定义一个矩阵类模板 **template <class T> class Matrix**

**数据成员（访问权限定义为protected）**：

指向存放矩阵内容的二维指针（**T** \*\*content）

矩阵的行和列（size\_t row; size\_t column）

**成员函数（访问权限定义为public）：**

* **带默认参数的构造函数**：根据参数规定的行和列的值初始化矩阵空间

**Matrix**(size\_t \_row = 5, size\_t \_column = 5);

* **拷贝构造函数**：初始化矩阵

**Matrix**(const **Matrix<T>** &matrix);

* **析构函数**：释放堆空间

**~Matrix**( );

* 初始化矩阵内容 void init(**T** \*\*mat);
* 获取矩阵的行（定义为const成员函数）size\_t getRow() const;
* 获取矩阵的列（定义为const成员函数）size\_t getColumn() const;
* 打印矩阵内容（定义为const成员函数）void print() const;
* 重载**赋值运算符=** （只能作为类成员函数重载）

**Matrix<T>**& operator=(const **Matrix<T>** &matrix);

* 重载**加法运算符+**，实现两个矩阵相加

**Matrix<T>** operator+(const **Matrix<T>** &matrix);

* 重载**函数调用运算符( )**，实现获取矩阵中某个指定位置的元素 （只能作为类成员函数重载）

**T**& operator()(size\_t rindex, size\_t cindex);

**要求**：将类的定义与类成员函数的实现分开。

定义一个函数模板**search**，实现查找一维数组中与指定的数据相等的元素第一次和最后一次出现的位置（若未找到，则将位置设置为**-1**）。

template <class **T**=int, class **S**=int> // 带默认类型参数

class **Pair**{

public:

**Pair**() { }

**Pair**(**T** \_first, **S** \_second): first(\_first), second(\_second) { }

**T** first;

**S** second;

};

template <class **T**>

**Pair**<int, int> search(const **T** \*vec, int n, const **T** &data);