Sam大叔

"if you ever want something badly,let it go.if it comes back to you,then it's yours forever.if it doesn't,then it was never yours to begin with."

导航
博客园
首页
新随笔
联系
订阅 XML
管理

< 2021年11月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类
iphone学习(1)
STL 容器(3)
设计模式(1)
数据结构-C/C++(6)

随笔档案
2013年12月(1)
2013年3月(1)
2013年2月(1)
2013年1月(6)
2012年12月(5)

阅读排行榜
1. STL之list容器详解(89983)
2. STL之vector容器详解(46193)
3. STL之deque容器详解(40207)

C++ 单链表基本操作分析与实现

链表

链表是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑顺序是通过链表中的指针链接次序实现的。链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。每个结点包括两个部分：一个是存储数据元素的数据域，另一个是存储下一个结点地址的指针域。 相比于线性表顺序结构，链表比较方便插入和删除操作。

创建头节点

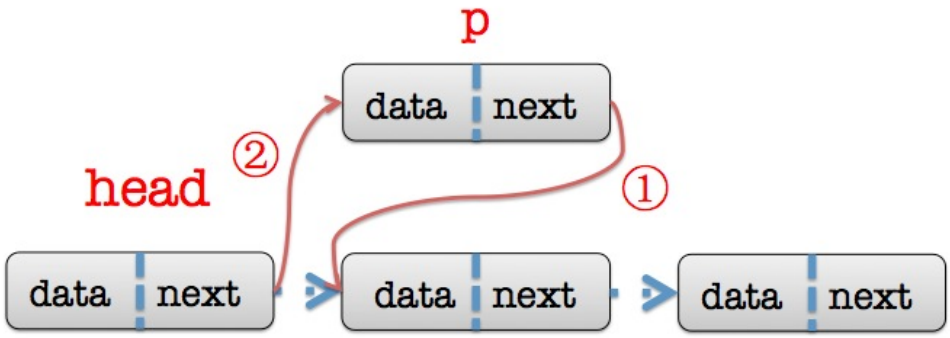
手动new一个新的Node，将Node的next置为NULL即可。

head = new Node(0);head->next = NULL;

从头插入一个新的节点

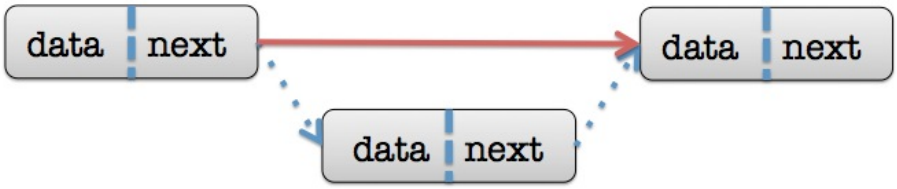
手动new出一个新的节点p，使p的next的指向head->next所指向的地址,然后将head->next从新指向p即可。

Node * p = new Node(int); p->next = head->next; head->next = p;



删除指定节点

先遍历到指定节点的前一个节点,然后通过将前一个节点的next指针指向指定节点的下一个节点,达到悬空指定节点的效果,然后删除指定节点即可。代码请参照后面的完整代码。



修改指定节点

遍历到指定节点的位置,将其data修改为要修改的值即可。修改代码请参考后面的完整代码。

4. C++ 单链表基本操作分析与实现(31613)
5. UIAlertView笔记(14540)

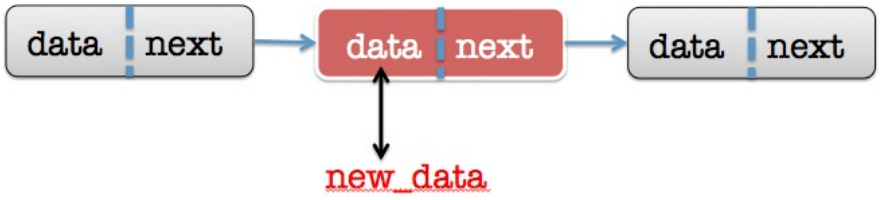
评论排行榜
1. STL之list容器详解(7)
2. STL之vector容器详解(6)
3. STL之deque容器详解(2)
4. Mac下Qt连接MySQL 驱动问题(1)
5. C++ 单链表基本操作分析与实现(1)

推荐排行榜
1. STL之list容器详解(11)
2. STL之vector容器详解(9)
3. STL之deque容器详解(3)
4. C++ 单链表基本操作分析与实现(3)
5. UIAlertView笔记(1)

最新评论
1. Re:STL之vector容器详解

谢谢老哥，着实搞明白了！

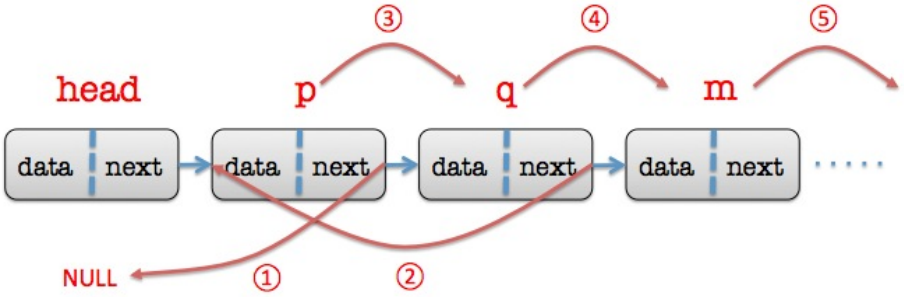
--whiteh
2. Re:Mac下Qt连接MySQL 驱动问题
读到最后一句，确实泪流满面
--findumars
3. Re:STL之deque容器详解
写的很棒，转一下
--张松超
4. Re:STL之list容器详解
int a[5] = {1,2,3,4,5}; list<int> a1;
list<int>::iterator it; a1.assign(2,10);
for(it = a1.begin();i...
--谨==
5. Re:STL之list容器详解
NICE!
--codersssss



链表反转

定义三个临时节点指向头结点之后的第1个节点p,第2个节点q和第3个节点m。将p->next置为空,然后将q->next = p,然后将p向后移动一个节点,即p = q,最后将q向后移动一个节点,即q = m,最后把m向后移动一个节点,即m = m->next;依此类推直到m等于NULL,然后将q->next = p,最后将head->next指向q(即目前第一个节点疑,也就是原本最后的一个节点)。

通过三个节点达到从头开始逐个逆序的目的。反转代码请参考后面的完整代码。



完整代码如下：

```
1 //
2 //  List.cpp
3 //  List
4 //
5 //  Created by scandy_yuan on 13-1-6.
6 //  Copyright (c) 2013年 Sam. All rights reserved.
7 //
8
9 #include <iostream>
10 using namespace std;
11
12 class List {
13 public:
14     List(){create_List();}
15     ~List(){clear();}
16     //创建头结点
17     void create_List();
18     //插入函数
19     void insert(const int& d);
20     //在指定位置插入
21     void insert_pos(const int& d,const int& d1);
22     //删除指定数据的节点
23     void erase(const int& d);
24     //修改指定数据
25     void updata(const int& d,const int& d1);
26     //反转链表函数
27     void reverse();
28     //打印
29     void print();
30 private:
31     //节点结构
32     struct Node{
33         int data;
34         Node * next;
```

```
35          Node(const int& d):data(d),next(NULL){}
36      };
37      Node * head;//头节点
38      //清理链表函数
39      void clear(){
40          Node * p = head;
41          //从头节点开始循环删除
42          while(p){
43              Node * q = p->next;
44              delete p;
45              p = q;
46          }
47      }
48      //查找数据d的上一个节点位置的函数
49      //为了方便后面删除操作
50      Node* find(const int& d){
51          Node * p = head;
52          for(;p;p=p->next){
53              if(p->next->data==d)
54                  break;
55          }
56          return p;
57      }
58  };
59
60  //创建头结点
61  void List::create_List()
62  {
63       head = new Node(0);
64  }
65  //从头插入一个节点
66  void List::insert(const int& d)
67  {
68      Node *p = new Node(d);
69      p->next = head->next;
70      head->next = p;
71  }
72  //打印函数
73  void List::print()
74  {
75      for(Node * p = head->next;p;p=p->next){
76          cout << p->data << endl;
77      }
78  }
79  //在d位置之前插入d1
80  void List::insert_pos(const int& d,const int& d1)
81  {
82      Node * p = find(d);
83      Node * q = new Node(d1);
84      q->next = p->next;
85      p->next = q;
86  }
87
88  //删除
89  void List::erase(const int& d)
90  {
91      Node * p = find(d);
92      //因为p是上一个节点的位置，用q来保存
93      //要删除的节点的地址
94      Node *q = p->next;
95      //通过将上一个节点的next指针指向要删除节点的next指
96      //针志向的节点实现断开要删除节点的目的
97      p->next = p->next->next;
98      //删除要删除的节点q
99      delete q;
100 }
101
102 //修改指定数据
103 void List::updata(const int& d,const int& d1)
104 {
105     Node * p = find(d);
106     p->next->data = d1;
```

```
}

//反转链表
void List::reverse()
{
    Node * p = head->next;//头结点之后的第1个节点
    Node * q = head->next->next;//头结点之后的第2节点
    Node * m = head->next->next->next;//头结点之后的第3个节点
    p->next = NULL;//将头接点之后的第1个节点的next指针置为空
    //根据m是否为空来判断 以此逆序每一个节点
    while(m){
        q->next = p;
        p = q;
        q = m;
        m = m->next;
    }
    //将最后一个节点逆序
    q->next = p;
    //将头从新指向新的的第1个节点(之前的最后一个节点)
    head ->next = q;
}

int main(int argc, const char * argv[])
{

    // insert code here...
    List list;
    list.insert(30);
    list.insert(20);
    list.insert(10);
    list.insert_pos(10, 5);
    list.print();
    cout << "----------------------" << endl;
    list.erase(10);
    list.print();
    cout << "----------------------" << endl;
    list.reverse();
    list.print();
    cout << "----------------------" << endl;
    list.updata(5, 8);
    list.print();
    return 0;
}
```

分类: 数据结构-C/C++

Sam大叔
关注 - 0
粉丝 - 33
+加关注

« 上一篇： 括号匹配程序
» 下一篇： STL之vector容器详解

posted on 2013-01-06 15:39 Sam大叔 阅读(31613) 评论(1) 编辑 收藏 举报