**线性表**

1. 上节我们提到的是顺序表现在我们来讲链式存储
   1. 我们现在用链式来存储，所以我们就不要担心内存不够的问题
   2. 那废话不多说，我们直接上代码
2. 具体操作

#include <iostream>

#include <stdio.h>

#include <iomanip>

#include <string>

#include <assert.h>

#include <time.h>

template <typename T>

class LinkList{

int n; // 链表的长度

struct LNode{

T data;

LNode \*next;

};

LNode \*head;

public:

LinkList(){

head = new LNode();

head->next = 0; // 认为是空链表

n = 0;

}

//获取到位置 i 的值赋值给e

bool get(int i, T &e){

assert(i>=0 && i <= n);

LNode \*p = head;

int j = 0;

while(p && j < i){

p = p->next;

}

if(!p) return false;

e = p->data;

return true;

}

// 在最后加上一个值

bool push\_back(T e){

LNode \*p = head;

while(p->next){

p = p->next;

}

p->next = new LNode(); //重新开辟一个节点来存储新push的数据

if(!p->next) return false; //空间开辟失败

p->next->next = 0; // 维持之前0是空的状态

p->next->data = e;

n++; // 数量加1

return true;

}

//用来接收打印的指针函数

bool traverse(void (\*func)(T &e)){

LNode \*p = head;

while(p){

func(p->data);

p = p->next;

}

std::cout << std::endl;

}

//获取长度

int getSize(){

return n;

}

//现在我们用下标操作符[]获取元素

T operator[](int i)const{

LNode \*p = head;

int j = 0;

while(p &&j++ < i){

p = p->next;

}

return p->data;

}

T& operator[](int i){

LNode \*p = head;

int j = 0;

while(p &&j++ < i){

p = p->next;

}

return p->data;

}

};

template <typename T>

void printLinkList(T &e){

std::cout << e << " ";

}

int main()

{

LinkList<char> list;

list.push\_back('A');

list.traverse(printLinkList);

int len = list.getSize();

std::cout << list[1];

return 0;

}

剩下的内容自己搞定就ok了，是不是