动态链表

作者：李俊烨 栾子骞

1. 实验目的：

要求：

链表是一种重要的数据结构，需要动态的进行存储分配，要求通过函数分别实现动态链表的建立、结点的插入、结点的删除以及链表的输出。

1. 实验内容：

编写链表、节点建立、插入、寻找、删除函数，以及链表的输出函数

1. 实验方案：

**基础编写**：

首先确定节点以及链表结构：（以储存数据为整型为例）

typedef struct \_node {

int value;

struct \_node\* next;

}Node;

typedef struct \_list {

Node\* head;

Node\* tail;

}List;

其次引入<stdlib.h>，以Node\* p = (Node\*)malloc(sizeof(Node));这行代码为核心，考虑特殊情况head==NULL，然后分情况接入p

再者利用循环for (p = list->head; p; p = p->next)，对列表进行遍历从而完成输出

**进阶及改进**：

1.链表升级

将单向链表升级成双向链表，即在Node中加入struct \_node\* last;，在add函数中添加对last的初始化。

2.函数编写

以free（）函数为核心编写删除节点函数，通过遍历和判断完成对返回特定节点值，节点的接入的实现

3.增加代码安全性

加入对某些函数参数的指针加入前缀const，如int getValue(const List\* list, int index)、void print(const List \* list)等。其次将删除函数delNode的返回值由void 改为int ，可以判断是否删除成功；

四，实验效果：



五，实验结论：

收获：

1. 了解和学习了c语言的一个新的数据储存结构；
2. 在添加节点或者删除节点过程中，由许多判读和复杂的地址赋值，通过过程中不断试错、理清思路、修改，对指针的使用有逐渐熟练；
3. 通过与java中容器的使用以及面向对象编程的特点相比较，我感受到c语言指针灵活但不安全的特点；

后续改进：

1. 丰富链表的类型和功能函数
2. 增强安全性，防止如连续使用指向NULL的指针等问腿。