完成日期：2016.10.25

**实验一**

* 问题描述：

1.分别写出下列函数的递归算法和迭代算法，并求出n=10时的函数值。

　Fib(n)= n 当n=0或n=1

Fib(n-2)+Fib(n-1) 当n≥2

* 算法思想：

递归算法运用递归函数，多次自我调用。

迭代算法运用for ，进行多次运算。

* 设计描述：

1. 因为最终结果可能值很大，所以用long long int存储返回的结果。
2. main函数中调用递归函数Fib算出递归出的结果，调用迭代函数Fib2算出迭代出的结果，两结果比较。
3. 算法描述：

long long int Fib(int n)//递归函数

{ if(n==0或n==1)

返回 n;

else

返回 Fib(n-2)+Fib(n-1);

}

long long int Fib2(int n)//迭代

{

if(n==0或n==1)

返回 n;

for(i=2;i<=n;i++)

{ s=first+second;//第三个数等于前两个相加

first=second;//往下取数

second=s;

}

返回 second;

}

main()

{

s=Fib(10);

r=Fib2(10);

打印s,r;

}

* 源代码

（文件名：递归迭代.c）

#include<stdio.h>

#include<stdlib.h>

long long int Fib(int n);

long long int Fib(int n)//递归

{ if(n==0||n==1)

return n;

else

return Fib(n-2)+Fib(n-1);

}

long long int Fib2(int n)//迭代

{ int first=0;

long long int second=1;

int s;

int i;

if(n==0||n==1)

return n;

for(i=2;i<=n;i++)

{ s=first+second;//第三个数等于前两个相加

first=second;//往下取数

second=s;

}

return second;

}

main()

{ long long int s,r;

int n;

printf("Please input the number:");

scanf("%d",&n);

s=Fib(n);//递归

r=Fib2(n);//迭代

printf("递归Fib %d=%lld\n",n,s);

printf("迭代Fib %d=%lld",n,r);

return 0;

system("pause");

}

* 测试结果

1.输入：不用输入(默认为10时)

输出：递归Fib 10=55

迭代Fib 10=55

2.测试两个算法时间：

因1000数据过大，运行很慢，所以测了30的

删除代码中迭代，得到递归Fib 50= 832040 2.293s

删除代码中递归，得到迭代Fib 50=832040 2.101s

* 用户使用说明：

运行环境：Dev C++

首先输入要计算的数据，然后静候结果。

* 心得体会

遇到的问题：迭代算法时开始没有把前两个数单独写出，所以结果不对。之后按照算法一步步手算时改正了。

缺点：数据大时程序运行过慢。

心得：处理问题时，递归和迭代算法应该有选择的使用，要考虑运行时间和占用内存，选择最优解。

**实验二**

* 问题描述

假设以带头结点的循环链表表示队列，并且只设一个指针指向队尾元素结点而不设头指针，试编写相应的队列初始化、入队列、出队列和判断队列状态的算法。

　利用上述算法完成下面的各操作，并在每一操作后输出队列状态。

　1)下列元素逐一入队：5,7,3,8,55 状态：5个元素

　2)3个元素出队　　　 状态：２个元素

3)再２个元素出队　　状态：队空

　4)再１个元素出队　　状态：队空（指示下溢）

* 算法思想：

用循环链表表示队列，尾指针指向头结点。入队列把新元素连在尾部，出队列通过尾指针找到头结点后的元素，出队列。用lenth计队列长度，判断状态。

* 设计描述

1. 数据结构：

typedef struct QNode{

int data;

struct QNode \*next;

}QNode,\*QueuePtr; //定义队列中元素

typedef struct{

QueuePtr rear;//队尾指针

int lenth; //队列长度

}LinkQueue,\*LinkQuPtr;//定义队列

1. 处理流程：把一个个数据EnQueue进队列，然后调用output函数，输出状态。先三个元素出队，main调用DeQueue函数出队，然后调用output函数。同理，两个元素出队。然后继续调用DeQueue函数，判断下返回值，若返回ERROR，则表示溢出，提示下溢。
2. 算法：

#define OVERFLOW -1

#define OK 1

#define ERROR 0

typedef struct QNode{

int data;

struct QNode \*next;

}QNode,\*QueuePtr;

typedef struct{

QueuePtr rear;//队尾指针

int lenth; //队列长度

}LinkQueue,\*LinkQuPtr;

函数类比书上的,头指针改成尾指针的next.

int InitQueue(&W)//构造一个空队列Q

int EnQueue(&W,int e)//插入元素e为Q的新的队尾元素

int DeQueue(&W)

//若队列不空，则删除Q的队头元素，用e返回其值，并返回OK；

//否则返回ERROR

void output(& W)

{

int count;

count=W->lenth;

printf("The queue is:\n");

while(count>0)

{打印元素 ;count--;

}//输出队列

if(W->lenth==0)

printf("NULL\n");

else

printf("have %d elements.\n",W->lenth);

}

main()

{ LinkQuPtr Q;

QueuePtr A;

Q=(LinkQuPtr)malloc(sizeof(LinkQueue));

int a,b,c,d,count,f,l;

char n;

a=InitQueue(Q);//初始化队列

if(a==0)

printf("InitQueue Error!\n");

else

{ printf("please input the number you want to save:\n");

count=0;

while(count<5)//按照题目要求，一共输入五个数

{ scanf("%d",&n);

EnQueue(Q,n);

count++;

}//元素依次入队

output(Q);

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

count=0;

while(count<3)

{ b=DeQueue(Q);

if(b==1) count++;

else printf("ERROR!\n");

}//三个元素出队

output(Q);

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

count=0;

while(count<2)

{ c=DeQueue(Q);

if(c==1) count++;

else printf("ERROR!\n");

}//两个元素出队

output(Q);

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

count=0;

while(count<1)

{ d=DeQueue(Q);

if(d==1) count++;

else if(d==0) {printf("OVERFLOW!\n"); count++;}//删除返回ERROR，提示下溢

else printf("ERROR!\n");

}//一个元素出队

output(Q);

}

}

* 源程序：

（文件名：循环链表队列.c）

#include<stdio.h>

#include<stdlib.h>

#define OVERFLOW -1

#define OK 1

#define ERROR 0

typedef struct QNode{

int data;

struct QNode \*next;

}QNode,\*QueuePtr;

QueuePtr p;

typedef struct{

// QueuePtr front;//不设头指针

QueuePtr rear;//队尾指针

int lenth; //队列长度

}LinkQueue,\*LinkQuPtr;

int InitQueue(LinkQuPtr W);

int EnQueue(LinkQuPtr W,int e);

int DeQueue(LinkQuPtr W);

void output(LinkQuPtr W);

int InitQueue(LinkQuPtr W)//构造一个空队列Q

{ W->rear=(QueuePtr)malloc(sizeof(QNode));

W->rear->next=(QueuePtr)malloc(sizeof(QNode));

if(!W->rear) exit(OVERFLOW);

W->rear=W->rear->next; //将队尾指针指向头结点

// (W->rear)->next=NULL;

W->lenth=0;//长度归零

return OK;

}

int EnQueue(LinkQuPtr W,int e)//插入元素e为Q的新的队尾元素

{ QNode \*j=(QueuePtr)malloc(sizeof(QNode));

if(!j) exit(OVERFLOW);

j->data=e;//值传入j

W->rear=j;

W->lenth++;//长度加一

return OK;

}

int DeQueue(LinkQuPtr W)

//若队列不空，则删除Q的队头元素，用e返回其值，并返回OK；

//否则返回ERROR

{ if(W->lenth==0)

return ERROR;

QueuePtr p;

p=W->rear->next;

if(W->rear==p)//只剩一个结点

{ W->rear=W->rear->next;

W->rear->next=p->next;

}

else

W->rear->next->next=p->next;

W->lenth--;

free(p);

return OK;

}

void output(LinkQuPtr W)

{

int count;

count=W->lenth;

printf("The queue is:\n");

while(count>0)

{printf("%d ",(W->rear-2\*count+2)->data); count--;

}//输出队列

printf("\n");

//QNode p;

if(W->lenth==0)

printf("NULL\n");

else

printf("have %d elements.\n",W->lenth);

}

main()

{ LinkQuPtr Q;

QueuePtr A;

Q=(LinkQuPtr)malloc(sizeof(LinkQueue));

int a,b,c,d,count,f,l;

char n;

a=InitQueue(Q);//初始化队列

if(a==0)

printf("InitQueue Error!\n");

else

{ printf("please input the number you want to save:\n");

count=0;

while(count<5)//按照题目要求，一共输入五个数

{ scanf("%d",&n);

EnQueue(Q,n);

count++;

}//元素依次入队

output(Q);

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

count=0;

while(count<3)

{ b=DeQueue(Q);

if(b==1) count++;

else printf("ERROR!\n");

}//三个元素出队

output(Q);

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

count=0;

while(count<2)

{ c=DeQueue(Q);

if(c==1) count++;

else printf("ERROR!\n");

}//两个元素出队

output(Q);

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

count=0;

while(count<1)

{ d=DeQueue(Q);

if(d==1) count++;

else if(d==0) {printf("OVERFLOW!\n"); count++;}//删除返回ERROR，提示下溢

else printf("ERROR!\n");

}//一个元素出队

output(Q);

}

system("pause");

return 0;

}

* 测试结果：

Please input the number you want to save:

输入: 5 7 3 8 55

输出：The queue is:

5 7 3 8 55

have 5 elements.

3 elements out!

The queue is:

8 55

have 2 elements.

2 elements out!

The queue is:

NULL

1 element out!

OVERFLOW!

The queue is:

NULL

* 用户使用说明：

由于程序默认设定及题目要求出队数目，所以设置用户只能输入五个数字，若输入多了的话，只存储前五个数字。

运行环境：Dev C++

* 心得体会：

之前一直对链表掌握不够，借此机会自己对照着书上的算法，研究了一下。

把尾指针的next代替头指针的方法是从网上查到的。

开始时DeQueue出队时，没有单独考虑只剩一个节点的情况，也是在网上看到别人的算法添加上的。

output函数中依次打印数据，开始用尾指针向后找循环的头结点输出，结果总是报错，然后问的同学，所以按着数出来的地址向前找头结点。

main函数中开始忘记给Q分配内存，所以一开始就停止运行了。

心得体会：链表要在脑中有个大概的结构，或者在纸上画出来才更好做。而且要善用调试，好多问题都是调试解决的。