**数据结构实验报告**

**实验一：线性表**

专 业： 网络工程

班 级： 1501

姓 名： 邓泽先

学 号： 150200325

指导教师： 王梅

日 期： 2016-10-14

**一、实验目的**

熟练掌握线性表的基本操作在两种存储结构上的实现，其中以各种链表的操作和应用作为重点内容。

**二、实验内容**

**长整数四则运算**

【问题描述】

设计一个实现任意长的整数进行加法运算的演示程序。

【基本要求】

整数的位数最好不要受限制。

**三、需求分析**

1.输入的形式和输入值的范围

输入的数字可以任意长度的整数，输入结束后按回车执行下面程序。

2.输出形式

数字相加后会带有符号输出结果。

3.程序所能达到的功能

将任意长度的整数进行相加并输入其结果。

4.测试数据：见第八部分。

**四、概要设计**

(1) 本程序包含的函数

主函数 main()

获取链表所表达数字有多少位的函数 getLength(pNode head)

链表翻转函数 convert(pNode head)

链表输出函数 printfList(pNode head)

链表建立函数 createList()

数字相加函数 add(pNode head1, pNode head2)

字母转化为大写的函数 toBigger(char c)

(2)各函数的功能设计为：

int getLength(pNode head) 传入一个链表的头节点，获取该链表除了头节点外其他节点的数量，将这个数量以int的形式返回

void convert(pNode head) 传入一个链表的头节点，将该链表进行翻转，不返回值

void printfList(pNode head) 传入一个链表，将该链表所表达的值进行输出，不返回值

pNode createList() 根据用户输入的数据建立一个链表，并将该链表的头节点返回

pNode add(pNode head1, pNode head2) 向该函数传入两个头节点，并将头节点指链表所表达的数字的值进行相加，返回相加后的结果的链表的头节点。

char toBigger(char c) 将小写字母转化为大写字母

int main (void) 直接或间接调用上述函数

**五、详细设计**

(1)结点类型和指针类型:

typedef struct Num \* pNode;

struct Num{

int num;

pNode next;

} Num;

(2)单链表的基本操作：

函数算法：

(1)int getLength(pNode head)

求长度函数：接受一个头节点，判断是否为空。如果不为空将这个链表通过while循环得到链表的长度，将长度值返回

(2) void convert(pNode head)

翻转函数：接受一个头节点，判断该节点是否为空。如果不为空通过循环将该头节点所指链表进行翻转，头节点在翻转后仍是头节点。

(3) void printfList(pNode head)

输出函数：接受一个头节点，判断是否为空。如果不为空，通过while循环将头节点指向的链表遍历输出。

(4) pNode createList()

尾部插入法创建链表的函数：首先建立head指针和p1指针，将这两个指针都指向一个新分配的空间，该空间为链表的头指针。通过head将头指针中的值赋值为1，代表数字默认为正的。然后通过while循环获取值直到输入内容为’\n’，p1指向的节点会随着while循环的执行而变化。如果输入的值为负的，头指针中的值改为0。这里获取值是通过字符逐个读取，然后再通过相应的转化为数字。最后将head返回。

(5) pNode add(pNode head1, pNode head2)

将两个整数进行相加的函数：函数接受两个链表的头指针。我们平时运算都是从低位到高位运算，这里也是，通过上面说到的convert(pNode head)函数将两个链表进行翻转，然后进行运算。在函数中定义p1 p2指向翻转后的头指针指向节点的下一个节点。然后就是通过比较head1和head2中的值判断两个数是否等号。然后进行相对应的加操作。

**六、调试分析**

调试的话时间大都用在了add函数的调试了，这个函数中包含的内容较多，自己对代码的优化能力还很差，是的这个函数里的代码在调试的时候显得比较麻烦。

**七、使用说明**

运行程序后就会立刻看到 **请输入数字：（键入回车键结束该数字的输入）：** 的输出，输入第一个数字后按回车键会输出和前面那个一样的**请输入数字：（键入回车键结束该数字的输入）：**，输入第二个数字后，按回车键就会将两个数字相加并输出结果。然后会输出 **是否还要继续？（输入Y为继续）** ，这里输入Y或者y都可以继续输入下一组数字，输入其他内容会退出程序。

**八、测试结果**

(1)0和0应输出“0”，输出结果为“0”。

(2)-2和2应输出“0”，输出内容为“0”。

(3)-23456789和-76543211应输出“-100000000”，输出内容为“-100000000”。

(4)-99999999和1000000000000;应输出“999900000001”，输出内容为“999900000001”。

(5)1000和1001应输出“2001”，输出内容为“2001”。

(6)-2000和1001应输出“-999”，输出内容为“-999”。

**九、附录：部分代码**

#include<stdio.h>

#include<malloc.h>

typedef struct Num \* pNode;

struct Num{

int num;

pNode next;

} Num;

int getLength(pNode head)

{

pNode p1 = head->next;

int length = 0;

while(p1 != NULL)

{

length++;

p1 = p1->next;

}

return length;

}

void convert(pNode head)

{

struct Num \* p1, \* p2, \*p3;

int n = 0;

p1 = head->next;

if(head == NULL)

{

printf("这是个空链表！！\n");

return;

}

p2 = p1->next;

p1->next = NULL;

while(p2 != NULL)

{

p3 = p2->next;

p2->next = p1;

p1 = p2;

p2 = p3;

}

head->next = p1;

}

void printfList(pNode head)

{

int flag = 0;

pNode p;

p = head;

if(head == NULL)

{

printf("该链表为空！！\n");

}

else

{

printf("“");

if(p->num == 0)

printf("-");

p = p->next;

while(p != NULL)

{

if(p->num == 0)

{

if(flag)

printf("%d", p->num);

}

else

{

printf("%d", p->num);

flag = 1;

}

p = p->next;

}

if(!flag)

{

printf("\r");

printf("“");

printf("%d", 0);

}

printf("”");

printf("\n");

}

}

pNode createList() //该函数将每一位的数字一字符的形式存入，然后根据对应的ascII码转化为数字。

{ //创建的链表有头节点，头节点中的值为0代表数字为负值，为1代表为正值

char a;

int i;

pNode head, p1;

p1 = head = (pNode) malloc (sizeof(struct Num));

head->num = 1;

head->next = NULL;

while((a = getchar()) != '\n')

{

if(a == '-')

{

head->num = 0;

continue;

}

if((a != '-') && (a < '0' || a > '9'))

{

printf("输入了错误的内容！！程序退出！！\n");

exit(1);

}

p1->next = (pNode) malloc (sizeof(struct Num));

p1->next->num = a - 48;

p1 = p1->next;

p1->next = NULL;

}

return head;

}

pNode add(pNode head1, pNode head2)

{

pNode p1, p2, head, p;

int flag = 0;//记录某一位两个数相加是否大于等于10,如果flag为1，则大于等于10；如果为0,则小于10

p = head = (pNode) malloc (sizeof(struct Num));

convert(head1);//这一行和下一行代码是用来将数字进行翻转，翻转后才能方便进行相加

convert(head2);

p1 = head1->next;

p2 = head2->next;

if(head1->num == head2->num)//判断两个数字的符号是否相等，如果同号的话就执行下面的代码

{

head->num = head1->num;

while(p1 != NULL && p2 != NULL)//循环执行的代码将相对应的位的数字相加，并判断是否该向高一位进一

{

p->next = (pNode) malloc (sizeof(struct Num));

p = p->next;

p->num = 0;

if(flag)

{

p->num += 1;

flag = 0;

}

p->num = p1->num + p2->num + p->num;

if(p->num >= 10)

{

flag = 1;

p->num %= 10;

}

p1 = p1->next;

p2 = p2->next;

}

if(p1 == NULL && p2 != NULL) //两个数字位数不一样的情况

{

while(p2 != NULL)

{

p->next = (pNode) malloc (sizeof(struct Num));

p = p->next;

p->num = 0;

if(flag)

{

p->num += 1;

flag = 0;

}

p->num += p2->num;

if(p->num >= 10)

{

flag = 1;

p->num %= 10;

}

p2 = p2->next;

}

}

else if(p1 != NULL && p2 == NULL) //两个数字位数不一样的情况

{

while(p1 != NULL)

{

p->next = (pNode) malloc (sizeof(struct Num));

p = p->next;

p->num = 0;

if(flag)

{

p->num += 1;

flag = 0;

}

p->num += p1->num;

if(p->num >= 10)

{

flag = 1;

p->num %= 10;

}

p1 = p1->next;

}

}

if(p1 == NULL && p2 == NULL) //当两个链表都执行到最后时，判断是否应该向最高位进一

{

if(flag) //如果进一的话，执行如下代码

{

p->next = (pNode) malloc (sizeof(struct Num));

p = p->next;

p->num = 1;

}

p->next = NULL;

}

}

else //接下来的代码的作用是将不等号的两个数字相加

{

if(getLength(head1) > getLength(head2)) //如果两个数字位数不一样，head1所指的数的位数大于head2，相加后数字的符号应该是head1所指的数字的符号

{

head->num = head1->num;

while(p2 != NULL)

{

p->next = (pNode) malloc (sizeof(struct Num));

p = p->next;

p->num = 0;

if(flag)

{

p->num -= 1;

flag = 0;

}

p->num = p1->num - p2->num + p->num;

if(p->num < 0)

{

p->num += 10;

flag = 1;

}

p1 = p1->next;

p2 = p2->next;

}

while(p1 != NULL)

{

p->next = (pNode) malloc (sizeof(struct Num));

p = p->next;

p->num = 0;

if(flag)

{

p->num -= 1;

flag = 0;

}

p->num = p1->num + p->num;

if(p->num < 0)

{

p->num += 10;

flag = 1;

}

p1 = p1->next;

}

p->next = NULL;

}

else if(getLength(head1) < getLength(head2)) //head2所指的数的位数大于head1，相加后数字的符号应该是head2所指的数字的符号

{

head->num = head2->num;

while(p1 != NULL)

{

p->next = (pNode) malloc (sizeof(struct Num));

p = p->next;

p->num = 0;

if(flag)

{

p->num -= 1;

flag = 0;

}

p->num = p2->num - p1->num + p->num;

if(p->num < 0)

{

p->num += 10;

flag = 1;

}

p1 = p1->next;

p2 = p2->next;

}

while(p2 != NULL)

{

p->next = (pNode) malloc (sizeof(struct Num));

p = p->next;

p->num = 0;

if(flag)

{

p->num -= 1;

flag = 0;

}

p->num = p2->num + p->num;

if(p->num < 0)

{

p->num += 10;

flag = 1;

}

p2 = p2->next;

}

}

else{ //两个数字位数相等，执行如下代码

while(p1 != NULL)

{

p->next = (pNode) malloc (sizeof(struct Num));

p = p->next;

p->num = 0;

if(flag)

{

p->num = -1;

flag = 0;

}

p->num = p1->num - p2->num + p->num;

if(p->num < 0)

{

p->num += 10;

flag = 1;

}

p1 = p1->next;

p2 = p2->next;

}

if(flag)

head->num = head2->num;

else

head->num = head1->num;

}

p->next = NULL;

}

convert(head); //再将运行后的数字翻转，得到想要的数字

return head;

}

char toBigger(char c)

{

if(c >= 'a' && c <= 'z')

{

c -= 32;

}

return c;

}

int main (void)

{

char c;

do{

printf("请输入数字：（键入回车键结束该数字的输入）\n");

pNode p1 = createList();

printf("请输入数字：（键入回车键结束该数字的输入）\n");

pNode p2 = createList();

printf("两个数字相加为：\n");

pNode head = add(p1, p2);

printfList(head);

printf("是否还要继续？（输入Y为继续）\n");

scanf(" %c", &c);

getchar();

c = toBigger(c);

}while(c == 'Y');

return 0;

}