**实 验 报 告**

**课程名称** 数据结构

**实验项目**  实验一 长整数加减

**系 别\_\_\_ \_计算机学院 \_ \_\_\_\_\_\_**

**专 业\_\_\_\_计算机科学与技术\_ \_\_**

**班级/学号\_\_\_\_1503/2015011227 \_\_**

**学生姓名 \_\_\_\_\_\_\_徐蕾 \_\_\_\_\_\_\_\_\_\_ \_**

**实验日期** \_ 2017年3月1日

**成 绩** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_

**指导教师** 黄改娟

**实验题目：实验一 长整数加减法**

1. **需求分析**

在计算机中有存储证书的数据类型，比如整型、长整型、无符号整型等，但他们的存储位数是有限的，这与具体的语言、具体的编译环境有关。一般C语言中，整型为16位长，长整型为32为长。但是在一些应用中要求的数据非常大，比如在公钥密码学中的模数要求是一个大素数，这个数如果太小会影响密码的安全性。当然，对于这些大数要进行运算，因此，有必要考虑一些超长整数的运算。

这里要求设计一个实现任意长的整数进行加减法运算的算法程序。具体的是要求设计一个存储大整数的数据结构及其运算的算法，能进行任意位数的整数加减法运算，这其中任意长整数可为正数，也可为负数，数据可以由键盘输入或者是由数据文件中读取。

1. **方案设计**

为了方便存储数据，可采用长度可以任意伸缩的链表来存储数据。又由于运算通常是从低位到高位进行，而输入数据通常是由高位到低位输入，那么存储数据的链表可以双方向进行操作，所以采用利用双向链表实现长整数的存储，每个结点可以含一个整型变量，也可以含多个整型变量，这里为了方便运算，采用一个结点存储一个整型变量的策略。

进一步讲，如果从链表尾部直接需要找到链表头部或者从链表头部可以方便的找到链表尾部，可以将链表首尾相接，形成双向循环链表[如果不想采用双向循环链表，也可以将头和尾分别设置指针(head, tail)指示，需要用到表头的时候直接通过head即可找到，需要用到表尾的时候通过tail就可以找到]。输入数据及结果形式可以按中国对于长整数的表示习惯，每四位一组，组间用逗号隔开。也可以按照国际惯例，每3位一组进行输入输出，组间用逗号隔开。

1. **系统实现：**

程序源码：

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

typedef struct Node // 双向链表的结构体定义

{

int data;

struct Node \*prior;

struct Node \*next;

}DLNode;

//链表初始化

void ListInitiate(DLNode \*\*head) //初始化一个头结点为head的双向循环链表

{

if((\*head=(DLNode \*)malloc(sizeof(DLNode)))==NULL) exit(0);

(\*head)->prior=\*head;

(\*head)->next=\*head;

}

//计算已知链表长度

int ListLength(DLNode \*head) //计算以head为头结点的链表的长度

{

DLNode \*p=head;

int size=0;

while(p->next!=head)

{

p=p->next;

size++;

}

return size;

}

//插入函数

int ListInsert(DLNode \*head,int i,int x) //双向链表的数据插入,i表示是插入的第几个元素，将节点数据为x的节点插到第i个位置上去。

{

DLNode \*p,\*s;//节点指针

int j;

p=head->next;//p指向head的下一个节点

j=0;

while(p!=head&&j<i)//p不能为空并且j<i的时候条件成立，while才继续执行

{

p=p->next;//p指向下一个节点

j++;

}

if(j!=i)

{

printf("\n插入位置不合法！");

return 0;

}

if((s=(DLNode \*)malloc(sizeof(DLNode)))==NULL) exit(0);

s->data=x;

s->prior=p->prior;//插入

p->prior->next=s;//将p的上一个节点和p的下一个节点连接起来

s->next=p;//s指向p

p->prior=s;

return 1;

}

//绝对值

int abs(int x) // 绝对值函数，返回x的绝对值。

{

if(x<0) return -x;

else return x;

}

//读入数据并插入对应链表

int InputNumber(DLNode \*head) //读入输入的数据,将从键盘中接收数据并把得到的数据存入以head为头结点的链表中

{

int input,i=0;//第i个节点

char c;

scanf("%d%c",&input,&c);

while(1)

{

if(input<0&&i==0)//输入数为负且是第一个节点

{

head->data=0;//将长整数的符号保存在头结点中

//input=abs(input);//取输入数字的绝对值

ListInsert(head,i,input);//插入数据

}

else if(input>=0&&i==0)//输入数为正且是第一个节点

{

head->data=1;//将长整数的符号保存在头结点中

ListInsert(head,i,input);//插入数据

}

else

{

if(head->next->data>=0)

ListInsert(head,i,input);//非第一个节点

else

{

//input=-1\*input;

ListInsert(head,i,input);

}

}

i++;

if(c==';') break;//遇到数据输入完成标志，跳出循环

scanf("%d%c",&input,&c);

}

return 1;

}

//输出

void OutputNumber(DLNode \*head,int sign) //从表尾输出数据元素

{

DLNode \*r=head->next;//头结点

while(r->data==0&&r!=head->prior)

{

r=r->next;

}

if(sign==1)

{

printf("结果是:");

}

else

{

printf("结果是: -");

}

printf("%d",r->data);

r=r->next;

while(r!=head)

{

if(r->data<10)

{

printf(",000");

printf("%d",r->data);//右移

}

else if(r->data<100)

{

printf(",00");

printf("%d",r->data);

}

else if(r->data<1000)

{

printf(",0");

printf("%d",r->data);

}

else

{

printf(",%d",r->data);

}

r=r->next;

}

printf("\n");

}

//正数加正数

void add(DLNode \*head1,DLNode \*head2,DLNode \*head3)//进位计数变量 ，加法操作

{

int z=0;

int e;

DLNode \*p1,\*p2;//节点

p1=head1->prior;

p2=head2->prior;

while(p1!=head1&&p2!=head2)

{

e=p1->data+p2->data+z;

if(e>=10000)

{

z=1;

e=e%10000;

}

else z=0;

ListInsert(head3,0,e);

p1=p1->prior;p2=p2->prior;

}

if(p1==head1&&p2!=head2)

{

while(p2!=head2)

{

e=p2->data+z;

if(e>=10000)

{

z=1;

e=e%10000;

}

else z=0;

ListInsert(head3,0,e);

p2=p2->prior;

}

if(z==1) ListInsert(head3,0,z);

}

else if(p1!=head1&&p2==head2){

while(p1!=head1)

{

e=p1->data+z;

if(e>=10000)

{

z=1;

e=e%10000;

}

else z=0;

ListInsert(head3,0,e);

p1=p1->prior;

}

if(z==1) ListInsert(head3,0,z);

}

else{

if(z==1) ListInsert(head3,0,z);

}

}

//判断两正数大小函数

int change(DLNode \*head1,DLNode \*head2)

{

int length1,length2,r=2;

length1=ListLength(head1);

length2=ListLength(head2);

DLNode \*p1,\*p2;

p1=head1->next;//指向头结点的下一节点

p2=head2->next;

if(length1>length2)

{

r=0;

return r;

}

else if(length1<length2)

{

r=1;

return r;

}

else

{

int i=0;

for(i=0;i<length1;i++)

{

if(p1->data>p2->data)

{

r=0;

return r;

break;//结束循环体

}

else if(p2->data>p1->data)

{

r=1;

return r;

break;

}

else

{

p1=p1->next;//p1所指节点中的指针next指向的节点地址赋给p1

p2=p2->next;

r=2;

}

}

}

return r;

}

//减法函数

void minus(DLNode \*head1,DLNode \*head2,DLNode \*head3)

{

int z=0,x=-1;

int e;

DLNode \*p1,\*p2;//节点指针

p1=head1->prior;

p2=head2->prior;

x=change(head1,head2);

if(x==0)

{

while(p1!=head1&&p2!=head2)

{

p1->data=p1->data+z;

if(p1->data>=p2->data)

{

e=p1->data-p2->data;

ListInsert(head3,0,e);

p1=p1->prior;p2=p2->prior;

z=0;

}

else

{

e=10000+p1->data-p2->data;

ListInsert(head3,0,e);

p1=p1->prior;p2=p2->prior;

z=-1;//借位

}

}

p1->data=p1->data+z;

while(p1!=head1)

{

e=p1->data;

ListInsert(head3,0,e);

p1=p1->prior;

}

}

else if(x==1)

{

p2=head1->prior;

p1=head2->prior;

while(p1!=head2&&p2!=head1)

{

p1->data=p1->data+z;

if(p1->data>=p2->data)

{

e=p1->data-p2->data;

ListInsert(head3,0,e);

p1=p1->prior;p2=p2->prior;

z=0;

}

else

{

e=10000+p1->data-p2->data;

ListInsert(head3,0,e);

p1=p1->prior;p2=p2->prior;

z=-1;

}

}

p1->data=p1->data+z;

while(p1!=head2)

{

e=p1->data;

ListInsert(head3,0,e);

p1=p1->prior;

}

head3->next->data=-1\*head3->next->data;

}

else

{

head3->next->data=0;

}

}

//整合

void yunsuan(DLNode \*head1,DLNode \*head2,DLNode \*head3,char ch)

{

DLNode \*p1,\*p2;//节点指针

p1=head1->next;//p指向head的下一个节点

p2=head2->next;

if(head1->data==1&&head2->data==1) //头结点 1的值为1与头结点2的值为1

{

if(ch=='+') add(head1,head2,head3);

else minus(head1,head2,head3);

}

else if(head1->data==1&&head2->data==0)//头结点 1的值为1与头结点2的值为0

{

if(ch=='+')

{

head2->next->data\*=-1;//2号结点的Next结点的数据乘本身等于-1

minus(head1,head2,head3);

}

else

{

head2->next->data\*=-1;//2号结点的Next结点的数据乘本身等于-1

add(head1,head2,head3);

}

}

else if(head1->data==0&&head2->data==1)

{

if(ch=='+')

{

head1->next->data\*=-1;

minus(head2,head1,head3);

}

else

{

head1->next->data\*=-1;//1号结点的Next结点的数据乘本身等于-1

head2->next->data\*=-1;//2号结点的Next结点的数据乘本身等于-1

add(head1,head2,head3);//函数

head3->next->data\*=-1;//3号结点的Next结点的数据乘本身等于-1。

}

}

else

{

if(ch=='+')

{

head1->next->data\*=-1;

head2->next->data\*=-1;

add(head1,head2,head3);

head3->next->data\*=-1;

}

else

{

head1->next->data\*=-1;

head2->next->data\*=-1;

minus(head2,head1,head3);

}

}

}

//主函数

int main()

{

char ch,ch1;

while(1)

{

//int w=-1;

DLNode \*a,\*b,\*c;

ListInitiate(&a);

ListInitiate(&b);

ListInitiate(&c);

printf("请输入数A(以分号结束):");

InputNumber(a);

//printf("\n");

printf("请输入数B(以分号结束):");

InputNumber(b);

//w=change(a,b);

printf("请选择操作符：<+,-,\*>:\n");

scanf("%s",&ch1);

if(ch1=='+'||ch1=='-')

{

yunsuan(a,b,c,ch1);

OutputNumber(c,1);

}

else printf("此版本不支持%c运算",ch1);

printf("要继续吗？(y/n) :");

scanf("%s",&ch);

if(ch=='Y'||ch=='y')

{

printf("\n");

continue;

}

else exit(0);

}

}

1. **结果测试**

