目录

[一. 万年历 2](#_Toc131071228)

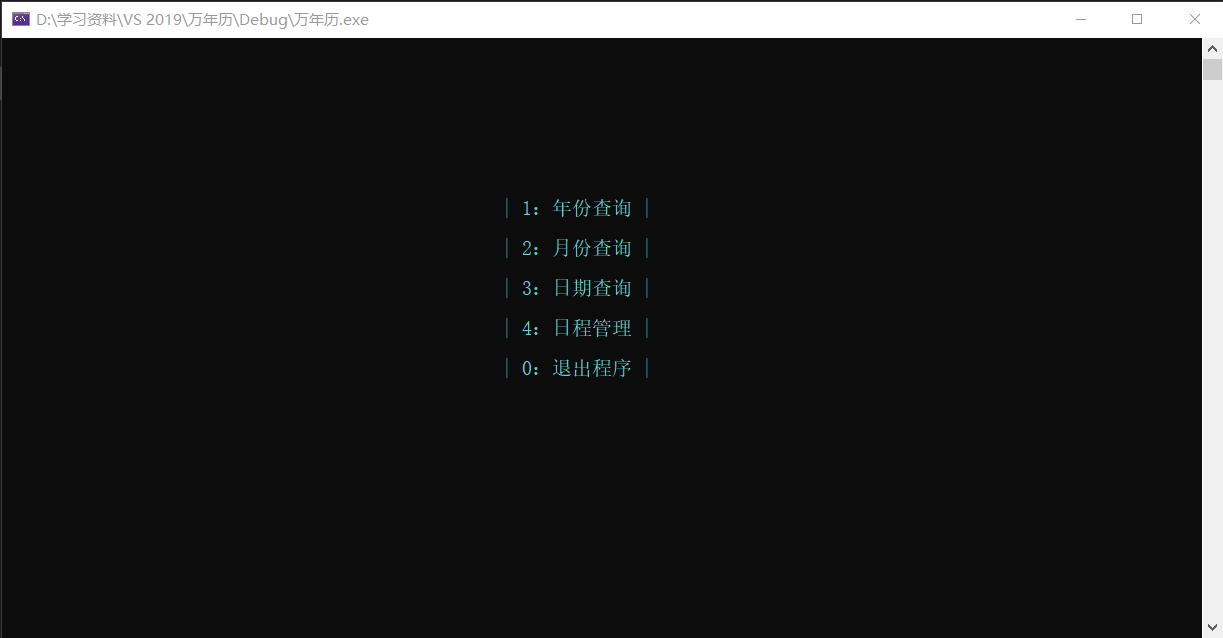
[2.1 项目简介 2](#_Toc131071229)

[2.2 设计思路 3](#_Toc131071230)

# 万年历

## 2.1 项目简介

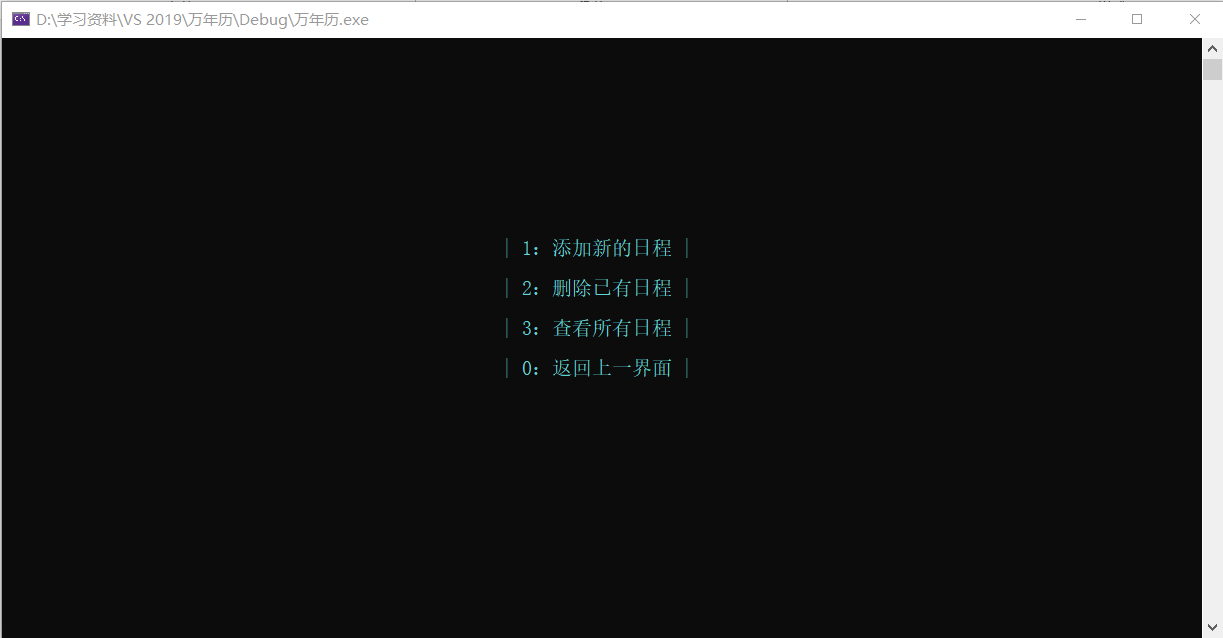
* 主菜单界面



* 年份、月份、日期查询（与文档要求一致，略）

注：万年历的查询范围理论上支持0到无穷大（int数据范围内）

* 日程管理（二级菜单）



1. 添加新的日程：用户输入年月日与事件来添加新的日程
2. 删除已有日程：通过打印所有已有的日程，用户通过输入日程前的编号来选择删除已有的日程
3. 查看所有日程：打印所已添加的日程

注：日程管理功能可将用户添加的日程保存到本地文档中

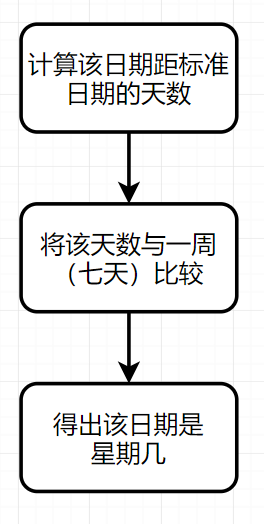
## 2.2 设计思路

* 总体思路：对于实现一个万年历，最主要的只有两个点：1. 年份是否是闰年的判断；2. 该日期是星期几的问题。

对于第一个问题比较简单，那么主要就是第二个问题：如何判断某个日期具体是星期几？

最简单的方法是找一个具体星期已知的日期，然后通过计算所求日期距已经日期的天数，再和一周（七天）进行比较，这样就可以计算出具体某一天是星期几。通过简单的百度，发现目前的万年历都是以1900年1月1日（星期一）作为标准日期来计算，那么该程序也将以此为基准（这也是确实可查的最早的已知日期）

* 计算具体某天是星期几

1. 对天数的计算则涉及到闰、平年的判断，以及该日期是在1900年1月1日之后还是之前（该程序支持0到无穷大的查找）
2. 将步骤一得出的天数进行数据处理，然后与标准日期比较，计算得出该日期具体是星期几。在比较过程中也存在判断该日期是在标准日期之前还是之后的问题（两者相对于标准日期的计算方式不同）

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: IsYear

\* 功能说明: 判断输入的年份是闰年还是平年

\* 形 参: year ：年份

\* 返 回 值: 1. 闰年

\* 0. 平年

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

Status IsYear(int year)

{

//1. 普通年能被4整除，但不能被100整除是闰年

//2. 世纪年能被400整除是闰年

//总结：能被4整除且不能被100整除或能被400整除的是闰年

if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)

return SUCCESS; //闰年

else

return ERROR; //平年

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: SumDayLater

\* 功能说明: 计算输入的日期距1900/01/01的天数（该日期位于1900/01/01之后）

\* 形 参: 1. year ：年

\* 2. month：月 [1,12]

\* 3. day ：日 [1,31]

\* 返 回 值: 当前日期距1900/01/01的天数（正数）；返回值类型为long long以支持尽量大的年份查询

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

long long SumDayLater(int year, int month, int day)

{

long long day\_y = 0;

int day\_m = 0, temp;

//计算day\_y

day\_y = (year - 1900) \* 365;

temp = (year - 1900) / 4;

if ((year - 1900) % 4 == 0 && year != 1900)

day\_y += temp \* 1 - 1;

else

day\_y += temp \* 1;

//计算day\_m

switch (month)

{

case 12: day\_m += 30;

case 11: day\_m += 31;

case 10: day\_m += 30;

case 9 : day\_m += 31;

case 8 : day\_m += 31;

case 7 : day\_m += 30;

case 6 : day\_m += 31;

case 5 : day\_m += 30;

case 4 : day\_m += 31;

case 3 : {

if (IsYear(year) == 1)

day\_m += 29;

else

day\_m += 28;

}

case 2 : day\_m += 31;

case 1 : break;

}

//计算总的天数

return day\_y + day\_m + day - 1;

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: SumDayBefore

\* 功能说明: 计算输入的日期距1900/01/01的天数（该日期位于1900/01/01之前）

\* 形 参: 1. year ：年

\* 2. month：月 [1,12]

\* 3. day ：日 [1,31]

\* 返 回 值: 当前日期距1900/01/01的天数；返回值类型为long long以支持尽量大的年份查询

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

long long SumDayBefore(int year, int month, int day)

{ //以1899/12/31为基准（-1）

long long day\_y = 0;

int day\_m = 0, temp;

//计算day\_y

day\_y = (1900 - year) \* 365;

temp = (1900 - year) / 4;

day\_y += temp \* 1 - 1;

//计算day\_m

switch (month)

{

case 12: day\_m += 30;

case 11: day\_m += 31;

case 10: day\_m += 30;

case 9: day\_m += 31;

case 8: day\_m += 31;

case 7: day\_m += 30;

case 6: day\_m += 31;

case 5: day\_m += 30;

case 4: day\_m += 31;

case 3: {

if (IsYear(year) == 1)

day\_m += 29;

else

day\_m += 28;

}

case 2: day\_m += 31;

case 1: break;

}

//计算总的天数

return day\_y - day\_m - day + 2;

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: WeekDay

\* 功能说明: 计算输入的日期是周几

\* 形 参: 1. year ：年

\* 2. month：月 [1,12]

\* 3. day ：日 [1,31]

\* 返 回 值: 0. 0：周日

\* 1. 1：周一

\* 2. 2：周二

\* 3. 3：周三

\* 4. 4：周四

\* 5. 5：周五

\* 6. 6：周六

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

int WeekDay(int year, int month, int day)

{

int temp;

if (year >= 1900)

{

temp = SumDayLater(year, month, day) % 7;

}

else

{

temp = SumDayBefore(year, month, day) % 7;

if(temp != 0)

temp = 7 - temp;

/\*

\*temp计算表

0 1 0

1 0 6

2 6 5

3 5 4

4 4 3

5 3 2

6 2 1

\*/

}

switch (temp)

{

case 0: return 1; break;

case 1: return 2; break;

case 2: return 3; break;

case 3: return 4; break;

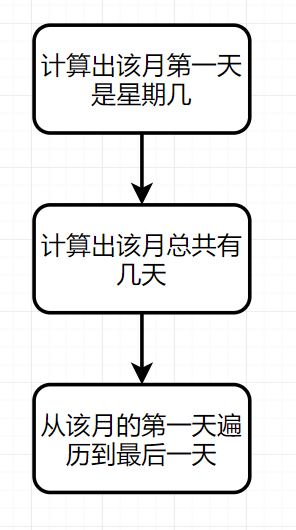
case 4: return 5; break;

case 5: return 6; break;

case 6: return 0; break;

}

}

* 打印具体一个月的日历

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: PrintMonth

\* 功能说明: 打印输入的某年某月的日历

\* 形 参: 1. year ：年

\* 2. month：月 [1,12]

\* 返 回 值: 打印成功返回1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

Status PrintMonth(int year,int month)

{

int week, day;

week = WeekDay(year, month, 1); //该月第一天的星期

//计算该月的天数

if (month == 2)

{

if (IsYear(year) == 1)

day = 29; //闰年二月29天

else

day = 28; //平年二月28天

}

else if (month == 4 || month == 6 || month == 9 || month == 11)

day = 30; //小月30天

else

day = 31; //大月31天

//打印日历模板

printf("---------------------------------------------------\n");

printf("\t\t %d年\t%d月\n", year, month);

printf("---------------------------------------------------\n");

printf("日\t一\t二\t三\t四\t五\t六\n");

//打印日历信息

int month\_day, month\_week, flag = 1;

for (month\_day = 1, month\_week = week; day > 0; month\_day++, day--, month\_week++)

{

//打印每一天的信息

if (flag == 1) //确定该月第一天在日历中的位置

{

int cnt = month\_week;

while (cnt--) //把第一天调整到合适的位置

{

printf("\t");

}

flag = 0; //标记第一天已经调整过

}

printf("%d\t", month\_day); //打印每一天的信息

//打印一周后换行（在周六之后换行）

if (month\_week == 6)

{

printf("\n");

month\_week = -1; //+1=0，重新在周日开始

}

}

printf("\n");

return SUCCESS;

}

* 打印一整年的日历（调用12次PrintMonth函数，略）
* 日程管理：添加、删除、查看

1. 通过单链表来保存日程的数据
2. 添加：通过用户输入的数据在单链表表尾中添加新的结点

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: AddEvent

\* 功能说明: 在指定日期添加日程

\* 形 参: 无

\* 返 回 值: 1. SUCCESS：添加成功

\* 2. ERROR ：添加失败（1.该日期已有日程 2.输入日期不合法）

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

Status AddEvent(LinkedList L)

{

int year=-1, month=-1, day=-1; //提前赋值原理同main函数处注释

char event[51] = { '\0' };

//输入年月日

print\_toxy(48, 12);

printf("依次输入年月日：");

scanf\_s("%d %d %d", &year, &month, &day);

while (getchar() != '\n'); //输入容错，吃掉非法字符

//输入日期范围判断

if (year < 0|| month < 1 || month>12||day < 1 || day>31)

{

print\_toxy(48, 14);

printf("输入日期范围有误！！！\n");

return ERROR; //输入日期不合法返回ERROR

}

//输入日程事件

print\_toxy(48, 14);

printf("输入要添加的日程（不多于25字）：");

scanf\_s("%s", event, 51);

//插入结点

char flag = InsertList(L,year, month, day, event);

save(L); //保存数据

if (!flag)

return ERROR; //结点插入失败

else

return SUCCESS;

}

1. 删除：通过用户输入的编号（该结点在单链表中的位置）在单链表中删除该结点

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: DeleteEvent

\* 功能说明: 在指定日期删除日程

\* 形 参: num：指定删除的日程的序号

\* 返 回 值: 1. SUCCESS：删除成功

\* 2. ERROR ：删除失败（该序号不合法）

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

Status DeleteEvent(LinkedList L, int num)

{

//根据序号找到待删除结点

LNode\* p = L;

while (num--)

{

p = p->next;

if (!p)

{

printf("输入序号范围有误！！！\n");

return ERROR;

}

}

//删除结点

DeleteList(L, p->year, p->month, p->day);

save(L); //保存数据

return SUCCESS;

}

1. 查看：遍历并打印单链表

*注：对单链表的基本操作功能函数不展示，详见文件LinkedList.c*

* 日程管理：数据的本地保存

1. 在程序开始运行时，首先从当前目录下寻找文档进行数据的读取，若找不到相关文档则新建一个空白文档

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: load

\* 功能说明: 1. 从文件中加载数据，若目录下不存在该文件，则新建一个文件用于保存数据

\* 2. 对单链表的结点数据进行保存，保存到一个二进制文件中（.bin）

\* 形 参: L：单链表，用于保存从文件中加载出来的数据

\* 返 回 值: 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void load(LinkedList L)

{

FILE\* fp;

errno\_t err;

err = fopen\_s(&fp, "calendar.bin", "rb"); //rb；只读

if (err) //没有文件则创建一个文件

{

err = fopen\_s(&fp, "calendar.bin", "wb"); //wb：只写，新建

if (!fp)

exit(1);

fclose(fp);

}

else

{

if (!fp)

exit(1);

LNode\* p = L;

LNode\* q = (LNode\*)malloc(sizeof(LNode));

if (!q)

exit(1);

//fread()：读取fp数据到q

while (fread(q, sizeof(LNode), 1, fp))

{

p->next = q; //把q插入在p后面

p = p->next;

q = q->next;

q = (LNode\*)malloc(sizeof(LNode));

if (!q)

exit(1);

}

fclose(fp);

}

}

1. 在用户进行日程的添加、删除操作后，程序将链表的数据重新写入文档中（由于步骤一，不存在找不到文档），进行数据实时保存

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: load

\* 功能说明: 1. 把单链表的数据保存到文件中，该函数必须在load调用后才能调用，以保证目标文件的存在

\* 2. 从头到尾重新保存一遍单链表节点的数据，将旧数据覆盖

\* 形 参: L：单链表，将该单链表的数据保存到文件中

\* 返 回 值: 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void save(LinkedList L)

{

FILE\* fp;

errno\_t err;

err = fopen\_s(&fp, "calendar.bin", "wb");

if (!fp)

exit(1);

LNode\* p = L->next;

while (p)

{

fwrite(p, sizeof(LNode), 1, fp);

p = p->next;

}

fclose(fp);

}