实验二：信号序列 C 语言实现

计算机学院 郭嘉睿 2413174

一、实验平台

编译器：gcc.exe (Rev8, Built by MSYS2 project) 15.2.0

版本控制：git version 2.48.1.windows.1

托管平台：GitHub

二、实验目的

1. 验证：验证通过 C 语言结构体实现任意起始下标信号序列（signal sequence）的正确性。
2. 观察：观察信号序列在固定长度与无限长度输入模式下的内存分配与索引映射行为。
3. 分析：分析 C 语言实现中逻辑下标到物理下标的转换机制及其对边界访问的影响。
4. 比照：比较固定长度与可扩展序列在内存使用和输入交互上的差异。

三、实验核心问题

实验的核心问题是：如何在 C 语言中实现一个支持任意起始下标且可无限扩展的信号序列结构。

核心挑战包括：

1. 索引映射：C 数组下标必须从 0 开始，而实验要求逻辑下标可以为任意整数（包括负数）。因此必须设计映射函数：

$$physical = logical - start$$

2. 动态扩容：为支持无限输入，使用 realloc() 动态调整数组大小。
3. 错误控制与日志：所有错误信息统一使用英文并输出至 stderr。
4. 交互式输入控制：实验通过命令行输入模式完成数据读入与显示，输入停止记号为显式字符串 STOP。

四、实验设计

输入（Input）

- 用户在命令行选择模式（1 = 固定长度，2 = 无限长度）；
- 设定起始下标（可为负数）；
- 输入信号值；在无限模式下，输入 STOP 终止输入。

输出（Output）

- 程序打印信号序列的起始下标、长度及所有元素值；
- 所有错误信息打印至标准错误输出（stderr）。

核心处理流程

1. 初始化信号序列结构 SignalSeq：记录起始下标、长度、是否可扩展等信息；
2. 逻辑下标到物理下标转换函数 logical_to_physical() 实现安全索引；
3. 动态扩容函数 signal_seq_append() 通过倍增策略进行内存扩展；
4. 用户输入函数 input_unbounded() 读取值直到遇到停止记号；
5. 最终由 print_sequence() 打印结果。

手动实现部分：结构体定义、索引映射、内存管理、错误处理、交互逻辑。

调库部分：标准库函数 malloc, realloc, fgets, strtod, fprintf 等。

五、代码与结果展示及分析

核心代码片段

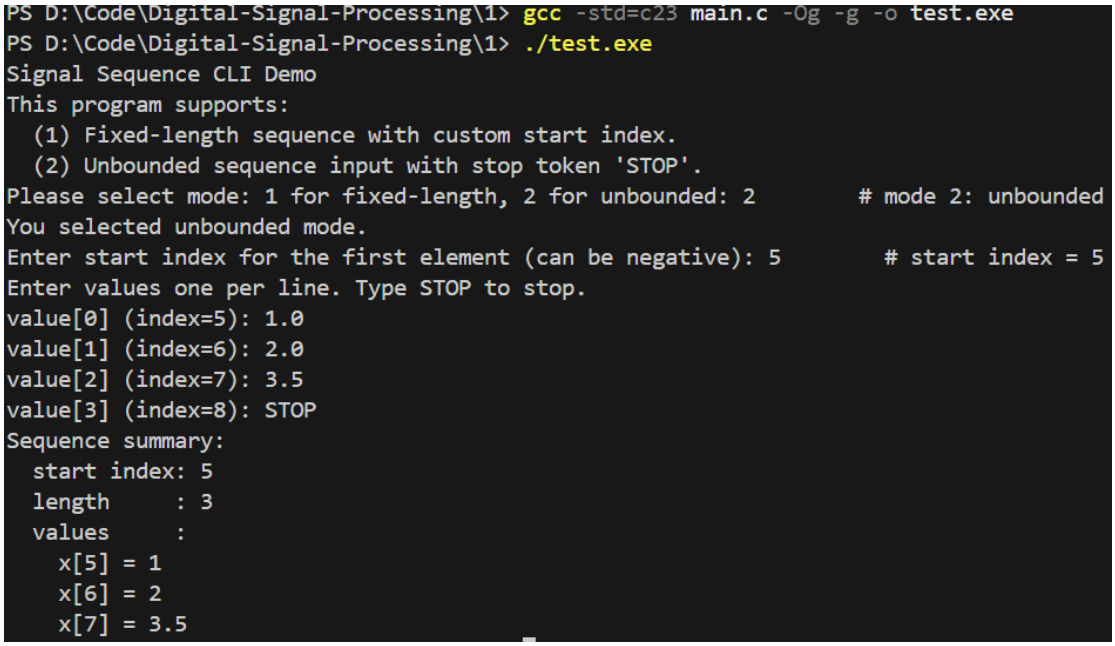
```
/**
 * @brief 将逻辑下标转换为物理下标。Map logical index to physical index.
 */
static int signal_seq_logical_to_physical(const SignalSeq *seq, int logical_index, int
*physical_index) {
    if (seq == NULL || physical_index == NULL) {
        fprintf(stderr, "Error: NULL pointer in logical_to_physical.\n");
```

```
        return -1;
    }
    int offset = logical_index - seq->start;
    if (offset < 0 || offset >= seq->length) {
        fprintf(stderr, "Error: logical index %d is out of range [%d, %d].\n",
                logical_index, seq->start, seq->start + seq->length - 1);
        return -1;
    }
    *physical_index = offset;
    return 0;
}
```

说明：

- 此段代码为实验的核心之一，用于保证逻辑索引与底层数组索引的安全映射。
- 它是本实验“任意起始下标”特性的实现关键。

六、控制台结果截图



七、结果分析

- 所有测试结果与设计预期一致；
- 支持固定长度与无限输入两种模式；
- 下标范围可为负数，映射稳定；
- 错误信息规范且输出位置正确（stderr）；
- 动态内存管理无泄漏（malloc/realloc/free 路径完整）。

改进方向：增加 set/get/shift/scale 等信号操作命令形成简易 REPL。

八、结论

通过本实验，验证了在 C 语言环境下使用结构体实现任意下标起点的信号序列的可行性，并成功实现了：

- 索引安全映射；
- 固定长度与动态扩容；
- CLI 人机交互输入；
- 统一的错误与日志输出系统。

实验结果完全符合预期，为后续信号处理函数（如卷积、加窗、移位等）的实现奠定了基础。