;

        case 6:

            cout << "Program ended!" << endl;

            break;

        default:

            cout << "Invalid input!" << endl;

            break;

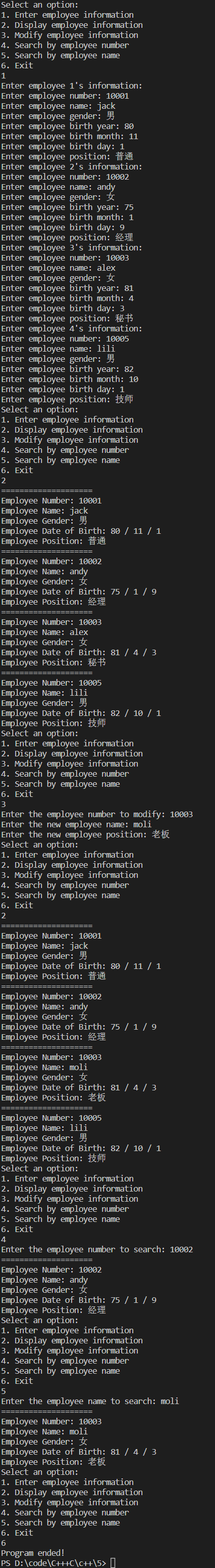
        }

    }

    return 0;

}

### 实验结果：



### 思考与扩展：

如何使用链表方式来保存已输入职工对象?查询函数该如何来写？

引入了一个新的Node结构体来表示链表节点。AddNode函数用于在链表末尾添加新节点，并返回新的链表头节点。SearchByNumber和SearchByName函数用于在链表中按员工编号和姓名进行查找。以下是修改后的全部代码：

#include <iostream>

#include <string>

using namespace std;

class Date

{

private:

    int year;

    int month;

    int day;

public:

    void SetYear(int year)

    {

        this->year = year;

    }

    void SetMonth(int month)

    {

        this->month = month;

    }

    void SetDay(int day)

    {

        this->day = day;

    }

    int GetYear()

    {

        return year;

    }

    int GetMonth()

    {

        return month;

    }

    int GetDay()

    {

        return day;

    }

};

class Employee

{

private:

    int number;

    string name;

    string sex;

    Date birth;

    string position;

public:

    Employee() {}

    Employee(int num, string n, string s, int y, int m, int d, string pos)

    {

        number = num;

        name = n;

        sex = s;

        birth.SetYear(y);

        birth.SetMonth(m);

        birth.SetDay(d);

        position = pos;

    }

    Employee(int num, string n, string pos)

    {

        number = num;

        name = n;

        position = pos;

    }

    void ModifyInfo(string n, string pos)

    {

        name = n;

        position = pos;

    }

    void DisplayInfo()

    {

        cout << "====================" << endl;

        cout << "Employee Number: " << number << endl;

        cout << "Employee Name: " << name << endl;

        cout << "Employee Gender: " << sex << endl;

        cout << "Employee Date of Birth: " << birth.GetYear() << " / " << birth.GetMonth() << " / " << birth.GetDay() << endl;

        cout << "Employee Position: " << position << endl;

    }

    friend void SearchByNumber(Employee \*empList, int n, int num);

    friend void SearchByName(Employee \*empList, int n, string name);

    int getNumber()

    {

        return number;

    }

};

struct Node

{

    Employee emp;

    Node \*next;

};

Node \*AddNode(Node \*head, Employee emp)

{

    Node \*newNode = new Node;

    newNode->emp = emp;

    newNode->next = NULL;

    if (head == NULL)

    {

        head = newNode;

    }

    else

    {

        Node \*temp = head;

        while (temp->next != NULL)

        {

            temp = temp->next;

        }

        temp->next = newNode;

    }

    return head;

}

void SearchByNumber(Node \*head, int num)

{

    Node \*temp = head;

    while (temp != NULL)

    {

        if (temp->emp.getNumber() == num)

        {

            temp->emp.DisplayInfo();

            return;

        }

        temp = temp->next;

    }

    cout << "Employee with number " << num << " not found!" << endl;

}

void SearchByName(Node \*head, string name)

{

    Node \*temp = head;

    while (temp != NULL)

    {

        if (temp->emp.name == name)

        {

            temp->emp.DisplayInfo();

            return;

        }

        temp = temp->next;

    }

    cout << "Employee with name " << name << " not found!" << endl;

}

int main()

{

    Node \*head = NULL; // 头节点

    int choice = 0;

    int num;

    string name, s;

    int y, m, d;

    string pos;

    while (choice != 6)

    {

        cout << "选择操作：\n1. 输入员工信息\n2. 显示员工信息\n3. 修改员工信息\n4. 按员工编号查找\n5. 按员工姓名查找\n6. 退出\n";

        cin >> choice;

        switch (choice)

        {

        case 1:

            int n;

            cout << "输入要添加的员工数量：";

            cin >> n;

            for (int i = 0; i < n; i++)

            {

                cout << "输入第 " << i + 1 << " 个员工的信息：" << endl;

                cout << "输入员工编号：";

                cin >> num;

                cout << "输入员工姓名：";

                cin >> name;

                cout << "输入员工性别：";

                cin >> s;

                cout << "输入员工出生年份：";

                cin >> y;

                cout << "输入员工出生月份：";

                cin >> m;

                cout << "输入员工出生日期：";

                cin >> d;

                cout << "输入员工职位：";

                cin >> pos;

                Employee emp(num, name, s, y, m, d, pos);

                head = AddNode(head, emp);

            }

            break;

        case 2:

            {

                Node \*temp = head;

                while (temp != NULL)

                {

                    temp->emp.DisplayInfo();

                    temp = temp->next;

                }

            }

            break;

        case 3:

            cout << "输入要修改的员工编号：";

            cin >> num;

            cout << "输入新的员工姓名：";

            cin >> name;

            cout << "输入新的员工职位：";

            cin >> pos;

            {

                Node \*temp = head;

                while (temp != NULL)

                {

                    if (temp->emp.getNumber() == num)

                    {

                        temp->emp.ModifyInfo(name, pos);

                        break;

                    }

                    temp = temp->next;

                }

            }

            break;

        case 4:

            cout << "输入要查找的员工编号：";

            cin >> num;

            SearchByNumber(head, num);

            break;

        case 5:

            cout << "输入要查找的员工姓名：";

            cin >> name;

            SearchByName(head, name);

            break;

        case 6:

            cout << "程序结束！" << endl;

            break;

        default:

            cout << "无效输入！" << endl;

            break;

        }

    }

    // 释放链表内存

    Node \*temp = head;

    while (temp != NULL)

    {

        Node \*current = temp;

        temp = temp->next;

        delete current;

    }

    return 0;

}