第五次作业分析

首先我要解释一下这次作业的问题，当做这次作业时，我以书，课件PPT，与上网为参考文件。这因为我在沙河没法去上机课，主要原因是我们留学生在学院路还有课，而且我课表与上机课的时间有冲突导致我没法去上上机课。因为当时我没有上上机课，所以当我做作业的时候，有些C语言提供的一些作用我不太明白怎么用，然后我查上网也可以说从上网学习的，可是上网只为参考文件。

1.猴子选王

void count(int n,int m,int q)

{

LinkList p, r, list = NULL;

int i;

for ( i = 1; i <= n; i++) //首先i的初值是1，i一直加起来至小于等于n

{

p = (LinkList)malloc(sizeof(LNode)); //在这里我们先申请一个新的结点

p -> data = i; //然后 数据域存放第i个结点的编号

if(list == NULL) //如果list等于空的时候

list = p; // 把p的值存放到list里边。

Else //如果list不等于空的时候

r->link = p; //把新的结点链接在链表尾部

r = p;

}

//到这里创建了n个循环表

p->link = list;

p = list;

//然后把链表的最后一个节点指向头节点,构成一个循环链表。

for (i = 1; i < q; i++) //i一直加到i不大于q

{

r = p;

p = p→link;

}

//这命令将p指向第1个出发结点

while(p->link != p)//当p的下一个链结点的值不等于p

{

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

{

r = p;

p = p->link;

}

//p指向第m个结点，r指向第m-1个结点

r->link = p→link; //删除第m个结点

free(p); //释放被删除的结点的空间

p = r→link; //p指向新的的出发点

}

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

}

2.多项相乘

void multi(struct node \*node1, struct node \*node2, struct node \*node\_res)

{

node\_res -> coef = (node1 -> coef) \* (node2 -> coef);

node\_res -> exp = (node1 -> exp) + (node2 -> exp);

}

//指定相乘的计算，把系数相乘，然后把幂数相加,把结果存放到node\_rs。

void poly\_multi(struct node \*poly1, struct node \*node2, struct node \*poly\_res)

//这函数计算poly1\*node2.然后存放到poly\_res//

{

struct node \*polynode\_p = poly1, \*resnode\_p = poly\_res;

for (;;)

{

if (polynode\_p -> link != NULL)

{

multi(polynode\_p, node2, resnode\_p);

resnode\_p -> link = (struct node \*)malloc(sizeof(struct node));

resnode\_p = resnode\_p -> link;

polynode\_p = polynode\_p -> link;

}

else

{

multi(polynode\_p, node2, resnode\_p);

resnode\_p -> link = NULL;

break;

}

}

}

//首先polynode\_p指向poly1,跟node2相乘，resnode\_p指向下面计算的结果。运行multi的函数，然后给resnode\_p给一个新的链结点，resnode\_p指向它的下一个结点，polynode也是。

struct node \*merge\_poly(struct node \*poly1, struct node \*poly2)

//这函数是把poly1和poly2放在一起。

{

struct node \*poly\_res = poly1, \*poly1\_p\_tmp = poly1, \*poly2\_p\_tmp = poly2, \*poly\_res\_tmp = poly1;

//poly\_res指向poly1的结果，poly1\_p\_tmp指向poly1拿来比较的，poly2\_p\_tmp也是，poly\_res\_tmp指向它最后的比较结果。

if (poly1\_p\_tmp -> exp > poly2\_p\_tmp -> exp) //命令poly1的幂数和poly2的幂数相比较，如果poly1大于poly2它的结果就是poly1.

{

poly\_res = poly1;

poly\_res\_tmp = poly1;

poly1\_p\_tmp = poly1\_p\_tmp -> link;

}

else if (poly1\_p\_tmp -> exp < poly2\_p\_tmp -> exp) 命令poly1的幂数和poly2的幂数相比较，如果poly1大于poly2它的结果就是poly2.

{

poly\_res = poly2;

poly\_res\_tmp = poly2;

poly2\_p\_tmp = poly2\_p\_tmp -> link;

}

else if(poly1\_p\_tmp -> exp == poly2\_p\_tmp -> exp) //如果他的幂数相等，把它的系数相加。

{

poly\_res = poly1;

poly1\_p\_tmp -> coef += poly2\_p\_tmp -> coef;

poly\_res\_tmp = poly1;

poly1\_p\_tmp = poly1\_p\_tmp -> link;

struct node \*tmp = poly2\_p\_tmp;

poly2\_p\_tmp = poly2\_p\_tmp -> link;

free(tmp);

}

while(poly1\_p\_tmp != NULL && poly2\_p\_tmp != NULL)//跟上面的一样，但是在这当它们不等于空的时候

{

if (poly1\_p\_tmp -> exp > poly2\_p\_tmp -> exp)

{

poly\_res\_tmp -> link = poly1\_p\_tmp;

poly\_res\_tmp = poly1\_p\_tmp;

poly1\_p\_tmp = poly1\_p\_tmp -> link;

}

else if (poly1\_p\_tmp -> exp < poly2\_p\_tmp -> exp)

{

poly\_res\_tmp -> link = poly2\_p\_tmp;

poly\_res\_tmp = poly2\_p\_tmp;

poly2\_p\_tmp = poly2\_p\_tmp -> link;

}

else if (poly1\_p\_tmp -> exp == poly2\_p\_tmp -> exp)

{

poly1\_p\_tmp -> coef += poly2\_p\_tmp -> coef;

poly\_res\_tmp -> link = poly1\_p\_tmp;

poly\_res\_tmp = poly1\_p\_tmp;

poly1\_p\_tmp = poly1\_p\_tmp -> link;

struct node \*tmp = poly2\_p\_tmp;

poly2\_p\_tmp = poly2\_p\_tmp -> link;

free(tmp);

}

}

struct node \*poly\_multi\_poly(struct node \*poly1, struct node \*poly2)//在这函数计算poly1和poly2的相乘结果

{

struct node \*node2\_tmp = poly2; //node2\_tmp指向poly2要跟poly1相乘

struct node \*poly\_p\_res = (struct node \*)malloc(sizeof(struct node));

poly\_multi(poly1, node2\_tmp, poly\_p\_res); //指向上面函数的最后放在一起的计算结果

struct node \*poly\_p\_tmp = (struct node \*)malloc(sizeof(struct node));

node2\_tmp = node2\_tmp -> link; //node2\_tmp指向它下一个的链结点

if (node2\_tmp != NULL)

{

for (;;)

{

poly\_multi(poly1, node2\_tmp, poly\_p\_tmp);

poly\_p\_res = merge\_poly(poly\_p\_res,poly\_p\_tmp);

if (node2\_tmp -> link != NULL)

{

poly\_p\_tmp = (struct node \*)malloc(sizeof(struct node));

node2\_tmp = node2\_tmp -> link;

}

else break;

}

}

return poly\_p\_res;

}

for(node\_tmp = poly1\_head;;)

{

scanf("%d %d%c" ,&coef\_tmp,&exp\_tmp,&lf);

node\_tmp -> coef = coef\_tmp;

node\_tmp -> exp = exp\_tmp;

if (lf != '\n')

{

node\_tmp -> link = (struct node \*)malloc(sizeof(struct node));

node\_tmp = node\_tmp -> link;

}

else

{

node\_tmp -> link = NULL;

break;

}

}

//命令输入poly1\_head的数据（第一个一元多项的数据）

for(node\_tmp = poly2\_head;;)

{

scanf("%d %d%c",&coef\_tmp,&exp\_tmp,&lf);

node\_tmp -> coef = coef\_tmp;

node\_tmp -> exp = exp\_tmp;

if (lf != '\n')

{

node\_tmp -> link =(struct node \*)malloc(sizeof(struct node));

node\_tmp = node\_tmp -> link;

}

else

{

node\_tmp -> link = NULL;

break;

}

}

// 命令输入poly2\_head的数据（第二个一元多项的数据）

res = poly\_multi\_poly(poly1\_head,poly2\_head);

while(res != NULL)

{

printf("%d %d ",res -> coef,res -> exp);

res = res -> link;

}

//res执行poly\_multi的函数，然后当res不等于空的时候，它输出它的结果，然后res指向它的下一个链结点。

3.输出文件的末尾行

if(argc == 3 && argv[1][0] == '-')

{

n = atoi(argv[1]+1); //将字符类型的n转换为整数类型的n

filename = argv[2];

}

//这命令行输入中指定打印行数时，获取行数及文件名，然后把文件名存放在filename,行数存放在n

else if(argc == 2)

{

filename = argv[1];

}

else

{

printf("Usage : tail [-n] filename \n");

return(1);

}

//这命令行输入中没有指定打印行数时，获取文件名，此时行数为缺省10，n的缺省值为10，从define定义的。

if((fp = fopen(filename,"r")) == NULL)

{

printf("can't open file: %s !\n",filename);

return(-1);

}

//这命令是检查文件里边是否空，空的话就输出 can’t open file...

list = ptr = (struct tail \*)malloc(sizeof(struct tail)); //在list和ptr一个链结点。

ptr -> line = NULL; //ptr->line为空

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

{

ptr -> link = (struct tail \*)malloc(sizeof(struct tail));在list申请一个链结点

ptr = ptr -> link; 指针ptr指向下一个链结点

ptr -> line = NULL;

}

ptr -> link = list;

ptr = list;

//创建循环链表,然后把链表的最后一个节点指向头节点,构成一个循环链表。

while(fgets(curline, MAXLEN, fp) != NULL)

{

if(ptr -> line != NULL)

free(ptr→line); //要是ptr→line 不是空的话，释放ptr→line 的空间

ptr->line = (char \*)malloc(strlen(curline)+1);

strcpy(ptr->line,curline);

ptr=ptr->link;

}

//从文本文件中读一行放入curline中，把读到的一行放入到链结点的数据域，然后ptr指向下一个链结点

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

{

if (ptr -> line != NULL)

{

printf("%s",ptr -> line);

ptr = ptr -> link;

}

}

//若ptr→line 不是空，命令打印文本文件的一行，ptr指向下一个链结点。

fclose(fp);

return 0;

}

//关闭文件。