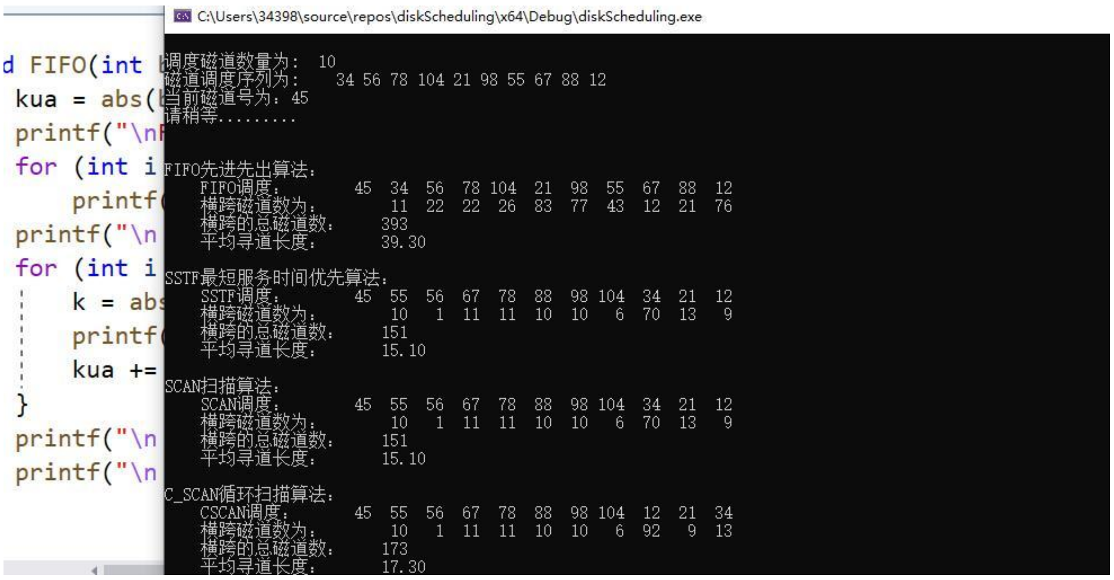
实验代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include <iomanip>
#include<Windows.h>
#define max 200
using namespace std;
int num; //磁盘数
//55 58 39 18 90 160 150 38 184
int request[100];     //请求磁盘序列

int kua;        //横跨的总数
int k;          //每次横跨的磁盘数
int re[100];        //复制初始序列
int r[100];         //记录每个算法执行后序列

void FIFO(int begin) {      //先进先出
    kua = abs(begin - request[0]);
    printf("\nFIFO 先进先出算法：\n      FIFO 调度：          %3d", begin);
    for (int i = 0; i < num; i++)
        printf(" %3d", request[i]);
    printf("\n      横跨磁道数为：          %3d", abs(begin - request[0]));
    for (int i = 1; i < num; i++) {
        k = abs(request[i - 1] - request[i]);
        printf(" %3d", k);
        kua += k;
    }
    printf("\n      横跨的总磁道数：       %3d", kua);
    printf("\n      平均寻道长度：          %.2f\n", 1.0 * kua / num);
}

int Smin(int b, int re[]) {     //返回离开始磁盘 b 最近的磁盘下标
    int min = abs(b - re[0]);
    int j = 0;
    for (int i = 1; i < num; i++)
        if (abs(b - re[i]) < min) {
            min = abs(b - re[i]);
            j = i;
        }
    return j;
```

```
}

void SSTF(int begin) {      //最短服务时间优先
    int c = 0, b = begin;
    printf("\nSSTF 最短服务时间优先算法：\n      SSTF 调度:           %3d", begin);
    for (int i = 0; i < num; i++) {
        c = Smin(b, re); //返回最近的磁道下标
        b = re[c]; //将最近的磁盘作为开始
        re[c] = 9999999; //将已经访问过的磁盘  设为很大值
        printf(" %3d", b);
        r[i] = b;
    }
    kua = abs(begin - r[0]);
    printf("\n      横跨磁道数为:           %3d", abs(begin - r[0]));
    for (int i = 1; i < num; i++) {
        k = abs(r[i - 1] - r[i]);
        printf(" %3d", k);
        kua += k;
    }
    printf("\n      横跨的总磁道数:         %3d", kua);
    printf("\n      平均寻道长度:           %.2f\n", 1.0 * kua / num);

}


void SCAN(int begin) {      //扫描算法
    int c = 0, b = begin;
    for (int i = 0; i < num; i++) //SSTF 时 re[]已改变
        re[i] = request[i];
    printf("\nSCAN 扫描算法：\n      SCAN 调度:           %3d", begin);
    for (int i = 0; i < num - 1; i++) {
        for (int j = 0; j < num - i - 1; j++) {
            if (re[j] > re[j + 1]) {
                re[j] = re[j] + re[j + 1];
                re[j + 1] = re[j] - re[j + 1];
                re[j] = re[j] - re[j + 1];
            }
        }
    }
    for (int i = 0; i < num; i++)
        if (re[i] > b) {
            printf(" %3d", re[i]);
            r[c++] = re[i];
        }
    for (int i = num - 1; i >= 0; i--)
        if (re[i] < b) {
            printf(" %3d", re[i]);
            r[c++] = re[i];
        }
    kua = abs(begin - r[0]);
    printf("\n        横跨磁道数为：          %3d", abs(begin - r[0]));
    for (int i = 1; i < num; i++) {
        k = abs(r[i - 1] - r[i]);
        printf(" %3d", k);
        kua += k;
    }
    printf("\n        横跨的总磁道数：        %3d", kua);
    printf("\n        平均寻道长度：          %.2f\n", 1.0 * kua / num);
}


void C_SCAN(int begin) {    //循环扫描
    int c = 0, b = begin;
    printf("\nC_SCAN 循环扫描算法：\n        CSCAN 调度：          %3d", begin);
    for (int i = 0; i < num; i++)
        if (re[i] > b) {
            printf(" %3d", re[i]);
            r[c++] = re[i];
        }
    for (int i = 0; i < num; i++)
        if (re[i] < b) {
            printf(" %3d", re[i]);
            r[c++] = re[i];
        }
    kua = abs(begin - r[0]);
    printf("\n        横跨磁道数为：          %3d", abs(begin - r[0]));
    for (int i = 1; i < num; i++) {
        k = abs(r[i - 1] - r[i]);
        printf(" %3d", k);
        kua += k;
    }
    printf("\n        横跨的总磁道数：        %3d", kua);
    printf("\n        平均寻道长度：          %.2f\n", 1.0 * kua / num);
}

int main() {
    int begin;    //开始磁盘位置
    int proceed;
```

```
	srand((unsigned int)time(NULL));
	while (true) {
		string data[4] = { "data0.txt","data1.txt","data2.txt","data3.txt" };
		printf("磁盘调度模拟实现\n\n 正在读取到从文件中数据.....\n\n");
		ifstream readData;
		int i = rand() % 4;
		readData.open(data[i]);
		Sleep(1200);
		printf("读取 data%d.txt 成功!!\n\n", i);

		readData >> num;
		printf("调度磁道数量为:   %d    ", num);

		printf("\n 磁道调度序列为:        ");
		for (int i = 0; i < num; i++) {
			readData >> request[i];
			re[i] = request[i];
			printf("%d ", request[i]);
		}
		printf("\n");

		readData >> begin;
		printf("当前磁道号为：%d          \n", begin);
		printf("请稍等.........\n\n");
		Sleep(1200);

		FIFO(begin);
		SSTF(begin);
		SCAN(begin);
		C_SCAN(begin);

		printf("\n 继续读取数据吗?(1 Y/0 N)");
		scanf_s("%d", &proceed);
		if (proceed == 0) {
			break;
		}
	}
	return 0;
}
```

实验结果：



```
d FIFO(int
 kua = abs(
 printf("\n
 for (int i
     printf(
 printf("\n
 for (int i
     k = abs
     printf(
     kua +=
 }
 printf("\n
 printf("\n
```

C:\Users\34398\source\repos\diskScheduling\x64\Debug\diskScheduling.exe

```
调度磁道数量为:  10
磁道调度序列为:    34 56 78 104 21 98 55 67 88 12
当前磁道号为: 45
请稍等.........


FIFO先进先出算法:
    FIFO调度:        45   34   56   78  104   21   98   55   67   88   12
    横跨磁道数为:          11   22   22   26   83   77   43   12   21   76
    横跨的总磁道数:      393
    平均寻道长度:        39.30

SSTF最短服务时间优先算法:
    SSTF调度:        45   55   56   67   78   88   98  104   34   21   12
    横跨磁道数为:          10    1   11   11   10   10    6   70   13    9
    横跨的总磁道数:      151
    平均寻道长度:        15.10

SCAN扫描算法:
    SCAN调度:        45   55   56   67   78   88   98  104   34   21   12
    横跨磁道数为:          10    1   11   11   10   10    6   70   13    9
    横跨的总磁道数:      151
    平均寻道长度:        15.10

C_SCAN循环扫描算法:
    CSCAN调度:       45   55   56   67   78   88   98  104   12   21   34
    横跨磁道数为:          10    1   11   11   10   10    6   92    9   13
    横跨的总磁道数:      173
    平均寻道长度:        17.30
```