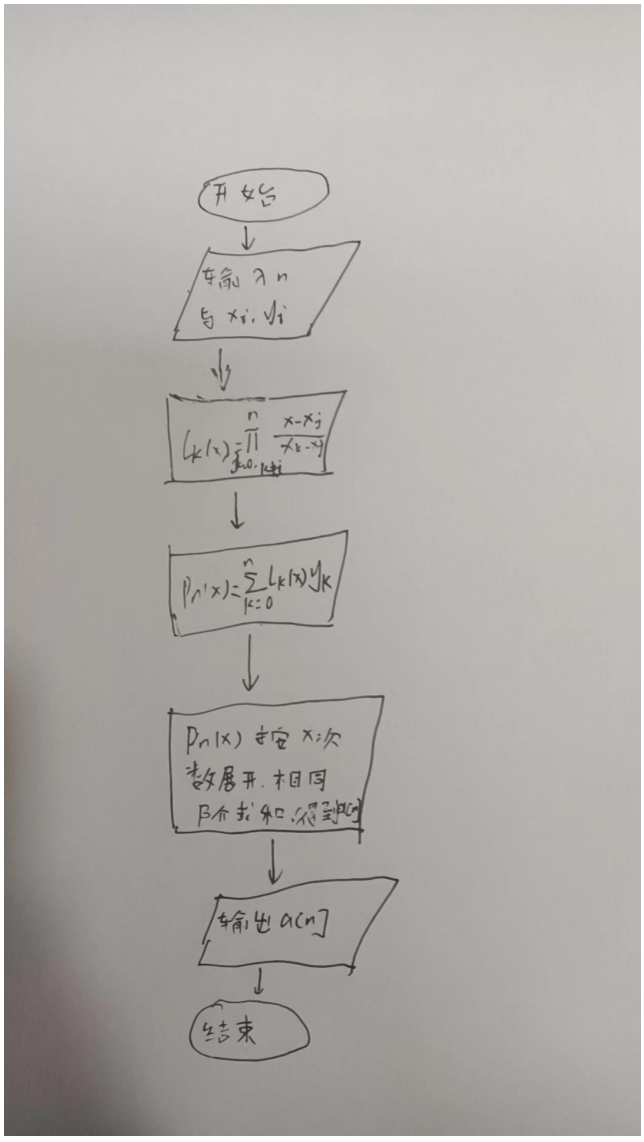
21374389 牛鹏军 210711

理论分析

【2】用C语言实现n次拉格朗日插值多项式$L(x)$的计算，插值函数原型为int lagPolynomial (int n, double * X, double * Y, double * a)，其中X为插值节点数组：$x_0<x_1<\cdots<x_n$，Y为函数值数组：y_0，y_1，…，y_n。函数返回n次插值多项式系数数组a：$L(x)=a[0]+a[1]x+\cdots+a[n]x^n$。

关于拉格朗日插值基本多项式求解，按照书上的公式写几个 for 循环就可以了，我感觉本题的难点在于如何从基本多项式得到最后的插值多项式，我选择了使用二维/一维指针，编写了求和、求积等函数，得到解答。

算法设计



编程实现

C 语言代码：

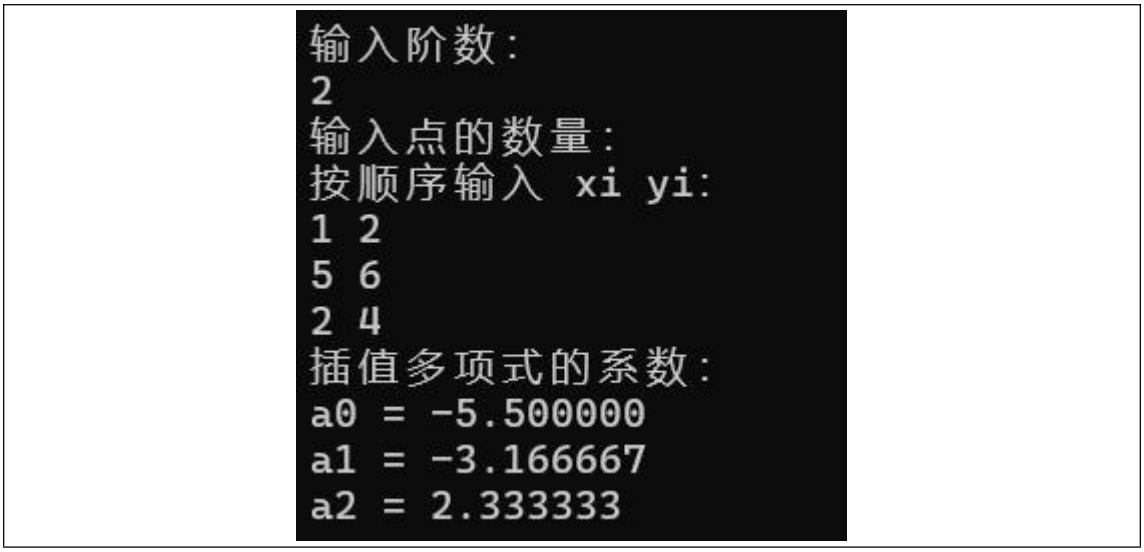
```
#include <stdio.h>
#include <stdlib.h>

void finalFactor(double *one, double *two, int numOfOne, int numOfTwo, double *ans, int n) {
	for (int i = 0; i <= numOfOne; i++) {
		for (int j = 0; j <= numOfTwo; j++) {
			ans[i + j] += one[i] * two[j];
		}
	}
}

void sum(double **ans, int n, double *qwer) {
	for (int i = 0; i <= n; i++) {
		for (int j = 0; j <= n; j++) {
			qwer[i] += ans[j][i];
		}
	}
}

void lagPolynomial(int n, double *X, double *Y, double *a) {
	double *fenmu = (double *)malloc((n + 1) * sizeof(double));
	double *temp = (double *)malloc((n + 1) * sizeof(double));
	double **ans = (double **)malloc((n + 1) * sizeof(double *));
	double *qwer = (double *)malloc((n + 1) * sizeof(double));

	if (fenmu == NULL || temp == NULL || ans == NULL || qwer == NULL) {
		printf("内存分配失败\n");
		exit(1);
	}

	for (int i = 0; i <= n; i++) {
		ans[i] = (double *)malloc((n + 1) * sizeof(double));
		if (ans[i] == NULL) {
			printf("内存分配失败\n");
			exit(1);
		}
	}

	for (int i = 0; i <= n; i++) {
		fenmu[i] = 1.0;
		temp[i] = 1.0;
```

```
            qwer[i] = 0.0;
        }

        for (int k = 0; k <= n; k++) {
            for (int j = 0; j <= n; j++) {
                if (j != k) {
                    double a[2] = {-X[j], 1};
                    finalFactor(a, temp, 1, 1, ans[k], n);
                    fenmu[k] *= (X[k] - X[j]);
                }
            }
            for (int i = 0; i <= n; i++) {
                ans[k][i] *= Y[k];
                ans[k][i] /= fenmu[k];
            }
            sum(ans, n, qwer);
        }

        for (int i = 0; i <= n; i++) {
            a[i] = qwer[i];
        }

        free(fenmu);
        free(temp);
        for (int i = 0; i <= n; i++) {
            free(ans[i]);
        }
        free(ans);
        free(qwer);
}

int main() {
        int n;
        double *x, *y, *a;
        printf("输入阶数：\n");
        scanf("%d", &n);

        if (n < 0) {
            printf("阶数必须为非负整数\n");
            return 1;
        }

        // 为 x 和 y 数组分配内存空间
        x = (double *)malloc((n + 1) * sizeof(double));
```

```
    y = (double *)malloc((n + 1) * sizeof(double));
    a = (double *)malloc((n + 1) * sizeof(double));

    // 检查内存分配是否成功
    if (x == NULL || y == NULL || a == NULL) {
        printf("内存分配失败\n");
        return 1;
    }

    printf("输入点的数量：\n");
    printf("按顺序输入 xi yi：\n");

    for (int i = 0; i <= n; i++) {
        scanf("%lf %lf", &x[i], &y[i]);
    }

    lagPolynomial(n, x, y, a);

    // 打印结果
    printf("插值多项式的系数：\n");
    for (int i = 0; i <= n; i++) {
        printf("a%d = %lf\n", i, a[i]);
    }

    // 释放内存
    free(x);
    free(y);
    free(a);

    return 0;
}
```

测试分析

当输入二阶求解时，得到了插值多项式系数输出，遗憾的是输出结果有一定谬误，应该是指针运算中间有问题，限于本人能力有限，只能做到这个程度。



结论

本题最大难点在于将基本多项式展开为插值多项式，我采用一种基础的方法，代码冗余性可能过多，以至于可维护性不高，还需要后续改正。