暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，实现基于定长数组的 List。

二、实验环境

计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

定义 List 数据类型，含一个定长数组存储实际数据，含一个整型变量记录实际数据个数。在此基础上实现 List 的基本功能。速度快，可容纳的元素个数受限。

可在此补充详细原理与代码结构的简要说明

四、程序代码

```
// 参考代码如下
#include <stdio.h>
#include <stdlib.h>

struct ArrayList {
    int a[ 1000 ];
    int n;
```

```
};

void addLast( ArrayList * pList, int value ) {
    pList -> a[ pList->n++ ] = value;
}

void addAt( ArrayList * pList, int index, int value ) {
    for( int i = pList->n-1; i>=index; i-- ) {
        pList->a[i+1] = pList->a[i];
    }
    pList->a[index] = value;
    pList->n++;
}

void show( ArrayList * pList ) {
    printf( "len=%d values=", pList->n );
    for( int i=0; i < pList->n; i++ ) {
        printf( "%d ", pList->a[i] );
    }
    printf( "\n" );
}

int main() {
    ArrayList * pList = (ArrayList*) malloc( sizeof(ArrayList) );
    pList -> n = 0;

    addLast( pList, 123 );
    addLast( pList, 456 );
    addLast( pList, 789 );
    addAt( pList, 1, 777 );

    show( pList );


}
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，实现基于单向链表的 List。

二、实验环境

计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

每个对象（称为节点 Node）内部有两个成员，一个存储用户指定的数据，一个是指针指向下一个对象，这些节点串成一条链，尾节点的指针为空。在此基础上实现 List 的基本功能。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int value;
    Node * next;
};
```

```
struct LinkedList {
    Node dummyHead;
    Node * tail;
};

void addAt( LinkedList * pList, int index, int value ) {
    Node * p = pList->dummyHead.next;
    for( int i=0; i!=index-1; p=p->next,i++ );

    Node * pN = (Node *) malloc(sizeof(Node));
    pN->value = value;
    pN->next=p->next;

    p->next = pN;
}

void addLast( LinkedList * pList, int value ) {
    Node * pN = (Node *) malloc(sizeof(Node));
    pN->value = value;
    pN->next=NULL;

    Node * p;
    for( p = & (pList->dummyHead); p->next; p=p->next );
    p->next = pN;
}

void show( LinkedList * pList ) {
    for( Node * p = pList->dummyHead.next; p; p=p->next ) {
        printf( "%d\n", p->value );
    }
}

int main() {
    LinkedList * pList = (LinkedList*) malloc( sizeof(LinkedList) );
    pList->dummyHead.next = NULL;

    addLast( pList, 123 );
    addLast( pList, 456 );
    addLast( pList, 789 );
    addAt( pList, 1, 777 );
    show( pList );
}
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，实现基于双向链表的 List。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

每个对象（称为节点 Node）内部有三个成员：一个存储用户指定的数据；一个是指针指向下一个对象，这些节点串成一条链，尾节点的指针为空；一个是指针指向上一个对象，同样串成一条链，首节点的指针为空。如果 head tail 固定指向额外的 Node，则一些代码可简化，且无需维护 head tail。在此基础上实现 List 的基本功能。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
package p;

public class DoubleLinkedList {

private static class Node {
```

```
    Node prev;
    int value;
    Node next;
    Node( Node prev_a, int value_a, Node next_a ) {
        prev = prev_a;
        value = value_a;
        next = next_a;
    }
}

private final Node head = new Node( null, 0, null );
private final Node tail = new Node( null, 0, null );
{ head.next = tail; tail.prev = head; }
private int n;

public void print() {
    for( Node p = head.next; p != tail; p = p.next )
        System.out.print( p.value + " " );
    System.out.println();
}

// index: [-1,n]
private Node nodeAt( int index ) {
    Node p=head;
    for( int i=-1; i<index; p=p.next,i++ );
    return p;
}

public int get( int index ) {
    if( index < 0 || index >= n )
        throw new IndexOutOfBoundsException( ""+index );
    return nodeAt(index).value;
}

public void add( int index, int value ) {
    if( index < 0 || index > n )
        throw new IndexOutOfBoundsException( ""+index );

    // 不会 NullPointer，因为 head tail 必存在
    Node prev = nodeAt( index - 1 );
    Node node = new Node( prev, value, prev.next );
    node.prev.next = node.next.prev = node;
```

```
        n++;
    }

    public void remove( int index ) {
        if( index < 0 || index >= n )
            throw new IndexOutOfBoundsException( ""+index );

        Node node = nodeAt( index );
        node.prev.next = node.next;
        node.next.prev = node.prev;
        n--;
    }

    public static void main(String[] args) {
        DoubleLinkedList list = new DoubleLinkedList();
        list.add( 0, 1 ); list.print(); // 1
        list.add( 1, 2 ); list.print(); // 1 2
        list.add( 2, 3 ); list.print(); // 1 2 3
        list.add( 0, 4 ); list.print(); // 4 1 2 3
        list.add( 2, 5 ); list.print(); // 4 1 5 2 3
        list.remove( 0 ); list.print(); // 1 5 2 3
        list.remove( list.n - 1 ); list.print(); // 1 5 2
        list.add( 0, 1 ); list.print(); // 1 1 5 2
        list.add( 1, 2 ); list.print(); // 1 2 1 5 2
        list.add( 2, 3 ); list.print(); // 1 2 3 1 5 2
        list.add( 0, 4 ); list.print(); // 4 1 2 3 1 5 2
        list.add( list.n, 5 ); list.print(); // 4 1 2 3 1 5 2 5
        list.remove( 3 ); list.print(); // 4 1 2 1 5 2 5
    }




    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，实现基于增长数组的 List。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

List 内含一个数组。在添加元素时，如果原数组不够用则新生成一个更大的数组，并把原数组的内容复制到新数组。新数组长度是原数组的多少倍，称为增长因子，参考值：Java 的 ArrayList 取 1.5，C++ 的 STL 取 2。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>
#include <stdlib.h>

struct List {
    int * a;
    int n;
```

```
    int c; // capacity
};

void addAt( List * pList, int index, int value ) {
}

void addLast( List * pList, int value ) {
    if( pList->n==pList->c ) {
        pList->c *= 2;
        int * b = (int*) malloc( pList->c * sizeof(int) );
        for( int i=0; i < pList->n; i++ )
            b[i] = pList->a[i];
        free( pList->a );
        pList->a = b;
    }
    pList->a[ pList->n++ ] = value;
}

void show( List * pList ) {
    for( int i=0; i<pList->n; i++ ) {
        printf( "%d\n", pList->a[i] );
    }
}

int main() {
    List * pList = (List*) malloc( sizeof(List) );
    pList -> c = 10;
    pList->a = (int*) malloc(sizeof(int) * pList -> c);
    pList->n = 0;

    for( int i=0; i<100; i++ ) {
        addLast( pList, i );
    }
    show( pList );
}
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

List 内含一个数组。在添加元素时，如果原数组不够用则新生成一个更大的数组，并把原数组的内容复制到新数组。新数组长度是原数组的多少倍，称为增长因子，参考值：Java 的 ArrayList 取 1.5，C++ 的 STL 取 2。数组看作是首尾循环相接的（注意不是说表中元素首尾循环相接），在 List 尾部添加时，若已到达数组尾部，则转为在数组头部添加；在 List 头部添加时，若已到达数组头部，则转为在数组尾部添加。因此 List 内需要两个整形变量，以标记实际数据的起点和终点，建议两个变量分别表示起点和长度，终点通过换算得出。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
// List.cpp
#include <stdlib.h>
```

```
struct List {
    int c; // capacity
    ELEMENT_TYPE * a; // array
    int s; // start, physical index s <=> logical index 0
    int n; // number
};

List * create() {
    List * p = (List*) malloc( sizeof(List) );
    p->c = 4;
    p->a = (ELEMENT_TYPE *) malloc( sizeof(ELEMENT_TYPE)*p->c );
    p->s = 0;
    p->n = 0;
    return p;
}

void destroy( List * p ) {
    free( p->a );
    free( p );
}

// logical index -> physical index
// index:[0,n)
int _( List * p, int index ) {
    // index:[0,n), n<=c
    // s:[0,c)
    // so x=s+index:[0,2c-1)
    int x = p->s + index;
    if( x >= p->c )
        x -= p->c;
    return x;
}

// index:[0,n)
ELEMENT_TYPE getAt( List * p, int index ) {
    return p->a[_(p,index)];
}

ELEMENT_TYPE getLast( List * p ) {
    return getAt( p, p->n-1 );
```

```
}

ELEMENT_TYPE getFirst( List * p ) {
    return getAt( p, 0 );
}

// index:[0,n)
void setAt( List * p, int index, ELEMENT_TYPE value ) {
    p->a[_(p,index)] = value;
}

void setLast( List * p, ELEMENT_TYPE value ) {
    setAt( p, p->n-1, value );
}

void setFirst( List * p, ELEMENT_TYPE value ) {
    setAt( p, 0, value );
}

void ensureCapacity( List * p, int limit ) {
    if( p->c >= limit )
        return;
    int newC = p->c*2;
    if( newC < limit )
        newC = limit;
    ELEMENT_TYPE * newA = (ELEMENT_TYPE *)
malloc( sizeof(ELEMENT_TYPE)*newC );
    for( int i=0; i<p->n; i++ )
        newA[i] = getAt(p,i);
    free( p->a );
    p->c = newC;
    p->a = newA;
    p->s = 0;
}

// index:[0,n+1), n means addLast
void addAt( List * p, int index, ELEMENT_TYPE value ) {
    ensureCapacity( p, p->n+1 );

    p->n++;
    for( int i = p->n-1; i>index; i-- )
        setAt( p, i, getAt(p, i-1) );
```

```
    setAt( p, index, value );
}

void addLast( List * p, ELEMENT_TYPE value ) {
    addAt( p, p->n, value );
}

void addFirst( List * p, ELEMENT_TYPE value ) {
    // do NOT call addAt() for effeciency
    ensureCapacity( p, p->n+1 );

    p->s--;
    if( p->s < 0 )
        p->s = p->c-1;
    p->n++;
    setFirst( p, value );
}

// index:[0,n)
ELEMENT_TYPE removeAt( List * p, int index ) {
    ELEMENT_TYPE result = getAt( p, index );
    for( int i = index; i<p->n-1; i++ )
        setAt( p, i, getAt(p, i+1) );
    p->n--;
    return result;
}

ELEMENT_TYPE removeLast( List * p ) {
    return removeAt( p, p->n-1 );
}

ELEMENT_TYPE removeFirst( List * p ) {
    // do NOT call removeAt() for effeciency
    ELEMENT_TYPE result = getFirst( p );
    p->s++;
    if( p->s >= p->c )
        p->s -= p->c;
    p->n--;
    return result;
}
```

```
// testList.cpp
#include <stdio.h>

#define ELEMENT_TYPE int
#include "List.cpp"

int main() {
    List * p = create();
    addLast( p, 123 );
    addLast( p, 456 );
    addLast( p, 789 );
    addFirst( p, 777 );
    addAt( p, 2, 888 );

    for( int i=0; i<p->n; i++ )
        printf( "%d\n", getAt(p,i) );
}
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，输入多项式各项的系数及对应的次数，对任意输入的变量取值，求多项式的最终值。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

非稀疏多项式的保存，可以用单个 List 组保存系数，下标对应次数。稀疏多项式的保存，可以用两个 List，一个保存系数，一个保存次数。计算时，根据保存的多项式，计算每项的幂部分，再和系数序列内积。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
package p;

import tools.list.List;

class Polynomial extends List<Polynomial.Term> {
```

```
static class Term {
    double c; // coefficient
    int e; // exponent
    Term( double c_a, int e_a ) {
        c = c_a;
        e = e_a;
    }
    public String toString() {
        return String.format( "%s%sx%d", c>=0?"+":"", c, e );
    }
}

Polynomial add( double c, int e ) {
    add( new Term(c, e) );
    return this;
}




public String toString() {
    return toString(" ");
}

Polynomial add( Polynomial that ) {
    Polynomial result = new Polynomial();
    int i1=0, i2=0;
    for( ; i1<this.length() && i2<that.length(); ) {
        Term t1 = this.get(i1);
        Term t2 = that.get(i2);
        if( t1.e < t2.e ) {
            result.add( t2.c, t2.e );
            i2++;
        } else if( t1.e == t2.e ) {
            result.add( new Term(t1.c+t2.c, t1.e) );
            i1++; i2++;
        } else {
            result.add( t1.c, t1.e );
            i1++;
        }
    }
```

```
        for( ; i1<this.length(); i1++ )
            result.add( this.get(i1).c, this.get(i1).e );
        for( ; i2<that.length(); i2++ )
            result.add( that.get(i2).c, that.get(i2).e );
        return result;
    }

    Polynomial multiply( Polynomial that ) {
        Polynomial result = new Polynomial();
        for( Term t1 : this ) {
            Polynomial p = new Polynomial();
            for( Term t2 : that )
                p.add( t1.c*t2.c, t1.e+t2.e );
            result = result.add( p );
        }
        return result;
    }




    public static void main(String[] args) {
        Polynomial p1 = new
Polynomial().add(2,5).add(-5,3).add(-10,1).add(9,0);
        Polynomial p2 = new Polynomial().add(7,4).add(6,2).add(1,1);
       System.out.printf( "%s%n%s%n%s%n%s%n", p1, p2, p1.add(p2),
p1.multiply(p2) );
    }

    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，基于字符串实现大整数的运算。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

用十进制字符串/字符列表表示大整数，空间存在浪费。加法的原理为，从低位到高位对应相加，同时产生进位，进位最多为 1，最高位的进位引起总位数的增长。乘法的原理为卷积运算。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>
#include <stdlib.h>

#define ELEMENT_TYPE char
#include "List.cpp"

List * createFromString( char * s ) {
    List * r = create();
```

```
    for( ; *s; s++ )
        addLast( r, *s );
    return r;
}

void show( List * p ) {
    for( int i=0; i<p->n; i++ )
        putchar( getAt(p,i) );
    putchar('\n');
}

List * add( List * p, List * q ) {
    while( p->n < q->n )
        addFirst( p, '0' );
    while( q->n < p->n )
        addFirst( q, '0' );
    List * r = create();

    int carry = 0;
    for( int i=p->n-1; i>=0; i-- ) {
        int t = getAt(p,i)-'0' + getAt(q,i)-'0' + carry;
        carry = t / 10;
        t = t % 10;
        addFirst(r, t + '0');
    }
    addFirst(r, carry+'0');
    while( getFirst(r)=='0' )
        removeFirst(r);
    return r;
}

List * multiplySingle( List * p, char c ) {
    List * r = create();
    int carry = 0;
    for( int i=p->n-1; i>=0; i-- ) {
        int t = (getAt(p,i)-'0') * (c-'0') + carry;
        carry = t / 10;
        t = t % 10;
        addFirst(r, t + '0');
    }
    addFirst(r, carry+'0');
    while( getFirst(r)=='0' )
```

```
            removeFirst(r);
        return r;
    }

    List * multiply( List * p, List * q ) {
        List * r = create();
        for( int i=q->n-1; i>=0; i-- ) {
            List * tmp1 = multiplySingle( p, getAt(q,i) );
            for( int j=0; j<q->n-1-i; j++ )
                addLast( tmp1, '0' );
            List * tmp2 = add( r, tmp1 );
            destroy( r );
            destroy( tmp1 );
            r = tmp2;
        }
        while( getFirst(r)=='0' )
            removeFirst(r);
        return r;
    }

    int main() {
        List * p = createFromString( "123456789" );
        //show(p);return 0;
        List * q = createFromString( "987654321" );

        List * r = add( p, q );
        show( r );
        destroy( r );

        r = multiply( p, q );
        show( r );
        destroy( r );

        return 0;
    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，给一个整数、小数，表示距离、时间、金额，将中文读法的字符串显示出来。注意小数点后各位的单位。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

按 4 位一组（万为单位）来读。注意零是如何处理的：碰到 0 不读，碰到非 0 时根据前一位是否是 0 补读零；一组如果是 4 个全 0，则本组的单位不读。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>

int main() {
    char a[] = "30    0000 0000 1000"; // arabian number
    int len=0;
    for( int i=0; a[i]; i++ )
```

```
        if( a[i] != ' ' )
            a[len++] = a[i];
    a[len] = 0;

    char * cnDigits[] = {"零","一","二","三","四","五","六","七","八
","九"};
    char * units[] = {"", "十", "百", "千"};

    for( int i=0; i<len; i++ ) {
        int digit = a[i] - '0';
        char * cnDigit = cnDigits[digit];
        int p = len - 1 - i; //  从右向左数第几位，从 0 开始数
        int off = p % 4; // 4 位一组，组内从右向左数第几位，从 0
开始数
        char * unit = units[off];

        if( digit != 0 ) {
            if( i>0 && a[i-1]=='0' )
                printf( "%s", "零" ); //  补读零
            printf( "%s%s", cnDigit, unit ); //  读数字+单位
        }

        if( p % 4 == 0 ) { //  读组单位
            int isCurrentGroupAllZero = 1; //  一组全零
            for( int j=0; j<4 && i-j>=0; j++ )
                if( a[i-j] != '0' ) {
                    isCurrentGroupAllZero = 0;
                    break;
                }
            if( isCurrentGroupAllZero )
                continue; //  则不读本组单位

            if( p % 8 )
                printf( "%s", "万" );
            for( int j=0; j<p/8; j++ )
                printf( "%s", "亿" );
        }
    }

    printf( "\n" );
```

```
    return 0;
}
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，给定语料库，生成一段文字，要求生成文字字符串的多元分布与给定的语料库保持一致。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

将若干中文文章（建议用 GB 系列编码）读入一个巨大的字符串 text，从其中一个随机位置开始，输出 3 个字符；每次根据输出的最后 3 个字符 xyz，看字符串 xyz 在 text 中出现了多少次，随机选择一次出现，将此次出现 xyz 后面的字符输出。这叫做语言的 n-gram 模型，此处 n=3。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
```

```
const int N = 3;

struct Ch {
    char a;
    char b;
};
#define ELEMENT_TYPE Ch
#include "List.cpp"

void putCh( Ch ch ) {
    if( ch.a )
        putchar( ch.a );
    putchar( ch.b );
}

void show( List * p ) {
    for( int i=0; i<p->n; i++ )
        putCh( getAt(p,i) );
}

List * getSubList( List * p, int off, int len ) {
    List * r = create();
    for( int i=0; i<len; i++ )
        addLast( r, getAt(p,off+i) );
    return r;
}

int findSubListFrom( List * p, int off, List * sub ) {
    for( int i=off; i+sub->n <= p->n; i++ ) {
        int j=0;
        for( ; j<sub->n; j++ ) {
            Ch ch1 = getAt(p,i+j);
            Ch ch2 = getAt(sub,j);
            if( ch1.a != ch2.a || ch1.b != ch2.b )
                break;
        }
        if( j==sub->n )
            return i;
    }
    return -1;
}
```

```
// uniform [0,1)
double random() {
    return rand() / (RAND_MAX + 1.0);
}

int main() {
    List * p = create();
    char * dir = "C:/g/Buffer/cnNovels";
    DIR * dp = opendir( dir );
    for( dirent * ep; ep=readdir(dp); ) {
        //printf( "%s\n", ep->d_name );
        if( ep->d_name[0] == '.' )
            continue;
        char path[1000] = {}; // 注意路径长度有上限，一般文件系统也有限制
        strcpy( path, dir );
        strcat( path, "/" );
        strcat( path, ep->d_name );
        FILE * fp = fopen( path, "r" );
        fprintf( stderr, "%s\n", path );
        for( int t; (t=fgetc(fp)) != EOF; ) {
            if( t == '\r' || t == '\n' )
                continue;
            if( t&0x00000080 ) { // GB 编码中的汉字
                int tt = fgetc(fp);
                addLast( p, (Ch){ (char)t, (char)tt } );
            } else { // ASCII 前 128 个编码
                addLast( p, (Ch){ 0, (char)t } );
            }
        }
        fclose( fp );
    }
    closedir( dp );

    // 为方便调试，可固定随机数种子 srand(0)，正式运行时改为 srand( time(NULL) )
    // srand( time(NULL) );
    int r = (int) ( random() * p->n ); // 有什么问题？应为 random() * (p->n - N)
    List * sub = getSubList( p, r, N );
```

```
        show( sub );

        for( int i=0; i<1000; i++ ) {
            // 出现了多少次
            int count = 0;
            for( int off=0; ; off++, count++ ) {
                off = findSubListFrom( p, off, sub );
                if( off == -1 )
                    break;
                if( off + N > p->n ) // 最后 N 个字后继无字，因此不算出现次数，以避免此情况
                    break;
            }
            // 随机找一次出现
            int choose = (int) ( random() * count );
            count = 0;
            // 重新找到这次出现
            for( int off=0; off = findSubListFrom( p, off, sub ); off++, count++ ) {
                if( count == choose ) {
                    destroy( sub );
                    sub = getSubList( p, off+1, N ); // 后继 1 个字
                    putCh( getLast(sub) );
                    break;
                }
            }
        }
        return 0;
    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，在一个表达式或源代码字符串中，检查括号是否都是配对的。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

从头到尾扫描字符串，对不同种类的括号，遇到开括号就入栈，遇到闭括号则检查是否与栈顶开括号类型相同，相同则出栈，否则报错括号不匹配。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>

#define ELEMENT_TYPE char
#include "List.cpp"

int check( char * s ) {
```

```
    List * p = create();
    for( ; *s; s++ ) {
        char c = *s;
        if( c=='(' || c=='[' || c=='{' )
            addLast( p, c );
        if( c==')' || c==']' || c=='}' ) {
            if( p->n == 0 )
                return -1;
            char d = removeLast( p );
            if( c==')'&&d=='(' || c==']'&&d=='[' || c=='}'&&d=='{' )
                ;
            else
                return -1;
        }
    }
    destroy( p );
    return 0;
}

int main() {
    char * s = "afnj;e(fa[z;]aioew{faw}wwwwf)aw;kf";
    printf( "%s", check( s ) ? "FAIL" : "OK" );
    return 0;
}
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，使用栈方法，对表达式字符串进行求值。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

扫描 token list：碰到数字则入数字栈；碰到运算符 t，则从符号栈中弹出 1 个运算符、从数字栈中弹出 2 个数字进行计算，结果入数字栈，如此重复，直到符号栈顶的优先级<t 的优先级，再将 t 入符号栈。

每次碰到(直接入栈或表；碰到运算符则按规则向前计算，但不越过 (；碰到)则强行向前计算直到(，然后删除这一对()。具体可通过如下方法实现：把(优先级设为最低，)优先级设为次低。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double numS[100];
int numN = 0;
```

```
void nums_push( double v ) {
  numS[ numN++ ] = v;
}
double nums_pop() {
  return numS[ --numN ];
}

char opS[100];
int opN = 0;
char ops_len() {
  return opN;
}
void ops_push( char v ) {
  opS[ opN++ ] = v;
}
char ops_pop() {
  return opS[ --opN ];
}
char ops_peek() {
  return opS[ opN-1 ];
}

int prio( char op ) {
  if( op=='+' ) return 1;
  if( op=='-' ) return 1;
  if( op=='*' ) return 2;
  if( op=='/' ) return 2;
  if( op=='^' ) return 3;
  if( op=='=' ) return -9;
  if( op=='(' ) return -9;
  if( op==')' ) return -8;
  return -99; // not an operator
}

int isNum( char c ) {
  return prio(c) == -99;
}

int dir( int prio ) {
  if( prio==3 ) return '<';
  return '>';
}

void calcOneStep() {
  char op = ops_pop();
  double right = nums_pop();
  double left = nums_pop();
  double r = 0;
  if( op=='+' ) r = left + right;
  if( op=='-' ) r = left - right;
  if( op=='*' ) r = left * right;
  if( op=='/' ) r = left / right;
  if( op=='^' ) r = pow( left, right );
  nums_push( r );
}

int main() {
```

```
    //char * expression = "1+2="; // 3
    //char * expression = "2^2^3="; // 256
    //char * expression = "1+2*3^4/5-6="; // 27.4
    //char * expression = "1-(2-3)=";
    //char * expression = "1.23+4*5^6/(7-8)="; // -62498.77
    char * expression = "(((1.23+4)*5)^6/(7-(8)))=";
//-319764435.325232640625;

    for( char *p=expression; *p; ) {
        if( isNum(*p) ) {
            nums_push( strtod(p,&p) );
            continue;
        }
        char op = *p;
        p++; // char op = *p++;

        if( op == '(' ) {
            ops_push( op );
            continue;
        }

        // 如下写法难读
        //while( ops_len() && (   prio(op)<prio(ops_peek()) ||
( prio(op)==prio(ops_peek()) && dir(prio(op))=='>' )   ) )
        // calcOneStep();
        for( ; ; ) {
            if( ops_len() == 0 )
                break;
            if( prio(ops_peek()) < prio(op) )
                break;
            if( prio(ops_peek()) > prio(op) ) {
                calcOneStep();
                continue;
            }
            // ==
            if( dir(prio(op)) == '>' )
                calcOneStep();
            break;
        }
        if( op == '=' ) {
            printf( "%f\n", nums_pop() );
            return 0;
        }
        if( op == ')' ) {
            ops_pop();
            continue;
        }
        ops_push( op );
    }
    return -1; // error
}
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，使用递归方法，对算术表达式字符串进行求值。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

按照运算符优先级从低到高：若有+，则递归计算左右两侧，结果相加；若有-，递归计算左右两侧并相减；对*/^依此类推。注意减、除运算的处理，应从后向前扫描。

碰到开括号时，寻找与之匹配的闭括号，对括号内的子字符串递归处理。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
package p;

import tools.list.List;

class EvalRecursively {
```

```
    public static void main( String[] args ) {
    //  String expressionString = " 1.2 +( 3.4 *5.6) ^7.8-987, 654    .321
"; // 9.57982093399658E9
    //  String  expressionString  =  "1.2+3.4*5.6^7.8-9.0";  //
2329945.4401809676
        String expressionString = "1-2-3";

        // 简单地解析非负小数及 5 种运算符
        List<Object> tokens = new List<>();
        for( String e = "("+expressionString+")"; ! e.isEmpty(); ) {
            int index = 0;
            for( ; index < e.length(); index++ )
                if( "+-*/^()".contains(""+e.charAt(index)) )
                    break;
            String s = e.substring(0,index).replaceAll("[\\s,]+", "");
            if( ! s.isEmpty() )
                tokens.add( new Double(s) );
            tokens.add( e.charAt(index) );
            e = e.substring(index+1);
        }

        System.out.println( calc(tokens) );
    }

    private static double calc(List<Object> tokens) {
        // 递归处理括号
        if( tokens.first().equals('(') ) {
            List<Object> left = new List<>();
            for( int level = 0; ; ) {
                Object token = tokens.removeFirst();
                left.add( token );
                if( token.equals('(') )
                    level ++;
                else if( token.equals(')') )
                    level --;
                if( level == 0 )
                    break;
            }
            tokens.addFirst( calc( left.subListOl( 1, left.length()-2 ) ) );
        }
```

```
        // 递归处理运算符
        int i;
        if( (i=tokens.indexOf('+')) >= 0 ) {
            double a = calc( tokens.subListOl( 0, i ) ); // [0,i-1]
            double b = calc( tokens.subList( i+1 ) ); // [i+1,len-1]
            return a + b;
        }
        if( (i=tokens.lastIndexOf('-')) >= 0 ) {
            double a = calc( tokens.subListOl( 0, i ) ); // [0,i-1]
            double b = calc( tokens.subList( i+1 ) ); // [i+1,len-1]
            return a - b;
        }
        if( (i=tokens.indexOf('*')) >= 0 ) {
            double a = calc( tokens.subListOl( 0, i ) ); // [0,i-1]
            double b = calc( tokens.subList( i+1 ) ); // [i+1,len-1]
            return a * b;
        }
        if( (i=tokens.lastIndexOf('/')) >= 0 ) {
            double a = calc( tokens.subListOl( 0, i ) ); // [0,i-1]
            double b = calc( tokens.subList( i+1 ) ); // [i+1,len-1]
            return a / b;
        }
        if( (i=tokens.indexOf('^')) >= 0 ) {
            double a = calc( tokens.subListOl( 0, i ) ); // [0,i-1]
            double b = calc( tokens.subList( i+1 ) ); // [i+1,len-1]
            return Math.pow( a, b );
        }
        return (Double) tokens.first();
    }




    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，给出汉诺塔问题的解：三根柱子，第一根柱子上存放有 64 个从小到大的圆盘。每次允许从一根柱子的最上方搬移一个圆盘放到另一根柱子上。任何时刻都必须保持小圆盘在大圆盘上方。给出步骤，将第一根柱子上的所有圆盘搬移到第三根柱子上。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

递归处理，要把 A 柱上的 n 个圆盘搬移到 C 柱上，可以先递归把 A 柱上的 n-1 个圆盘搬移到 B 柱上，再把 A 柱的第 n 个圆盘搬移到 C 柱上，最后再递归把 B 柱上的 n-1 个圆盘搬移到 C 柱上。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>

void f( char x, int n, char y ) {
    if( n==1 ) {
```

```
            printf( "%c => %c\n", x, y );
            return;
        }
        char z = 3 - (x-'A') - (y-'A') + 'A';
        f(x, n-1, z);
        f(x, 1, y);
        f(z, n-1, y);
    }

    int main() {
        f( 'A', 3, 'B' );
        return 0;
    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，基于指针实现树结构。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

定义节点数据类型，每个节点含有一个指针列表，其中的指针指向本节点的各个子节点。为方便，子节点也可以有指针指向父节点。在此基础上完成树结构的基本操作。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>

//typedef struct Node * ELEMENT_TYPE;
#define ELEMENT_TYPE struct Node *
#include "List.cpp"

struct Node {
    char value;
```

```
        struct Node * parent;
        struct List * children;
    };

    Node * createNode( char value, Node * parent ) {
        Node * p = (Node *) malloc( sizeof(Node) );
        p->value = value;
        p->parent = parent;
        p->children = create();
        if( parent )
            addLast( parent->children, p );
        return p;
    }

    void destroyNode( Node * p ) {
        free( p->children );
        free( p );
    }

    int main() {
        Node * a = createNode( 'a', NULL );
        Node * b = createNode( 'b', a );

        createNode( 'x', createNode('y', createNode('z', a ) ) );

        printf( "%c\n", getAt( getAt(a->children,1)->children,
0 )->value );

        // 程序退出时自动关闭已打开的文件、动态分配的内存
        return 0;
    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，用递归求树的各项信息。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

用递归法求树的各项信息，每个节点的信息通过递归调用在各个子节点上，然后对每个子节点统计得到。例如：节点、叶节点的个数、总值、平均值、最大值、最小值、总深度、平均深度、最大深度，等等。求得的信息可以考虑存放到各个节点，供后续使用。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>

//typedef struct Node * ELEMENT_TYPE;
#define ELEMENT_TYPE struct Node *
#include "List.cpp"
```

```
struct Node {
    int value;
    struct Node * parent;
    struct List * children;
};

Node * createNode( int value, Node * parent ) {
    Node * p = (Node *) malloc( sizeof(Node) );
    p->value = value;
    p->parent = parent;
    p->children = create();
    if( parent )
        addLast( parent->children, p );
    return p;
}

// 节点个数
int f1( Node * p ) {
    int sum = 1;
    for( int i=0; i<p->children->n; i++ )
        sum += f1( getAt(p->children,i) );
    return sum;
}

// 叶节点个数
int f2( Node * p ) {
    if( p->children->n == 0 )
        return 1;
    int sum = 0;
    for( int i=0; i<p->children->n; i++ )
        sum += f2( getAt(p->children,i) );
    return sum;
}

// 节点总深度
int h1( Node * p, int height ) {
    int sum = height;
    for( int i=0; i<p->children->n; i++ )
        sum += h1( getAt(p->children,i), height+1 );
    return sum;
}
```

```
int main() {
    Node * r = createNode( 1, NULL );
    Node * t = createNode( 2, r );
    createNode( 7, createNode(8, createNode(9, r ) ) );
    //printf( "%d\n", f1(r) );
    //printf( "%f\n", g1(r) );
    printf( "%d\n", h1(r,0) );
    return 0;
}
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，用递归实现二叉树三序遍历。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

对每个节点，递归遍历左子节点、右子节点，本节点的访问可以在这两者之前、中间、之后，由此形成三序遍历。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>
#include <stdlib.h>

struct Node {
    char value;
    Node * parent;
    Node * left;
    Node * right;
};
```

```
Node * createNode( char value, Node * left, Node * right ) {
    Node * node = (Node *) malloc( sizeof(Node) );
    node->value = value;
    node->left = left;
    if( left )
        left->parent = node;
    node->right = right;
    if( right )
        right->parent = node;
    return node;
}

void preRecursively( Node * node ) {
    if( node == NULL )
        return;
    printf( "%c ", node->value );
    preRecursively( node->left );
    preRecursively( node->right );
}

void inRecursively( Node * node ) {
    if( node == NULL )
        return;
    inRecursively( node->left );
    printf( "%c ", node->value );
    inRecursively( node->right );
}

void postRecursively( Node * node ) {
    if( node == NULL )
        return;
    postRecursively( node->left );
    postRecursively( node->right );
    printf( "%c ", node->value );
}

int main() {
    Node * r = createNode( 'a',
        createNode('b',
            createNode('d',NULL,NULL),
            createNode('e',NULL,NULL)
```

```
            ),
            createNode('c',
                createNode('f',NULL,NULL),
                createNode('g',NULL,NULL)
            )
        );

        preRecursively( r ); printf( "\n" );
        inRecursively( r ); printf( "\n" );
        postRecursively( r ); printf( "\n" );

        return 0;
    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，用任务列表实现二叉树三序遍历。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

用任务列表可以将任意递归方法转为非递归方法，相当于模拟了程序运行时的函数调用栈。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>
#include <stdlib.h>

struct Node {
    char value;
    Node * parent;
    Node * left;
    Node * right;
};
```

```
Node * createNode( char value, Node * left, Node * right ) {
    Node * node = (Node *) malloc( sizeof(Node) );
    node->value = value;
    node->left = left;
    if( left )
        left->parent = node;
    node->right = right;
    if( right )
        right->parent = node;
    return node;
}

struct Task {
    char type; // 'P' print, 'E' expand
    Node * node;
};

#define ELEMENT_TYPE Task
#include "List.cpp"

void preByTasks( Node * r ) {
    List * tasks = create();
    addFirst( tasks, (Task){ 'E', r } );
    while( tasks->n ) {
        Task t = removeFirst( tasks );
        if( t.node == NULL )
            continue;
        if( t.type == 'P' ) {
            printf( "%c ", t.node->value );
        } else { // t.type == 'E'
            addFirst( tasks, (Task){ 'E', t.node->right } );
            addFirst( tasks, (Task){ 'E', t.node->left } );
            addFirst( tasks, (Task){ 'P', t.node } );
        }
    }
    destroy( tasks );
}

void inByTasks( Node * r ) {
    List * tasks = create();
    addFirst( tasks, (Task){ 'E', r } );
```

```
    while( tasks->n ) {
        Task t = removeFirst( tasks );
        if( t.node == NULL )
            continue;
        if( t.type == 'P' ) {
            fprintf( stderr, "%c ", t.node->value );
        } else { // t.type == 'E'
            addFirst( tasks, (Task){ 'E', t.node->right } );
            addFirst( tasks, (Task){ 'P', t.node } );
            addFirst( tasks, (Task){ 'E', t.node->left } );
        }
    }
    destroy( tasks );
}

void postByTasks( Node * r ) {
    List * tasks = create();
    addFirst( tasks, (Task){ 'E', r } );
    while( tasks->n ) {
        Task t = removeFirst( tasks );
        if( t.node == NULL )
            continue;
        if( t.type == 'P' ) {
            fprintf( stderr, "%c ", t.node->value );
        } else { // t.type == 'E'
            addFirst( tasks, (Task){ 'P', t.node } );
            addFirst( tasks, (Task){ 'E', t.node->right } );
            addFirst( tasks, (Task){ 'E', t.node->left } );
        }
    }
    destroy( tasks );
}

int main() {
    Node * r = createNode( 'a',
        createNode('b',
            createNode('d',NULL,NULL),
            createNode('e',NULL,NULL)
        ),
        createNode('c',
            createNode('f',NULL,NULL),
            createNode('g',NULL,NULL)
```

```
            )
        );

        preByTasks( r ); printf( "\n" );
        inByTasks( r ); printf( "\n" );
        postByTasks( r ); printf( "\n" );

        return 0;
    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果

暨南大学本科实验报告专用纸

课程名称 数据结构 成绩评定

实验项目名称 **错误!未找到引用源。** 指导教师 干晓聪

实验项目编号 实验项目类型 设计性 实验地点 数学系机房

学生姓名 **错误!未找到引用源。** 学号 **错误!未找到引用源。**

学院 信息学院 系 数学系 专业 **错误!未找到引用源。**

实验时间 年 月 日 午 ~ 月 日 午 温度 ℃湿度

一、实验目的

写一个程序，用列表实现树的广度优先遍历。

二、实验环境

计算机：计算机：PC X64 / PC X86
操作系统：Windows / Linux / MacOS
编程语言：C / C++ / Java
IDE：C-Free / C++-Dev / Visual C++ / Eclipse

三、程序原理

根节点入队，每次出队一个节点，访问之后，将其所有子节点入队。循环直到队为空。

可在此补充详细原理与代码结构的简要说明

```
// 参考代码如下
#include <stdio.h>
#include <stdlib.h>

struct Node {
    char value;
    Node * parent;
    Node * left;
    Node * right;
};
```

```
Node * createNode( char value, Node * left, Node * right ) {
    Node * node = (Node *) malloc( sizeof(Node) );
    node->value = value;
    node->left = left;
    if( left )
        left->parent = node;
    node->right = right;
    if( right )
        right->parent = node;
    return node;
}

#define ELEMENT_TYPE Node *
#include "List.cpp"

void breadFirst( Node * r ) {
    List * nodes = create();
    addLast( nodes, r );
    while( nodes->n ) {
        Node * node = removeFirst( nodes );
        if( node == NULL )
            continue;
        printf( "%c ", node->value );
        addLast( nodes, node->left );
        addLast( nodes, node->right );
    }
    destroy( nodes );
}

int main() {
    Node * r = createNode( 'a',
        createNode('b',
            createNode('d',NULL,NULL),
            createNode('e',NULL,NULL)
        ),
        createNode('c',
            createNode('f',NULL,NULL),
            createNode('g',NULL,NULL)
        )
    );
```

```
        breadFirst( r );

        return 0;
    }
```

五、出现的问题、原因与解决方法

六、测试数据与运行结果