1 找到指定位置

判断this->next== NULL

来确认位置

head

2 保存this->next

val next

val next

val next

val next

3 new node 并使:

this->next = node->this

node->next = 原this->next

val next

tamplate <class T>

class LineList

{

private:

T val;

LineList\* next;

public:

LineList(T headVal = 0):

val(headVal), next(NULL);

void LineList::addNodeLoop(int data, int pos) //在pos下一个插入 使用迭代

{

LineList\* head;

int i;

if (pos == -1) //头部添加 其实只是在头部后添加 然后交换成员变量

{

LineList\* buffer = this->next;

LineList\* newNode = new LineList(this->data);

head->next = newNode;

newNode->next = buffer;

this->data = data;

}

for (head = this, i = 0; i < pos; head = head->next, i++)

{

if (head->next == NULL) //判断空指针

{

std::cout << "NULL next!" << std::endl;

return;

}

}

//迭代至pos节点

if (head->next == NULL) //尾部添加

{

LineList\* newNode = new LineList(data);

head->next = newNode;

}

else //中间插入

{

LineList\* buffer = head->next;

LineList\* newNode = new LineList(data);

head->next = newNode;

newNode->next = buffer;

}

}

void LineList::addNodeCall(int data, int pos) //递归方式

{

if (pos == 0) //头部添加 其实只是在头部后添加 然后交换成员变量

{

LineList\* buffer = this->next;

LineList\* newNode = new LineList(this->data);

this->next = newNode;

newNode->next = buffer;

this->data = data;

}

if (pos != 0)

{

if (this->next == NULL) //判断空指针

{

std::cout << "NULL next!" << std::endl;

return;

}

this->next->addNodeCall(data, pos - 1);

//进入递归

}

else

{

if (head->next == NULL) //尾部添加

{

LineList\* newNode = new LineList(data);

head->next = newNode;

}

else //中间插入

{

LineList\* buffer = head->next;

LineList\* newNode = new LineList(data);

head->next = newNode;

newNode->next = buffer;

}

}

}

}