## 实验3 线性结构的综合应用

学号：2022217587 姓名：党存远 专业：物联网工程

|  |  |  |  |
| --- | --- | --- | --- |
| **“数据结构课程实验”验收细则** | | | |
| **成绩等级** | **具体表现** | **教师评分**  **（百分制）** | |
| A（100-90] | 1）能较快完成实验；2）编码正确；3）能完整正确地回答教师提问； | □ |  |
| B（90-80] | 1）能够在规定时间内完成实验；2）编码正确；3）能正确地回答教师提问； | □ |  |
| C（80-70] | 1）能够在规定时间内完成实验； | □ |  |
| D（70-60] | 1）虽完成实验但未能在规定时间内完成验收的； | □ |  |
| E（<60） | 1）未能进行验收的； | □ |  |

**3.1 实验目的**

（1）理解三种线性结构栈、队列与线性表的特点；

（2）熟练掌握基于顺序表与基于链表的两种存储结构及有关算法的设计。

（3）根据具体问题的需要，设计出合理的线性结构，并设计相关算法。

**3.2设计算法实现下列问题的求解。**

<1>设计计算器系统，能对包含四则运算及括号的表达式计算其结果。

<2> “约瑟夫环问题”：约瑟夫，是一个古犹太人，曾经在一次罗马叛乱中担任将军，后来战败，他和朋友及另外39个人躲在一口井里。被发现后，罗马人表示只要投降就不死，约瑟夫和朋友想投降，可是其他人坚决不同意。怎么办呢? 让41个人围成一个圆圈，从第一个人开始报数，数到3的那个人被旁边的人杀死。这样就可以避免自杀了，因为犹太人的信仰是禁止自杀的。约瑟夫和朋友站在什么位置才保住了性命呢？

根据上述要求，设计算法输出约瑟夫和朋友所在位置。

**3.3 验收内容：对三种线性结构栈、队列与线性表的理解及算法实现。**

**3.4 运行结果截图及说明**

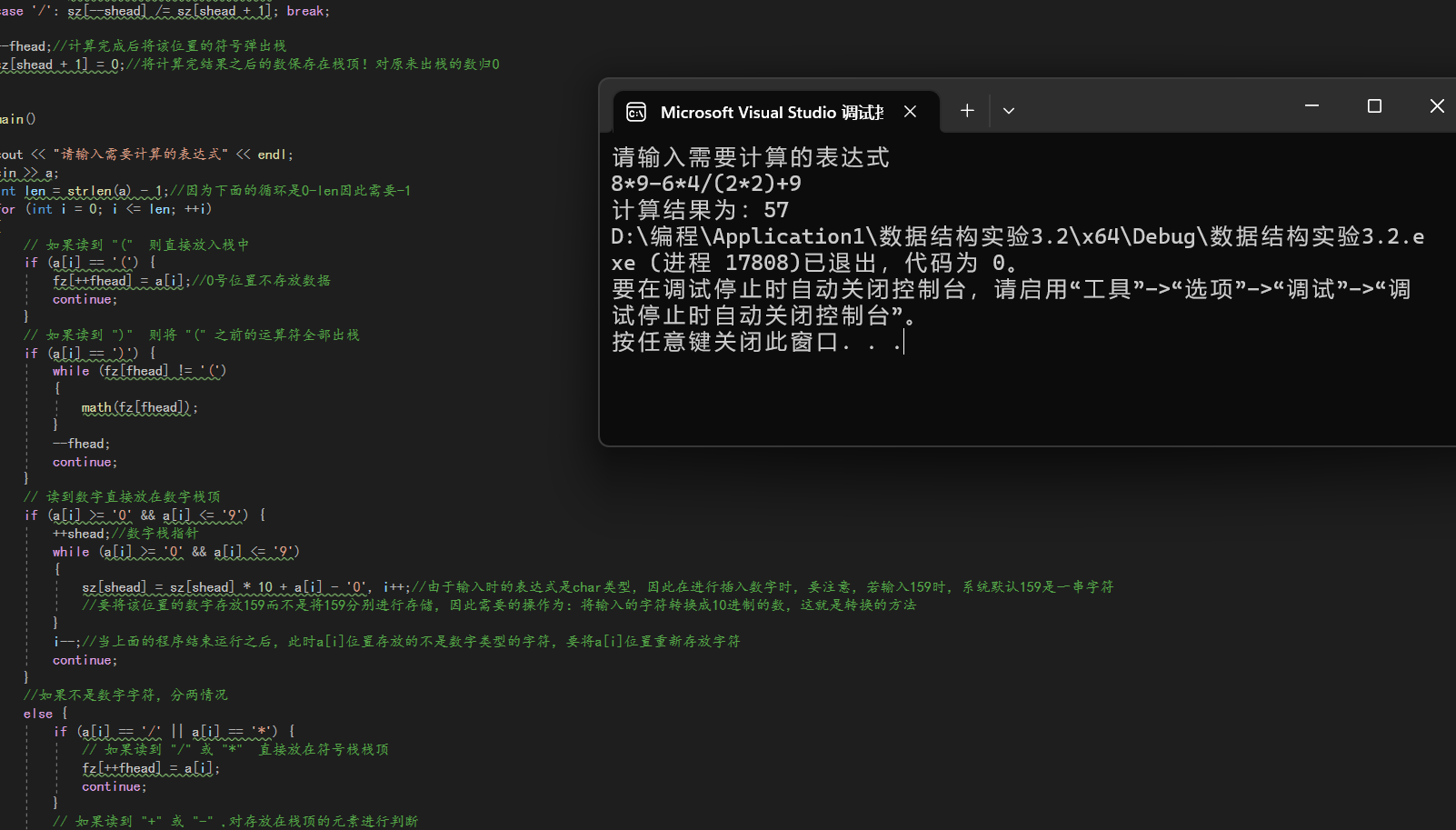


图1：简单计算器的实现

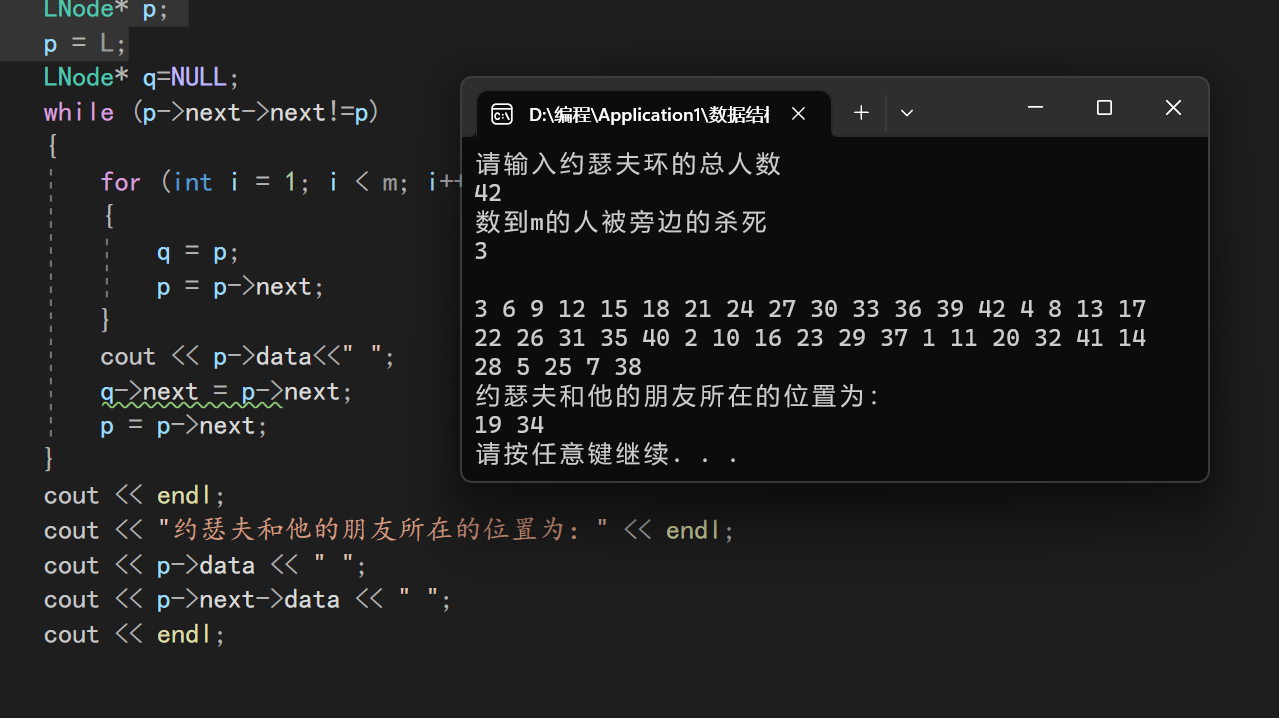


图2：约瑟夫问题单循环链表实现

**3.5 附源代码**

1、

#include<iostream>

using namespace std;

char a[100];

char fz[100]; // 符号栈

int sz[100]; // 数字栈

int fhead = 0; // 符号栈指针

int shead = 0; // 数字栈指针

void math(char f) // 从数字栈中取出栈顶的两个数字 进行 f 运算 结果继续放在栈中

{

switch (f)

{

//计算时，一定是后出栈的数，对先出栈的数进行计算

case '+': sz[--shead] += sz[shead + 1]; break;

case '-': sz[--shead] -= sz[shead + 1]; break;

case '\*': sz[--shead] \*= sz[shead + 1]; break;

case '/': sz[--shead] /= sz[shead + 1]; break;

}

--fhead;//计算完成后将该位置的符号弹出栈

sz[shead + 1] = 0;//将计算完结果之后的数保存在栈顶！对原来出栈的数归0

}

int main()

{

cout << "请输入需要计算的表达式" << endl;

cin >> a;

int len = strlen(a) - 1;//因为下面的循环是0-len因此需要-1

for (int i = 0; i <= len; ++i)

{

// 如果读到 "(" 则直接放入栈中

if (a[i] == '(') {

fz[++fhead] = a[i];//0号位置不存放数据

continue;

}

// 如果读到 ")" 则将 "(" 之前的运算符全部出栈

if (a[i] == ')') {

while (fz[fhead] != '(')

{

math(fz[fhead]);

}

--fhead;

continue;

}

// 读到数字直接放在数字栈顶

if (a[i] >= '0' && a[i] <= '9') {

++shead;//数字栈指针

while (a[i] >= '0' && a[i] <= '9')

{

sz[shead] = sz[shead] \* 10 + a[i] - '0', i++;//由于输入时的表达式是char类型，因此在进行插入数字时，要注意，若输入159时，系统默认159是一串字符

//要将该位置的数字存放159而不是将159分别进行存储，因此需要的操作为：将输入的字符转换成10进制的数，这就是转换的方法

}

i--;//当上面的程序结束运行之后，此时a[i]位置存放的不是数字类型的字符，要将a[i]位置重新存放字符

continue;

}

//如果不是数字字符，分两情况

else {

if (a[i] == '/' || a[i] == '\*') {

// 如果读到 "/" 或 "\*" 直接放在符号栈栈顶

fz[++fhead] = a[i];

continue;

}

// 如果读到 "+" 或 "-" ,对存放在栈顶的元素进行判断

else

{

// 则将栈顶 "/" "\*" 全部弹出，因为乘号与除号的优先级前，需要优先进行运算，运算结束的标志是读到下一个'+'或'-'

while (fz[fhead] == '\*' || fz[fhead] == '/' )

{

math(fz[fhead]);

}

fz[++fhead] = a[i];//将'\*'与'/'运算完成以后，在将新的符号放置在符号栈的栈顶

}

}

}

while (fhead != 0) {

math(fz[fhead]);//当前面的操作做完以后，可能此时符号栈中还有一些符号未被计算，直到符号栈中没有剩余的元素操作才结束

}

cout << "计算结果为：";

cout << sz[shead];//最后存放在数字栈中的元素就是最终的结果

return 0;

}

2、

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define OVERFLOW -2

typedef int Status;

typedef char ElemType;

typedef class LNode

{

public:

Status data;

LNode\* next;

}LNode,\*LinkList;

Status InitList(LinkList& L)

{

L = new LNode;

if (L == NULL)

{

cout << "内存申请失败" << endl;

return OVERFLOW;

}

L->next = NULL;

return OK;

}

//尾插法创建单链表

void CreateLinkList\_R(LinkList& L, int n)//创建不具有头结点的单链表

{

LinkList r;

r = L;

LNode\* p;

r->data = 1;

for (int i = 1; i < n;i++)

{

p = new LNode;

p->data = i+1;

r->next = p;

p->next = L;

r = p;

}

}

void ShowList(LinkList L)

{

LNode\* p;

p = L;

cout << L->data<<" ";

p = p->next;

while (p!= L)

{

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

void Joseph\_LinkList(LinkList L, int n, int m)//n代表总人数，m代表数到第m个数死亡

{

LNode\* p;

p = L;

LNode\* q=NULL;

while (p->next->next!=p)

{

for (int i = 1; i < m; i++)

{

q = p;

p = p->next;

}

cout << p->data<<" ";

q->next = p->next;

p = p->next;

}

cout << endl;

cout << "约瑟夫和他的朋友所在的位置为：" << endl;

cout << p->data << " ";

cout << p->next->data << " ";

cout << endl;

}

int main()

{

LinkList L;

InitList(L);

cout << "请输入约瑟夫环的总人数" << endl;

int n = 0;

cin >> n;

CreateLinkList\_R(L, n);

//cout << GetLenth(L) << endl;

cout << "数到m的人被旁边的杀死" << endl;

int m = 0;

cin >> m;

cout << endl;

Joseph\_LinkList(L, n, m);

//ShowList(L);

system("pause");

return 0;

}

**3.6 调试过程中出现的bug及总结**

1、在进行简单计算器编写的过程中，首先创建了两个栈，一个栈用于存放数字，另一个栈用于存放符号，对于(‘+’ ‘-’号分为一种情况，‘\*’ ‘/’号一种情况， ‘(’ ‘)’分为另一种情况)

2、首先从左至右依次扫描算术式，遇到数字存放进数字栈，特别注意：【由于输入的算术式是符号类型的，因此对于算数式存放进数字栈时要将字符类型的数字转换成10进制整数，也就是while(a[i]>= ‘0’&&a[i] <=‘9’){num=num\*10+a[i]- ‘0’}】

3、当出现‘+’ ‘-’时，将‘+’ ‘-’存入符号栈。

4、当出现‘\*’ ‘/’时，将‘\*’ ‘/’放入符号栈的栈顶。之后继续放入符号

此时：若出现符号为‘\*’ ‘/’则重复操作，若出现符号为‘+’ ‘-’时，则先将栈顶的‘\*’ ‘/’进行运算，运算过程：【从数字栈中弹出两个数，后出的数对前出的数进行‘\*’ ‘/’运算】，直到将所有的‘\*’ ‘/’运算完毕之后，再将‘+’ ‘-’入栈

5、当出现‘(’时，对后面的数依次进行访问直到出现‘)’停止访问，在访问过程中同时对‘()’中数进行运算。

6、当符号栈中所有的元素全部运算完毕之后，最后留在数字栈中的结果就是运算结果。

7、在进行约瑟夫问题的求解过程中，第一次设置了头结点，头结点中不存放数据，导致循环途中访问时出现错误。在取消头结点之后问题解决。

8、当循环结束时，条件是该链表中仅剩下两个结点，此时结束条件是：p->next->next==p;//此时问题得到解决