Array 动态数组模块

本模块实现了一个通用的动态数组模板类 `Array<T>`，支持插入、删除、查找、扩容等常用操作，接口风格参考工业界标准，适用于 C++ 项目。

特性

- 支持任意类型元素（模板实现）
- 支持动态扩容
- 提供常用数组操作接口
- 边界检查与异常安全
- 代码风格规范，接口注释详细
- 附带交互式测试程序

主要接口

- `explicit Array(int capacity)`：构造函数，初始化指定容量的数组
- `Array(const Array& other)`：拷贝构造
- `Array& operator=(const Array& other)`：赋值操作符
- `~Array()`：析构函数
- `T get(int index) const`：获取指定索引的元素
- `void set(int index, const T& value)`：设置指定索引的元素
- `void insert(int index, const T& value)`：在指定位置插入元素
- `void remove(int index)`：删除指定位置的元素
- `void extend(int enlarge)`：扩展数组容量
- `int search(const T& value) const`：查找元素首次出现的位置
- `int size() const`：获取当前元素个数
- `bool isEmpty() const`：判断数组是否为空
- `bool isFull() const`：判断数组是否已满

详细接口说明见 ../include/array.hpp。

用法示例

```
#include "array.hpp"
#include <iostream>

int main() {
    Array<int> arr(5);
    arr.insert(0, 10);
    arr.insert(1, 20);
    arr.insert(2, 30);
    arr.set(1, 25);
    std::cout << "arr[1] = " << arr.get(1) << std::endl;
    arr.remove(0);
    std::cout << "size = " << arr.size() << std::endl;
    std::cout << "index of 30 = " << arr.search(30) << std::endl;
    arr.extend(5);
    std::cout << "isFull = " << arr.isFull() << std::endl;
    return 0;
}
```

交互式测试

本模块提供了交互式测试程序，支持命令行操作数组，便于学习和调试。**测试程序所有交互均为中文，适合中文用户体验。**

编译并运行：

```
g++ -std=c++11 test/test_array.cpp -o test_array
./test_array
```

示例交互（中文版）：

```
请输入数组初始容量: 3

====== 动态数组交互测试菜单 ======
命令列表:
  insert <下标> <值>    : 在下标插入值
  remove <下标>         : 删除指定下标的元素
  set <下标> <值>       : 设置指定下标的值
  get <下标>            : 获取指定下标的值
  search <值>           : 查找值，返回下标
  extend <扩容数>       : 扩展数组容量
  size                  : 当前元素个数
  isEmpty               : 判断数组是否为空
  isFull                : 判断数组是否已满
  print                 : 打印数组内容
  help                  : 显示菜单
  exit / 0              : 退出程序
------------------------------------
请输入命令: > insert 0 42
已在下标 0 插入 42。
> print
数组内容: [42]
> insert 1 99
已在下标 1 插入 99。
> get 1
下标 1 的值为: 99
> exit
程序结束，再见!
```

常见问题

- **Q: 插入/删除越界或数组已满怎么办?**
  A: 会抛出异常并提示错误信息。
- **Q: 如何扩容?**
  A: 使用 `extend <扩容数>` 命令或 `extend()` 方法。
- **Q: 支持哪些类型?**
  A: 支持任意可赋值类型（模板实现）。

模板使用说明

本模块为模板实现，建议直接包含 `array.hpp` 头文件，无需单独编译 cpp 文件。例如：

```
g++ -std=c++11 your_code.cpp -o your_program
```

相关文档

- doc/ADT.md：数组抽象数据类型说明
- ../include/array.hpp：接口定义与注释