实验报告电子版

**数据结构实训**

**——多维数字信号的线性表表示及操作**

姓 名： 刘远明

学 号： 2220212113

指导老师： 韩凤

实验日期： 2024年3月25日

**大连海事大学电子信息科学技术专业**

**Academic Honesty Violations in Practice of Data Structures (PDS)**

In **PDS**, the MINIMUM penalty recommended for a violation of the Academic Honesty Policy will be a ZERO ON THE ASSIGNMENT, PROJECT or EXAM and a LOWERING OF YOUR FINAL GRADE for below what is otherwise earned. You may NOT withdraw from the course if found guilty. Some examples of academic misconduct in **PDS** include but are not limited to the following actions:

1. Picking up and using or discarding another student's written or computer output;

2. Using the computer account of another student;

3. Representing as one's own the work of another on assignments, quizzes, and projects;

4. Giving another student a copy of one's work on an assignment before the due date.

5. Copying work from online resources (Baidu,Chegg, google forums, etc.)

6. Posting work to online resources where other students can view your work.

All submissions will be checked for similarity. This code will check each submission for similarity between other student submissions, past student submissions, the solution manual, and online resources and postings. If your submission is flagged for a high level of similarity, it will be turned in for an academic honesty violation if deemed appropriate.

NOTE: Changing variable names, adding comments, or spacing will l result in a violation.

1. **实验目的**

1、深入理解线性表的存储结构及操作的基本原理和方法；2、掌握线性表的各种操作（建立、插入、删除等）的实现方法；3、能够实现多维信号的简单运算和操作，如均值滤波、上采样和下采样等。

1. **实验内容与要求**

（1）为一个不定长多维数字信号设计一个数据结构（形如下图示） ，并实现对其进行基本操作，包括初始化、求数据元素个数操作、插入、删除、取数据元素、判非空操作。

（2）能够实现多维信号的简单运算和操作（数字信号处理课讲授的基本方法），如均值滤波、上采样和下采样等*。*

1. **算法描述与流程**
   * + 1. 为一个不定长多维数字信号设计一个数据结构，并实现对其进行基本操作，包括初始化、输入数据、插入、删除、取数据元素。
2. 初始化

生成新节点作为头节点，用头指针指向头节点，然后将头点的指针域置空。

1. 根据输入长度建立新链表

指针p指向第一个节点，根据输入节点个数从首元结点始，利用尾插法建立新的链表。

1. 插入

输入要前插的位置i，并由指针p指向该节点的上一个节点，生成一个新节点\*s，将节点\*p的指针指向新节点\*s，节点\*s成功插入链表

1. 删除

输入要删除的位置i并由指针p指向该节点的上一个结点，将指针s指向要删除的节点，将指针p的指针指向指针s的下一个节点，从链表中删除\*s节点，保存删除节点的数据域，释放删除节点的空间。

1. 查找

输入要查找的数据位置的值，用指针p指向首元节点，从首元节点开始以此顺着链域next向下查找，只要j<i且\*p非空，则循环执行以下操作：p指向下一个节点，输出指针p所指向节点的值。

* + - 1. 能够实现多维信号的简单运算和操作，如均值滤波、上采样和下采样等。

1. 均值滤波

用pa,pb指针指向原函数进行三个数的均值滤波，创建新链表表头newlist，将得到得均值赋给节点\*p的data，利用指针p和s进行尾插生成新的链表。

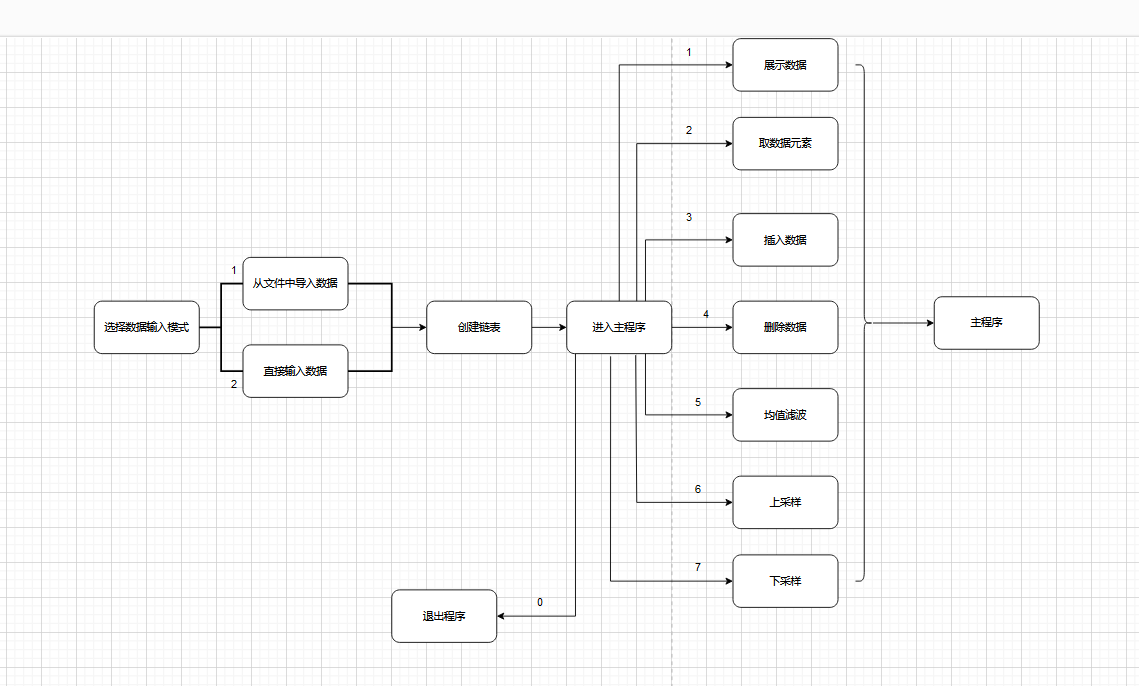
1. 上采样

对节点\*p1和\*p2取均值，生成一个新节点\*p3，将得到得均值赋给节点\*p3的data，将节点\*p3插入原链表节点\*p1和\*p2当中。

* + - 1. 能够对文件进行读取操作。

1. 读文件
2. 取文件

（4）流程图



1. **算法实现**

**#include"stdio.h"**

**#include"stdlib.h"**

**#include"string.h"**

**#define LEN sizeof(struct zuobiao)**

**typedef struct zuobiao**

**{**

**int y\_zhou;**

**int x\_zhou;**

**struct zuobiao\* next;**

**}ZB;**

**int n = 0;**

**//通过读取文件的方式来读取数据创建链表**

**ZB\* create\_1(const char\* filename) {**

**ZB\* head = NULL;**

**ZB\* tail = NULL;**

**FILE\* file = fopen(filename, "r");**

**if (file == NULL) {**

**perror("Error opening file");**

**return NULL;**

**}**

**int x, y;**

**while (fscanf(file, "%d,%d", &x, &y) == 2) {**

**ZB\* newNode = (ZB\*)malloc(sizeof(ZB));**

**if (newNode == NULL) {**

**perror("Error allocating memory");**

**break;**

**}**

**newNode->x\_zhou = x;**

**newNode->y\_zhou = y;**

**newNode->next = NULL;**

**if (head == NULL) {**

**head = newNode;**

**tail = newNode;**

**} else {**

**tail->next = newNode;**

**tail = newNode;**

**}**

**}**

**fclose(file);**

**return head;**

**}**

**//直接输入数据来创建链表**

**ZB\* create\_2()**

**{**

**ZB\* p1, \* p2, \* head;**

**int y\_zhou;**

**int x\_zhou;**

**head = NULL;**

**p1 = p2 = (ZB\*)malloc(LEN);//开创一个空间用于存放结构体内的数据**

**printf("请输入x轴与y轴：（格式：x轴，y轴）\n");**

**scanf("%d,%d", &p1->x\_zhou, &p1->y\_zhou);**

**while (p1->x\_zhou != 0)//x\_zhou=0时，链表创建结束**

**{**

**n++;//输入了多少个坐标的数据**

**if (n == 1)**

**head = p1;//把第一个结点的地址赋给head指针**

**else**

**p2->next = p1;//如果输入的不是第一个结点的数据，就让其内的next指针指向下一个结点的开头**

**p2 = p1;//p2指针移动到p1指针处（下一个结点的开头）**

**p1 = (ZB\*)malloc(LEN);//p1又指向新开创结点的开头**

**scanf("%d,%d", &p1->x\_zhou, &p1->y\_zhou);**

**}**

**p2->next = NULL;//如果输入的x\_zhou=0，跳出循环，最后一个结点的next指针指向空指针（即指向NULL值）**

**return head;//返回第一个结点的地址（这样才能找到后面的所有结点）**

**}**

**ZB\* insert\_link(ZB\* head)//插入结点的函数**

**{**

**ZB\* p;**

**printf("请输入需要插入数据的位置：\n");**

**int m;**

**scanf("%d", &m);**

**p = head;//p与head指向第一个结点的开头**

**int i = 1;//插入位置**

**printf("请输入需要插入的数据 : （格式：x轴，y轴）\n");**

**ZB\* j;**

**j = (ZB\*)malloc(LEN);**

**scanf("%d,%d", &j->x\_zhou, &j->y\_zhou);**

**if (m == 1)//如果要在第一个位置插入数据**

**{**

**j->next = head;//新的结点的next指针直接指向头结点，新的数据就会成为第一个结点**

**head = j;//head指针又指向第一个结点的开始**

**return head;//返回插入数据后的头结点**

**}**

**else**

**{**

**while (i < m - 1)**

**/\*m-1意味着：如果要在第5个位置插入数据，**

**则需要让第4个结点的next指向插入数据的开头\*/**

**{**

**p = p->next;//循环找到需要插入数据的上一个结点的next所在位置**

**i++;**

**}**

**}**

**j->next = p->next;//把p的next地址赋给j的next地址，即新的结点可插入两个结点的中间**

**p->next = j;//p的next指向j的地址**

**return head;//返回第一个结点的地址**

**}**

**ZB\* delete\_link(ZB\* head)//删除结点的函数**

**{**

**ZB\* p1, \* p2;**

**printf("请输入需要删除数据的位置：\n");**

**int m;**

**scanf("%d", &m);**

**p1 = head;**

**if (m == 1) //如果删除的是第一个结点**

**{**

**p2 = p1->next; //直接把下一个地址作为返回值**

**free(p1); //释放内存**

**return p2;**

**}**

**else**

**{**

**int i = 1;**

**while (i < m - 1)**

**{**

**p1 = p1->next;**

**i++;**

**}**

**p2 = p1->next;**

**p1->next = p1->next->next;//p1的next指向了需要删除的结点的next指向的结点（即越过了中间需要删除的结点）**

**free(p2);//释放需要删除的结点的空间**

**return head;//返回第一个结点的地址**

**}**

**}**

**//均值滤波函数**

**ZB\* mean\_filter(ZB\* head, int window\_size) {**

**ZB\* current = head;**

**ZB\* newHead = NULL;**

**ZB\* newTail = NULL;**

**while(current != NULL) {**

**int sumX = 0;**

**int sumY = 0;**

**int count = 0;**

**ZB\* windowEnd = current;**

**for (int i = 0; i < window\_size && windowEnd != NULL; i++) {**

**sumX += windowEnd->x\_zhou;**

**sumY += windowEnd->y\_zhou;**

**windowEnd = windowEnd->next;**

**count++;**

**}**

**ZB\* newNode = (ZB\*)malloc(sizeof(ZB));**

**if (newNode == NULL) {**

**perror("Error allocating memory");**

**break;**

**}**

**newNode->x\_zhou = sumX / count;**

**newNode->y\_zhou = sumY / count;**

**newNode->next = NULL;**

**if (newHead == NULL) {**

**newHead = newNode;**

**newTail = newNode;**

**} else {**

**newTail->next = newNode;**

**newTail = newNode;**

**}**

**current = current->next;**

**}**

**return newHead;**

**}**

**//上采样函数**

**ZB\* upsampling(ZB\* head) {**

**ZB\* current = head;**

**ZB\* newHead = NULL;**

**ZB\* newTail = NULL;**

**while (current != NULL && current->next != NULL) {**

**// 复制当前节点值**

**ZB\* newNode = (ZB\*)malloc(sizeof(ZB));**

**if (newNode == NULL) {**

**perror("Error allocating memory");**

**break;**

**}**

**newNode->x\_zhou = current->x\_zhou;**

**newNode->y\_zhou = current->y\_zhou;**

**newNode->next = NULL;**

**if (newHead == NULL) {**

**newHead = newNode;**

**newTail = newNode;**

**} else {**

**newTail->next = newNode;**

**newTail = newNode;**

**}**

**// 计算并插入新节点**

**ZB\* nextNode = current->next;**

**//newNode = (ZB\*)malloc(sizeof(ZB));**

**if (nextNode = NULL) {**

**printf("仅一个元素，无法上采样");**

**break;**

**}**

**newNode->x\_zhou = (current->x\_zhou + nextNode->x\_zhou) / 2;**

**newNode->y\_zhou = (current->y\_zhou + nextNode->y\_zhou) / 2;**

**newNode->next = NULL;**

**newTail->next = newNode;**

**newTail = newNode;**

**// 移动到下一个原始节点**

**current = nextNode;**

**}**

**return newHead;**

**}**

**//下采样函数**

**ZB\* downsampling(ZB\* head, int factor) {**

**ZB\* current = head;**

**ZB\* newHead = NULL;**

**ZB\* newTail = NULL;**

**int count = 0;**

**while (current != NULL) {**

**if (count % factor == 0) {**

**// 复制当前节点值**

**ZB\* newNode = (ZB\*)malloc(sizeof(ZB));**

**if (newNode == NULL) {**

**perror("Error allocating memory");**

**break;**

**}**

**newNode->x\_zhou = current->x\_zhou;**

**newNode->y\_zhou = current->y\_zhou;**

**newNode->next = NULL;**

**if (newHead == NULL) {**

**newHead = newNode;**

**newTail = newNode;**

**} else {**

**newTail->next = newNode;**

**newTail = newNode;**

**}**

**}**

**count++;**

**current = current->next;**

**}**

**return newHead;**

**}**

**void print(ZB\* head)//输出坐标信息的函数**

**{**

**ZB\* p;**

**p = head;//p取head的地址之后，二者同时指向第一个结点的开头**

**n == 0;**

**printf("\n");**

**if (head != NULL)**

**{**

**do**

**{**

**printf("x轴：%d y轴：%d\n", p->x\_zhou, p->y\_zhou);**

**p = p->next;//把next的地址（即下一个结点的开头）赋给p**

**n++;**

**} while (p != NULL);**

**}**

**printf("\n现在共有%d个坐标的信息：\n", n);**

**}**

**//输出到文件**

**void fprint(const char\* filename, ZB\* head) {**

**FILE\* file = fopen(filename, "w");**

**if (file == NULL) {**

**perror("Error opening file");**

**return;**

**}**

**ZB\* current = head;**

**while (current != NULL) {**

**fprintf(file, "%d,%d\n", current->x\_zhou, current->y\_zhou);**

**current = current->next;**

**}**

**fclose(file);**

**}**

**// 取出链表中第i个节点的值**

**int getValue\_1(ZB\* head, int i) {**

**ZB\* p = head;**

**int count = 0;**

**while (p != NULL) {**

**count++;**

**if (count == i) {**

**return p->x\_zhou;**

**}**

**p = p->next;**

**}**

**return -1; // 如果没有找到第 i 个节点，返回一个错误值**

**}**

**int getValue\_2(ZB\* head, int i) {**

**ZB\* p = head;**

**int count = 0;**

**while (p != NULL) {**

**count++;**

**if (count == i) {**

**return p->y\_zhou;**

**}**

**p = p->next;**

**}**

**return -1; // 如果没有找到第 i 个节点，返回一个错误值**

**}**

**int pp()**

**{**

**printf(" ╪╪╪╪╪╪╧╧╧╧╧╧╧╧╪╪╪╪╪╪\n");**

**printf("╔═══ 数据结构课程实验一 ╧╧═╗\n");**

**printf("║※1.展示数据 ║\n");**

**printf("║※2.插入数据 ║\n");**

**printf("║※3.删除数据 ║\n");**

**printf("║※4.均值滤波 ║\n");**

**printf("║※5.上采样 ║\n");**

**printf("║※6.下采样 ║\n");**

**printf("║※0.退出系统 ║\n");**

**printf("╚═══════════════════════════╝\n");**

**return 0;**

**}**

**int main()**

**{**

**system("title c语言定时关机程序");**

**system("mode con cols=42 lines=30");**

**system("color 0B");**

**system("date /T");**

**system("TIME /T");**

**ZB\* pointer;**

**ZB\* filteredList = NULL; // 将 filteredList 初始化为 NULL**

**ZB\* upList = NULL;**

**ZB\* downList = NULL;**

**int c;**

**printf("请选择数据导入模式以方便接下来操作：\n");**

**printf("※1.从文件中读取数据并创建链表\n");**

**printf("※2.手动输入数据并创建链表\n");**

**int a;**

**scanf("%d", &a);**

**getchar();**

**if (a == 1) {**

**pointer = create\_1("data.txt"); // 创建动态链表，返回第一个结点的地址赋给pointer**

**} else if (a == 2) {**

**pointer = create\_2();**

**} else {**

**printf("error!\n");**

**}**

**do{**

**pp();**

**scanf("%d",&c);**

**getchar();**

**if (c == 1) {**

**print(pointer);**

**} else if (c == 2) {**

**pointer = insert\_link(pointer);**

**print(pointer);**

**n = 0;**

**} else if (c == 3) {**

**pointer = delete\_link(pointer);**

**print(pointer);**

**n = 0;**

**} else if (c == 4) {**

**filteredList = mean\_filter(pointer, 2);**

**print(filteredList);**

**fprint("out\_average.txt", filteredList);**

**n = 0;**

**} else if (c == 5) {**

**if (upList != NULL) {**

**free(upList); // 释放之前分配的内存**

**}**

**upList = upsampling(pointer);**

**print(upList);**

**fprint("out\_up.txt", upList);**

**n = 0;**

**} else if (c == 6) {**

**if (downList != NULL) {**

**free(downList); // 释放之前分配的内存**

**}**

**downList = downsampling(pointer, 2); // 添加第二个参数，例如 2**

**print(downList);**

**fprint("out\_down.txt", downList);**

**n = 0;**

**} else if (c == 0) {**

**// 退出选项，无需执行任何操作**

**} else {**

**printf("错误！\n");**

**}**

**}while (c != 0); // 当输入不为0时，继续循环**

**if (pointer != NULL) {**

**free(pointer); // 释放占用的内存**

**}**

**if (filteredList != NULL) {**

**free(filteredList); // 释放分配的内存**

**}**

**if (upList != NULL) {**

**free(upList); // 释放分配的内存**

**}**

**if (downList != NULL) {**

**free(downList); // 释放分配的内存**

**}**

**// ZB\* pointer;**

**// pointer = create\_1("data.txt");//创建动态链表，返回第一个结点的地址赋给pointer**

**// n = 0;//n清零，为了保证之后函数里，从第一个结点开始遍历**

**// printf("%d,%d",getValue\_1(pointer,1),getValue\_2(pointer,1));**

**// ZB\* filteredList=mean\_filter(pointer,3);**

**// print(filteredList);**

**// fprint("out.txt",filteredList);**

**// print(pointer);//输出链表**

**// pointer = insert\_link (pointer);**

**// n = 0;**

**// print(pointer);//输出链表**

**// pointer = delete\_link(pointer);**

**// n = 0;**

**// print(pointer);//输出链表**

**//free(pointer);//释放占用的内存**

**system("pause");**

**exit(0);**

**return 0;**

**}**

1. **实验结果与分析**

**5.1测试用例**

对下列二维数字信号数据进行测试

1，2

3，4

5，6

7，8，

9，10

11，12

13，13

分别对其进行链表的创建 查找 插入 删除 输出等操作

**5.2测试程序**

**int pp()**

**{**

**printf(" ╪╪╪╪╪╪╧╧╧╧╧╧╧╧╪╪╪╪╪╪\n");**

**printf("╔═══ 数据结构课程实验一 ╧╧═╗\n");**

**printf("║※1.展示数据 ║\n");**

**printf("║※2.插入数据 ║\n");**

**printf("║※3.删除数据 ║\n");**

**printf("║※4.均值滤波 ║\n");**

**printf("║※5.上采样 ║\n");**

**printf("║※6.下采样 ║\n");**

**printf("║※0.退出系统 ║\n");**

**printf("╚═══════════════════════════╝\n");**

**return 0;**

**}**

**int main()**

**{**

**system("title c语言定时关机程序");**

**system("mode con cols=42 lines=30");**

**system("color 0B");**

**system("date /T");**

**system("TIME /T");**

**ZB\* pointer;**

**ZB\* filteredList = NULL; // 将 filteredList 初始化为 NULL**

**ZB\* upList = NULL;**

**ZB\* downList = NULL;**

**int c;**

**printf("请选择数据导入模式以方便接下来操作：\n");**

**printf("※1.从文件中读取数据并创建链表\n");**

**printf("※2.手动输入数据并创建链表\n");**

**int a;**

**scanf("%d", &a);**

**getchar();**

**if (a == 1) {**

**pointer = create\_1("data.txt"); // 创建动态链表，返回第一个结点的地址赋给pointer**

**} else if (a == 2) {**

**pointer = create\_2();**

**} else {**

**printf("error!\n");**

**}**

**do{**

**pp();**

**scanf("%d",&c);**

**getchar();**

**if (c == 1) {**

**print(pointer);**

**} else if (c == 2) {**

**pointer = insert\_link(pointer);**

**print(pointer);**

**n = 0;**

**} else if (c == 3) {**

**pointer = delete\_link(pointer);**

**print(pointer);**

**n = 0;**

**} else if (c == 4) {**

**filteredList = mean\_filter(pointer, 2);**

**print(filteredList);**

**fprint("out\_average.txt", filteredList);**

**n = 0;**

**} else if (c == 5) {**

**if (upList != NULL) {**

**free(upList); // 释放之前分配的内存**

**}**

**upList = upsampling(pointer);**

**print(upList);**

**fprint("out\_up.txt", upList);**

**n = 0;**

**} else if (c == 6) {**

**if (downList != NULL) {**

**free(downList); // 释放之前分配的内存**

**}**

**downList = downsampling(pointer, 2); // 添加第二个参数，例如 2**

**print(downList);**

**fprint("out\_down.txt", downList);**

**n = 0;**

**} else if (c == 0) {**

**// 退出选项，无需执行任何操作**

**} else {**

**printf("错误！\n");**

**}**

**}while (c != 0); // 当输入不为0时，继续循环**

**if (pointer != NULL) {**

**free(pointer); // 释放占用的内存**

**}**

**if (filteredList != NULL) {**

**free(filteredList); // 释放分配的内存**

**}**

**if (upList != NULL) {**

**free(upList); // 释放分配的内存**

**}**

**if (downList != NULL) {**

**free(downList); // 释放分配的内存**

**}**

**// ZB\* pointer;**

**// pointer = create\_1("data.txt");//创建动态链表，返回第一个结点的地址赋给pointer**

**// n = 0;//n清零，为了保证之后函数里，从第一个结点开始遍历**

**// printf("%d,%d",getValue\_1(pointer,1),getValue\_2(pointer,1));**

**// ZB\* filteredList=mean\_filter(pointer,3);**

**// print(filteredList);**

**// fprint("out.txt",filteredList);**

**// print(pointer);//输出链表**

**// pointer = insert\_link (pointer);**

**// n = 0;**

**// print(pointer);//输出链表**

**// pointer = delete\_link(pointer);**

**// n = 0;**

**// print(pointer);//输出链表**

**//free(pointer);//释放占用的内存**

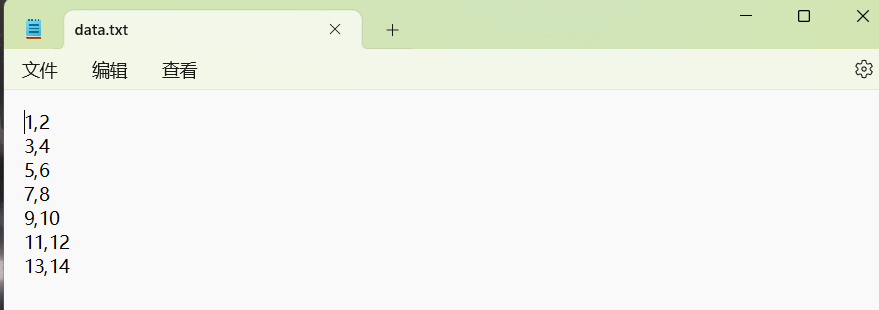
**system("pause");**

**exit(0);**

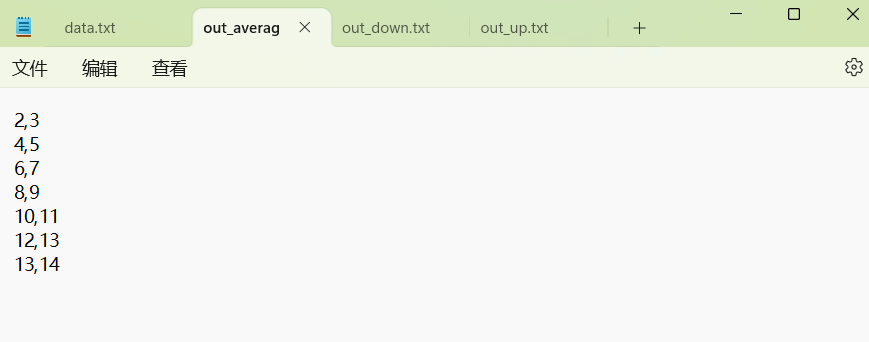
**return 0;**

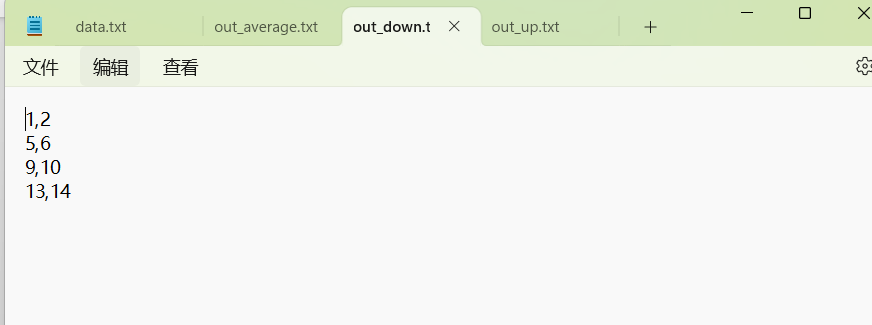
**}**

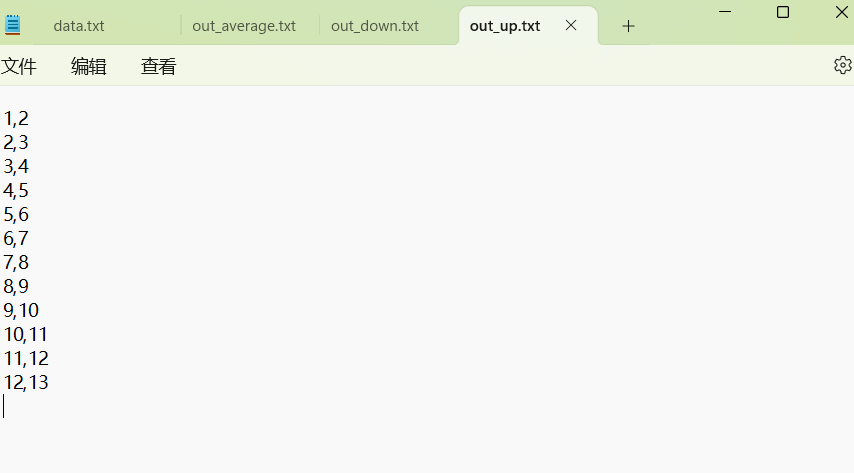
**5.3初始参数设置**

**

**5.4实验结果**

****

****

****

**5.5实验分析**

（1） 时间复杂度

①初始化函数：根据读取的数据个数执行n次操作，因此时间复杂度为O(N)。

②求数据元素个数函数：需要遍历链表的所有元素，因此时间复杂度为O(N)。

③插入、删除、取数据元素的函数：在链表中查找指定节点并执行操作，时间复杂度为O(N)。

④均值滤波：需要对每个结点进行滤波操作，因此时间复杂度为O(N)。

⑤上采样：需要对原数据的每两个相邻数据进行插值，遍历链表一次，时间复杂度为O(N)。

⑥下采样：需要对原数据进行间隔取样，循环次数为n/2，时间复杂度O(N)。

（2）边界条件

①初始化函数：需考虑所读取的数据个数边界。

②求元素个数函数：在链表为空或指针域指向NULL时结束，考虑链表为空的情况。

③插入、删除、取数据元素函数：需确保链表中存在指定节点。

④均值滤波、上采样、下采样：在链表为空或指针域指向NULL时结束，需要保证链表的结点个数大于等于1。另外，对第一个结点和最后一个结点可能需要进行特殊处理。